FACULDADES INTEGRADAS DE CARATINGA

FACULDADE DE CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UMA API PARA INTEGRAR FRAMEWORKS PARA IMPLEMENTAÇÕES DE APLICAÇÕES WEB

JOÃO PAULO CONSTANTINO

Caratinga

2013

João Paulo Constantino

DESENVOLVIMENTO DE UMA API PARA INTEGRAR FRAMEWORKS PARA IMPLEMENTAÇÕES DE APLICAÇÕES WEB

Monografia apresentada à banca examinadora da Faculdade de Ciência da Computação das Faculdades Integradas de Caratinga, como requisito parcial para obtenção do título de bacharel em Ciência da Computação, sob orientação do professor Msc.Glauber Luis da Silva Costa.

Caratinga

2013

João Paulo Constantino

DESENVOLVIMENTO DE UMA API PARA INTEGRAR FRAMEWORKS PARA IMPLEMENTAÇÕES DE APLICAÇÕES WEB

Monografia submetida à Comissão examinadora designada pelo Curso de Graduação em Ciência da Computação como requisito para obtenção do grau de Bacharel.

_____

Prof. Msc. Glauber Luís da Silva Costa
Faculdades Integradas de Caratinga

_____

Prof. Wanderson Silva
Faculdades Integradas de Caratinga

_____

Prof. Jonilson Batista
Faculdades Integradas de Caratinga

Caratinga, 12 /2013

Agradecimentos

Agradeço primeiramente a Deus, pois ele me deu força nas horas mais difíceis e cuidou de min quando mais precisei. Agradeço principalmente a minha mãe por ter sido uma guerreira todos esses anos, trabalhando muito para que esse sonho se concretizasse, agradeço ao meu pai por não medir esforços para me ajudar a concluir os estudos.

Meus amigos também foram importantes, mas existem quatro pessoas na qual eu não posso deixar de cita-las diretamente, pois se elas não tivessem estendido a mão quando eu mais precisei hoje eu não estaria concluindo o curso, começando por Ciminio Pontes que me ajudou logo no início da faculdade, já no 3º período quando as minhas esperanças de continuar estudando já não existiam, a minha coordenadora Fabricia Pires Tiola junto com meu grande amigo Flavio Costa me deram uma oportunidade que possibilitou que continuasse a estudar. Outra pessoa que me deu força pra continuar e me ajudou a tomar decisões que parecem ter sido as corretas foi José Roberto de Souza, além do professor, ex-chefe e um amigo Jacson Rodrigues da Silva que me mostrou que eu era capaz de muita coisa até quando eu mesmo duvidei.

Outras pessoas que foram circunstancias para a conclusão do trabalho foi meu diretor Hudson Silva, que apostou no trabalho que viria a ser desenvolvido e bancou alguns custos, como cursos para que me aperfeiçoasse no trabalho a ser desenvolvido, e meu orientador Glauber Costa que muitas vezes puxou o freio de mão com as ideias que nós tínhamos, que estavam além de um trabalho acadêmico, à minha coorientadora Fabrícia Pires Tiola que me atendeu em diversos momentos que precisei de auxilio com o trabalho.

Obrigado!

“Controlar os outros é força, controlar-se a si próprio é verdadeiro poder.”

Lao Tsé

Resumo

Com o aumento da demanda e exigências por aplicações Web, as empresas e setores de Tecnologia da Informação (TI) precisam atender às necessidades de seus clientes no menor tempo possível, mas sem perder a qualidade de seus serviços. Mas aplicações Web são complexas de serem desenvolvidas, além das tecnologias usadas nessas aplicações estarem atualizando constantemente.

Este trabalho teve como objetivo desenvolver uma *Application Programming Interface* (API) para facilitar a fase de desenvolvimento de software para Web. A API desenvolvida utiliza alguns componentes do *Twitter Bootstrap* (TB2) que podem ser manipulados através de objetos e auxiliar o desenvolvedor na criação de interfaces de uma página Web. Esses componentes agregados aos componentes do *Zend Framework 2* (ZF2), fazem com que os desenvolvedores possam reutilizar uma gama de funcionalidades para criar aplicações Web. Essa API foi responsável por reunir em um único ambiente de desenvolvimento Web, ferramentas que agilizam a criação de componentes que uma aplicação Web contém.

Para propor esse ambiente foi estudado sobre o ZF2 e TB2, conhecendo assim o que cada um podia agregar de valor para o ambiente. Para desenvolver a API foi utilizado a metodologia *Hot Spot*, para desenvolver os componentes que compõe a API, foram coletados os requisitos e feita uma modelagem completa dos componentes, em seguida foi realizada a implementação da API.

Com o estudo desse trabalho foi possível concluir que desenvolver aplicações para Web é um processo muito complexo, mas com o ambiente proposto pelo trabalho, a complexidade dessas aplicações pode diminuir, pelo fato do ambiente já agregar uma gama de componentes prontos para serem reutilizados pelos desenvolvedores. Além de contar com uma API que pode facilitar o processo de manipular os componentes do TB2.

Palavras-chave: Web; API; *Software; Framework*.

Abstract

With the increase of the demand and and requirements for web applications, the Information Technology (IT) companies and sectors must attend to the needs of their clients in the shortest time possible, but without the quality of their services. But web applications are complex to be developed, in addition to the technologies used in these applications are being constantly updated.

This work had as an objective develop an Application Programming Interface (API) to facilitate the development fase of software for Web. The developed API uses some Twitter Bootstrap (TB2) components wich can be manipulated through objects and support the developer in the creation of Web page interfaces.

These components aggregated to the Zend Framework 2 (ZF2) components, make that developers may reuse a gamma of functionalities to create Web applications. This API was responsible to reunite in a single web development environment, tools that speeds up the creation of components contained in a web application.

To propose this environment, was studied about the ZF2 and TB2, thus knowing what value which one could aggregate to the environment. To develop the API it was used the Hot Spot methodology, to develop the components wich compose the API, the requirements were colected and was made a complete modeling of the components, hereafter was realised the implementation of the API. With the study of this work it was possible to conclude that develop We applications is a very complex process, but with the environment proposed by this work, the complexity of these applications can decrase, by the fact that the environment already aggregates a gamma of components ready to be used by the developers. Besides o countig with an API wich can facilitate the process of manipulate the TB2 components.

Keywords: Web; API; Software; Framework;

Lista de siglas

API – *Application Programming Interface* (Interface de programação de aplicativos)
CSS - *Cascading Style Sheets*
HP - (Hewlett-Packard)
HTML - Hyper Text Markup Language (Linguagem de Marcação de Texto)
HTTP - *Hypertext Transfer Protocol* (Protocolo de Transferência de Dados)
IBM - *International Business Machines*
MVC - Model - View – Controller (Modelo – Visão - Controladores)
PHP – *Hypertext Preprocessor* (Processado de Hipertextos)
POO – Programação Orientada a Objetos
TB - Twitter Bootstrap
TB2 – Twitter Bootstrap 2
TI – Tecnologia da Informação
UML - *Unified Modeling Language* (Linguagem Unificada para Modelagem)
URL - *Uniform Resource Locator*
W3C - *Word Wide Web Consoritun*
ZF2 – Zend Framework 2

Lista de Ilustrações

Figura 1 - Apresenta um exemplo de barra de navegação. 27
Figura 2 - Apresenta a utilização do componente TabBar 27
Figura 3 - Apresentação da BarraNavegacao em funcionamento. 30
Figura 4 - Apresentação do componente TabBar.....31
Figura 5 - Apresentação das opções de cores dos botões do Twitter Bootstrap.....33
Figura 6 - Apresentação das opções de tamanho dos botões. 34
Figura 7 - Apresentação de menu desenvolvido utilizando Acordeom.....35
Figura 8 - Apresentação da Arquitetura entre os frameworks e a API. 37
Figura 9 - Apresentação da modelagem do componente Elemento. 38
Figura 10 - Representação da modelagem do componente BarraNavegacao. 40
Figura 11 - Apresentação do componente TabBar.....41
Figura 12 - Representação da modelagem do componente botões. 42
Figura 13 - Apresentação da modelagem do componente Acordeom. 43

SUMÁRIO

1. Introdução 7
2. Referencial Teórico 9
2.1. A internet e a web 9
2.2. Softwares 10
2.3. Programadores de *Softwares* 11
2.4. Aplicações Web 11
2.4.1. Linguagem de Marcação 12
2.4.2. Linguagem de Formatação 12
2.4.3. Linguagem de Script 13
2.4.4. Linguagens de Programação 13
2.5. Vantagens e Desvantagens Sobre Aplicações Web 14
2.5.1. Vantagens 14
2.5.2. Desvantagens 15
2.6. UML 16
2.7. Programação Orientada a Objetos 16
2.7.1. Classe 17
2.7.2. Objeto 17
2.7.3. Herança 18
2.7.4. Polimorfismo 18
2.8. Reusabilidade 19
2.9. Padrões de Projeto 20
2.11. *Frameworks* 21
2.12. Twitter Bootstrap 23
2.13. Zend Framework 2 23
2.14. Hot Spot 24
3. Metodologia 25
3.1. Estudo e escolha das ferramentas 25
3.1.1. Zend Framework 2 25
3.1.2. Twitter Bootstrap 26
3.2. Componentes para Composição da API 28
3.3. Requisitos Para os componentes da API 29
3.3.1. Requisitos Básicos 29
3.3.2. BarraNavegacao 29
3.3.3. Tab Bar 31

3.3.4. Botao 32
3.3.5. Acordeom 34
3.4. Implementação da API 35
4. Resultados 37
4.1. Implementações Básicas 38
4.2. Componente BarraNavegacao 39
4.3. Tab Bar 40
4.4. Botões 42
4.5. Acordeom 43
4.6. Conclusão da API Desenvolvida 43
5. Conclusão 45
6. Trabalhos Futuros 46
Referências 47

1. INTRODUÇÃO

A internet deixou de ser um meio só de entretenimento e virou um dos grandes veículos de comunicação, sendo usada por grandes e pequenas empresas para auxiliar o seu negócio e suas tarefas como:

- Divulgar os seus produtos e serviços;
- Realizar vendas;
- Conhecer mais sobre perfil de seus clientes por meio de redes sociais.

Algumas empresas usam a internet como principal meio de negócio, podendo citar *Google*, *Facebook*, *Yahoo*, *Amazon*, sendo estas estrangeiras e algumas empresas brasileiras como o Submarino, *Netshoes*, Saraiva, lojas de produtos comercializados pela internet.

A internet nem sempre é usada diretamente como meio de negócio pelas empresas. Pode ser usada para auxiliar e agilizar tarefas do dia-a-dia através de *softwares*, melhorando o desempenho de seus funcionários e colaborando nas tomadas de decisões, por meio de relatórios de dados financeiros, entre outros tipos de dados de acordo com o domínio da empresa. Esses *softwares* que utilizam a internet para transmissão dos dados também são conhecidos como aplicações Web.

Outro grande fator para utilizar aplicações Web, é que as pessoas estão acessando por mais tempo na internet e isso pode ser uma ponte de acesso entre as empresas e os seus consumidores. Segundo uma pesquisa do Ibope "*O Brasil ocupa a terceira posição em quantidade de usuários ativos na internet (52,5 milhões)*". "*Se por um lado, o Brasil fica em terceiro lugar em número de usuários, por outro, o país é o primeiro colocado quando considerado o tempo de acesso de cada internauta*", (IBOPE, 2013).

Com essa grande expansão da internet, a demanda de aplicações Web para as empresas e setores de TI cresceu muito. Por isso é necessário acompanhar o mercado para que estas empresas não percam o seu espaço ou deixem de conquistar novos consumidores. Porém, o desenvolvimento de aplicações para plataforma Web é complexo, por ser necessário o uso de várias tecnologias para se criar uma aplicação e realizar a junção dessas tecnologias usadas no

desenvolvimento para formar uma aplicação final fica ainda mais complexo (BUSTAMANTE, 2008).

Outro fator a ser considerado com relação ao desenvolvimento de aplicações Web diz respeito a questões como: acessibilidade em diversos dispositivos, plataformas, compatibilidade entre os navegadores, segurança da informação, usabilidade, desempenho e integridade dos dados (DANTAS, 2009).

Uma possível solução para esses problemas é a criação de um ambiente integrado para o desenvolvimento. Basicamente esse ambiente é composto por diferentes *frameworks,* cada um responsável por uma camada da aplicação Web. O *framework* fica então com a responsabilidade de auxiliar o desenvolvedor a agilizar o desenvolvimento e obter soluções para problemas recorrentes existentes em cada camada.

Esses *frameworks* são formados por diversos componentes que são reutilizados para desenvolver aplicações, sendo assim não existe a necessidade dos desenvolvedores criar soluções para problemas já solucionados, bastando-se apenas fazer o reúso do mesmo.

Com esse ambiente criado, alguns problemas básicos mais difíceis de resolver já seriam solucionados, como: incompatibilidade entre navegadores com novas tecnologias, acessibilidade em diversos dispositivos, base da aplicação já estaria pronta, algumas funcionalidades prontas para serem reutilizadas, aplicação mais confiável por estar usando tecnologias que são atualizadas constantemente, diminuição do tempo de desenvolvimento, entre outras vantagens.

Realizar a junção de diferentes *frameworks* não é uma tarefa simples, pois exige do desenvolver experiência em utilizar os *frameworks* individualmente. Neste trabalho foi proposto o desenvolvimento de uma API para facilitar a comunicação entre os *frameworks* trazendo benefícios como, manipular os componentes do TB2 com objetos em PHP, evitando a manipulação desses componentes direta com as tags HTML que exibe mais escrita de código, além de não exigir experiência do desenvolver sobre os *frameworks*.

Para desenvolver essa API foi utilizada a metodologia *Hot Spot*. Seguindo os processos ditados pela mesma, foram coletados os requisitos básicos para API, requisitos específicos dos componentes a serem criados. Foram efetuadas as modelagens sobre os requisitos, e por fim a implementação base da API, seguido de seus componentes.

2. REFERENCIAL TEÓRICO

Nas próximas sub sessões serão abordados os temas nos quais estre trabalho científico tem fundamentado seu referencial teórico. Com essas abordagens descritas no trabalho facilitará o entendimento do trabalho a ser desenvolvido.

2.1. A INTERNET E A WEB

A Internet segundo Deitel (2003, p.5) "*uma rede global de computadores*". Para Tanenbaum (2003, p.54) a Internet é "*conjunto de redes que utilizam certos protocolos comuns e fornecem determinados serviços comuns*". A internet teve início na década de 1960 nos Estados Unidos da América em seu departamento de defesa, o seu proposito original era apenas conectar universidades espalhadas no território Americano, mas com o surgimento da Web a internet se expandiu, e atualmente é acessível a bilhões de computadores e dispositivos com acesso a internet (DEITEL, 2003).

A Internet permite que computadores troquem informações entre si. Seguindo a arquitetura cliente e servidor um programa cliente em uma máquina pode acessar informações do programa servidor em outra máquina possibilitando assim a comunicação dos mesmos (NIELSEN, 1998).

Web é uma arquitetura que faz uso da internet para executar suas aplicações, teve início da década de 90 em Genebra na Suíça. No início era destinado a físicos e cientistas e foi desenvolvida com o intuito de facilitar o acesso a informação de qualquer lugar do mundo. Mesmo com todo esse avanço a Web ainda continua baseada em seus princípios fundamentais, que é a computação cliente e servidor, ou seja, os servidores armazenam os documentos e os clientes acessam os mesmos (BUSTAMANTE, 2008).

Para transmitir essas informações além da internet, a Web utiliza quatro ferramentas, sendo elas: HTTP, Arquivos Hipertextos, URL e *Browser*.

HTTP é o protocolo para transmissão de dados utilizado em toda Web (TANENBAUM, 2003).

> "*O protocolo de transferência utilizado em toda a World Wide Web é o HTTP (HyperText Transfer Protocol). Ele especifica as mensagens que os clientes podem enviar aos servidores e que respostas eles receberão. Cada interação consiste em uma solicitação ASCII, seguida por uma resposta RFC 822 semelhante ao MIME. Todos os clientes e todos os servidores devem obedecer a esse protocolo*". Tanenbaum (2003, p. 493).

Hipertextos são arquivos que contém todo conteúdo de uma página de internet ou de uma aplicação Web, possuindo elementos para marcação do conteúdo da página, podendo ser estáticos ou podem ser gerados dinamicamente pelo servidor responsável pela página Web (JAZAYERI, 2007).

URL são nomes responsáveis por referenciar documentos na internet, esses nomes equivalem aos endereços originais de cada computador na rede, conhecidos como IP. Essas URLs são usadas para referenciar além de aplicações na Web também são usadas para identificar documentos na internet (JAZAYERI, 2007).

Browser ou navegador são programas executados nos computadores dos usuários de internet, sendo responsáveis por fazer as requisições pelas páginas que usuários solicitam e além disso interpreta o conteúdo retornado pelo servidores das páginas, o conteúdo é recebido em forma de documentos hipertextos e o navegador exibem esses arquivos de forma amigável para o usuário final, ou seja com uma interface contendo botões, tabelas, listas entre outras estruturas para exibir o conteúdo da página (JAZAYERI, 2007).

2.2. SOFTWARES

Computadores processam dados sob controle de um conjunto de instruções chamadas de programa de computador Deitel (2010). Esses programas também são conhecidos como software.

Na atualidade, os *softwares* estão cada vez mais presentes no dia a dia das pessoas. Quando usuários de internet acessam redes sociais a página acessada é composta por um *software,* além disso, para construir a página são utilizados *softwares.* E não é só no meio de entretenimento que os *softwares* aparecem, quando clientes de um banco acessam caixas eletrônicos, aplicativos para celular sendo de qualquer finalidade, como um jogo, uma agenda ou até mesmo algum aplicativo que é usado a trabalho.

2.3. PROGRAMADORES DE *SOFTWARES*

Programadores são pessoas que trabalham no desenvolvimento de *software*, como dito no tópico anterior esses *softwares* são compostos por instruções que executam uma determinada tarefa no sistema, os programadores são responsáveis por implementar essas instruções, que também são conhecidas como rotinas ou algoritmos

Em um sistema de computador os algoritmos são responsáveis por executar a lógica da aplicação ou regra de negócio, e para cada componente desse sistema existe uma regra de negócio, para no final todas as instruções definidas compor um *software* (BINI, 2009).

2.4. APLICAÇÕES WEB

Aplicações Web são construídas sobre o conceito original da Web. São compostas por documentos hipertextos que exibem o conteúdo da aplicação para o usuário (JAZAYERI, 2007).

Essas aplicações deixaram de ser apenas documentos de texto e passaram a ter mais interatividade com o usuário, essa evolução é devido às novas tecnologias agregadas a esses documentos. Essas tecnologias, além de textos estáticos possuem programas executando em tempo real no servidor da aplicação, que a cada requisição gera novos conteúdos para essas páginas.

Além dos programas executando no servidor existem scripts que são executados no próprio navegador do cliente, esses scripts também são programas e são os maiores responsáveis pela evolução das aplicações Web (BARESI, 2007).

Havia anteriormente um último parágrafo que eu sugeri que você alterasse para que servisse de base para criar um link entre essa seção e as seções subsequentes.

2.4.1. Linguagem de Marcação

Linguagem usada para fazer a marcação de conteúdo das páginas Web, contendo vários elementos que são usados pelos desenvolvedores, como: tabelas, listas, parágrafos, imagens, links para outras páginas Web e formulários. Esses elementos são responsáveis por marcar os conteúdos gerados pelas páginas. Umas das linguagens que pertencem a esse domínio, é o *Hyper Text Markup Language* (HTML), ou seja, linguagem de marcação de texto, essa linguagem foi criada na década de 90 com o intuito de ser utilizada para gerar arquivos contendo as informações a serem acessadas pelos usuários.

Em janeiro de 2008 a *Word Wide* Web *Consoritun (*W3C). Órgão regulamentador dos padrões a serem utilizados nas tecnologias e métodos utilizados nas aplicações e páginas Web publicou uma nova versão dessa tecnologia, chamada de HTML 5. Essa versão trouxe muitos avanços para as aplicações Web, como novos elementos sendo eles, adicionar vídeos em páginas sem ter que usar plug-ins de outras tecnologias, fazer desenhos em 3D sem utilizar imagens e mais dinamismo e interatividade a essas páginas (W3C, HTML, 2013).

2.4.2. Linguagem de Formatação

As *Cascading Style Sheets* (CSS), Folhas de Estilo em Cascata, é uma linguagem usada para formatar conteúdos das páginas, como: lista, tabelas, textos, fontes, links, formulários e todos os elementos HTML.

A linguagem CSS fornece uma variedade de propriedades para estilização de cada conteúdo da página, como: largura, altura, alinhamento, disposição entre um elemento e outro, permitindo também que os desenvolvedores possam adaptar suas páginas para serem acessadas em diversos aparelhos eletrônicos com telas distintas, exemplos: desktops, notebooks, celulares e até mesmo televisões entre outras telas (W3C, CSS, 2013).

2.4.3. Linguagem de Script

Script são códigos que podem estar embutidos no próprio documento HTML ou estar em arquivos separados, mas inclusos no arquivo HTML. Segundo a W3C SCRIPT (2013) "*Scripting normalmente se refere ao código de programa escrito em JavaScript que é executado pelo navegador quando a página é baixada*".

Scripts permitem aos desenvolvedores aumentar a interatividade das páginas, por meio de eventos disparados quando o usuário pressiona uma tecla, podem ser eventos de movimento do mouse ou click. Além de eventos, Scripts podem ser usados para validar dados do usuário quando o mesmo preenche os campos de um formulário (BUSTAMANTE, 2008).

2.4.4. Linguagens de Programação

Essas linguagens são responsáveis por construir as páginas que são acessadas pelos usuários, exemplo, quando um usuário faz uma pesquisa em um site ele envia uma requisição para o servidor o mesmo encaminha para aplicação e a linguagem de programação que foi usada para implementar a aplicação processa os dados e retorna a nova página para o usuário final.

Essas linguagens também são responsáveis por permitir o desenvolver escrever todas as regras de negócio do lado do servidor (JÚNIOR, 2005).

2.5. VANTAGENS E DESVANTAGENS SOBRE APLICAÇÕES WEB

Nas duas próximas sessões será abordado sobre as principais vantagens e desvantagens sobre utilizar e desenvolver aplicações Web.

2.5.1. Vantagens

Quando um *software* é produzido para executar sobre à arquitetura Web ele possui algumas vantagens.

As aplicações Web não precisam ser instaladas em cada máquina que for fazer uso da aplicação, pois, elas são instaladas em uma máquina que provê essa aplicação de forma on-line, bastando-se apenas que as máquinas clientes tenham um navegador para acessar essas aplicações, (BUSTAMANTE, 2008).

Quando uma aplicação executa sobre a plataforma Web logo ela não precisa de versões distintas para diferentes sistemas operacionais, isso evita que os desenvolvedores implementem versões para cada sistema operacional existente no mercado utilizado pelos usuários, Bustamente (2008). Além dessas vantagens qualquer manutenção ou atualização da aplicação é realizada em um único ponto, atingindo assim a todos os clientes da aplicação (NIELSEN, 1998).

Uma aplicação que executa na plataforma Web tem um poder de alcance maior, pois, qualquer usuário que tenha acesso à internet logo tem acesso à aplicação (BUSTAMANTE, 2008).

Exige muito menos de poder de processamento das máquinas clientes ao executar a aplicação, pois, a aplicação é executada em uma máquina servidora que já faz todo trabalho pesado para as máquinas clientes (NIELSEN, 1998).

2.5.2. Desvantagens

Uma das principais desvantagens é o fato da arquitetura da Web não ter sido projetada para executar aplicações e sim documentos de texto. Mas devido ao sucesso da internet deixar de executar aplicações na plataforma Web seria um grande desperdício de uma dos maiores avanços da tecnologia (BUSTAMANTE, 2008).

Essas aplicações podem ser acessadas de diversos dispositivos sendo que cada um possui características distintas como resolução, alguns tem uma resolução muito alta como televisões, já outros tem uma resolução baixa como celulares e, ainda temos a incompatibilidade nos navegadores instalados nesses dispositivos com as novas tecnologias utilizadas na Web (BUSTAMANTE, 2008).

Em uma aplicação cliente-servidor tradicional, o cliente pode usar a mesma conexão e utiliza-la para executar outros processos, mas em aplicações Web para cada requisição uma nova conexão é iniciada com o servidor (CHANG, 2004).

A Web funciona por meio de requisições HTTP, os parâmetros dessas requisições transportam somente dados do tipo *String.* Ou seja, texto ou uma sequência de caracteres, logo esses dados tem que ser convertidos para outros tipos (BUSTAMANTE 2008).

HTML oferece um conjunto limitado de componentes para escrever interfaces para essas aplicações, logo são usadas outras linguagens como de Script para auxiliar no processo de criar essas interfaces, além da linguagem de formatação conhecida como CSS, esses pontos faz com que os desenvolvedores tenham que conhecer três linguagens totalmente distintas para criar aplicações Web (BUSTAMANTE, 2008).

Aplicações Web podem ser acessadas de qualquer parte do mundo bastando-se apenas que usuários tenham acesso à internet, logo elas estão mais propiciais a invasões, com isso a segurança dessas aplicações requer um cuidado muito maior que aplicações executam em máquinas locais (BUSTAMANTE, 2008).

Mas algumas dessas desvantagens os desenvolvedores conseguem contorna-las camuflando assim alguns desses problemas para o usuário final, além disso as tecnologias atualmente estão evoluindo muito nesse sentido auxiliando

assim os desenvolvedores criarem essas aplicações mais robustas (BUSTAMANTE, 2008).

2.6. UML

A *Unified Modeling Language* (UML) Linguagem Unificada para Modelagem, é uma linguagem de modelagem gráfica voltada para representação conceitual e física de um sistema computacional utilizado por engenheiros, arquitetos, gerentes e desenvolvedores de *software*.

Na década de 1980, um grande número de aplicações que começaram a utilizar o paradigma Orientação a Objetos, diversos autores como Booch, Rumbaugh e Jacobson, desenvolveram processos para descrever os componentes do paradigma de orientação a objetos, Deitel (2003). Já em 1994 James Rumbaugh associou-se a Grady Booch na *Rational Software Corporation,* hoje um grupo da IBM, e começaram a unificar os seus processos para descrever aplicações utilizando orientação a objetos e já em 1996 foi liberada as primeiras versões da UML (DEITEL, 2003).

Uma de suas grandes vantagens é sua flexibilidade, pois, permite que os modeladores de UML tenham liberdade para utilizar vários processos para projetar sistemas. Eles podem utilizar de diagramas, que são um conjunto de padrões para notações gráficas para representar alguns elementos do paradigma orientação a objetos.

2.7. PROGRAMAÇÃO ORIENTADA A OBJETOS

Em meados da década de 90 surgiu a necessidade de criar um novo paradigma para linguagens de programações, pois na época os computadores já tinham se tornado populares em vários países e muitas pessoas já usavam para executar as tarefas de trabalho (LISBOA, 2012).

No entanto um dos paradigmas mais utilizados na época que era a programação estruturada já não atendia as novas necessidades dos *softwares*, que precisavam fornecer mais qualidade e velocidade. Então com esses problemas foi criado o paradigma de Programação Orientada a Objetos (POO), (LISBOA, 2012).

Orientação a Objetos faz com que os sistemas sejam enxergados como uma coleção de objetos que trocam informações entre si, sendo assim, esse aspecto facilita a reutilização de código, além de englobar conceitos como abstração, hierarquização, classificação, modularização, relacionamento, simultaneidade e persistência, (DEITEL, 2010).

2.7.1. Classe

Classe é um conjunto de características de um elemento representado no mundo real dentro de um sistema e essas informações são definidas através de atributos também conhecidos como variáveis ou campos que armazenam as informações das características de uma classe (DEITEL, 2010).

Além de atributos, classe contém métodos que representam as ações que esse objeto pode executar, exemplo: um objeto que representa um carro contém os método de ligar, desligar, andar para frente e para trás, passar marchas entre outras ações (DEITEL, 2010).

2.7.2. Objeto

Objetos é um termo utilizado para representar um elemento do mundo real dentro de um sistema, exemplo: um objeto avião no sistema de gerenciamento de aviações áreas, esse objeto contém informações como números de vagas para passageiros, informações técnicas como quantidade de litros de combustível que comporta e, a quantos quilômetros por hora ele consegue chegar entre outras informações.

Esses objetos são instâncias de classes definidas pelos sistemas, onde todas as informações que forem necessárias armazenar em um objeto tem que ser declarada dentro de uma classe (DEITEL, 2010).

2.7.3. Herança

Herança é uma das formas de reutilizar *software*. Uma classe herda informações e características de uma classe pai; a classe que serve de base para outras classe é conhecida como superclasse; as classes que herdam de uma superclasse é conhecida como subclasse; em uma subclasse o desenvolvedor consegue reaproveitar todas as informações da superclasse além de poder criar novas funcionalidades ou modifica-las (DEITEL, 2012).

Uma subclasse pode ter suas propriedades separadas da sua superclasse porém a subclasse é utilizada para representar características mais específicas de um objeto, já uma superclasse trabalha de forma mais genérica para servir de base para um número maior de classes (DEITEL, 2012).

2.7.4. Polimorfismo

Polimorfismo permite que dados de tipos diferentes sejam tratados de forma homogênea, permitindo que o programador possa generalizar operações e tratar vários conjuntos de informações ao invés de unidades isoladas (LISBOA, 2012).

O polimorfismo permite de uma forma geral ou genérica ao invés de problemas específicos, facilitando a programação além de aumentar a reusabilidade de código. Outra grande vantagem é a extensibilidade que o polimorfismo agrega a programação de sistemas, isso faz com que a manutenção desses sistemas fique mais fácil e concisa. Um exemplo de polimorfismo: é necessário criar um sistema que armazene informações e ações de vários animais, sendo assim a grande maioria dos animais tem algumas características comuns, aproveitando essas características pode-se criar uma superclasse que contenha essas informações e

ações que eles têm em comum e tendo um reaproveitamento das funcionalidades na hora de desenvolver aos outras informações das classes específicas para cada espécie ou animal em específico (DEITEL, 2012).

2.8. REUSABILIDADE

A reusabilidade é uma abordagem de desenvolvimento que tenta maximizar o reúso de *softwares* existentes ou componentes individuais que compõem um sistema por completo e até mesmo por objetos e métodos específicos. Além dessas componentes, a reusabilidade pode ser aplicada a padrões de projetos e padrões de análise. Um padrão de projeto muito utilizado e agrega para a reusabilidade em *software* é o MVC descrito na próxima sessão (SOMERVILLE, 2007).

Uma das vantagens mais consideradas na reusabilidade é a redução de custos, como poucos componentes precisam ser especificados e projetados, implementados e validados. Um sistema que é desenvolvido utilizando a abordagem de reusabilidade agrega vantagens como:

- Confiança: *software* reusado já foi testado e corrigido em um ambiente real de trabalho (SOMERVILLE, 2007).
- Uso eficiente: ao invés de refazer trabalhos que já foram feitos por outros desenvolvedores, eles reaproveitam e agregam seus conhecimentos (SOMERVILLE, 2007).
- Desenvolvimento acelerado: o reúso diminui o tempo de desenvolvimento de um *software* porque tanto o tempo de implementação quanto o tempo de validação são diminuídos (SOMERVILLE, 2007).

O reúso de *software* não acontece de uma forma simples devendo ser planejado e implantado por meio de um programa que envolva toda a organização (SOMERVILLE, 2007).

2.9. PADRÕES DE PROJETO

De acordo com Sweat (2005, p, 18) o termo padrões de projeto foi criado na arquitetura de edificações e não na arquitetura de *software*. O arquiteto Christopher Alexander em 1977 publicou o livro *A Pattern Language* (Linguagens de Padrões) nesse livro ele abordava a ideia de agregar padrões já existentes na criação de novas edificações com algumas que já foram construídas, e está ideia de Alexandre foi trazida para a arquitetura de *software* por se encaixar exatamente nos conceitos de construir *softwares* (LISBOA, 2012).

Para Fowler (2006, p. 31), "*um elemento-chave dos padrões são aqueles que estão enraizados na prática*", ou seja utilizar conhecimentos já adquirido em outras experiências Lisboa (2012). Gutmans et all (2005, p.59) afirma que "*padrões de projetos são problemas que já foram abordados pela comunidade desenvolvedora de software e recebe soluções geralmente aceitas*". Segundo Gutmans et al (2005, p.59) "*a vantagem de conhecer e usar padrões de projetos é não somente economizar tempo, em vez de reinventar a roda, mas também dar aos desenvolvedores uma linguagem comum para design de software*. "

Padrões de Projetos também servem para facilitar a comunicação entre desenvolvedores, pois um nome dado ao um conjunto de informações como: qual o problema, base para a solução, quando usar e quais vantagens e desvantagens e com um nome dado a essas informações agiliza o processo de transmissão dessas informações entre os desenvolvedores (LISBOA, 2012).

Padrões de Projetos consistem na essência de uma solução para um problema especifico, na qual não envolve implementação direta, podendo-se usar um padrão de projeto em qualquer linguagem. Além disso, cada Padrão de Projeto consiste em resolver um problema específico (LISBOA, 2012). Para Lisboa (2012, p.23) "*um sistema de informação pode ter muitos problemas não definidos e bem complexos. Isso exige uma combinação de vários padrões de projetos*", na próxima sessão é abordado sobre um dos mais famosos padrões de projeto o MVC.

2.10. MVC

O *Model View Controller* (MVC), modelo – visão - controlador – é um padrão de projeto utilizado para modelar arquiteturas de *software,* muito comum em aplicações Web e muito utilizado por *frameworks*. MVC se utiliza de outros padrões de projeto para propor uma solução para o problema (GAMMA, 1998).

O MVC propõem que um sistema seja composto por três camadas: *Model* (Modelo), *Controllers* (Controladores) e *View* (Visão), cada camada é responsável por organizar as implementações feitas para o sistema.

O Modelo é o local onde são organizadas as implementações de regras de negócio, acesso e manipulação de banco de dados. Esses dados podem ser buscados de aplicações externas através de componentes fornecidos por empresas como *Facebook, Gmail, Twitter* (SOMERVILLE, 2007).

A Visão é camada que organiza e contém todas as implementações da interface de uma aplicação Web, sejam implementações internas que são produzidas especificamente para aplicação ou APIs externas que são utilizadas para auxiliar a implementação da interface (DUARTE, 2011).

Controladores são responsáveis por processar as requisições feitas pelos usuários. O processamento consiste em receber a requisição, que as repassa para a camada de modelo (se for necessária acesso aos dados da aplicação) , que, por sua vez devolve os dados requeridos ao controlador que devolve esses dados para a camada de visão (DUARTE, 2011).

O MVC começou a ser utilizado em aplicações Web a fim de melhorar a fase de implementação, propondo uma boa prática para organizar a arquitetura das aplicações Web, ele faz com que a arquitetura fique mais organizada, e auxilia o processo de manutenção e desenvolvimento de novas funcionalidades, além de facilitar a reutilização de código (DUARTE, 2011).

Na sessão 2.11 é abordado sobre os *frameworks* que são compostos por uma infinidade de padrões de projeto que é a próxima sessão e pelo MVC.

2.11. *FRAMEWORKS*

Um *framework* é um projeto de subsistema composto por um conjunto de classes abstratas e concretas Wirfs-Brock e Johnson (1990). Para Gamma 2011, "*frameworks nada mais são que combinações de padrões de projetos, e estão no mais alto nível de reuso*".

Frameworks no domínio de desenvolvimento de aplicações Web, é uma ferramenta que contém uma diversidade de funcionalidades prontas para serem usadas, essas funcionalidades geralmente são funcionalidade comuns em todas aplicações Web (BUSTAMANTE, 2008).

Para Johnson-Foote (1997) e Fayad (2000), *Frameworks* "são aplicações semi-completas, reutilizáveis que podem ser especializadas para produzir novas aplicações". Para Venturine (2011) "*O desenvolvimento de aplicações* Web *utilizando framework se expandiu de tal forma que se tornou imprescindível o uso delas para a construção de sistemas voltados para internet, sejam eles pequenos ou sistemas que atendem a uma multinacional*".

Quando é feito o uso de *frameworks* para desenvolvimento de *softwares*, detalhes específicos desse sistema são agregados aos componentes do *framework* criando assim o *software* desejado. *Frameworks* raramente são os *softwares* em si, geralmente um *software* é composto por vários *frameworks* (SOMERVILLE, 2007).

Construir um framework implica em um alto risco e investimento significativo (Gamma, 2005). Devido a esses problemas foram escolhidos dois *framework* de grande adesão no mercado: o Zend Framework 2 (ZF2) e o Twitter Bootstrap 2 (TB2). O ZF2 voltado para *back-end* e TB2 voltado para *front-end.* Nas duas próximas sessões e abordados sobre essas duas camadas de uma aplicação Web, *front-end* e *back-end.*

2.12. TWITTER BOOTSTRAP

TB é um *framework* voltado para criar interfaces de aplicações Web. Ele é composto por diversos componentes que auxiliam desenvolvedores a implementar a camada de *front-end* de uma aplicação (TB2, 2013).

Front – End é uma camada onde é implementada toda interface das páginas de uma aplicação Web. Nessa camada são definidos os arquivos responsáveis pela marcação, formatação e interatividade das páginas da aplicação (OUSTERHOUT, 2009).

O TB foi desenvolvido pela equipe que desenvolveu a rede social Twitter, e foi disponibilizado gratuitamente para os desenvolvedores utilizarem em Outubro de 2009 e atualmente está na versão 3.0, diferente da versão utilizada no trabalho que foi à versão 2.3 (TB, 2013).

2.13. ZEND FRAMEWORK 2

ZF2 é um *framework* voltado para a camada de *back-end* de aplicações Web, sendo desenvolvido por uma comunidade de desenvolvedores na qual são os mesmos que desenvolvem o *Hypertext Preprocessor* (PHP).

Back – End é uma camada onde é implementada todas as regras de negócio da aplicação, responsável por processar todos os dados que envolvem a aplicação, podendo ser da própria aplicação ou buscados de fontes externas como Facebook, Twitter (OUSTERHOUT, 2009).

Atualmente o *framework* é mantido pela empresa *Zend Technologies*, além de grandes empresas estarem envolvida no projeto como Google, IBM, Microsoft, Adobe, SERPRO e BBC, mas não deixou de ser um *software* livre. Atualmente ZF2 está na versão 2.2.4 e sua primeira versão foi disponibilizada em meados de 2006 (Zend, 2013). Atualmente o ZF2 é um dos *frameworks* mais utilizados para desenvolver aplicações Web (PHP *Frameworks*, 2013).

2.14. HOT SPOT

Hot Spot é uma metodologia utilizada para construir *frameworks,* sua essência consiste em identificar os “hot spots”, partes do *framework* que devem ser mantidas flexíveis para serem estendidas por uma aplicação e seu domínio. Essa metodologia é dividida em quatro etapas a descrição de cada uma é detalhada a seguir de acordo com (Silva, 2000).

1. São identificadas as classes do *framework*, após esse processo é definido a estrutura de classes do *framework*.
2. São identificados os aspectos que diferem de aplicação para aplicação. Ou seja, diferenciar a estrutura de classes para o *framework* em seguida definir o grau de flexibilidade dessas classes para serem estendidas.
3. É realizada uma remodelagem modificando a estrutura de classes do *framework* proposta inicialmente para que essa estrutura atenda as novas necessidades de flexibilidades. Após esse processo é definido as formas de quais os processos que o usuário deve seguir para estender o *framework* em sua aplicação.
4. Última etapa consiste em fazer um refinamento das classes desenvolvidas e verificar se o *framework* é satisfatório, caso contrário é retornado na segunda fase do projeto.

3. METODOLOGIA

Este trabalho teve como objetivo desenvolver uma API que possa facilitar a integração de dois *frameworks* de domínios diferentes para potencializar o desenvolvimento de aplicações no ambiente Web: Um *back-end* para a camada de regras de negócio de uma aplicação e outro *front-end* para a camada que compõe a interface de uma aplicação. A escolha por *frameworks* é devido a sua gama de componentes já desenvolvidos que estão prontos para serem usados no desenvolvimento de novas aplicações Web. Já o paradigma Programação Orientados a Objetos é devido aos seus recursos disponibilizados para implementações de *software* e por já estar presente nos *frameworks.*

3.1. ESTUDO E ESCOLHA DAS FERRAMENTAS

Para desenvolver esse ambiente foi necessário o estudo do Zend Framework 2 (ZF2) e Twitter Bootstrap 2 (TB2).

3.1.1. Zend Framework 2

ZF2 teve seu projeto anunciado em meadosde 2005, sua primeira versão disponibilizada em Março de 2006, quando ainda era somente um conjunto de bibliotecas, ainda não havia se tornado um *framework*. Mas no final de 2006 já foi implementada a primeira versão da camada MVC do *framework*. Em Julho de 2007 foi disponibilizada já como um *framework*. Atualmente o ZF2 é se encontra na versão 2.2 (ZEND, 2013).

O ZF2 é responsável por atender as necessidades da camada de *back-end* de uma aplicação Web. Essa camada deve possuir uma estrutura robusta e fácil de ser estendida, permitindo aos desenvolvedores o reaproveitamento de toda sua

estrutura bastando-se apenas desenvolver as funcionalidades da aplicação Web, logo os desenvolvedores não precisam se preocupar com o desenvolvimento de uma estrutura para comportar a aplicação a ser desenvolvida, ao utilizar o ZF2 ele contém as características necessárias para compor a camada de *back-end* (ZF2, 2013).

A escolha do ZF2 como *framework* de *back-end* levou em consideração fatores como: comunidade ativa, utiliza tecnologias atualizadas, tem uma alta performance (ZF2, 2013). Outras características desejáveis de um *framework* também são atendidas pelo ZF2 arquitetura modelada utilizando MVC, suporta múltiplos bancos de dados, contém em sua estrutura componentes internos para trabalhar com banco de dados, possui suporte a *templates*, trabalha com sistema de *caching* possui componentes para validação de formulário, tem suporte à tecnologia *Ajax* e sua arquitetura é totalmente modular (PHP *Frameworks*, 2013).

Além destas vantagens outro grande benefício a ser considerado, é o fato de seus desenvolvedores serem os mesmos que desenvolvem o PHP, linguagem na qual o ZF2 é desenvolvido. Com os desenvolvedores do *framework* conhecendo da melhor forma possível a linguagem que eles utilizam, eles exploram o máximo possível de recursos do PHP, deixando o *framework* mais estável e robusto e afetando positivamente no seu desempenho.

Outra consideração a ser feita é o fato de mesmo ser um *software* livre, ou seja qualquer um pode baixar e usar sem custo algum, existe uma empresa financiadora mantendo o *framework*. A empresa responsável por manter o ZF2 está localizada nos Estados Unidos da América e leva o nome de *Zend Technologies,* essa mesma empresa certifica profissionais para trabalhar com o ZF2 e com PHP (ZEND, 2013).

3.1.2. Twitter Bootstrap

TB2 foi desenvolvido pela mesma equipe que desenvolve a rede social *Twitter*, ele é um projeto *open source,* ou seja qualquer um pode usar da forma que quiser, atualmente se encontra da versão 3.0, lançada em Setembro de 2013 (TB2, 2013)

O TB2 foi escolhido por conter uma gama de componentes que podem auxiliar no desenvolvimento de interfaces. Alguns componentes bem comuns que são encontrados em quase todos os sites ou sistema Web, é a barra de navegação, onde são exibidos os principais links da página ou da aplicação. A figura a seguir apresenta uma barra de navegação na qual foi citado no parágrafo.

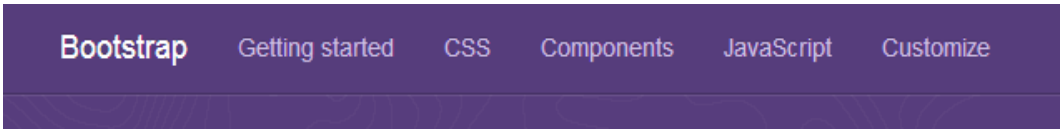

Figura 1 - Apresenta um exemplo de barra de navegação Fonte: próprio autor.

Como apresentado na FIGURA 1, essas barras são utilizadas para disponibilizar para o usuário os principais links que pode acessar dentro do site ou aplicação que está navegando, no exemplo da FIGURA 1, é a barra de navegação do site onde contém as informações sobre o *Twitter Bootstrap*.

O componente TabBar também é muito encontrado nos sites e aplicações Web, esse componente frequentemente utilizado para exibir conteúdo separado por sessões. A figura seguir apresenta a utilização do componente do TabBar.

Figura 2 - Apresenta a utilização do componente TabBar

A FIGURA 2 apresenta um exemplo do componente TabBar do TB2, onde os dados são separados pelas sessões, *home, Profile* e *Dropdown*.

Uma vantagem do TB2 a ser considerada é responsividade de seus elementos, permitindo que os desenvolvedores criem uma aplicação única que pode ser acessada por diversos dispositivos eletrônicos com resoluções distintas como,

notebooks, desktops, celulares *tablets* e até mesmo televisores (TB, 2013). Outro ganho é o tratamento que o *Twitter Bootstrap* interno que ele já faz para as incompatibilidades entre diversos navegadores que usam APIs diferentes e outros que estão desatualizados (TB, 2013).

3.2. COMPONENTES PARA COMPOSIÇÃO DA API

Para compor a API foram selecionados alguns componentes do TB, os mesmos foram desenvolvidos considerando os aspectos: dificuldade de implementação utilizando somente o TB; grande quantidade de tags HTML para desenvolver o componente; dificuldade em dar manutenção no componente e importância do mesmo para uma aplicação Web.

Os componentes “BarraNavegacao” e “TabBar” foram selecionados por serem muito utilizados em aplicações Web, além de serem de grande importância para um sistema Web. A grande maioria dos sistemas contém um barra de navegação, exibindo para o usuário os links para acessar determinadas funcionalidades da aplicação. O TabBar é de fundamental importância, pois, faz com que a interface de uma aplicação se torne mais amigável, pelo fato de separar o conteúdo a ser visualizado pelo usuário em seções, sendo assim ele escolhe somente o que deseja visualizar. Algumas características indesejáveis da BarraNavegacao e TabBar é a quantidade de código que necessário para criar esses componentes utilizando somente o TB2.

O componente “Botoes” foi selecionado para facilitar a criação de botões em aplicações Web. Como o TB2 disponibiliza vários estilos de botões para serem utilizados e implementados de forma diferente, surgiu a necessidade de agrupar todos os estilos dos botões em um local somente, facilitando assim a manipulação e criação destes.

O componente “Acordeom” foi selecionado devido ao grande número de tags HTML a serem descritas para formar o mesmo, além de agregar a usabilidade para interface de uma aplicação.

Na próxima seção é feito uma descrição sobre os requisitos de cada componente desenvolvido.

3.3. REQUISITOS PARA OS COMPONENTES DA API

Está seção descreve sobre a coleta de requisitos dos componentes que foram desenvolvidos. Essa coleta foi necessária para que os componentes não perdessem sua forma original fornecida pelo TB2. Esses requisitos ditam de que forma os componentes tiveram que ser implementados afim de obter os mesmo resultados a nível de interface, ou seja a interface de um componente desenvolvido para API tem que ser a mesma do formato original do TB2.

Cada componente desenvolvido disponibiliza as mesmas funcionalidades dos originais, mas utilizando estrutura de classes desenvolvidas em PHP e não formadas por várias tags descritas em HTML. Os requisitos de cada componente foram coletados nas suas funcionalidades originais disponibilizadas no site do *framework* TB2.

3.3.1. Requisitos Básicos

Estes requisitos são denominados básicos, pois todos componentes desenvolvidos tem como base esses requisitos. Para o desenvolvimento da API foi desenvolvido uma classe que agrega as características de uma *tag* HTML. Os componentes desenvolvidos estendem essa classe como base para criar seus componentes principais além das *tags* que o compõe.

3.3.2. BarraNavegacao

Esse componente representa o *Navbar,* componente original disponibilizado pelo TB2. Ele tem como objetivo disponibilizar em uma determinada posição do

layout de uma aplicação Web, os principais links que podem ser acessadas pelo usuário, um exemplo foi apresentado na FIGURA 3.

O objetivo de criar esse componente é propiciar aos desenvolvedores facilidade ao se desenvolver e manipular os dados da barra de navegação. Os links disponibilizados para o usuário muitas vezes é de acordo com as permissões que o usuário tem acesso no sistema, sendo assim manipular esses dados utilizando componentes desenvolvidos em PHP facilita o trabalho, pelo fato do PHP ser responsável por comunicar com o banco de dados e coletar as permissões que o usuário tem acesso.

As funcionalidades implementadas no componente segue na listagem abaixo.

- Especificar ou não um título.
- Adicionar links, onde cada link contém o texto do link e pode ser adicionado um ícone ao link.
- Adicionar links a direita, padrão a esquerda da barra.
- Adicionar sub links aos links adicionados, formando assim menus em *dropdowns* que são links que quando clicados exibe uma nova listagem de links que podem ser acessados.
- Definir componente como estático, ou seja independente do conteúdo da página ele permanece na mesma posição.
- Fixar o componente no topo do layout ou no rodapé do mesmo.

A FIGURA 3 apresenta uma barra de navegação do TB.

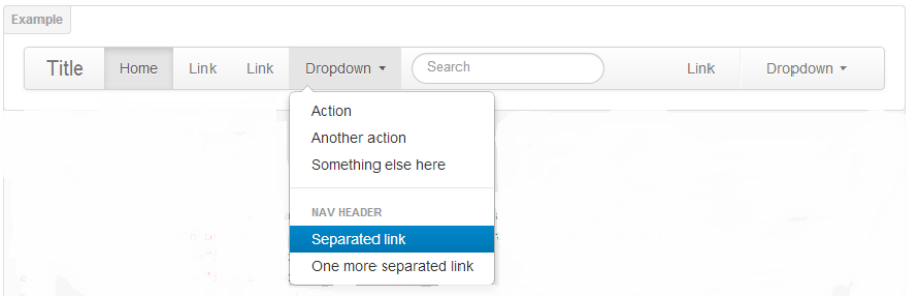

Figura 3 - Apresentação da BarraNavegacao em funcionamento Fonte: do próprio autor.

A FIGURA 3 apresenta as funcionalidades sendo utilizadas na criação da barra de navegação, onde são exibidos os links, o título, os links posicionados a direita da barra, e seu posicionamento ao topo.

3.3.3. Tab Bar

Tab Bar é um componente da API e foi desenvolvido para representar o componente *Tab,* original do TB2, geralmente é usado para criar menus, formulários separados por sessões, ou exibir dados para o usuário em sessões. Utilizando o Tab Bar da API é possível criar menus e fazer uma listagem de dados separando as informações em sessões. Essas sessões citadas são representas por abas que quando selecionadas exibem o conteúdo que a mesma representa.

O objetivo de desenvolver o Tab Bar é facilitar o trabalho dos desenvolvedores, utilizando o Tab Bar é possível criar o componente com poucas linhas de código, utilizando o *Tab* do TB2, é necessário varias linhas de código, pois o componente é descrito utilizando varias tags HTML.

O Tab Bar, contém todas funcionalidades do original que consiste em quando o usuário clicar em uma aba das abas, exibir o conteúdo que representa aba clicada. A FIGURA 3 apresenta um menu utilizando a Tab Bar.

As seguinte funcionalidades são disponibilizadas para o desenvolvedor.

- Adicionar sessão, consiste em criar uma aba para representar um conteúdo a ser exibido quando clicada.
- Adicionar conteúdo a uma sessão, consiste em adicionar o conteúdo que será exibido quando a sessão for clicada.

A FIGURA 4 apresenta a utilização de um *Tab* do TB.

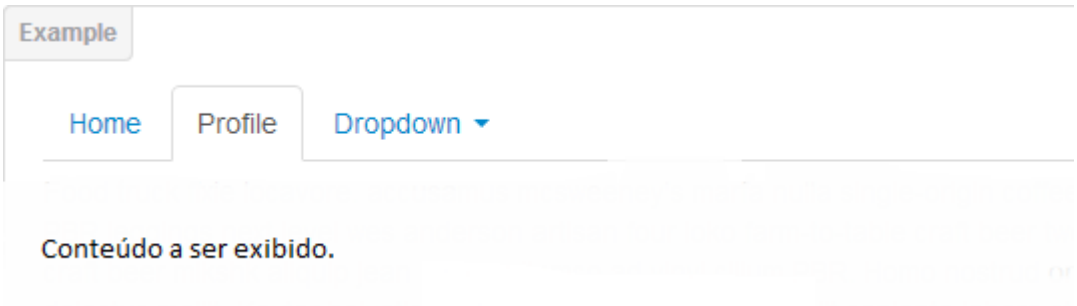

Figura 4 - Apresentação do componente TabBar Fonte: próprio autor.

Como apresentado na Figura 4, utilizando o Tab Bar é possível criar menus para navegação e exibição de algum conteúdo, ou exibir informações separados por sessões indicadas pelas abas.

3.3.4. Botao

O componente “Botao” foi desenvolvido para representar o componente *Button* do TB2, esse componente permite aos desenvolvedores estilizar os botões da sua aplicação, podem ser definidos tamanhos, cores, e até varias ações dentro de mesmo botão.

O objetivo desse componente é facilitar a criação desses botões, pois, ele contém muitas opções para estilização do mesmo, mas agregando todas as opções em um único componente pode facilitar o processo de criar e definir o estilo do botão.

As funcionalidades disponibilizadas pelo Botao são as seguintes.

- Definir tamanho do botão, os tamanhos dos botões podem ser visualizados na FIGURA 5.
- Definir a cor do botão, as cores e seus respectivos significados são apresentados na FIGURA 6.
- Definir ação do botão.
- Definir ações secundárias para os botões, ou seja, uma lista com as ações será apresentada para o usuário quando botão for clicado.
- Definir título do botão.

A FIGURA 5 a seguir apresenta as cores dos botões que podem ser definidas para o mesmo.

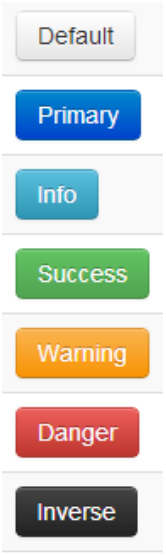

Figura 5 - Apresentação das opções de cores dos botões do Twitter Bootstrap

Como apresentado na FIGURA CORES BOTÔES, cada estilo por si só tem algum significado, o estilo *default* representa os botões sem nenhuma definição, o estilo *primary* pode ser utilizada para definir um botão que envia dados ao servidor do aplicação, exemplo: logar em um sistema, enviar dados de uma formulário entre outras opções.

Os estilos *info, success e warning* podem serem utilizados para indicar ao usuário que ação executada pelo botão não é uma ação simples, exemplo: o estilo *info* pode ser usado para exibir informações de um usuário, *success* pode ser usado para confirmar que ação foi realizada com sucesso, *warning* pode representar que ação que ele ira executar requer a atenção do usuário.

Os estilos *danger* e *inverse* podem ser utilizados para representar que a ação a ser executada pode causar danos as informações cadastradas no sistema, exemplos: *danger* e *inverse* podem ser utilizados para criar um botão que a ação é excluir dados da aplicação.

A FIGURA 6 apresenta as opções de tamanho dos botões.

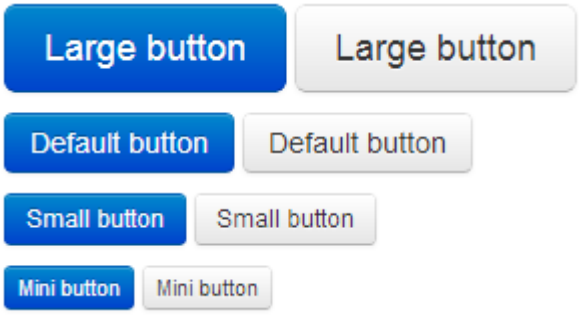

Figura 6 - Apresentação das opções de tamanho dos botões Fonte: próprio auto.

Como apresentado da FIGURA 7, os tamanhos dos botões possuem estilos, onde *large* representa o botão com tamanho maior, *default* representa o tamanho padrão dos botões, *small* representa o segundo menor e o *mini* o menor botão que pode ser estilizado.

3.3.5. Acordeom

Acordeom foi desenvolvido para representar o componente *Collapse* do TB2. Esse componente é utilizado para exibir informações de um jeito bem interativo, quando o usuário clicar em um link do mesmo, ele abre o conteúdo a ser apresentado, e fecha o último componente aberto, mas esse processo é feito em forma de slide, ou seja ele fecha e abre os conteúdos suavemente.

O objetivo do desenvolvimento desse componente é bem parecido com o componente TabBar, pois o Acordeom para ser criado é necessário muita escrita de código HTML, isso pode ser melhorado utilizando objetos em PHP, na qual utilizando o componente desenvolvido com algumas linhas de código é possível gerar esse mesmo componente em uma página Web.

As funcionalidades referentes ao acordeom são básicas segue a lista.

- Adicionar link, onde é definido título do link que representa o conteúdo a ser apresentado quando clicado.
- Adicionar conteúdo a um link, esse conteúdo quando o link do mesmo for clicado.

A FIGURA 7 apresenta um Acordeom sendo utilizado.

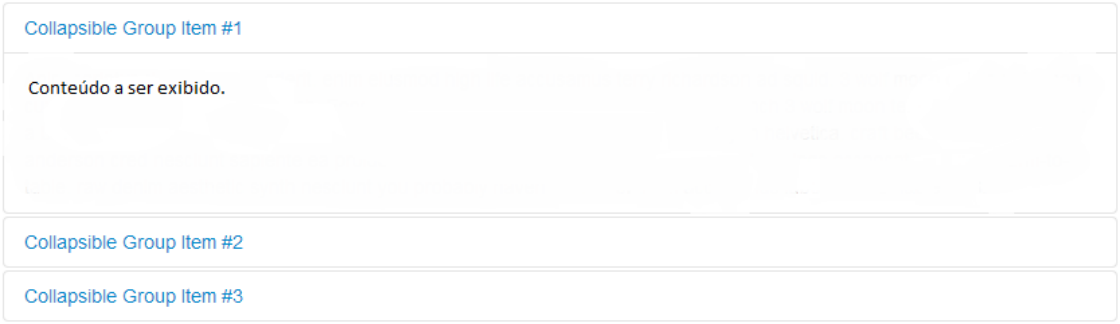

Figura 7 - Apresentação de menu desenvolvido utilizando Acordeom Fonte: próprio autor.

Como apresentado na FIGURA 7, o link clicado possui o seu conteúdo apresentado para o usuário e os demais permanecem escondidos.

3.4. IMPLEMENTAÇÃO DA API

Após a coleta de requisitos da API foram utilizadas algumas técnicas e metodologias para desenvolver a API de uma forma bem padronizada e concisa. Para desenvolvimento da API foi utilizada a metodologia *Hot Sopt*, apesar de ser uma metodologia utilizada para desenvolver *frameworks* também pode ser usada para organizar o desenvolvimento de APIs bibliotecas de *software* entre outros tipos de *softwares*.

Hot Spot foi utilizada na implementação da API, que foi fundamental para um bom andamento e boa organização da implementação da API. Utilizando a metodologia foram coletados os requisitos principais da API, onde foi feito a modelagem apresentado os requisitos que haveriam em todos componentes desenvolvidos, esse processo refere-se a e primeira etapa do *Hot Spot*.

Com esse processo feito, pode ser feito uma classe que propunha todos esses requisitos em uma só implementação, podendo assim servir de base para os componentes desenvolvidos. Após a implementação da classe base, foram coletados os requisitos dos demais componentes desenvolvidos, e feito uma modelagem dos requisitos coletados, esse passo refere-se a etapa 2 do *Hot Spot*.

Na etapa terceira foi feito uma analise entre a classe base que foi desenvolvida, com a modelagem dos componentes criados, esse passo consistiu em identificar se algum requisito do componente tinha algo em comum com os demais para ser transferido para a classe base. Na quarta etapa foram desenvolvidos os componentes da API.

Além da metodologia citada no texto acima foi utilizado diagramas UML para modelar a API e seus componentes de uma forma bem padronizada, esses diagramas além de documentar a API e seus componentes facilita ao desenvolvedor implementar as classes pelo fato de descreverem as classes a serem implementadas de forma bem simples e concisa.

Para desenvolver a API com os requisitos coletados foi usado o diagrama de classes para descrever a API de uma maneira geral e cada componente possui o seu diagrama em especial.

No próximo capitulo será descrito como desenvolvido a API e seus componentes.

4. RESULTADOS

Nesta sessão, será abordado sobre os resultados obtidos pelo desenvolvimento de cada componente que foi desenvolvido para compor a API. Foram desenvolvidos quatro componentes sendo eles: BarraNavegacao, Tab Bar Botões e Acordeom além de um componente base para auxiliar a criação dos componentes e manipular elementos HTML.

Para criar os componentes usando somente o *framework* do TB, é necessário incluir sua API na página que será utilizado ele será utilizado, além de implementar todas as tags HTML para formar o componente final.

Utilizando a API desenvolvida não a necessidade de implementar todas as tags para construir o componente, basta que o desenvolvedor instancie um objeto da classe que define suas características. Essas classes descrevem todas as *tags* HTML que são usadas para construir os componentes, onde são definidas suas características. Utilizando a API junto ao ZF2, não existe a necessidade de incluir a biblioteca do TB2, pois, o ZF2 já inclui a mesma por padrão.

A FIGURA 8 faz uma apresentação da arquitetura proposta pelos *frameworks*, ZF2 e TB2 mais a API desenvolvida.

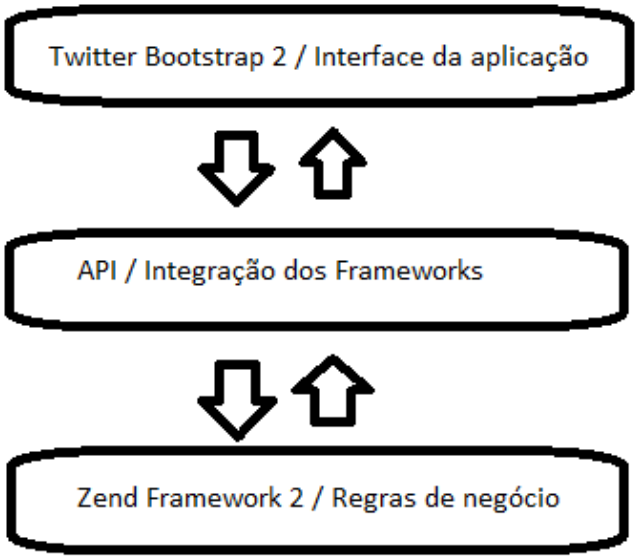

Figura 8 - Apresentação da Arquitetura entre os frameworks e a API Fonte: do próprio autor.

A FIGURA 8 apresenta como o ambiente funciona utilizando o TB2, a API desenvolvida no trabalho e o ZF2, os estilos dos componentes são definidos na biblioteca do TB2, a API desenvolvida implementa esses componentes utilizando classes PHP que retorna o código HTML para ser impresso na aplicação Web, já com os estilos definidos.

Os componentes podem ser utilizados na camada de Visão do ZF2, bastando apenas incluir a API desenvolvida onde for utilizada.

4.1. IMPLEMENTAÇÕES BÁSICAS

O primeiro requisito desenvolvido não foi nenhum componente em si do TB2, mas sim uma interface e uma classe na qual ela serve de base para os demais componentes e para as *tags* HTML. Esta classe tem como principal objetivo fornecer tudo o que for em comum entre os componentes, ou seja, ela é uma superclasse que contém os métodos e atributos para as suas sub classes.

Está classe implementa alguns métodos de uma interface que foi desenvolvida para especificar os atributos e métodos que são comuns nos componentes. A FIGURA 9 apresenta a modelagem da interface e da classe citada acima.

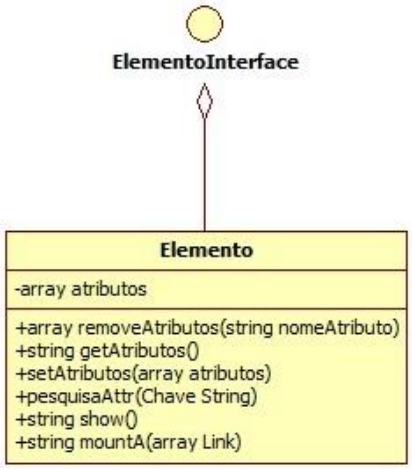

Figura 9 - Apresentação da modelagem do componente Elemento Fonte: próprio autor.

A FIGURA 9 apresenta a modelagem feita sobre a classe base(Elemento) para compor a API. Essa classe contém uma lista de funcionalidades.

- Manipular atributos de uma tag HTML
    - Adicionar
    - Remover
    - Pesquisar por atributo
- Gerar o HTML através das definições e atribuições que foram feitas sobre o componente.

Com essas funcionalidades é possível manipular os atributos de qualquer tag HTML utilizada para compor o componente, além de gerar o HTML de qualquer componente criado e retornar para o documento da página Web.

4.2. COMPONENTE BARRANAVEGACAO

O componente BarraNavegacao foi desenvolvido com base nos requisitos coletados e com na superclasse da API Elemento. Para desenvolver esse componente foi desenvolvido o diagrama da figura 7 para especificar o conteúdo da classe que contém os requisitos implementados. A FIGURA 10 apresenta a modelagem do componente BarraNavegacao.

Figura 10 - Representação da modelagem do componente BarraNavegacao Fonte: próprio autor.

Como apresentado na FIGURA 10, a classe BarraNavegacao, possui sete atributos que são utilizados para definir o comportamento do componente, como, posicionamento, largura, alinhamento, links, título e todas as características do componente.

Cada atributo possui seus métodos para manipular os valores do mesmo, além dos métodos showMenu e mountLinks, que são usados para gerar o HTML do componente e ser utilizado no documento da página Web.

4.3. TAB BAR

Esse componente foi desenvolvido com base na superclasse Elemento e com base nos requisitos coletados, essa classe tem como objetivo facilitar a construção de menus e forma de tabs, assim foram criados atributos e métodos dentro da classe TabBar para facilitar o desenvolvimento desses menus. A FIGURA 11 apresenta a modelagem do componente TabBar.

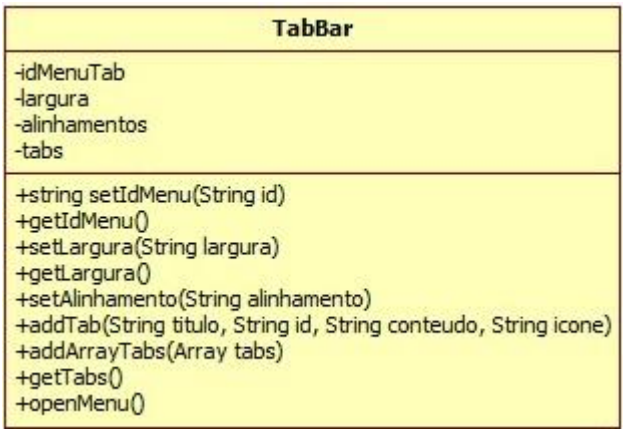

Figura 11 - Apresentação do componente TabBar Fonte: próprio autor.

Como apresentado na FIGURA 11 esse componente possui algumas atribuições a serem feitas, como o id da TabBar pois podem ter várias TabsBar em uma única página e esse id vai fazer com que elas não entrem em conflito, para definir o posicionamento da TabBar é utilizado o atributo alinhamento, podendo conter quatro valores, sendo eles: *top* onde os links da TabBar vão ficar posicionados no topo; *left* assim sendo posicionados a esquerda; *bottom* posicionados do rodapé da TabBar e *right* que posiciona os links a direita. O atributo largura define a largura da TabBar, esse atributo é passado na unidade de pixels, já o atributos *tabs* armazena todas as *tabs* adicionadas para a TabBar.

Os métodos definidos para a classe são os *gets* e *sets* que são utilizados para definir e acessar os valores dos atributos, e mais um método em especial que é responsável por gerar todo o HTML da TabBar, ele percorre as *tabs* que foram definidas e cria o cabeçalho da TabBar e por fim cria o conteúdo que foi definido para cada *tab.*

4.4. BOTÕES

O componente Botões foi desenvolvido com base na superclasse Elemento e nos requisitos coletados, sendo o mais simples do que foram desenvolvidos, pois ele contém poucas funcionalidades, a FIGURA 12 apresenta a modelagem do mesmo.

Figura 12 - Representação da modelagem do componente botões Fonte: próprio autor.

Como exibido na FIGURA 12 esse componente possui seis atributos sendo eles: título que contém o título do botão ou seja a mensagem que contém no botão; tipo que define a cor do botão, os seguintes tipos podem ser definidos, *default* que o padrão, *primary* que possui uma cor azul escuro, *info* possui a cor azul claro, *success* possui a cor verde claro, *warning* possui a cor amarelo claro, *danger* possui a cor vermelho e *inverse* possuindo a cor preta.

O atributo tamanho também possui vários valores a serem definidos, sendo elas, *mini* o tamanho mínimo que pode definir para um botão, *small* botão pequeno porém, maior que o mini, *default* é tamanho padrão que definido pela W3C e por último o *large* é o tamanho utilizado para definir botões grandes, todos os tamanhos são definidos pelo TB2.

Os próximos atributos são mais simples de serem utilizados, *split* define que o botão vai ter mais de uma ação, ou seja além da ação principal ele irá conter um menu *dropdown* com uma lista de outros links, e esses links são definidos no atributo links. E por último o atributo ações que define a ação principal do botão.

4.5. ACORDEOM

Esse componente é bem simples por ser utilizado apenas para exibição de conteúdo, consistindo em apenas definir o id do Acordeom para que ele não entre em conflito com outros acordeons que podem ser utilizados na mesma página, sendo assim, para definir esse id é usado o atributo idAcordeom, e para definir o conteúdo do acordeom é utilizado o *Array* também conhecidos como vetores armazenam várias informações, de acordeons que cada item do *Array* possui o título do acordeom, e o conteúdo a ser exibido. A FIGURA 13 apresenta a modelagem do componente Acordeom.

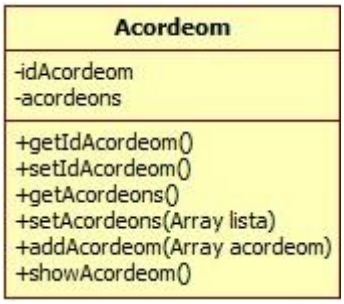

Figura 13 - Apresentação da modelagem do componente Acordeom.

Como a FIGURA 13 exibe além dos métodos *gets* e *sets* o componente Acordeom possui um método com o nome de showAcordeom que é responsável por gerar o HTML das informações passadas e por fim exibir esse HTML na página criando assim o Acordeom.

4.6. CONCLUSÃO DA API DESENVOLVIDA

A API desenvolvida disponibiliza para os desenvolvedores os componentes citados nas sessões, 4.2, 4.3, 4.4 e 4.5. Com esses componentes podem ser criados

interfaces para exibição de conteúdo, criar menus, barra de navegação, esses componente são originalmente do TB2.

Para descrever os mesmo componentes sem utilizar a API toda a interface tem ser criada através de diversas tags HTML e suas definições de classes e outros atributos para formar o componente final.

Mas utilizando API pode se criar os mesmo componentes, mas manipulando as informações de cada um utilizando objetos em PHP, cada componente da API possui um método que gera o HTML do componente automaticamente de acordo com as definições feitas sobre o componente.

5. CONCLUSÃO

Com os estudos realizados sobre aplicações Web foi possível identificar alguns problemas recorrentes enfrentados por desenvolvedores Web. Neste trabalho foi proposta e desenvolvida uma abordagem que contribui para que o desenvolvedor consiga contornar estes problemas por meio da aplicação de uma API que utiliza recursos de frameworks que fornecem recursos para sanar estes problemas.

Utilizando o ambiente proposto pelo trabalho, composto pelo Zend Framework 2 e Twitter Bootstrap 2, os seguintes problemas da camada de *front-end* podem ser contornados: incompatibilidade dos navegadores com as novas tecnologias, constantemente usadas em aplicações Web; criar interfaces com estilos bem definidos e de alta qualidade; dificuldade em criar componentes para serem utilizados em aplicações Web; criar páginas Web que se adaptam a diversas resoluções de telas; domínio sobre o CSS; dificuldade em design e alto custo para manutenção da camada de *front-end*. O TB2 possui em sua biblioteca a solução para esses problemas, entre outros problemas que ele pode contribuir.

Alguns problemas na camada de *back-end* são: organizar a estrutura de código e pastas de uma aplicação Web; montar uma arquitetura robusta e estendível; implementar padrões de projetos, como o MVC; reusabilidade de código; manipular e acessar banco de dados; confiabilidade da aplicação; criar estrutura de classes para atender as necessidades básicas de uma aplicação . O ZF2 é capaz de resolver os problemas citados e contribuir para outros problemas.

Para facilitar a utilização do TB2 junto ao ZF2 foi desenvolvido uma API, capaz de manipular os componentes do TB2, a API possui uma estrutura de classes desenvolvida em PHP, facilitando assim a criação e a manipulação dos componentes na camada de Visão do ZF2.

Com base na solução de alguns problemas resolvidos pelos *frameworks* mais a API desenvolvida, além da reusabilidade constante quando se usa *frameworks* para desenvolver sistemas, o desenvolvimento de aplicações Web utilizando o ambiente pode se tornar menos complexo.

6. TRABALHOS FUTUROS

Algumas sugestões para possíveis trabalhos futuros, a primeira é a continuação do mesmo, mas agregando todos os componentes disponibilizados pelo TB, além de algumas melhorias a serem implementadas na API desenvolvida.

- Uma sugestão de trabalho é realizar teste sobre o desempenho da API desenvolvida, deverá ser testado qual implementação deixa à página Web mais rápida e eficaz, ou seja testar uma página Web utilizando a API e os *frameworks* é mais rápida que a utilização dos *frameworks* sem a API integrando o mesmo.
- Outro trabalho, é agregar componentes de outros *frameworks* focados em *front-end* de uma aplicação, exemplos, *Jquery UI, Aloy Js, Zurb Foundation*, todos com vários componentes muito interessantes que podem vim a enriquecer a API desenvolvida.
- Outro trabalho é realizar a integração do ambiente que contém o *back-end* composto pelo ZF2 e o *front-end* composto pelo TB2, mais uma camada poderia ser vinculada no ambiente, utilizando a ferramenta *doctrine* que já tem módulos prontos para serem usados junto ao ZF2 poderia ser agregado no ambiente assim sendo o ambiente iria conter todas as camadas da aplicação completas, *front-end, back end,* e a camada de persistência dos dados da aplicação, ou seja camada de banco de dados.

REFERÊNCIAS

BARESI, L., Morasca, S., **Three Empirical Studies on Estimating the Design Effort of Web Applications**, ACM Transactions on *Software* Engineering and Methodology, Vol. 16, No. 4, Article 15, September 2007.

BUSTAMANTE, T. R. **Crux um arcabouço de *software* para desenvolvimento de aplicações Web**. 2008. 153f. Tese (Doutorado em Arquitetura de *Software*) - Universidade Federal de Minas Gerais, Belo Horizonte, 2008.

CHANG, P., Kim, W., Agha, G., **An Adaptive Programming Framework for Web Applications**, Symposium on Applications and the Internet, 2004.

DANTAS, Filho, F,O. **Proposta de um *Framework* para aplicações Web**, Edição 11, Dezembro 2009.

DUARTE, **Metodologia Rails: Análise Da Arquitetura Model View Controller Aplicada**, Universidade Federal de Minas Gerais,Instituto de Ciências Exatas Departamento de Ciências da Computação.

DEITEL, Paul J. DEITEL, Harvey M. **Java – Como Programar**. Prentice Hall. 8 edição, 2010.

GAMMA, Erich et al. **Design Patterns: Elements of Reusable Object-Oriented *Software***. Reading: Addison-Wesley, 1998.

GAMMA ,E. et al. **Padrões de Projetos: soluções reutilizáveis de *software* orientado a objetos**. Porto Alegre: Bookman, 2005.

GINIGE, A., Murugesan, S., **Guest editors' introduction: Web engineering—an introduction**. IEEE MultiMedia 8, 1, 14–18, 2001.

GUTMANS, A. et ali. **PHP 5 Programação Poderosa**. Rio de Janeiro: Alta Books, 2005.

FAYAD, M. E, **Introduction to the Computing Surveys' Electronic Symposium on Object - Oriented Application *Frameworks*, ACM Computing Surveys**, Vol.32, No. 1, March 2000.

FOWLER, M. **Padrões de Arquitetura de Aplicações Corporativas**. Porto Alegre: Bookman, 2006.

IBOPE, 2013. **Brasil é o terceiro país em número de usuários ativos na internet** Disponível em: <http://www.ibope.com.br/pt-br/noticias/paginas/brasil-e-o-terceiro-pais-em-numero-de-usuarios-ativos-na-internet.aspx> Acesso em: 16 abr. 2013

JAZAYERI, M., **Some Trends in Web Application Development, Future of *Software* Engineering** (FOSE'07), 2007.

JOHNSON, R. E, ***Frameworks* = (Components + Patterns), Communications of the ACM**, October 1997/ Vol. 40, No. 10.

JÚNIOR, J. H. C. R. **Linguagens de Programação para Internet. 2005**. 98f. Monografia (Bacharelado em Ciência da Computação) - Pontifícia Universidade Católica de Minas Gerais, Belo Horizonte - MG, 2005.

LISBOA, F. **Criando Aplicações PHP com Zend Framework e Dojo**, 2 edição, 2012.

OUSTERHOUT, J. K. Fiz: **A Component Framework for Web Applications**, Department of Computer Science Stanford University, 9 January 2009.

PHP *Frameworks*, 2013. **PHP *Frameworks* Disponível** em <http://www.php*frameworks*.com/index.php> Acessado: Maio de 2013

TANENBAUM, Andrew S. **Computer Networks**, 4 edição, 2003

TB, 2013, **Bootstrap *Powerful Front-End***. Disponível em: <http://getbootstrap.com/2.3.2/> Acessado em: Abril de 2013

NIELSEN, J., 1998, **Does Internet=Web?**, Disponível em: < http://www.useit.com/alertbox/980920.html > Acessado: Maio de 2013

SILVA, R, Pereira. **Suporte ao desenvolvimento e uso de *frameworks* e componentes**. Tese de doutorado Universidade Federal do Rio Grande do Sul, Instituto de Informática Programada de Pós-Graduação em Ciência da Computação.

SWEAT, J. E. ***php/arquitects's Guide to PHP Desings Patterns***. Marco Tabine e Associates, 2005.

W3C HTML, 2013. **HTML & CSS – W3C** Disponível em: <http://www.w3.org/standards/webdesign/htmlcss> Acesso em: 30 mai. 2013

W3C, CSS, **2013. HTML & CSS – W3C** Disponível em: <http://www.w3.org/standards/webdesign/htmlcss> Acesso em: 15 mai. 2013

W3C, SCRIPT, 2013. **JavaScript Web APIs – W3C** Disponível em: <http://www.w3.org/standards/webdesign/script> Acesso em: 20 mai. 2013

W3C, 2013. **Sobre o W3C.** Disponível em <http://www.w3c.br/Sobre> Acessado em Setembro de 2013.

ZEND, 2013. **Zend The PHP Company**. Disponível em: <http://www.zend.com/en/> Acessado em Julho de 2013.

ZF2, 2013. **About Zend *Frameworks***. Disponível em <http://framework.zend.com/> Acessado em Junho de 2013.