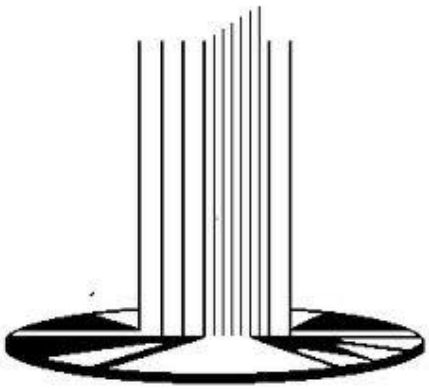
HOMERO: UM FRAMEWORK DE APOIO AO DESENVOLVIMENTO DE INTERFACES DE APLICAÇÕES WEB ACESSÍVEIS

Roberto Cícero de Oliveira

Dissertação de Mestrado

Orientação: Profa. Dra. Hana Karina S. Rubinsztejn

Área de Concentração: Engenharia de Software



Faculdade de Computação – FACOM

Universidade Federal de Mato Grosso do Sul – UFMS

Agosto/2013

DISSERTAÇÃO DE MESTRADO

HOMERO: UM FRAMEWORK DE APOIO AO DESENVOLVIMENTO DE INTERFACES DE APLICAÇÕES WEB ACESSÍVEIS

Roberto Cícero de Oliveira

Orientação: Profa. Dra. Hana Karina S. Rubinsztejn

Área de Concentração: Engenharia de Software

Faculdade de Computação – FACOM

Universidade Federal de Mato Grosso do Sul – UFMS

27 de Agosto de 2013

HOMERO: UM FRAMEWORK DE APOIO AO DESENVOLVIMENTO DE INTERFACES DE APLICAÇÕES WEB ACESSÍVEIS

Este exemplar corresponde á redação final da dissertação devidamente corrigida e defendida por Roberto Cícero de Oliveira e aprovada pela Banca Examinadora.

Campo Grande, 27 de agosto de 2013.

Banca Examinadora:

- Profa. Dra. Hana Karina S. Rubinsztejn (FACOM/UFMS) – orientadora
- Profa. Dra. Débora Maria Barroso Paiva (FACOM/UFMS)
- Profa. Dra. Maria Istela Cagnin Machado (FACOM/UFMS)
- Prof. Dr. André Pimenta Freire (UFLA)

DEDICATÓRIA

Difícil dedicar esse trabalho a um ou outro, pois minha força
e minhas convicções vieram de tantas pessoas...
Dedico esse trabalho a minha mãe Idimar, minha irmã Suzana e cunhado
Valdeir, meu irmão Neto, cunhada Sandra e sobrinha Letícia pelo apoio, por acreditarem
no meu sonho e entenderem minha ausência em tantos momentos importantes.
À minha esposa Deliane, pelo amor diário e amparo nos momentos de desespero,
e meus sogros Noestor e Darsoni por me receberem tão bem na família.
Dedico também a todos os meus amigos, pela compreensão, estímulo e ajuda
em todos os momentos, pois sem todos eles certamente esse sonho seria impossível.

Ao meu pai Benedito...

"Têm dia que meu almoço, é um pão com mortadela.
Mas lá no meu ranchinho a mulher e os filhinhos, tem franguinho na panela..."

AGRADECIMENTOS

Agradeço à minha esposa Deliane pelo amor, dedicação, cumplicidade e força, que mesmo nos momentos difíceis não me deixou fraquejar.

Aos meus pais, Benedito e Idimar, por minha educação, pelos ensinamentos e conselhos, pelo exemplo de vida, vitória e caráter que sempre nortearam meus princípios.

A minha irmã Suzana e cunhado Valdeir, que mesmo com a pouca diferença de idade, sempre agiram como porto seguro me permitindo assim, voar mais alto.

Ao meu irmão Benedito, Cunhada Sandra e sobrinha Letícia pelos momentos de conversa e pelo apoio prestado a meus pais na minha ausência.

Aos meus sogros Noestor e Darsoni pelo carinho, acolhimento e por estarem cotidianamente ao meu lado, agindo como verdadeiros pais.

A minha orientadora, professora Hana, por ter aceitado o desafio de me orientar em um tema relativamente novo e ter feito isso com maestria, como uma verdadeira Doutora.

Ao grupo de estudo de acessibilidade da Universidade Federal de Mato Grosso do Sul, formado pelas professoras Maria Istela e Débora, por terem me ajudado nortear o trabalho.

Ao colegiado do Curso de Mestrado em Ciência da Computação por terem acatado meus requerimentos.

A todos os professores e técnicos da FACOM/UFMS por contribuírem para o meu desenvolvimento e para a conclusão deste trabalho.

Agradeço eternamente a Deus que, mesmo às vezes sendo "*um homem de pouca fé*", sempre esteve ao meu lado guiando meu caminho e abençoando minhas escolhas.

RESUMO

A expansão da internet tornou-se evidente nos últimos anos, tanto pelo número de usuários, que atingiu a marca de 2,4 bilhões de pessoas no final de 2012, quanto pela quantidade de serviços disponíbilizados na rede, como por exemplo, boletim de ocorrência *on-line*, *e-bank*, *e-commerce*, entre outros.

Com tamanha expansão é essencial que o conteúdo seja acessível a todos os usuários, independente das dificuldades e limitações de cada um. Para isso é necessário desenvolver *software* com acessibilidade.

A concepção do *framework* Homero foi impulsionada pela necessidade de propiciar acessibilidade aos sistemas desenvolvidos e simplificar a utilização das diretrizes de acessibilidade propostas pelo W3C.

Desenvolvido utilizando a linguagem PHP, e de acordo com as diretrizes automatizáveis da *Web Content Accessibility Guidelines* (WCAG) 2.0, o *framework* Homero facilita a criação de interfaces *web* acessíveis em conformidade com o nível AAA na categorização da WCAG.

Buscando indícios da qualidade do *framework* proposto, um estudo empírico foi realizado junto a uma amostra controlada da população. Os resultados obtidos durante o estudo comprovaram a eficácia do Homero em auxiliar o desenvolvimento *web* acessível.

Palavras-chaves: *Framework*, Acessibilidade, Interfaces *Web*, WCAG.

ABSTRACT

The expansion of the internet has become apparent in last years, both by the number of users, that reached the milestone of 2.4 billion people in late 2012, as well as by the services available on the network, such as online police report, e-bank, e-commerce, among others.

With such expansion it is essential that the content be accessible to all users, regardless of the difficulties and limitations of each one. For this, it is necessary to develop software with accessibility.

The design of the Homero framework was driven by the need to provide accessibility to the developed systems and to simplify the use of the accessibility guidelines proposed by the W3C.

Developed using PHP language and in accordance with the automatable guidelines of Web Content Accessibility Guidelines (WCAG) 2.0, the Homero framework promotes that web interfaces developed through their classes have accessibility level AAA in the categorization of WCAG 2.0.

Seeking evidence of the quality of the proposed framework, an empirical study was conducted with a sample of the population controlled. The results obtained proved the effectiveness of Homero to assist the affordable web development.

Keywords: Framework, Accessibility, Web Interfaces, WCAG.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

1. INTRODUÇÃO

1.1. Contextualização

É evidente nos últimos anos o crescimento significativo da internet em todas as áreas do conhecimento e ramos de negócio. De acordo com o relatório anual do serviço de monitoramento de *websites* Pingdom (Pingdom, 2013), em dezembro de 2012 o número de usuários da internet chegou a 2,4 bilhões de pessoas e o número de endereços de *e-mails* atingiu a marca de 4,3 bilhões, com mais de 144 bilhões de *e-mails* trocados. O número de *sites* no mundo passou de 555 milhões em 2011 para 634 milhões em 2012. O Facebook é a rede social mais utilizada no mundo com aproximadamente 1 bilhão de usuários ativos, destes, 200 milhões foram criados no ano passado. Devido à tamanha expansão é necessário que os profissionais da área de Tecnologia da Informação (TI) se preocupem cada vez mais em planejar investimentos em soluções *on-line*, inteligentes e acessíveis, a fim de possibilitar que todas as pessoas tenham acesso a informações e/ou serviços mediados pela *web*.

Cada vez mais, empresas têm disponibilizado seus serviços de forma *on-line*, como o caso das lojas de *e-commerce* no Brasil, que no ano de 2012 tiveram faturamento de 22,5 bilhões de reais, com previsão de crescimento de 24% para o ano de 2013, segundo o e-bit, empresa especializada em dados do setor (Abril, 2013).

Adicionalmente, os órgãos governamentais também aderiram à disponibilização de informações e serviços na *web*, como o caso das declarações de imposto de renda, portais de transparência e boletins de ocorrência *on-line*. A partir do ano 2011 as declarações de imposto de renda, referente ao ano anterior, puderam ser entregues somente via internet (Camargo, 2013). Essa decisão foi adotada pelo fato das declarações em papel serem muito custosas de processar quando comparados ao envio das declarações pela internet, além de reduzir o número de declarações preenchidas erroneamente graças à validação de dados propiciadas pelas aplicações disponibilizadas (IRPF, 2013). Segundo a Lei Complementar 131, de maio de 2009, as prefeituras, governadorias e presidência da república começaram a disponibilizar as informações de despesas e receitas em portais, conhecidos como portais de transparência (Brasil, 2009). O poder executivo também inovou nas delegacias e disponibilizou o serviço de boletim de ocorrência *on-line* para ocorrências simples. No primeiro trimestre de 2013 foram registradas mais de 9,5mil ocorrências virtuais somente no estado de Mato Grosso do Sul (G1, 2013).

Com a internet alcançando tamanha gama de pessoas deve-se atentar ao fato que algumas delas possuem deficiências, permanentes ou temporárias. Segundo o censo 2010 do IBGE (Instituto Brasileiro de Geografia e Estatística) (IBGE, 2010), aproximadamente 45,6 milhões de brasileiros possuem algum tipo de deficiência, o que representa quase 24% da população brasileira.

1.2. Motivação e Justificativa

Com tamanha expansão do número de usuário utilizando os serviços *on-line* chega-se a conclusão de que para uma aplicação alcançar uma quantidade maior de pessoas é necessário atentar-se às dificuldades e limitações dos potenciais usuários a fim de prover os mesmos serviços e as mesmas informações a todos. Para isso é necessário desenvolver *software* com acessibilidade.

A Norma Brasileira NBR 9050 (ABNT. NBR 9050, 2004) adota a seguinte definição de acessibilidade: "Possibilidade e condição de alcance para utilização, com segurança e autonomia, de edificações, espaço, mobiliário e equipamentos urbanos". Pela definição entende-se que um objeto é acessível quando este pode ser alcançado e seu uso é seguro e autônomo. O quão fácil de ser utilizado não é mais uma questão de acessibilidade, e sim de usabilidade.

No contexto *web* a Norma ISO 9241-171 – *Ergonomics of Human-System Interaction Guidance on Software Accessibility* – define acessibilidade como a possibilidade de uso de um produto, serviço, ambiente ou equipamento por pessoas com os mais diferentes níveis de deficiências. Tal norma é destinada a desenvolvedores de *software* e fornece orientações sobre como elaborar o projeto de *software* a fim de alcançar um nível elevado de acessibilidade (ISO 9241-171, 2008). Ademais, a frase vencedora do concurso Jornadas de Conhecimento sobre Acessibilidade na *Web*, escrita por Carla Nascimento, afirma que "acessibilidade na *web* é tornar todos os serviços, assuntos e publicações tão fáceis de serem utilizados por todas as pessoas, que até esquecemos que há diferenças" (W3C, 2009), ou seja, é a prática de desenvolver aplicações que possam ser utilizadas igualmente por todos, independente de possuírem, ou não, alguma deficiência.

Para que os conteúdos disponibilizados sejam acessíveis existem vários mecanismos sendo pesquisados e desenvolvidos em várias linhas de pesquisa, como o teclado adaptado RCT-BARBAN desenvolvido pela Neo Luzz Brasil destinado a usuários com dificuldades motoras (Neo Luzz Brasil, 2013), o uso e evolução dos *softwares* de leitura de

tela, como o JAWS for Windows – versão 14, desenvolvido pela empresa Freedom Scientific (Freedom Scientific, 2013) e o uso de marcações específicas que proveem semântica, como as propostas pela WCAG – *Web Contents Accessibility Guidelines* (Diretrizes de Acessibilidade para Conteúdo da *Web*) que atualmente está na versão 2.0 (WCAG, 2008). No âmbito nacional também foi criada uma diretriz de acessibilidade, aderente a WCAG 2.0, denominada e-MAG – Modelo de Acessibilidade de Governo Eletrônico (Brasil, 2010), atualmente na versão 3.0. Esses e vários outros mecanismos funcionam individualmente, mas também podem ser usados em conjunto, garantindo que pessoas com várias deficiências consigam utilizar os sistemas de forma independente e autônoma, como preza a norma brasileira de acessibilidade.

Mesmo com iniciativas nacionais e internacionais de prover semântica aos conteúdos disponibilizados na *web*, a acessibilidade ainda permanece um pouco distante, visto que tais recursos ainda não são utilizados com frequência durante o desenvolvimento de aplicações *web*. Como prova disso temos que apenas 4,5% dos *sites* das instituições públicas possuem um nível mínimo de acessibilidade, segundo Núcleo de Acessibilidade e Inclusão – NAIS, do Ministério do Planejamento, Orçamento e Gestão (NAIS, 2013). Tal fato pode ser explicado pela falta de processos de desenvolvimento que abordem diretamente as questões referentes à acessibilidade. Adicionalmente, implementar elementos, um a um, de forma acessível pode demandar muito tempo devido às análises necessárias para tal. Uma forma de amenizar esse problema é prover mecanismos para que os desenvolvedores possam reutilizar o *design* das diretrizes de acessibilidade. Uma técnica de reutilização que pode ser utilizada para isso é o *framework* de *software*.

Framework de *software* (Fayad e Johnson, 2000) é uma técnica de reutilização que pode ser aplicada para auxiliar os desenvolvedores a produzirem *softwares* com tempo e esforço reduzidos. Baseado em suas características de instanciação e integração são organizados em caixa-branca, caixa-preta e caixa-cinza, cada qual com vantagens e desvantagens inerentes a sua concepção. Os *framework*s são classificados conforme a utilização (Fayad e Schmidt, 1997), sendo distribuídos em *frameworks* de infraestrutura, de integração de "*middleware*" e de aplicação.

No contexto de acessibilidade, Maia (2010) adaptou o *framework* Pantaneiro (Sandim *et al*., 2006), específico para a geração de *websites e-gov*, a fim de permitir que as páginas geradas atendessem a alguns critérios de acessibilidade *web*, porém, não foi encontrado na literatura nenhum *framework* para o desenvolvimento de camada de interface de aplicações *web* acessíveis, em qualquer domínio de aplicação.

1.3. Objetivos

O objetivo geral deste trabalho é disseminar o uso das diretrizes de acessibilidade propostas pela WCAG 2.0 e criar uma cultura de acessibilidade no âmbito de aplicações *web*, garantindo que as informações e/ou serviços disponibilizados na internet sejam acessíveis a todas as pessoas, independente dessas possuírem deficiências ou não. Para tangibilizar tal objetivo foi proposto um *framework* de *software* denominado Homero, com o intuito de apoiar o desenvolvimento da camada de interface de aplicações *web* acessíveis em conformidade com o nível AAA, que é o nível mais alto de acessibilidade, proposto pela WCAG 2.0 (WCAG, 2008). A linguagem de programação utilizada na codificação do *framework* e que também deve adotada na utilização foi o PHP (um acrônimo recursivo para "*PHP: Hypertext Preprocessor"*) (PHP, 2010), visto sua popularidade na comunidade de desenvolvedores. O Homero procura difundir a utilização das diretrizes da WCAG 2.0 através da simplicidade na utilização e reuso dos esforços empregados na criação de componentes acessíveis, sendo os cuidados referentes a acessibilidade responsabilidade do *framework* e o fornecimento das informações corretas reponsabilidade dos utilizadores do mesmo.

Com o objetivo de explicitar evidências da melhoria no processo de desenvolvimento de aplicações acessíveis com a utilização do *framework*, bem como prover indícios da qualidade e nível de acessibilidade dos produtos desenvolvidos um estudo empírico foi conduzido. Neste observou-se, especialmente, o processo de desenvolvimento da camada de interface de aplicações *web* acessíveis utilizando, ou não, o Homero, e o nível de acessibilidade obtido em ambos os casos.

1.4. Organização da Escrita da Dissertação

A escrita desta dissertação está organizada em seis capítulos. No Capítulo 2 são apresentados os conceitos de acessibilidade, acessibilidade *web* e as diretrizes de acessibilidade, e juntamente com o Capítulo 3, que apresenta os modelos de desenvolvimento de *software* mais utilizados e conceitos de *framework* de *software*, formam a base teórica deste trabalho. No Capítulo 4 é apresentado o *framework* desenvolvido, seu processo de criação e forma de utilização. O Capítulo 5 apresenta o estudo empírico conduzido com o intuito de avaliar a eficiência do *framework* no desenvolvimento de interfaces *web* acessíveis. No Capítulo 6 são apresentadas as conclusões, contribuições e limitações deste trabalho e são discutidas as possibilidades de trabalhos futuros.

2. ACESSIBILIDADE

Neste capítulo são apresentadas as principais concepções de acessibilidade, discutindo desde o surgimento do movimento por igualdade de direitos até a acessibilidade no âmbito digital, que este trabalho visa apoiar. Inicialmente é apresentado o histórico da evolução dos movimentos por acessibilidade no Brasil, desde as primeiras iniciativas até a aplicação do conceito no âmbito *web*. Na sequencia são apresentadas as principais diretrizes internacionais de acessibilidade, com maior enfoque para a diretriz WCAG 2.0, que é foco deste trabalho disseminar. A iniciativa nacional de criar diretrizes de acessibilidade também é apresentada, finalizando com o levantamento bibliográfico de trabalhos relacionados.

2.1. Acessibilidade e a Sociedade

No final de 1979 e começo de 1980 foi fundada a Coalizão Pró-Federação Nacional de Entidades de Pessoas Deficientes, na qual pela primeira vez, organizações de diferentes estados e que trabalhavam com diferentes tipos de deficiência se reuniram para tratar estratégias de luta por igualdade de direitos. As discussões nesse primeiro momento levantavam os problemas e soluções de barreiras arquitetônicas, comunicação e transporte. Esse movimento ganhou tamanha força que o ano de 1981 foi declarado como ano internacional das pessoas com deficiência. Já no final dessa década passou-se a discutir a eliminação de barreiras de atitudes, visto que a constituição federal promulgada em 1988 trazia a seguinte proposta de redação na emenda popular para o capítulo "Dos Direitos Individuais": "Art. 5° – Todos são iguais perante a lei, sem distinção de sexo, raça, trabalho, credo religioso e convicções políticas ou por ser portador de deficiência de qualquer ordem". A intenção era inserir a explícita igualdade de direitos para as pessoas com deficiência, mas na redação final da constituição determinou-se: "Todos são iguais perante a lei, sem distinção de qualquer natureza" (Junior, 2010).

No final do século passado a preocupação deixou de ser a eliminação de obstáculos e tornou-se garantia de acesso, tanto que a Lei Federal nº 10.098, de 19 de dezembro de 2000, regulamentada pelo Decreto nº 5.296, de 02 de dezembro de 2004, normatizou, em linhas gerais, os critérios de acessibilidade para as frentes de mobiliário urbano, elementos da urbanização, construção e reforma de edifícios e para os meios de transporte e de comunicação (Brasil, 2004).

Com o grande crescimento da internet nos últimos anos, visto que o número de internautas nos últimos três anos teve um aumento de aproximadamente 40% (Pingdom, 2013), surgiu um novo rumo nas discussões: a garantia de acessibilidade em todas as suas dimensões e para todos os cidadãos. Os movimentos não são mais em torno da eliminação de obstáculos, mas ao direito de ingresso, permanência e usufruto de todos os bens e serviços sociais, inclusive no contexto digital. Na Figura 2.1 é apresentada tal evolução dos movimentos de acessibilidade.

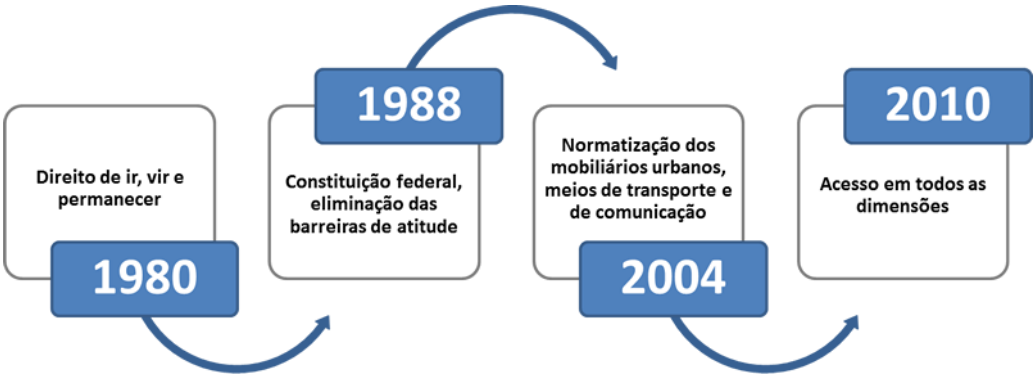


Figura 2.1 – Evolução dos movimentos de acessibilidade no Brasil.

Quando se fala em deficiência, muitas vezes surgem dúvidas do que pode ser chamado de deficiência, porém para esclarecer tal dúvida basta atentar-se ao Decreto nº 3298 de 20 de dezembro de 1999 (Brasil, 1999), que rege: "É considerada pessoa com deficiência a que se enquadra nas seguintes categorias: deficiência física, deficiência auditiva, deficiência visual e deficiência mental".

Após diversas discussões nos anos 80 e 90, e com uma definição clara do que é deficiência, foi regulamentada no artigo 8º do Decreto-Lei 5296 de 2 de dezembro de 2004 (Brasil, 2004) a Lei de nº 10.048 de 8 de novembro de 2000, priorizando atendimento às pessoas com deficiências, e a Lei 10.098 de 19 de dezembro de 2000, que estabelece normas gerais e critérios básicos para a promoção da acessibilidade.

> "Artigo 8º - Para os fins de acessibilidade considera-se (Decreto-lei 5296, 2004):
>
> I. acessibilidade: condição para utilização, com segurança e autonomia, total ou assistida, dos espaços, mobiliários e equipamentos urbanos, das edificações, dos serviços de transporte e dos dispositivos, sistemas e meios de comunicação e informação, por pessoa portadora de deficiência ou com mobilidade reduzida;
>
> II. barreiras: qualquer entrave ou obstáculo que limite ou impeça o acesso, a liberdade de movimento, a circulação com segurança e a possibilidade de as pessoas se comunicarem ou terem acesso à informação, classificadas em:

a) barreiras urbanísticas: as existentes nas vias públicas e nos espaços de uso público;
b) barreiras nas edificações: as existentes no entorno e interior das edificações de uso público e coletivo e no entorno e nas áreas internas de uso comum nas edificações de uso privado multifamiliar;
c) barreiras nos transportes: as existentes nos serviços de transportes; e
d) barreiras nas comunicações e informações: qualquer entrave ou obstáculo que dificulte ou impossibilite a expressão ou o recebimento de mensagens por intermédio dos dispositivos, meios ou sistemas de comunicação, sejam ou não de massa, bem como aqueles que dificultem ou impossibilitem o acesso à informação."

Com a necessidade de eliminar as barreiras de informações, e a crescente quantidade destas disponibilizadas na internet chegou-se a um novo conceito, o de acessibilidade *web*.

Segundo a WCAG (2010), acessibilidade *web* significa que pessoas com deficiências sejam capazes de usar a internet. Mais concretamente, significa uma *web* projetada a fim de que as pessoas possam perceber, entender, navegar e interagir de maneira efetiva, bem como criar e contribuir com conteúdos para a mesma. Portanto, é preciso que as ferramentas e páginas sejam acessíveis para os utilizadores, independente de possuírem deficiências ou não. Para alcançar tal objetivo órgãos, tanto nacionais quanto internacionais, propuseram diretrizes para nortear o desenvolvimento de forma que o conteúdo fosse acessível.

2.2. Diretrizes Internacionais de Acessibilidade Web

Dentre os órgãos internacionais que apresentaram iniciativas em busca de alinhar o desenvolvimento à acessibilidade destacam-se a *Section* 508 (Section 508, 2013), do governo americano, a *Irish Guidelines for Web Accessibility* (CEUD, 2013), do governo irlandês, a WAI-ARIA – *Accessible Rich Internet Applications* (WAI-ARIA, 2011) e a WCAG – *Web Contents Accessibility Guidelines* 1.0 (WCAG, 1999) e 2.0 (WCAG, 2008), disponibilizados pelo W3C (WWWC - *World Wide Web Consortium*) através de seu departamento WAI (*Web Accessibility Initiative).*

A *Section* 508 foi uma lei promulgada nos Estados Unidos em 1998, que ficou conhecida como lei da acessibilidade. Junto à lei foi criado um portal de acessibilidade,

trazendo as diretrizes propostas, que deveriam ser seguidas pelos desenvolvedores para que os sistemas desenvolvidos provessem acessibilidade, além de informações gerais tanto para profissionais da área de tecnologia, quanto para o público em geral, sobre leis, fóruns, ferramentas, entre outros (Section 508, 2013). As diretrizes propostas pela *Section* 508 foram criadas com base nas propostas pela WCAG 2.0 (WCAG, 2008).

O governo irlandês, através da Autoridade Nacional para a Deficiência (*National Disability Authority - NDA*) também apresentou uma proposta para diretrizes de acessibilidade conhecidas como *Irish National IT Accessibility Guidelines*. Essas diretrizes foram norteadas pelas descritas na WCAG 1.0 (WCAG, 1999), apresentando como diferenças entre ambas o acréscimo de explicações para cada diretriz e a inclusão de uma lista simplificada das declarações das diretrizes e de um sistema de numeração baseado na prioridade (CEUD, 2013).

O WAI-ARIA (*Accessible Rich Internet Applications*) é uma iniciativa, ainda não concluída, do W3C para prover acessibilidade a aplicações ricas em recursos. As diretrizes da ARIA propõem a adição de atributos específicos, assim como as da WCAG, técnicas de navegação, marcação de regiões e outras técnicas para prover semântica ao conteúdo disponibilizado. O foco de tais diretrizes é a geração de conteúdo dinâmico e criação de interfaces de usuário avançadas (WAI-ARIA, 2011). Como as especificações do HTML5 e do WAI-ARIA ainda não estão completamente definidas, algumas marcações ficam sobrepostas (Tarr, 2011).

A WCAG (*Web Contents Accessibility Guidelines*) 1.0 (WCAG, 1999) e 2.0 (WCAG, 2008) são documentos disponibilizados pelo departamento de acessibilidade *web* - *Web Accessibility Initiative* (WAI) do W3C. Conhecidos como Diretrizes de Acessibilidade para Conteúdo da *Web*, ou Recomendações de Acessibilidade do W3C, desde a criação da versão 1.0, em 1999, até os dias de hoje, estas diretrizes têm sido a base para o estudo e a prática de acessibilidade para a *web* em todo o mundo. Desde o final de 2008 a versão 1.0 começou a ser substituída pela sua atualização, a WCAG 2.0 do próprio W3C. Segundo o consórcio W3C, as diretrizes da WCAG 2.0 deveriam ser preferencialmente recomendadas e utilizadas desde o lançamento, embora seja possível seguir tanto as diretrizes da WCAG 1.0, como as da WCAG 2.0, ou ambas. Contudo, o W3C sugere que os conteúdos novos e atualizados utilizem as diretrizes da versão 2.0, pois são mais aderentes as tecnologias utilizadas atualmente.

As diretrizes da WCAG 2.0 explicam como tornar o conteúdo das páginas *web* acessíveis a pessoas com deficiência e destinam-se a todos os desenvolvedores de aplicações

web e programadores de ferramentas para criação de conteúdo *web*. Seu principal objetivo é prover acessibilidade, mas sua utilização também fará com que os conteúdos *web* se tornem de mais fácil acesso, e otimiza a navegação independente dos agentes utilizados pelos usuários, como por exemplo, navegadores comuns, por voz, celulares, leitores de tela, etc. Além disso, a utilização destas diretrizes irá ainda contribuir para que os motores de busca *on-line*, por exemplo, o Google ou o Bing, ranqueiem corretamente a informação facilitando a busca das informações na rede (WCAG, 2008).

Na Seção 2.2.1 é apresentada, em detalhes, a versão 2.0 da WCAG. Tal diretriz foi utilizada na concepção do *framework* proposto e a escolha se deu pelo fato da mesma apresentar aderência as demais diretrizes utilizadas no mundo.

2.2.1. Web Contents Accessibility Guidelines – WCAG 2.0

As Diretrizes de Acessibilidade para Conteúdo da W*eb* – WCAG 2.0 abrangem uma vasta gama de recomendações para tornar o conteúdo da *web* acessível. O cumprimento dessas diretrizes faz com que os conteúdos fiquem acessíveis a um maior número de pessoas com deficiências, incluindo cegueira e baixa visão, surdez e perda de audição, incapacidades ao nível da aprendizagem, limitações cognitivas, movimentos limitados, incapacidades ao nível da fala, fotossensibilidade e ainda combinações dessas incapacidades. O cumprimento dessas diretrizes facilita também a utilização do conteúdo da *web* pelos usuários em geral. A estrutura e os elementos da WCAG 2.0 estão ilustrados na Figura 2.2.

Princípios	Diretrizes	Nível A	Nível AA	Nível AAA
1. Perceptível	1.1 - Alternativas em Texto	1.1.1		
	1.2 - Mídias com base no tempo	1.2.1 - 1.2.3	1.2.4- 1.2.5	1.2.6 - 1.2.9
	1.3 - Adaptável	1.3.1 - 1.3.3		
	1.4 - Discernível	1.4.1 - 1.4.2	1.4.3 - 1.4.5	1.4.6 - 1.4.9
2. Operável	2.1 - Acessível por Teclado	2.1.1 - 2.1.2		2.1.3
	2.2 - Tempo Suficiente	2.2.1 - 2.2.2		2.2.3 - 2.2.5
	2.3 - Ataques Epiléticos	2.3.1		2.3.2
	2.4 - Navegável	2.4.1 - 2.4.4	1.4.1 - 2.4.4	2.4.8 -
3. Compreensível	3.1 - Legível	3.1.1	3.1.2	3.1.3 - 3.1.6
	3.2 - Previsível	3.2.1 - 3.2.2	3.2.3 - 3.2.4	3.2.5
	3.3 - Assistência de Entrada	3.3.1 - 3.3.2	3.3.3 - 3.3.4	3.3.5 - 3.3.6
4. Robusto	4.1 - Compatível	4.11 - 4.1.2		

Figura 2.2 – Estrutura da WCAG 2.0 (WCAG, 2008).

As diretrizes e os critérios de sucesso a elas associadas estão organizados em torno de quatro princípios, que apresentam a informação básica para um usuário acessar e utilizar os conteúdos da *web*. Um usuário que pretende utilizar a *web* tem de dispor de conteúdo que seja (WCAG, 2008):

1. **Perceptível** - A informação e os componentes da interface do usuário têm de ser apresentados aos usuários de formas perceptíveis;
2. **Operável** - Os componentes da interface do usuário e a navegação têm de ser operáveis;
3. **Compreensível** - As informações e a operação da interface do usuário têm de ser compreensíveis; e
4. **Robusto** - O conteúdo tem de ser suficientemente robusto para ser interpretado, com precisão, por uma grande variedade de usuários, incluindo tecnologias de apoio.

Se algum destes princípios não for seguido, os usuários com deficiências não serão capazes de utilizar a w*eb* plenamente.

Com o uso das doze diretrizes da WCAG 2.0 o conteúdo disponibilizado é acessível a todos os usuários, independente destes possuírem alguma deficiência. Para cada diretriz são fornecidos modos de cumprí-la, chamados de critérios de sucesso, e testá-la podendo assim avaliar a conformidade desta. Para satisfazer as necessidades dos diferentes grupos e situações, são definidos três níveis de conformidade: A (mais baixo), AA (intermediário) e AAA (mais elevado).

A seguir é descrita uma das diretrizes da WCAG 2.0, sendo o que esta e as demais estão disponibilizadas no *site* da WCAG (WCAG, 2008).

- **Diretriz 1.1 - Alternativas em Texto (Princípio 1: Perceptível)**

Fornecer alternativas em texto para qualquer conteúdo não textual permitindo, assim, que o mesmo possa ser alterado para outras formas mais adequadas à necessidade do indivíduo, tais como impressão em caracteres ampliados, braille, fala, símbolos ou linguagem mais simples.

1.1.1. Conteúdo Não Textual

Todo o conteúdo não textual que é apresentado ao usuário tem uma alternativa em texto que serve um propósito equivalente, exceto para as situações indicadas abaixo:

- Controles, Entrada: Se o conteúdo não textual for um controle ou aceitar a entrada de dados por parte do usuário, então deve ter um nome que descreva a sua finalidade.
- Mídias com base no tempo: Se o conteúdo não textual corresponder a mídia baseada no tempo, então as alternativas em texto fornecem, no mínimo, uma identificação descritiva do conteúdo não textual.
- Teste: Se o conteúdo não textual for um teste ou um exercício, inválidos se apresentados em texto, então as alternativas em texto fornecem, no mínimo, uma identificação descritiva do conteúdo não textual.
- Sensorial: Se a finalidade do conteúdo não textual for, essencialmente, criar uma experiência sensorial específica, então as alternativas em texto fornecem, no mínimo, uma identificação descritiva do conteúdo não textual.
- CAPTCHA[1]: Se a finalidade do conteúdo não textual for confirmar que o conteúdo está sendo acessado por uma pessoa e não por um computador, então são fornecidas as alternativas em texto que identificam e descrevem a finalidade do conteúdo não textual, e são fornecidas as formas alternativas do CAPTCHA que utilizam modos de saída para diferentes tipos de percepção sensorial, para atender diferentes incapacidades.
- Decoração, Formatação, Invisível: Se o conteúdo não textual for meramente decorativo, for utilizado apenas para formatação visual, ou não for apresentado aos usuários, então é implementado de uma forma que pode ser ignorada pelas tecnologias assistivas[2].

2.3. Diretrizes Nacionais de Acessibilidade Web

A construção de *websites* acessíveis é uma exigência do Decreto 5296, publicado em dezembro de 2004 (Brasil, 2004), que torna obrigatória a acessibilidade nos portais e *sites* eletrônicos da administração pública na rede mundial de computadores para o uso das pessoas

[1] Acrônimo da expressão "*Completely Automated Public Turing test to tell Computers and Humans Apart*" (teste de Turing público completamente automatizado para diferenciação entre computadores e humanos).
[2] São definidas como "*uma ampla gama de equipamentos, serviços, estratégias e práticas concebidas e aplicadas para minorar os problemas encontrados pelos indivíduos com deficiências*".

com deficiência, garantindo-lhes o pleno acesso aos conteúdos disponíveis. Isso motivou a criação de uma diretriz de acessibilidade nacional que pudesse nortear o desenvolvimento acessível. Tal diretriz é denominada Modelo de Acessibilidade de Governo Eletrônico e-MAG (Brasil, 2007).

O e-MAG consiste de um conjunto de recomendações a ser considerado para que o processo de acessibilidade dos *sites* e portais do governo brasileiro seja conduzido de forma padronizada e de fácil implementação. É coerente com as necessidades brasileiras e em conformidade com os padrões internacionais estabelecidos pela WCAG 2.0. Igualmente aos padrões internacionais, foi formulado para orientar profissionais que tenham contato com publicação de informações ou serviços na internet a desenvolver, alterar e/ou adequar páginas, *sites* e portais, tornando-os acessíveis ao maior número de pessoas possível.

A primeira versão do e-MAG foi disponibilizada para consulta pública em 18 de janeiro de 2005 e a versão 2.0, já com as alterações propostas, em 14 de dezembro do mesmo ano. Atualmente o e-MAG está na versão 3.0 (Brasil, 2011). Foi embasada na versão anterior e em novas pesquisas na área de acessibilidade na *web*. Apesar de utilizar a WCAG como referência, o e-MAG 3.0 foi desenvolvido e pensado para as necessidades locais, visando atender as prioridades brasileiras e mantendo-se alinhado ao que existe de mais atual neste segmento. A versão 3.0 é apresentada em apenas um documento, voltado para a população em geral, diferente da versão 2.0 que era apresentado em dois, sendo a cartilha técnica voltada a desenvolvedores e a visão do cidadão voltada aos cidadãos brasileiros. Outra decisão foi o abandono dos níveis de prioridade (A, AA e AAA) proposto na versão anterior, não sendo permitidas exceções com relação ao cumprimento das recomendações. Além disso, no e-MAG 3.0 foi incluída a seção chamada “Padronização de acessibilidade nas páginas do governo federal”, com o intuito de padronizar elementos de acessibilidade que devem existir em todos os *sites* e portais do governo.

A Portaria nº 3, de 7 de maio de 2007 (BRASIL, 2007), institucionalizou o e-MAG no âmbito do sistema de Administração dos Recursos de Informação e Informática – SISP, tornando sua observância obrigatória nos *sites* e portais do governo brasileiro. Mesmo com essa determinação o Ministério do Planejamento, Orçamento e Gestão através do Núcleo de Acessibilidade e Inclusão – NAIS constatou que apenas 4,5% dos *sites* das instituições públicas possuem um nível mínimo de acessibilidade (NAIS, 2013). Já a pesquisa "Dimensões e características da *Web* brasileira: um estudo do .gov.br" feito pelo grupo de trabalho de acessibilidade na *web* da W3C Brasil afirma que esse valor atingiu somente 2% das páginas *web* governamentais(W3C, 2012).

2.4. Trabalhos Relacionados

Com a necessidade de prover acessibilidade às aplicações disponibilizadas na internet diversas iniciativas foram propostas, dentre estas se podem elencar o levantamento de métricas de acessibilidade web, as propostas de processos de construção de aplicações web acessíveis e a criação ferramentas de apoio à construção acessível. As buscas do levantamento bibliográfico foram feitas em três bibliotecas on-line, tendo sucesso em encontrar trabalhos relevantes na ACM Digital Library (http://dl.acm.org). Os trabalhos também foram buscados nos *sites* das edições da W4A – *Disciplinary Conference on Web Accessibility* (Conferência Disciplinar Sobre Acessibilidade *Web*) que se encontra na 10ª edição. Nas buscas foram encontrados um grande número de trabalhos que abordaram propostas de métricas, ou então análises de determinados cenários, porém trabalhos que propusessem formas de otimizar o desenvolvimento acessível de interfaces *web* foi mais escasso, e nenhuma proposta de reuso das diretrizes da WCAG 2.0, em formato de *framework*, foi encontrada.

A preocupação em facilitar o uso das diretrizes da WCAG começou a ser discutida quando esta ainda estava na versão 1.0. Em 2006 Sloan *et al*. (2006) discutiram a importância de se existir um conjunto de diretrizes que norteassem, eficientemente, o desenvolvimento *web* acessível. Na época os níveis de acessibilidade na rede mundial de computadores foram considerados como decepcionantes pelos autores, e estes argumentaram sobre a dificuldade de aplicar, na prática, as diretrizes da WCAG 1.0 de maneira direta. Como solução foi proposto um modelo holístico chamado Trangram visando facilitar tal abordagem. Os autores chegaram à conclusão que o conceito de acessibilidade não possui um caminho absoluto, podendo ser alcançado através de várias alternativas e o uso do modelo proposto levaria a uma maior eficácia no desenvolvimento de aplicações com acessibilidade, permitindo que as informações e serviços se adaptassem as necessidades dos usuários.

O processo de desenvolvimento acessível, como um todo, foi abordado por Martin *et al*. (2011). Tratando principalmente da responsabilidade de prover acessibilidade nas fases iniciais dos projetos, para ser exato na fase de *design*, propuseram uma ferramenta de apoio. A ferramenta UI Design é composta por um diagrama de pontos de integração e gráficos de interdependência, onde após mapear os locais dos pontos de integração entre sistema e usuário são elaborados os gráficos de interdependência, que visa especificar quais diretrizes devem ser implementadas nessa integração, diminuindo a lacuna entre especialistas em acessibilidade, que têm pouca experiência em programação, e programadores, que em sua maioria, não possui conhecimento sobre acessibilidade.

Dentre os trabalhos analisados o *framework Alipi* (Dinesh *et al*., 2012), apresenta uma abordagem diferenciada pelo fato de propor uma abordagem distribuída e participativa, onde voluntários fornecem descrições alternativas para a página e os elementos desta. O *Alipi* se baseia em três subsistemas principais: (a) um subsistema para narradores criarem conteúdo, (b) um subsistema de indexação de páginas da *web* a suas narrativas, e, (c) um subsistema que narra a página *web*. O *Alipi* mantém o controle da origem, destino e linguagem de cada narração, não permitindo que o conteúdo se perca ou se torne confuso. A descentralização e a multiplicidade das narrações eliminam a abordagem normativa e top-down das diretrizes da WCAG, permitindo que as comunidades de narradores criem conteúdos específicos em torno das necessidades dos usuários.

O modelo *MIPAW – Modele of a Progressive Implementation of Web Accessibility* – (Villain *et al*., 2012) permite o estabelecimento e tratamento progressivo das diretrizes da WCAG 2.0, em todas as fases de um projeto. O principal objetivo do MIPAW é servir como um quadro para a elaboração de melhoria progressiva do nível real de acessibilidade, bem como otimizar a gestão da acessibilidade desses produtos de forma adequada às limitações da equipe. Tal modelo é uma adaptação da estrutura da WCAG 2.0 para quatro fases de implementação sucessivas, estruturados pela noção de acesso à informação: i) Proteção do acesso a informações; ii) Garantia de acesso à informação; iii) Melhoria do impacto ao usuário; e iv) Melhoria da experiência do usuário. Esse modelo propõe melhoria em três áreas: i) metodologia: apresenta uma visão clara dos objetivos e norteia por onde começar, distribuindo melhor os esforços e recursos do projeto; ii) sistemas de medição: representa de forma adaptada e em nível real a acessibilidade dos conteúdos; e iii) gerenciamento de projetos: propicia uma visão das questões essenciais do projeto.

Dentre os trabalhos que apresentaram avaliações de acessibilidade a proposta por Oikonomou *et al*.(2011), chamou a atenção pela forma que a avaliação é feita. Nas abordagens tradicionais a avaliação da acessibilidade é desenvolvida de uma forma monolítica, testando diferentes diretrizes de acessibilidade de domínio específico. A ferramenta proposta permite a avaliação de acessibilidade de conteúdo *web* sob diferentes perfis de deficiência, tecnologias e dispositivos de assistência, bastando configurar a ferramenta para a análise que deseja ser feita.

A linha de estudo apresentada por Thiessen (2011), é muito interessante para o contexto de estudo do Homero, pois assim como a WCAG 2.0, o WAI-ARIA é uma especificação que permite ao desenvolvedor prover semântica à ambientes *web* complexos. No estudo a preocupação do autor foi analisar o comportamento da especificação do ARIA

funcionando juntamente com o HTML 5. Tanto o HTML 5 quanto o ARIA são suportados, ao menos parcialmente, pela maioria dos navegadores recentes, porém a combinação das duas especificações pode levar a conflitos de semântica. Tais conflitos foram provados pelo estudo empírico feito no artigo. Segundo o autor o problema pode estar na imaturidade da combinação entre HTML 5, ARIA, navegadores e leitores de tela, e é agravado pela divergência entre os fornecedores.

A dificuldade de acesso a tecnologias assistivas, cotidianamente, foi discutida por Wald *et al.* (2011), que analisaram a dificuldade em utilizar tal tecnologia independente de local, pois muitas vezes são instaladas localmente nos computadores. Como solução propuseram um conjunto de ferramentas que fornecem acessibilidade. São eles: i) barra de ferramentas, que trabalha com todos os navegadores e com a maioria dos *sites* acessíveis, permitindo ações básicas de acessibilidade (redimensionamento de texto, alteração de cores, leitura de texto, entre outros); ii) menu acessível desenvolvido para verificar a acessibilidade e usabilidade de *sites* e aplicações; e iii) *site* que permite aos usuários testar qualquer aplicação *web* nas diretrizes da WCAG 2.0.

Com o aumento da *cloud computing, ou* computação na nuvem, Mangiatordi *et al.* (2011) apresentaram uma ferramenta chamada *Farfalla*, que propicia tecnologia assistiva baseada na nuvem, sendo as soluções de acessibilidade implementados junto ao conteúdo. *Farfalla* oferece aos usuários um conjunto de ferramentas (*plugins*) flexíveis, leves e que não precisam ser instalados na máquina. Este *framework* não trabalha com conformidade de diretrizes e sim com conhecimento empírico dos desenvolvedores de aplicações *web* e de *plugins*.

No âmbito nacional, Maia (2010) propôs um Modelo de Tarefas de Acessibilidade (MTA) inseridas nos subprocessos do Processo de Desenvolvimento da Norma ISO/IEC 12207 (Padrão para Tecnologia da Informação – Processos do Ciclo de Vida do *Software*) para auxiliar a análise, projeto, implementação e avaliação de *softwares* acessíveis. Para apoio ferramental aos subprocessos definidos no MTA, a autora adaptou um *framework* existente para a geração de *sites e-gov*, denominado Pantaneiro (Sandim *et al.*, 2006), de tal forma que as páginas geradas pelo mesmo atendessem a critérios de acessibilidade *web* contemplados pelo MTA.

Como constatado nos trabalhos, existem várias linhas de pesquisa buscando propiciar acessibilidade em aplicações *web* e evidenciando a dificuldade em alcançar esta, porém não foi encontrado na literatura nenhum artigo apresentando uma solução que facilite a aplicação das diretrizes da WCAG 2.0 em aplicações de uso geral.

2.5. Considerações Finais

Neste capítulo foram apresentados os principais conceitos de acessibilidade, o histórico das lutas para obtenção da mesma, as leis promulgadas em favor da inclusão de pessoas com necessidades especiais e as progressões da área. Foram apresentados também os conceitos de acessibilidade *web* que o trabalho proposto busca disseminar através da simplificação do uso das diretrizes da WCAG 2.0.

O levantamento bibliográfico conduzido revelou o estado da arte na área das pesquisas feitas sobre acessibilidade *web* e comprovou o interesse dos pesquisadores por essa área. Observou-se também que, apesar da iniciativa, um número muito reduzido de trabalhos propuseram formas eficientes de facilitar o emprego dos conceitos de acessibilidade *web* e nenhuma iniciativa de reúso das diretrizes da WCAG 2.0 foi encontrada. Devido a essa carência uma abordagem que cubra esta lacuna, como feito pelo trabalho proposto, é muito interessante.

3. DESENVOLVIMENTO DE SOFTWARE

A forma encontrada para disseminar os conceitos de acessibilidade *web* foi propor um *framework* de *software* que pudesse ser utilizado para facilitar a criação de camadas de interface de aplicações *web* acessíveis, utilizando as diretrizes da WCAG 2.0. Neste capítulo são apresentados os conceitos envolvidos nessa proposta. Primeiramente são apresentadas as definições de *frameworks* e os principais modelos de desenvolvimento de *software*, já que o trabalho proposto visa facilitar a inserção de conceitos de acessibilidade nesses processos. Na sequência são discutidos os modelos de construção de *frameworks*, visto que estes possuem uma abordagem própria, com estruturas e análises não aderentes aos modelos tradicionais de desenvolvimento.

3.1. Framework de Software

Em desenvolvimento de *software* são muitas as definições para *frameworks* encontradas na literatura, dentre elas, podem ser analisadas a defendida por Appleton (1997), que apresenta um *framework* de *software* como uma arquitetura reusável que fornece comportamento e estrutura genérica para uma família de abstrações de *software*, ou a de Fayad e Johnson (2000) que afirma que um *framework* é mais do que uma arquitetura, é uma aplicação semi-completa contendo componentes estáticos e dinâmicos que podem ser adaptados para produzir aplicações específicas do usuário. De acordo com Braga (2003), independente de qual definição seja abordada na análise, todas têm um enfoque em comum, que é o de facilitar o reuso, entendendo que tal reuso não é somente o aproveitamento de códigos e programas, mas também dos esforços realizados em todas as fases do desenvolvimento de *software*.

De acordo com Fayad e Johnson (2000) existem três tipos de *framework*:

- Caixa-branca: a integração é feita através de implementação de classes que herdam de classes do *framework* ou que implementem interfaces pré-definidas. O *framework* caixa-branca tem a vantagem de ser mais flexível, porém o desenvolvedor deve conhecer detalhes de implementação dificultando o processo de aprendizado e uso.
- Caixa-preta: a integração é feita através da ligação de objetos da aplicação a objetos do *framework*. Essa estrutura apresenta uma interface de comunicação

relativamente simples, não sendo necessário conhecer a estrutura do *framework* o que torna seu uso menos flexível.

- Caixa-cinza: é a combinação das características do caixa-preta e caixa-branca.

Os *framework*s são classificados por Fayad e Schmidt (1997) em três grupos:

- Infraestrutura: simplificam o desenvolvimento da infraestrutura de sistemas, como por exemplo os sistemas operacionais, sistemas de comunicação, interfaces com o usuário e ferramentas de processamento de linguagem. Em geral são usados internamente em uma organização de *software* e não são vendidos a clientes diretamente.
- Integração de "*middleware*": em geral, são usados para integrar aplicações e componentes distribuídos. Eles são projetados para melhorar a habilidade de desenvolvedores em modularizar, reutilizar e estender sua infraestrutura de *software* para funcionar em um ambiente distribuído.
- Aplicação: voltados a domínios de aplicação mais amplos e são aplicados em atividades de negócios nas empresas, como por exemplo, sistemas de telecomunicações, aviação, manufatura e engenharia financeira.

3.2. Processos de Desenvolvimento de Software

O Guia PMBOK®[3] (*Project Management Body of Knowledge)* define processo como sendo um conjunto de atividades inter-relacionadas realizadas para obter um conjunto específico de produtos, resultados ou serviços (PMBOK, 2008). Para o CMMI[4] (*Capability Maturity Model – Integration)*, um processo é definido quando se tem uma descrição que é mantida, ou seja, tem documentação que detalha o que é feito (produto), quando (etapas), por quem (papéis), os itens utilizados (insumos) e os itens produzidos (resultados) (CMMI, 2006). Focando no desenvolvimento de *software*, Ian Sommerville define um processo de *software* como um conjunto de atividades que leva à produção de um produto de *software* (Sommerville, 2007). Roger S. Pressman define processo de *software* como um arcabouço para as tarefas que são necessárias para construir *software* de alta qualidade (Pressman, 2006).

[3] Livro publicado pelo *Project Management Institute* que apresenta um conjunto de práticas em gestão de projetos e constitui a base do conhecimento em gerenciamento de projetos do PMI.
[4] É uma abordagem desenvolvida pelo *Software Engineering Institute* – SEI da Universidade Carnegie Mellon, que busca a de melhoria de processos de software, fornecendo às organizações elementos essenciais para que estes sejam eficazes.

Um processo de *software* tem, quatro objetivos fundamentais Booch *et al*. (2005):

- Providenciar orientação sobre a sequencia de realização das atividades envolvidas.
- Especificar os modelos descritivos do sistema que devem ser desenvolvidos.
- Dirigir as tarefas dos participantes e da equipe como um todo.
- Providenciar critérios para o monitoramento e a avaliação dos modelos e atividades do projeto.

Muitas vezes identificados como paradigmas, os modelos de desenvolvimento de *software* surgiram no final dos anos 60 para reverter à chamada crise do *software*[5]. Essa noção de crise foi referenciada por Edsger Dijkstra em apresentação feita na a *ACM - Association for Computing Machinery* (Associação para Maquinaria da Computação) em 1972 (Dijkstra, 1972). Os modelos de desenvolvimento mais difundidos são apresentados nas seções subsequentes.

3.2.1. Modelo Cascata

O Modelo Cascata, apresentado na Figura 3.1, também conhecido como modelo clássico, foi criado por Royce (1970). A principal característica desse modelo é o fato das atividades estarem distribuídas de forma linear e sequencial. A proposta inicial trazia como característica a retroalimentação, que tornava o modelo menos linear, porém a prática deturpou essa ideia e o modelo passou a ser sequencial, exigindo que uma etapa esteja totalmente completa para desencadear a próxima.

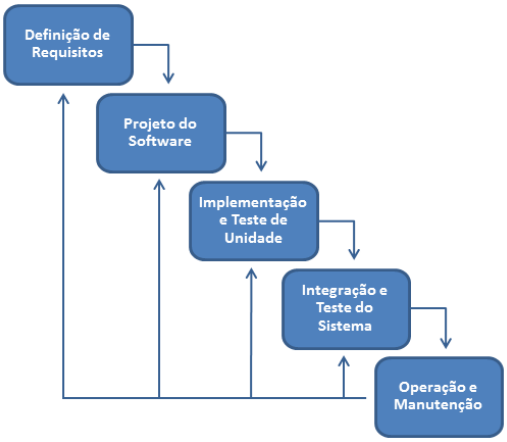


Figura 3.1 – Modelo Cascata.

[5] O termo Crise do Software surgiu para descrever as dificuldades enfrentadas no desenvolvimento de software no fim da década de 60. Entre suas características podemos encontrar problemas complexos, a ausência de técnicas bem estabelecidas e a crescente demanda por novas aplicações que começavam a se tornar um problema sério (Sommerville, 2007).

3.2.2. Modelo Espiral

O Modelo Espiral, proposto por Boehm (1988), é delimitado por uma sequencia de fases, em formato espiral, onde cada ciclo resulta em versões incrementais do *software*. Conforme apresentado na Figura 3.2, o ciclo inicia com a delineação do objetivo esperado e termina com o planejamento da próxima fase do desenvolvimento do *software*. Esforços de análise e engenharia são aplicados em cada fase, focado sempre no objetivo do projeto.

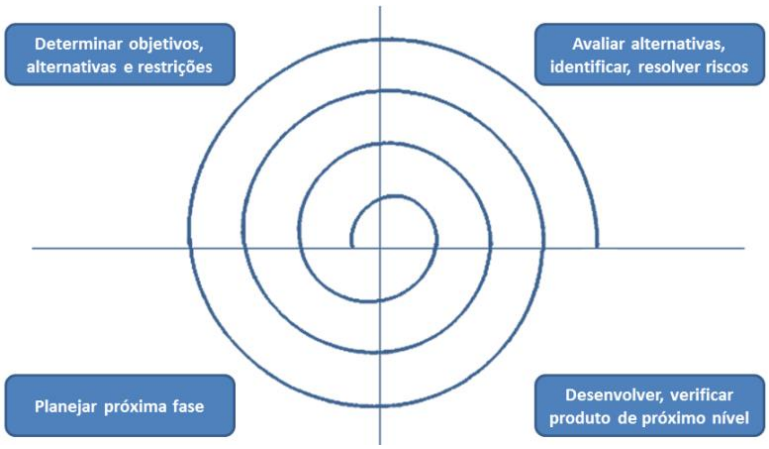


Figura 3.2 – Modelo Espiral.

3.2.3. Modelo Incremental

O Modelo Incremental (Pfleeger, 2004), assim como o Espiral, parte do princípio que o produto final não precisa ser entregue como um único pacote, podendo possuir várias partes desenvolvidas separadamente, conforme a prioridade. Segundo esse modelo, Figura 3.3, o *software* é desenvolvido em partes que são priorizadas, e a partir disso inicia-se o desenvolvimento em iterações, evoluindo o produto em versões, através da construção incremental e iterativa das novas funcionalidades até a versão final do sistema.

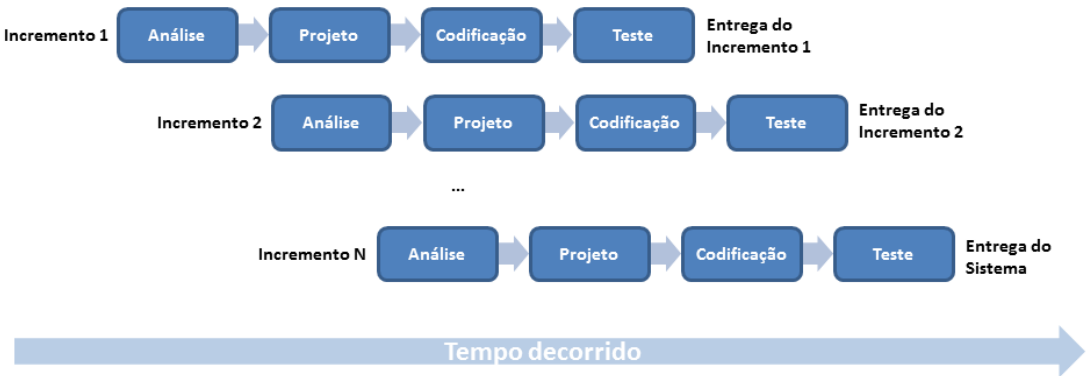


Figura 3.3 – Modelo Incremental.

3.2.4. Prototipação

Na Prototipação, Figura 3.4, o desenvolvedor interage diretamente com o usuário, escutando seus pedidos e desenvolvendo, imediatamente, um protótipo do produto desejado. O usuário, então, utiliza esse protótipo e fornece a equipe de desenvolvimento novas informações, levando a atualizações, adaptações e implementações no *software*, em tempo de projeto e desenvolvimento. Essa técnica pode ser utilizada em conjunto com outros modelos de desenvolvimento reforçando o momento de análise (Pressman, 2006).

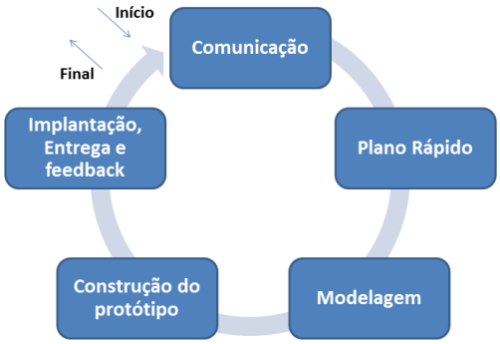


Figura 3.4 – Prototipação.

3.2.5. Rational Unified Process (RUP)

Criado em 1998 pela *Rational Software Corporation* baseado nas melhores práticas de vários outros processos de desenvolvimento que o antecederam, o processo unificado consiste em uma abordagem do ciclo de vida de um processo composto por 4 fases Booch *et al*. (2005). O RUP apresenta modelos de artefatos e roteiros para o cumprimento de cada atividade. Um diferencial da abordagem do RUP é o fato de apresentar a modelagem de negócios durante o início do ciclo de desenvolvimento do produto, que torna o entendimento das necessidades mais claras. O RUP também é incremental e iterativo, garantindo entregas de versões do produto no decorrer do processo de desenvolvimento.

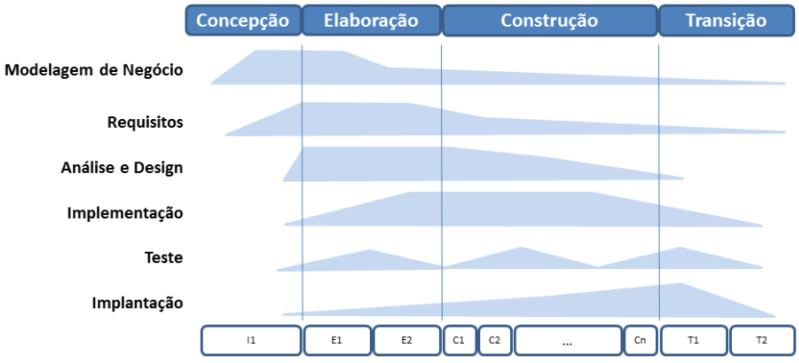


Figura 3.5 – Rational Unified Process (RUP).

Na Figura 3.5 estão representadas as disciplinas da engenharia de *software* indicando o que será feito durante o desenvolvimento (linhas), e as fases do desenvolvimento (colunas). Na intersecção entre linha e coluna é representado o esforço empregado em cada momento.

3.2.6. Métodos Ágeis

Fowler (2005) apresenta as metodologias ágeis como uma reação do mercado aos modelos tradicionais de desenvolvimento, que tiveram dificuldade em ser utilizados na indústria por serem muito burocráticos e exigirem documentação excessiva. As principais características, que diferenciam os métodos ágeis dos tradicionais são:

- Métodos ágeis são mais adaptativos, pois acolhem a mudança a qualquer momento, a ponto de adaptar a própria metodologia para serem bem sucedidos, enquanto os métodos tradicionais procuram planejar em detalhes longos períodos, apresentando resistência à mudança.
- Métodos ágeis baseiam-se nas pessoas, pois asseguram que nenhum processo definido possa sobrepor as habilidades da equipe, já os tradicionais procuram definir processos que vão funcionar com qualquer um que o execute.

Existem diversos processos e metodologias consideradas ágeis, dentre elas tem-se o Scrum – Figura 3.6 e o *Extreme Programming* (XP) – Figura 3.7.

O Scrum foi idealizado por Schwaber e Beedle (2002) e possui como objetivo fornecer um processo conveniente para a gerência de projetos. É um processo iterativo e incremental que pode ser aplicado a qualquer produto, ou no gerenciamento de qualquer atividade complexa. O método é incremental, iterativo e baseia-se em princípios como: equipes pequenas, requisitos estáveis ou desconhecidos e interações curtas.

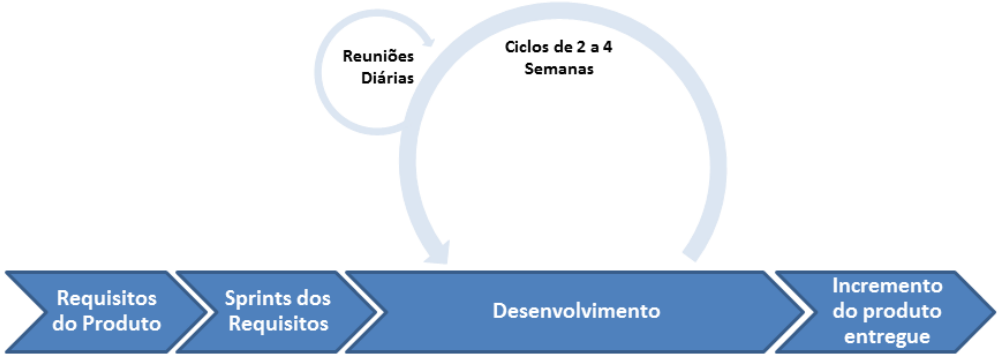


Figura 3.6 – Scrum.

O XP é um método ágil que foi pensado para desenvolvimento de *software* com requisitos vagos ou em constante mudança. Utilizando a estratégia de acompanhamento, realiza vários ajustes durante o desenvolvimento de *software*. Os cinco valores fundamentais da metodologia XP são: comunicação, simplicidade, *feedback*, coragem e respeito. A partir desses valores, possui como princípios básicos: *feedback* rápido, presumir simplicidade, mudanças incrementais, abraçar mudanças e trabalho de qualidade (Pressman, 2006).

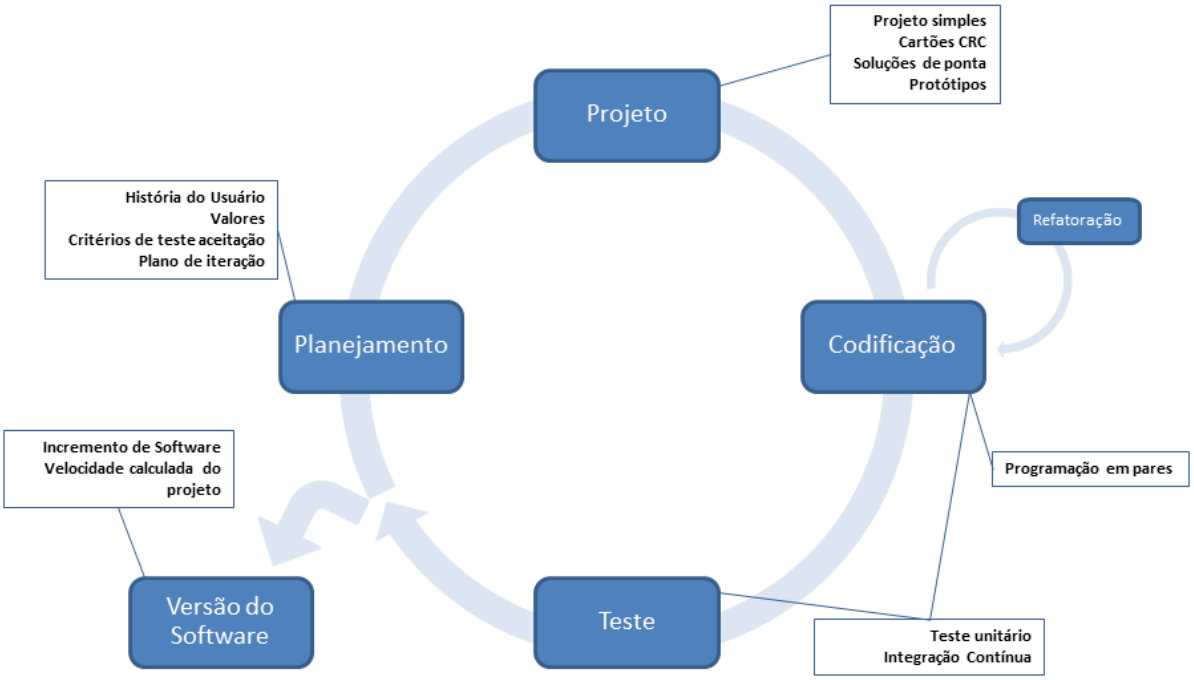


Figura 3.7 – eXtreme Programming.

3.2.7. Fases Comuns

Cada processo de desenvolvimento de *software* proposto apresenta características próprias e adequadas a determinadas situações, porém algumas atividades fundamentais são comuns a todos eles, estas podem ser vistas na Figura 3.8 (Sommervile, 2007).

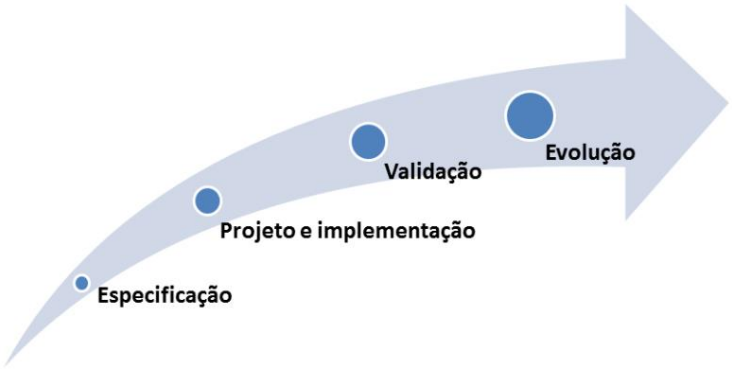


Figura 3.8 – Fases comuns.

Na fase de especificação é definida a funcionalidade do *software* e as restrições sobre sua operação. Na fase de projeto e implementação o *software* é efetivamente construído, de tal que atenda a especificação feita. Após ser construído o *software* passa pela fase de validação, garantindo que este faça o que o cliente deseja. A fase de evolução é responsável por atender os novos requisitos que naturalmente surgirão.

3.3. Processos de Construção de Framework

A construção de *frameworks* de *software* não é uma tarefa trivial, possuindo análises e estruturas diferenciadas das utilizadas corriqueiramente no desenvolvimento de aplicações, tanto que processos de construção de *software* tradicionais não são suficientes. Na literatura foram encontradas algumas propostas para a construção de *frameworks*, destacando-se dentre elas a de Pree *et al*. (1999), a de Schmid (1997) e a de Bosch *et al*.(1999).

Pree *et al*. (1999), conforme pode ser analisado na Figura 3.9, propõe um processo que se inicia pela definição de um modelo de objetos de uma aplicação específica que é refinado por meio de um ciclo de construção repetido sucessivamente. A abordagem de Pree inicia-se identificando e documentando os pontos variáveis do *framework*, que são então projetados e implementados. O *framework* é testado para garantir que satisfaça aos requisitos do domínio. Novos pontos variáveis podem ser encontrados e o ciclo se repetir. No fim de cada interação, o *framework* é avaliado para determinar se pode ser liberado para o uso ou se deve passar por outro ciclo de refinamento.

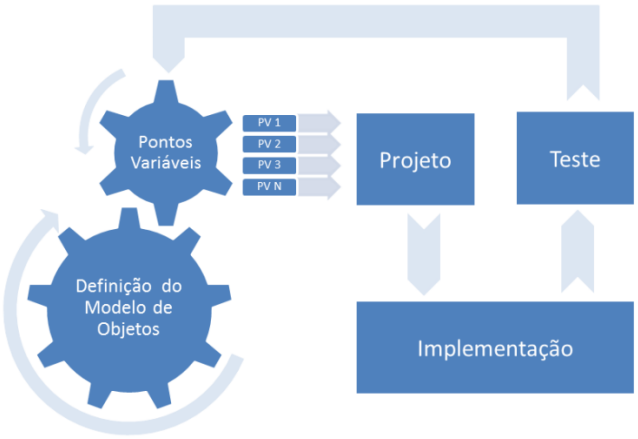


Figura 3.9 – Construção de frameworks proposta por Pree.

A técnica de Schmid (1997), elucidada na Figura 3.10, sugere que o *framework* seja desenvolvido pela generalização sistemática com base em um modelo de classes de uma aplicação. Inicialmente faz-se uma análise de alto nível para determinar os pontos variáveis e

estabelecer os principais aspectos do sistema que precisam ter flexibilidade. A seguir, cada ponto variável é analisado produzindo sua especificação, que ainda passa pelo projeto de alto nível. Por último, as classes da aplicação são transformadas em classes do *framework*, através da generalização sistemática, ou seja, generaliza-se o comportamento das classes de aplicação até obter-se classes com o comportamento do domínio da aplicação.

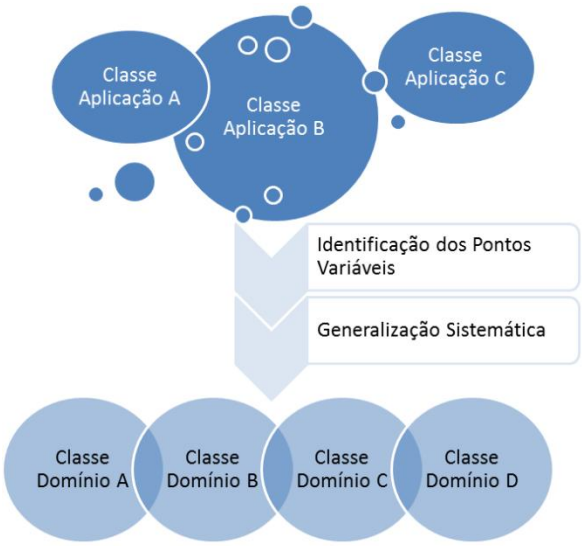


Figura 3.10 – Construção de framework proposta por Schmid.

O modelo de Bosch *et al*. (1999) é composto por seis fases, que podem ser vistas na Figura 3.11. A primeira fase trata da análise do domínio, que é executada para descrever o domínio coberto pelo *framework* e para captar seus principais conceitos e requisitos. Na segunda, cria-se o projeto arquitetural do *framework*. Na terceira fase, refina-se esse projeto, que é implementado na quarta fase. Na quinta são feitos os testes do *framework* para avaliar tanto sua funcionalidade quanto usabilidade. Finalmente, na sexta fase o *framework* é documentado e é criado um guia do usuário. Bosch *et al*. (1999) apresenta em sua abordagem uma preocupação maior que as demais com a usabilidade do *framework*, visto que nos testes validação da usabilidade e na documentação é confeccionado um guia para o usuário.

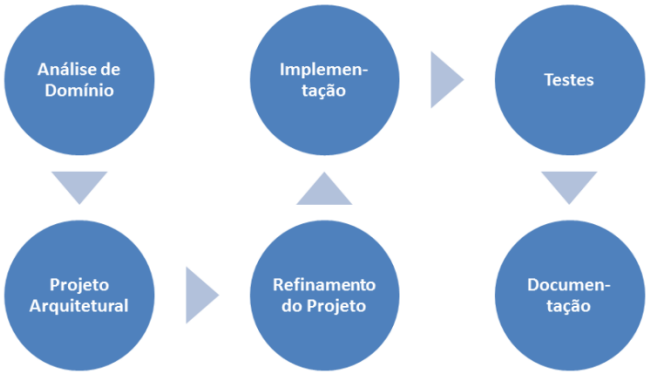


Figura 3.11 – Construção de frameworks proposta por Bosch.

3.4. Considerações Finais

Neste capítulo foram apresentados os conceitos e definições de *frameworks* pelo fato de que, como a proposta do trabalho é desenvolver um, precisa-se entender os conceitos envolvidos nessa abordagem, seu funcionamento e sua classificação. Os processos de desenvolvimento foram discutidos por questões de entendimento de em que fase utilizar a ferramenta. No caso do trabalho proposto, este deve ser utilizado na fase de implementação.

Os processos de desenvolvimento de *software* tradicionais não são suficientes para desenvolver *frameworks*, sendo assim foram analisadas propostas próprias para construção de *frameworks*. Para a concepção do trabalho foi utilizada a abordagem de Bosch com algumas modificações, apresentadas na Seção 4.2.

No próximo capítulo é apresentado o *framework* desenvolvido, o processo de construção adotado e a forma de utilização do mesmo.

4. HOMERO: UM FRAMEWORK DE APOIO AO DESENVOLVIMENTO DE INTERFACES DE APLICAÇÕES WEB ACESSÍVEIS

Neste capítulo é apresentado o *framework* Homero, que busca facilitar o desenvolvimento de aplicações *web* acessíveis, através da simplificação do uso das diretrizes da WCAG 2.0. A princípio é apresentada a versão *Beta* do *framework*, desenvolvida em 2010 pelo autor deste trabalho (Oliveira, R. C. 2010). Na sequencia é discutido o processo utilizado na criação e evolução do Homero para que as diretrizes da WCAG 2.0 fossem contempladas com êxito. A versão final é então explicitada, discutindo a forma como o Homero trabalha com acessibilidade, a fase do processo de desenvolvimento que o mesmo deve ser empregado e a forma de utilizar o *framework*, desde a instanciação até o uso de suas classes.

4.1. Homero: Um Framework de Apoio ao Desenvolvimento de Aplicações Web Acessíveis – Versão Beta

O *framework* Homero surgiu em 2010 pelo entendimento da necessidade de inserir acessibilidade nas aplicações *web*, tanto nas já existentes como nas que viessem a surgir. A versão *Beta* foi desenvolvida pelo autor deste trabalho (Oliveira, R. C. 2010), como trabalho de conclusão do curso de Ciência da Computação da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul – FACOM/UFMS.

A linguagem de programação utilizada na criação do Homero foi o PHP, uma linguagem livre, interpretada, híbrida (possibilita programação orientada a objetos e procedimental), permissiva (conversão de tipos é implícita) e rica em funções (PHP, 2013). Adicionalmente, o PHP é uma das linguagens mais utilizadas para gerar conteúdo dinâmico na *World Wide Web*. Desde o desenvolvimento da versão *Beta* do Homero até a conclusão da versão atual do *framework*, o PHP sempre esteve entre as 6 linguagens de programação mais utilizadas no mundo, segundo o *site* TIOBE Software, que faz análises estatísticas sobre o uso das linguagens de programação disponíveis no mercado (Tiobe, 2013). Na análise de 2013 o PHP ocupa o sexto lugar, como pode ser analisado na Figura 4.1.

Position May 2013	Position May 2012	Delta in Position	Programming Language	Ratings May 2013	Delta May 2012	Status
1	1	=	C	18.729%	+1.38%	A
2	2	=	Java	16.914%	+0.31%	A
3	4	↑	Objective-C	10.428%	+2.12%	A
4	3	↓	C++	9.198%	-0.63%	A
5	5	=	C#	6.119%	-0.70%	A
6	6	=	PHP	5.784%	+0.07%	A
7	7	=	(Visual) Basic	4.656%	-0.80%	A
8	8	=	Python	4.322%	+0.50%	A
9	9	=	Perl	2.276%	-0.53%	A
10	11	↑	Ruby	1.670%	+0.22%	A
11	10	↓	JavaScript	1.536%	-0.60%	A
12	12	=	Visual Basic .NET	1.131%	-0.14%	A
13	15	↑↑	Lisp	0.894%	-0.05%	A
14	18	↑↑↑↑	Transact-SQL	0.819%	+0.16%	A
15	17	↑↑	Pascal	0.805%	0.00%	A
16	24	↑↑↑↑↑↑↑↑	Bash	0.792%	+0.33%	A
17	14	↓↓↓	Delphi/Object Pascal	0.731%	-0.27%	A
18	13	↓↓↓↓↓	PL/SQL	0.708%	-0.41%	A
19	22	↑↑↑	Assembly	0.638%	+0.12%	B
20	20	=	Lua	0.632%	+0.07%	B

Figura 4.1 – Estatística do uso das linguagens de programação (Tiobe, 2013).

O *framework* desenvolvido era caixa-preta, visto que os usuários não precisavam conhecer o funcionamento interno do mesmo – Seção 3.1, orientado a objetos e atendia as diretrizes automatizáveis de acessibilidade do nível A da WCAG 2.0. Outra característica importante da versão *Beta* do Homero é o fato da mesma ter sido projetada para não gerar *tags* HTML em desuso, estando alinhada com o HTML 4.1 e o XHTML 1.1, que eram as versões mais atuais na época do desenvolvimento da versão *Beta*, e do mesmo forçar a utilização de classes CSS[6] ao invés de permitir que o estilo de cada elemento fosse definido no corpo do código, o que é desaconselhado pela W3C, pois perde-se em reuso de código e dificulta a manutenção. A linguagem que devia ser empregada na instanciação e uso da versão *Beta* do Homero também era o PHP.

Um dos principais problemas de desenvolvimento em PHP é a mistura de código PHP com *tags* HTML, pois para iniciar a programação PHP basta sinalizar início, utilizando a *tag* inicial de PHP (`<?php`), codificar o necessário e sinalizar final com a *tag* final de PHP (`?>`). Quando isso é feito muitas vezes gera-se um código complexo e com manutenibilidade comprometida. A versão *Beta* do *framework* diminuía tal necessidade, pois o desenvolvedor

[6] *Cascading Style Sheets* ou Folha de Estilo em Cascata

utilizava somente o PHP para trabalhar com os elementos contemplados pelo Homero e os métodos definidos nas classes geravam o HTML de maneira limpa e de fácil manutenção.

A estrutura da versão *Beta* do Homero era composta por sete classes, como pode ser visto na Figura 4.2, funcionando como elementos autocontidos que poderiam ser integrados em alguns momentos, mas que funcionavam basicamente sozinhas. A classe `Image` foi criada para tratar as imagens que fazem parte do contexto da aplicação *web*. A classe `Text` foi desenvolvida com o intuito de tratar os textos semanticamente e a `DivText` provia meios de mudar seu tamanho. A classe `Lst` foi criada para tratar as listagens, a `Table` para tratar as tabelas e a `Group` para trabalhar com formulários, tornando-os simples e previsíveis. A classe `Header` tratava cabeçalho da página.

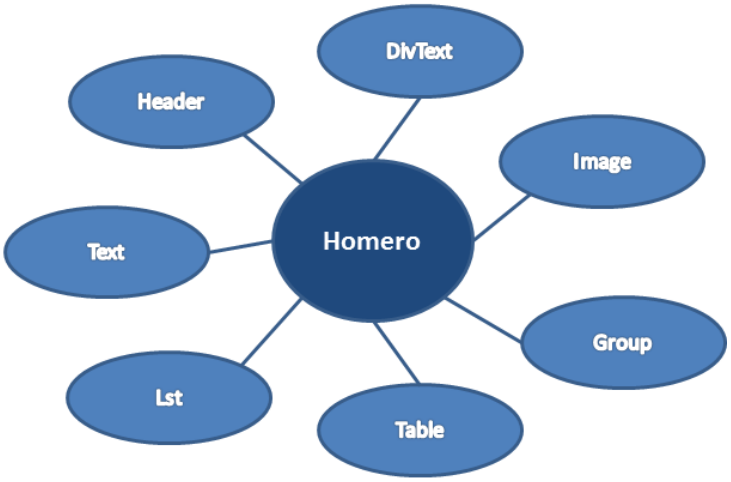


Figura 4.2 – Estrutura do framework Homero – versão Beta

O diagrama de classes simplificado da versão *Beta* pode ser observado na Figura 4.3, o completo está apresentado no Apêndice A. Pelo fato de se comportarem como componentes autocontidos, as classes não possuíam relacionamento entre si. Os atributos das classes foram definidos como privados para garantir o encapsulamento e baixo acoplamento e para acessa-los foram desenvolvidos métodos *setters* e *getters*. Como o PHP é uma linguagem fracamente tipada as validações dos tipos são feitas nos métodos *setters*, evitando que tal validação se tornasse repetitiva.

As classes da versão *Beta* foram definidas como finais, evitando que novas classes sejam derivadas. Tal decisão de projeto foi tomada para impedir que os métodos das classes fossem sobrescritos, pois caso a sobrescrita não se atentasse as validações de acessibilidade o conteúdo gerado poderia ter dados inacessíveis.

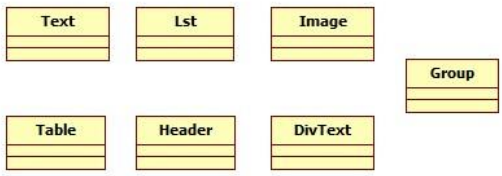


Figura 4.3 – Diagrama de classes do framework Homero – versão Beta.

Para a documentação no corpo das classes foi utilizada a notação em PHPDoc (PHPDoc, 2013), que é um padrão formal para comentar código PHP. Tal notação permite que, através de comandos de sistema ou aplicações disponíveis no mercado crie-se documentação para o código, e ainda, possibilitar a integração com ferramentas de desenvolvimento, como o caso da ferramenta Zend Studio (Zend Studio, 2013), que integra as classes desenvolvidas com o recurso de *auto-complete*, conforme ilustrado na Figura 4.4.

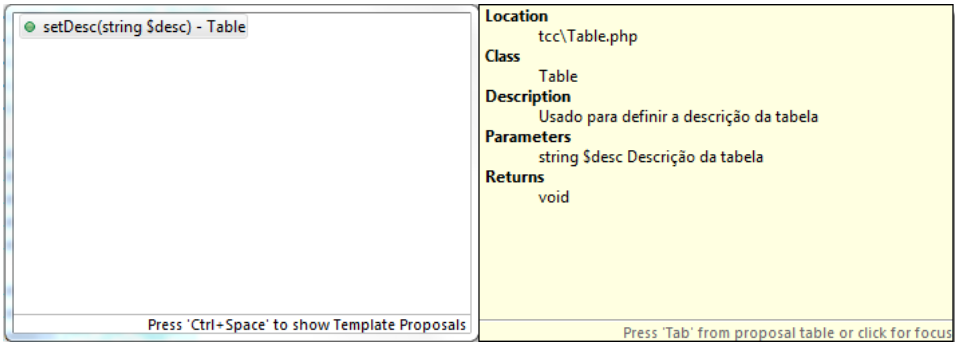


Figura 4.4 – Integração do Zend Studio com o PHPDoc.

A documentação gerada pelo PHPDoc comporta-se como um *site* e pode ser utilizada como um guia para entendimento das classes e formas de acessar as funcionalidades do *framework*. Tal documentação também é conhecida como CookBook. Na Figura 4.5 pode ser vista parte da documentação gerada.

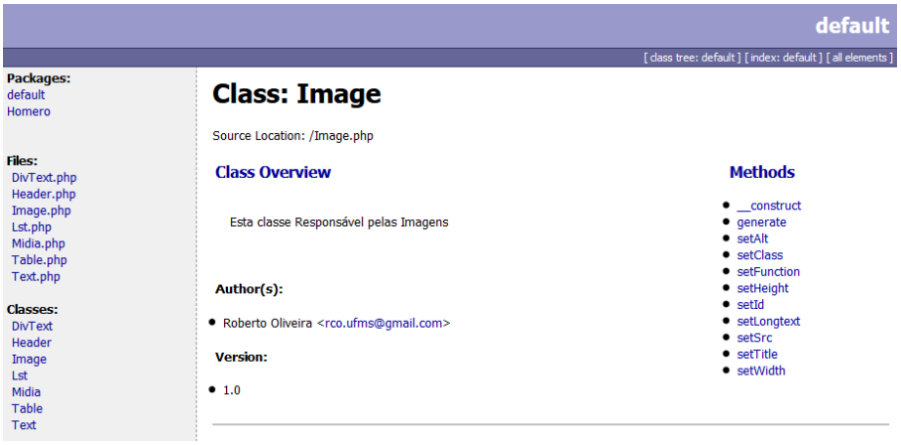


Figura 4.5 – Documentação do Homero – versão Beta gerado pelo PHPDoc.

O *framework* trabalhava de forma restritiva, obrigando o usuário a fornecer os dados de acessibilidade como parâmetros dos métodos de suas classes. Caso tais dados não fossem fornecidos, avisos de erros eram presentados durante a execução da aplicação *web*. Na Figura 4.6 – lado esquerdo, pode-se analisar um código utilizando a classe `Image` para criação de uma imagem. Na linha 9, quando o objeto foi instanciado, não se definiu o texto alternativo, que seria o quarto parâmetro do construtor. Como esse parâmetro, necessário para que o conteúdo seja acessível, também não foi definido posteriormente através de um *setter* correspondente, ao ser executado, o *framework* apresentou um "*Warning*" informando a falta deste parâmetro, como pode ser visto na Figura 4.6 – lado direito.

```
1  <?
2  include ("Homero.php");
3  $imp = new Header(null, "teste");
4  $imp->setLang("pt-br");
5  echo $imp->generate();
6  ?>
7  <body>
8  <?
9  $img2 = new Image("icones/wai.png","Menos",Menos,
10                   null,null,null,null,"desc.html");
11 echo $img2->generate();
12 ?>
13 </body>
```

Warning: O texto auxiliar da Imagem não foi especificado.
in **D:\www\Homero\Image.php** on line **319**

Figura 4.6 – Exemplo de utilização do Homero – versão Beta.

As principais limitações da versão *Beta* do Homero eram: i) o fato do mesmo ter contemplado somente as diretrizes automatizáveis do nível A da WCAG 2.0; e ii) ser necessário programar em PHP juntamente com HTML.

4.2. Processo de Criação, Adaptação e Evolução do Framework Homero

Na construção do Homero – versão *Beta* não foi utilizado nenhum processo de desenvolvimento estruturado, pois como as diretrizes analisadas eram somente as do nível A da WCAG 2.0, o número de classes geradas era reduzido, permitindo que a construção fosse feita de forma *ad-hoc*. Na evolução da versão *Beta* foi necessário analisar métodos de construção de *framework*, pois o número de diretrizes analisadas seria muito maior e consequentemente o número de classes também. Como visto na Seção 3.3 a construção de *frameworks* é complexa, com análises e estruturas diferenciadas das utilizadas em produtos finais de *software*. Dentre as abordagens discutidas, a de Bosch *et al*. (1999) foi a que apresentou maior aderência às necessidades da construção do Homero. Tal modelo é organizado em seis fases: análise de domínio, projeto arquitetural, refinamento,

implementação, teste e documentação. Mesmo sendo aderente, o modelo de Bosch *et al.* necessitou de algumas adaptações para que conseguisse contemplar com sucesso todas as diretrizes da WCAG 2.0. A Figura 4.7 apresenta o processo de construção do Homero.

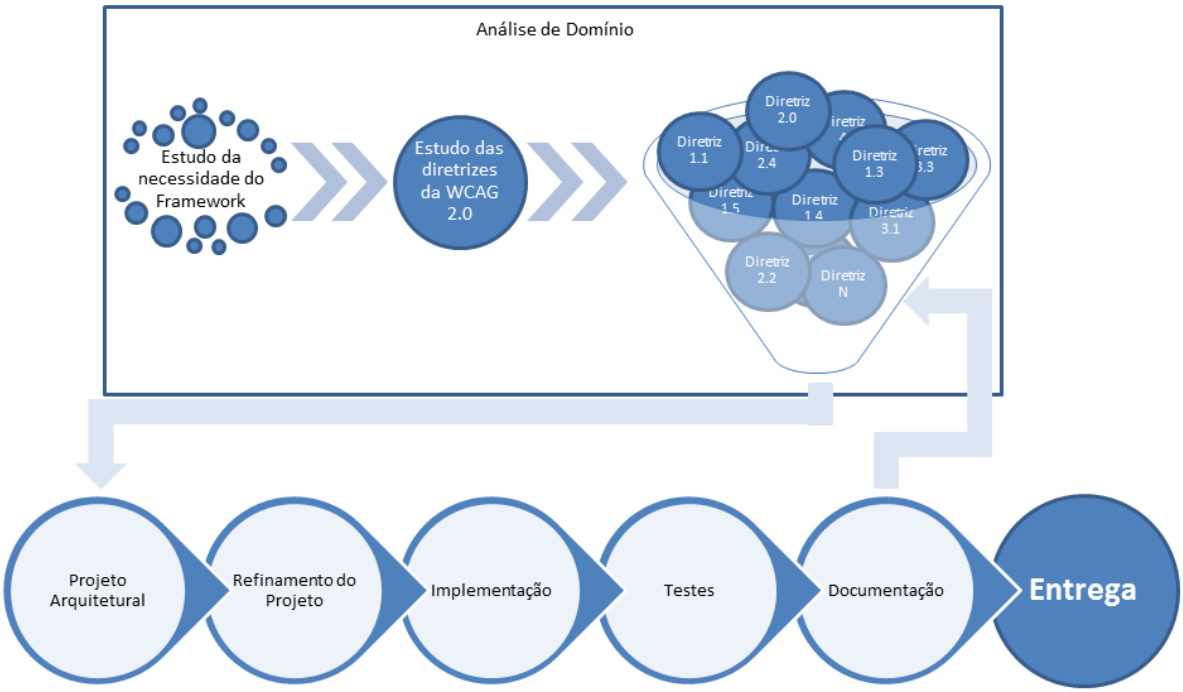


Figura 4.7 – Modelo utilizado na concepção do Homero.

Na fase de análise de domínio, foram verificadas as iniciativas de criar diretrizes que normalizassem o desenvolvimento de aplicações *web* acessíveis. Chegou-se a conclusão do uso das diretrizes da WCAG 2.0, pelo fato de, conforme visto na Seção 2.2, estas serem aderentes, e até terem servido de base, para as demais diretrizes de acessibilidade. No final da fase de análise de domínio as diretrizes e os critérios de sucesso foram levantados. As diretrizes representam o que deve ser feito e os critérios de sucesso apresentam como fazer. Nessa análise inicial não foi possível estimar o número de classes que seriam necessárias, e a partir daí deu-se início aos ciclos de construção do *framework*. No início de cada ciclo de desenvolvimento uma diretriz era selecionada para ser inserida no *framework*. Uma análise era então feita para verificar se essa diretriz poderia ser automatizada. Em caso afirmativo, iniciava-se o ciclo do desenvolvimento, caso contrário a mesma era descartada e escolhia-se outra diretriz.

Na fase de projeto arquitetural foi verificado onde a diretriz e o critério de sucesso se encaixariam na arquitetura do *framework*. Nesse momento foi analisado a mesma referia-se a um elemento já contemplado nas classes criadas ou se seria necessário implementar nova(s) classe(s). Na fase de refinamento do projeto foi analisado se a evolução ou criação da(s)

classe(s) iriam gerar algum comportamento similar a outras classes existentes, caso o resultado fosse positivo, uma nova superclasse era então criada, contemplando esse comportamento e essas classes similares começavam a herdar o comportamento desta. Na quarta fase as decisões tomadas no projeto e no refinamento foram implementados e o código documentado utilizando a notação PHPDoc (PHPDoc, 2013).

Os testes, elaborados na quinta fase, foram feitos utilizando a técnica de teste funcional, ou seja, aplicando-se uma entrada e observando sua saída. Esses testes foram divididos em dois tipos de testes: i) testes unitários: para cada diretriz implementada foi construída uma aplicação semi-completa utilizando a(s) classe(s) que contemplava(m) a diretriz e submetidas no validador *Total Validator* (2013), e ii) testes de integração: após o teste unitário obter o resultado positivo, uma aplicação completa foi desenvolvida, e evoluída, utilizando todas as classes existentes para verificar se essas novas classes desenvolvidas apresentariam conflito com as anteriores e também submetia-as no validador *Total Validator*. Caso algum conflito fosse apresentado este era tratado. Como as diretrizes contempladas no *framework* eram somente as automatizáveis, os testes foram feitos somente de forma automática (utilizando o validador).

A sexta fase, de documentação, foi relativamente simples, pois com o código documentado utilizando o PHPDoc (PHPDoc, 2013), foi utilizada uma ferramenta de apoio para geração automática da documentação e do guia do usuário (CookBook), no caso o Zend Studio (Zend Studio, 2013), que possui um *plugin* próprio para trabalhar com documentação. Essa documentação está localizada na raiz do *framework*, na pasta PHPDoc. Ao final da sexta fase, em cada ciclo, tinha-se uma versão concisa do *framework* que poderia ser utilizada, e uma nova diretriz era então escolhida para reiniciar o ciclo de desenvolvimento do *framework.*

4.3. Homero: Um Framework de Apoio ao Desenvolvimento de Interfaces de Aplicações Web Acessíveis

Procurando suprir as deficiências encontradas no *framework* Homero – Versão *Beta* foi proposta uma nova versão do Homero, que trabalhasse com todas as diretrizes automatizáveis de todos níveis de acessibilidade propostos pela WCAG 2.0. O nome do *framework* na nova versão foi alterado de “Homero: Um Framework de Apoio ao Desenvolvimento de Aplicações Web Acessíveis” para “Homero: Um Framework de Apoio ao Desenvolvimento de Interfaces de Aplicações Web Acessíveis” pelo fato de que ao

maturar o entendimento da funcionalidade do Homero constatou-se que este não apoia o desenvolvimento de aplicações e sim de interfaces de aplicações, sendo o novo nome mais aderente ao propósito do *framework.*

A nova versão do *framework*, assim como a versão *Beta*, é caixa-preta, orientado a objetos e foi desenvolvido utilizando as melhores práticas de desenvolvimento em PHP em organização e documentação do código-fonte (iMasters, 2013). As diretrizes atendidas pelo Homero passaram do nível A da WCAG 2.0 para AAA. Além disso, essa nova versão foi criada buscando alinhamento com o HTML 5.0, o que foi conseguido em quase todas as classes construídas, exceto na classe `Table`, pois a WCAG 2.0 sugere o uso de uma *tag* chamada `caption`, que descreve a natureza da tabela, mas esta não é suportada no HTML 5.0. Outro conflito entre o HTML e WCAG 2.0 levou a exclusão de uma classe do *framework*, pois a WCAG faz uma série de apreciações referentes ao elemento `applet`[7], porém este também não é mais suportado no HTML 5. Isso se deu pelo fato de as diretrizes da WCAG 2.0 terem sido propostas em 2008, momento que a versão corrente do HTML era a 4.01. A versão 5.0 do HTML foi apresentada posteriormente, com previsão de conclusão da especificação para o final de 2014. Para resolver essas incoerências, foi criada uma força tarefa do W3C para descobrir e resolver conflitos na especificação do HTML 5 e da WCAG 2.0 (W3C, 2011).

O Homero manteve o comportamento de forçar a utilização de classes CSS[8] para definição dos estilos dos elementos. O uso de estilos no corpo dos elementos é desaconselhado pela W3C, pois prejudica o reuso de estilos e polui o código, dificultando a manutenção. Também buscando melhorar a manutenibilidade das aplicações criadas a nova versão do Homero permite que os desenvolvedores utilizem somente o PHP para criação das interfaces *web*, gerando um código limpo e simplificando o aprendizado das equipes de desenvolvimento, que não terão mais que se preocupar com o entendimento do HTML, que é gerado automaticamente.

O número de classes do Homero passou de sete para trinta e nove. Para organizar melhor a estrutura do *framework* foi proposto um conceito de grupo de classes. Existem 11 grupos de classes, cada grupo composto por uma ou mais classes, como ilustrado na Figura 4.8.

[7] Um software aplicativo que é executado no contexto de outro programa, geralmente executando funções bem específicas.
[8] *Cascading Style Sheets* ou Folha de Estilo em Cascata

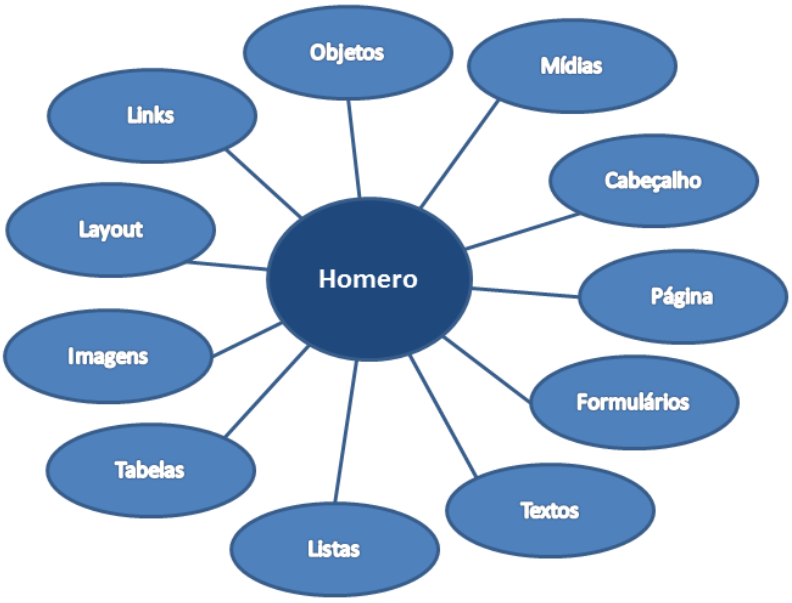


Figura 4.8 – Estrutura do framework Homero.

O grupo de classes Mídias é responsável pelas mídias do sistema (áudio e vídeo) e é formado pelas classes `Media` e `Source`. O grupo de classes Objetos representa os objetos do sistema, formado pelas classes `Object` e `Parameter`. O grupo Link, composto pela classe `Link`, trabalha com os *links* ou âncoras do sistema. O grupo Imagens é composto pela classe `Image`, e foi criada para tratar as imagens que fazem parte do contexto da aplicação. O grupo de classes Layout, composto pela classe `Div`, trabalha com as divisões de *layout* do *site*. As tabelas do sistema são tratadas pelo grupo Tabelas, que é composto pelas classes `Cell`, `Line` e `Table`. O grupo Listas, composto pela classe `Lst`, foi criado para tratar as listagens. O grupo de classes Textos é formado por 10 classes, `Superscript`, `Subscript`, `Emphasis`, `Cite`, `Span`, `LongExplanatory`, `ShortExplanatory`, `Strong`, `Paragraph` e `Title`, e trabalha com os diversos tipos de textos do sistema. O grupo Formulário, que trata dos formulários do sistema é formado pelas classes abstratas `ContainerData`, `Button` e `Input` e as classes concretas `Group`, `Textarea`, `InputPassword`, `InputCheckbox`, `InputRadio`, `InputFile`, `InputText`, `Select`, `Options`, `ButtonImage`, `ButtonReset` e `ButtonSubmit`. O grupo Cabeçalho, formado pela classe `Header`, trata do cabeçalho da página HTML. O grupo Página, composto pela classe `Page`, representa a página a ser desenvolvida.

Mesmo PHP sendo uma linguagem híbrida, que permite programação orientada a objetos e procedimental, o *framework* Homero é totalmente orientado a objetos, onde um objeto da classe `Page` é a representação da página *web*. Este objeto tem um atributo

cabeçalho, que aceita um objeto da classe `Header` e um atributo corpo, que aceita instâncias de outras classes. Esse comportamento fica claro no diagrama de classes simplificado, que pode ser observado na Figura 4.9, que foi adaptado de forma a apresentar as classes organizadas em seus respectivos grupos. O diagrama de classes completo, também organizado em grupos e constando seus métodos e atributos, pode ser visto no Apêndice B.

Mantendo o isolamento e acoplamento proposto na versão *Beta* as novas classes também tiveram seus atributos definidos como privados, com métodos *setters* e *getters*, e validações dos tipos de dados feitas nos *setters*. As classes do Homero foram definidas como finais, pelo mesmo motivo da versão *Beta*, que é evitar que sobrescritas de métodos não se atentem às validações de acessibilidade, prejudicando a acessibilidade das informações geradas.

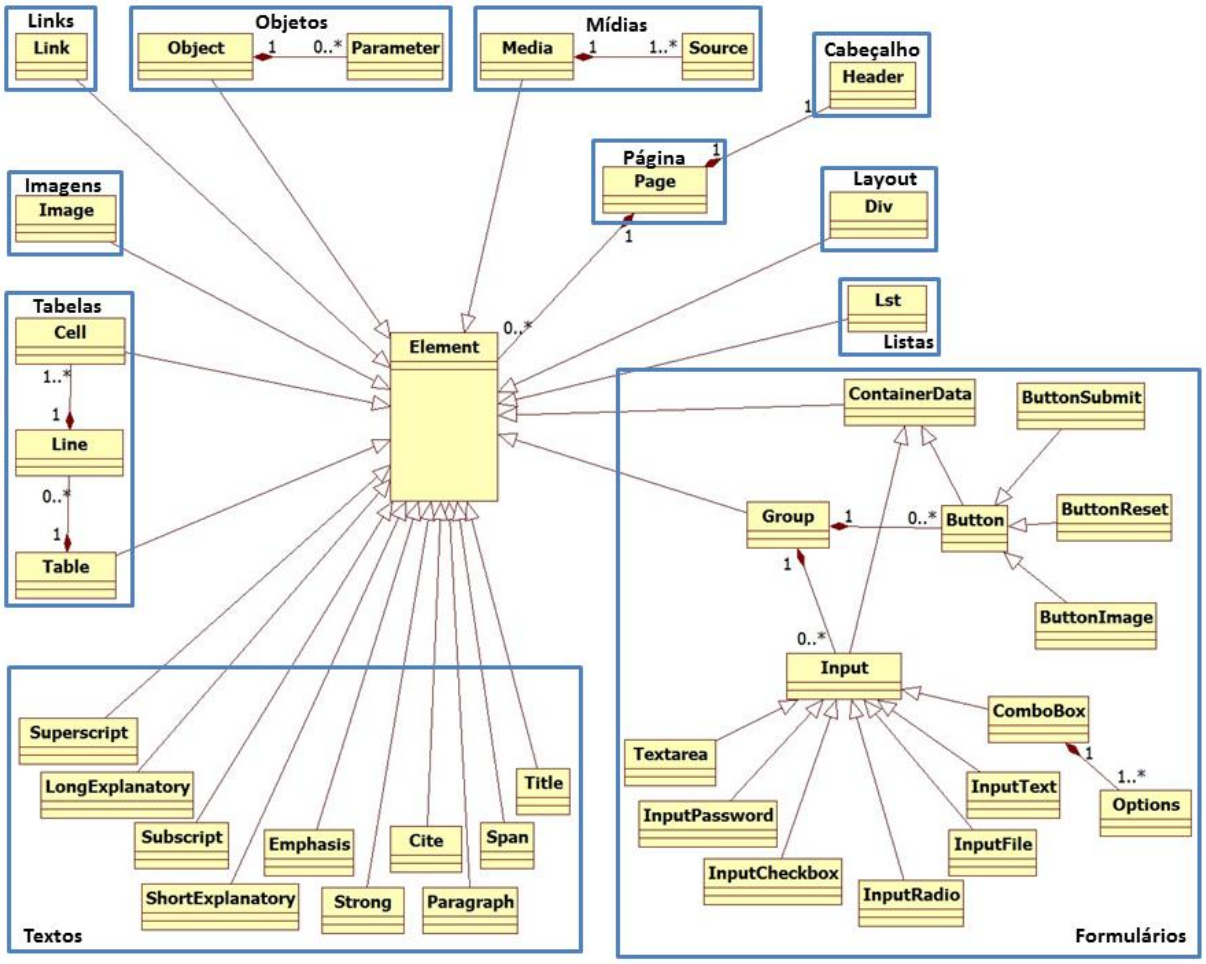


Figura 4.9 – Diagrama de classes do Homero.

Devido ao grande número de mudanças na estrutura e arquitetura do Homero, nada de codificação foi aproveitado da versão *Beta*, sendo as classes geradas do início, seguindo o processo de desenvolvimento proposto, como pode ser visto na Seção 4.2. Os

grandes benefícios trazidos da versão *Beta* foram os conceitos empregados na sua concepção, tanto os de desenvolvimento, como ser caixa-preta, orientado a objetos, alinhado com a versão mais atual do HTML, entre outros, quanto a de documentação, que manteve a utilização do PHPDoc, garantindo a integração com ferramentas de desenvolvimento e a criação automática de documentos do sistema. Como o número de classes da versão final do Homero é muito superior à versão *Beta* a documentação é muito mais extensa, porém como existe uma versão criada em forma de *site* pelo PHPDoc a navegação é simples e direta, como pode ser visto na Figura 4.10.

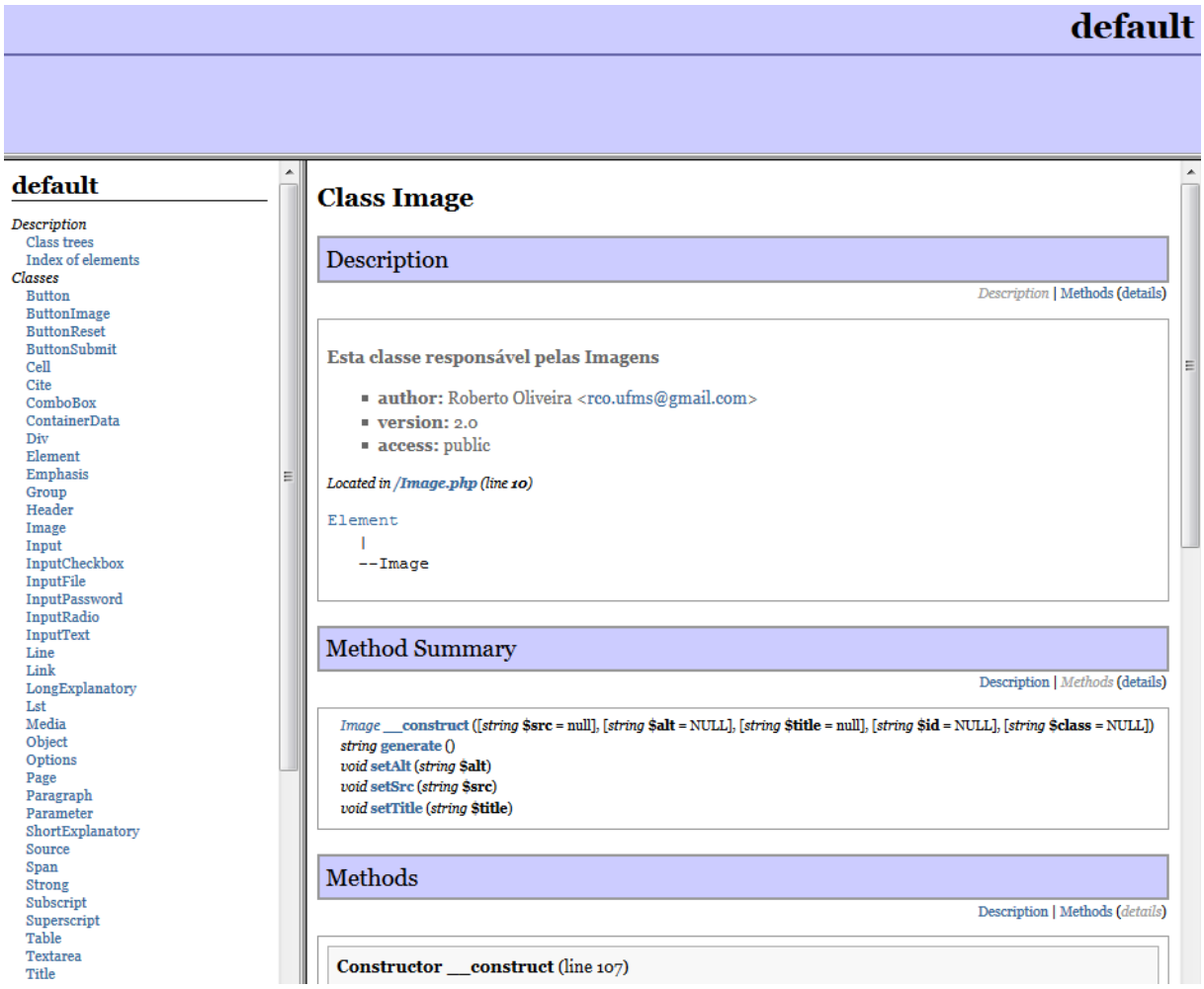


Figura 4.10 – Documentação do Homero gerada pelo PHPDoc.

O comportamento restritivo da versão *Beta* também foi mantido, cobrando sempre que o usuário forneça todos os dados de acessibilidade nos métodos das classes, sob pena de apresentar erros ("*Warning*") no momento da execução da aplicação *web*. Na Figura 4.11 - lado esquerdo, pode-se analisar um código escrito de uma pseudo-aplicação utilizando o Homero. Seguindo o mesmo exemplo utilizado na versão *Beta*, ao ser instanciado um objeto

da classe `Image`, na linha 9, não foi definido o texto alternativo da mesma, visto que o segundo parâmetro do construtor foi definido com valor nulo, o que causou um erro de acessibilidade na aplicação final, como pode ser visto na Figura 4.11 - lado direito.

```
1  <?
2  include ("Homero.php");
3  $header = new Header("Apresentação Homero");
4  $header->setMeta('http-equiv="Content-Type"
5          content="text/html; charset=iso-8859-1"');
6
7  $interface = new Page("pt-br");
8  $interface->setCabecalho($header->generate());
9  $img = new Image("icones/logo.png",null,
10                    "Homero", null,null);
11 $interface->setCorpo($img->generate());
12 $interface->generate();
13 ?>
```

Warning: O texto auxiliar da Imagem não foi especificado. in **D:\www\Homero\Image.php** on line **319**

Figura 4.11 – Exemplo de utilização do Homero.

Como mencionado anteriormente, pode-se verificar na codificação, apresentada na Figura 4.11 – lado esquerdo, que não é preciso o desenvolvedor se preocupar com a linguagem HTML, gerando um código simples, intuitivo e de fácil manutenção.

4.4. Processos de Desenvolvimento e o Homero

Como pôde ser analisado na Seção 3.2, existem vários processos de desenvolvimento de *software* que podem ser adotados, cada qual com características próprias e adequadas a determinadas situações, porém algumas atividades fundamentais são comuns a todos: especificação, projeto e implementação, validação de *software* e evolução.

O Homero é focado na fase de implementação, garantindo que os componentes levantados na fase de projeto sejam bem formados, com todas as informações de sintaxe cabíveis para serem acessíveis. Todavia, garantir que os componentes possuam elementos de sintaxe não é o único determinante para a acessibilidade de uma aplicação, pois as informações contidas nesses elementos devem possuir a informação correta, semanticamente bem formada.

Existem algumas decisões tomadas em nível de projeto que impactam diretamente na acessibilidade, como as cores dos elementos, ou o uso de determinados elementos, como títulos ou parágrafos. Tais decisões não podem ser mudadas no desenvolvimento, pois não cabe ao desenvolvedor alterar um projeto aprovado. Assim, para que o Homero gere uma interface de aplicação com informações sintaticamente acessíveis, o projeto desta também deve ser acessível.

Sendo assim surge a necessidade de tratar da acessibilidade em todas as fases do desenvolvimento, como se segue:

- Especificação: Levantar a necessidade do usuário, analisando os elementos que a aplicação deve ter e as informações para que estes sejam acessíveis.
- Projeto: Projetar a aplicação, organizando os elementos de forma que o conjunto seja acessível.
- Implementação: Desenvolver os elementos da aplicação garantindo que estes sejam acessíveis.
- Validação: Nos testes de sistema, verificar também quesitos de acessibilidade.
- Evolução: Ao evoluir o *software*, manter o cuidado com a acessibilidade, evitando que esta seja perdida.

Em cada fase do processo de desenvolvimento podem ser propostas várias ações para prover acessibilidade ao mesmo. Na Figura 4.12 é apresentado um exemplo, onde para cada fase é apresentada uma possível ação para buscar acessibilidade e evidenciar a atuação do Homero na fase de implementação.

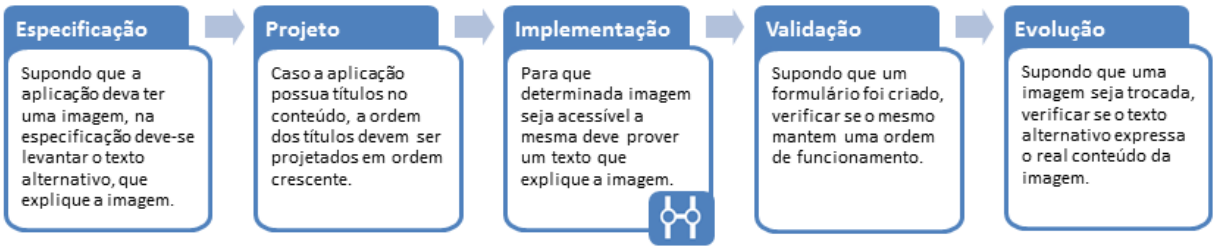


Figura 4.12 – Exemplo de ações de acessibilidade.

4.5. Instanciação e Uso do Homero

Para a instanciação do *framework* foi utilizada a propriedade da linguagem PHP, em que para a utilização de classes basta declará-las no corpo do arquivo ou incluí-las via função. Sendo assim, foi criado um arquivo chamado `Homero.php`, onde são feitas as inclusões cabíveis das classes para o correto funcionamento do *framework*, e o desenvolvedor deve incluí-lo no arquivo onde construirá a interface da aplicação (Figura 4.13).

```php
<?php
include("../Homero/Homero.php");
?>
```

Figura 4.13 – Instanciação do framework Homero.

Após tal inclusão basta declarar objetos das classes do *framework* para utilização das funcionalidades destes, incluindo geração dos códigos e validação da acessibilidade. Todo arquivo deve ter um objeto da classe `Page`. Esse objeto representa a interface da aplicação desenvolvida. Esse objeto possui atributo que recebe as informações do cabeçalho da página (inclusão de arquivos de scripts e de estilo e de meta dados) e outro que recebe os componentes da página (listas, parágrafos, formulários, etc). Para gerar a interface deve-se invocar o método `generate()` do objeto da classe `Page`.

```
1  <?php
2  include("../Homero/Homero.php");
3  $header = new Header("Apresentação Homero");
4  $header->setMeta('http-equiv="Content-Type" content="text/html; charset=iso-8859-1"');
5
6  $interface = new Page('pt-br');
7  $interface->setCabecalho($header->generate());
8
9  $imagem = new Image('imagens/logo.png', NULL, 'HOMERO',  NULL, NULL);
10 $imagem->setAlt('Logo do Homero');
11
12 $interface->setCorpo($imagem->generate());
13
14 $interface->generate();
15 ?>
```

Figura 4.14 – Exemplo detalhado do uso do framework Homero.

A Figura 4.14 exemplifica o uso do *framework*. O detalhamento de cada linha pode ser analisado a seguir:

- Linha 1: *Tag* de início de PHP.
- Linha 2: Instanciação do *framework*.
- Linha 3: Declaração de um objeto da classe `Header`, responsável pelo cabeçalho da página, passando o título da página como parâmetro.
- Linha 4: As informações de metadados da página são setados utilizando o método `setMeta()` do objeto `header`.
- Linha 6: Declaração de um objeto da classe `Page`, responsável pela interface da página, passando o idioma da página como parâmetro.
- Linha 7: O cabeçalho da página é inicializado utilizando a função `setCabeçalho()` do objeto `interface`. Como o `setCabeçalho()` aceita somente *strings*, como pode ser visto na documentação do *framework*, o método `generate()` do objeto `header` é invocado.
- Linha 9: Declaração de um objeto do tipo `Image`, passando o caminho da imagem e o título desta como parâmetro, os outros parâmetros não foram declarados.

- Linha 10: O texto alternativo é setado utilizando o método `setAlt()` do objeto `imagem`.
- Linha 12: A imagem é inserida no corpo da página utilizando o método `setCorpo()` do objeto `interface`. Como o `setCorpo()` aceita somente *strings*, como pode ser visto na documentação do *framework*, o método `generate()` do objeto `imagem` é invocado.
- Linha 14: É invocado o método `generate()` do objeto `interface` que efetivamente mostra a interface.
- Linha 15: *Tag* de final de PHP.

A maneira de utilizar cada classe, juntamente com os métodos e explicações dos mesmos encontram-se na documentação contida na pasta PHPDoc localizada na raiz do *framework.*

4.6. Considerações Finais

Neste capítulo foi apresentado o *framework* Homero, resultado deste projeto de mestrado. Tal framework busca disseminar a utilização dos conceitos de acessibilidades no desenvolvimento de aplicações *web*, através da simplificação do uso das diretrizes da WCAG 2.0, propostas pelo consórcio W3C.

O Homero foi concebido utilizando a linguagem de programação PHP, uma linguagem livre, interpretada, permissiva, rica em funções e uma das mais utilizadas no mundo. Mesmo PHP sendo uma linguagem híbrida, que permite programação orientada a objetos e procedimental, o Homero foi criado de forma que fosse utilizado somente através dos conceitos de orientação a objetos, bastando declarar objetos das classes do *framework* e acessar seus atributos e métodos para utilização das suas funcionalidades. Para concepção do Homero um processo de criação de *framework* foi proposto, onde tal processo pode ser utilizado para a criação de outros *frameworks* de *softwares*, aderentes às diretrizes automatizáveis da WCAG 2.0.

Com o intuito de avaliar a eficiência do Homero, foi realizado um estudo empírico. A proposta do estudo e seus resultados são apresentados em detalhes no Capítulo 5.

5. ESTUDO EMPÍRICO: INSTANCIAÇÃO DE UMA INTERFACE DE APLICAÇÃO WEB ACESSÍVEL UTILIZANDO O HOMERO

Neste capítulo é apresentado um estudo planejado para avaliar a aplicação dos conceitos de acessibilidade no desenvolvimento de interfaces *web* utilizando o Homero em relação a interfaces criadas sem o uso do *framework*.

Primeiramente são apresentadas as definições do estudo, seguido de seu planejamento e execução. Durante o estudo os participantes foram organizados em equipes e estas alocadas em dois grupos. Foram fornecidas as equipes um modelo da interface que deveriam ser desenvolvidas em um tempo determinado. As equipes de um grupo participaram utilizando o Homero e as do outro grupo participaram sem o uso do *framework*. Por fim, os dados referentes à execução do estudo são apresentados e discutidos, evidenciando o comportamento das equipes durante o desenvolvimento de aplicações *web* acessíveis.

5.1. Definições do Estudo Empírico

O estudo empírico foi planejado de acordo com Wholin *et al.* (2000) e teve como objetivo avaliar a eficácia do Homero no desenvolvimento de uma interface *web* acessível em relação ao desenvolvimento sem o uso do f*ramework*. Tal eficácia foi medida em termos do número de elementos implementados em determinado tempo e nível de acessibilidade da WCAG 2.0 atingido em testes automatizados (usando validador de acessibilidade).

O estudo foi conduzido com um grupo de estudantes do mestrado profissionalizante em Ciência da Computação da Faculdade de Computação – FACOM/UFMS, organizados em equipes e alocados em dois grupos: GH e GnH. Estas equipes participaram do estudo desenvolvendo uma interface, cujo modelo foi fornecido, em determinado tempo. As equipes do grupo GH participaram do estudo utilizando o Homero e as equipes do grupo GnH participaram sem o uso do *framework*. Para que nenhum imprevisto ocorresse no momento do estudo, um piloto foi realizado previamente.

Como parte do estudo foi realizada uma pesquisa qualitativa para levantar observações e sugestões para melhorias no Homero por parte dos participantes.

5.2. Planejamento do Estudo Empírico

Seleção do contexto: O estudo foi realizado com alunos do mestrado profissionalizante em Ciência da Computação da Faculdade de Computação – FACOM/UFMS.

Formulação de hipóteses: Foram elaborados dois tipos de hipóteses para o estudo a fim de analisar o efeito do uso do Homero. As hipóteses são relativas ao número de elementos implementados em determinado tempo e nível de acessibilidade da WCAG 2.0 atingido.

No Quadro 5.1 são apresentadas as hipóteses elaboradas de cada tipo e também a métrica usada para avaliar cada hipótese. No primeiro caso a métrica foi obtida através de contagem do número de elementos do *layout* implementados e no segundo as aplicações foram submetidas ao validador indicado pelo W3C, *Total Validator*, e obtido o nível de acessibilidade.

Quadro 5.1 - Hipóteses e Métricas Associadas.

Número de elementos desenvolvidos em um tempo determinado	
H_0	Não há diferença no número de elementos desenvolvidos utilizando, ou não, o Homero.
H_{a0}	Há diferença no número de elementos desenvolvidos utilizando, ou não, o Homero.
Nível de acessibilidade da WCAG 2.0	
H_1	Não há diferença no nível de acessibilidade alcançado utilizando, ou não, o Homero.
H_{a1}	Há diferença no nível de acessibilidade alcançado utilizando, ou não, o Homero.
H: hipótese nula, H_a: hipótese alternativa	

Seleção de variáveis: Considerando as variáveis independentes, seus "fatores" estão na entrada do estudo são todas aquelas que são controladas, sendo para esse estudo o *framework* proposto e o modelo da interface a ser desenvolvida. Por outro lado, os "tratamentos" são os valores utilizados para controlar os "fatores", ou seja, são a causa que afeta o resultado do estudo. Para esse estudo, os tratamentos foram as abordagens utilizadas no desenvolvimento (um grupo utilizou o Homero, o outro não) e a experiência dos participantes em programação orientada a objetos, PHP e HTML.

As variáveis dependentes estão na saída do estudo e são aquelas que estão sob análise, sendo assim afetadas durante o estudo. As suas variações devem ser observadas com base nas mudanças feitas nas variáveis independentes. As variáveis dependentes do estudo

realizado são o número de elementos desenvolvidos em determinado tempo e o nível de acessibilidade da WCAG 2.0 obtido em testes automatizados.

Seleção dos participantes: Os participantes do estudo foram selecionados por conveniência, sendo estes 25 alunos do curso de mestrado profissional em Ciência da Computação da FACOM/UFMS. No estudo piloto participaram 2 alunos do curso de mestrado acadêmico em Ciência da Computação da FACOM/UFMS. O objetivo do estudo piloto foi analisar a possibilidade de ocorrerem erros durante o estudo, e corrigi-los antes do estudo definitivo, logo os dados do estudo piloto não foram considerados nas análises.

Treinamento: Antes da condução do estudo foi realizado um levantamento para analisar o nível de conhecimento, teórico e prático, dos participantes em programação orientada a objetos, desenvolvimento *web* utilizando a linguagem PHP e HTML, além de identificar o nível de experiência dos participantes com conceitos de acessibilidade *web* e com diretrizes da WCAG 2.0. Esse levantamento foi feito por meio de um questionário (Apêndice C) respondido pelos participantes. A compilação das informações do conhecimento teórico dessa coleta de dados pode ser vista no Quadro 5.2.

Quadro 5.2 - Nível de Experiência Teórica dos Participantes.

Nível de conhecimento teórico	Nenhum	Pouco	Razoável	Alto
Orientação a Objetos	0	6	16	3
PHP	3	11	7	4
HTML	1	9	11	4
Acessibilidade *web*	5	14	6	0
Diretrizes WCAG 2.0	22	3	0	0

A coleta de informações referentes a conhecimento prático levantou se os participantes já tinham trabalhado em projetos de *software*, tanto na indústria quanto na academia. A compilação dessas informações pode ser analisada no Quadro 5.3.

Quadro 5.3 - Nível de Experiência Prática dos Participantes.

Nível de conhecimento prático	Nenhum	Estudado em aula, ou a partir de um livro	Praticado em um projeto de classe	Usado em um projeto ou na indústria	Usado em vários projetos na indústria
PHP	3	6	5	4	7
HTML	3	5	8	2	7
Acessibilidade *web*	15	5	4	0	1
Diretrizes WCAG 2.0	22	3	0	0	0

O levantamento demonstrou que a maioria dos participantes não possuíam conhecimento dos princípios de acessibilidade *web* e dos que conheciam, apenas três

afirmaram ter noção superficial das diretrizes da WCAG 2.0. Com essa constatação, prover um meio de disseminar tais conceitos tornou-se ainda mais imprescindível.

Buscando nivelar os participantes do estudo, duas apresentações distintas foram elaboradas, apresentando os conceitos referentes aos temas relevantes às abordagens utilizadas por cada grupo: i) Com grupo GH foi discutida a orientação a objetos, a linguagem PHP e o *framework* Homero, suas classes e forma de instanciação do mesmo. ii) Com grupo GnH foi abordado o desenvolvimento *tableless*[9], a linguagem HTML e a WCAG 2.0, suas diretrizes e critérios de sucessos.

Projeto do estudo empírico realizado: Os participantes foram organizados em equipes de 2 ou 3 pessoas. Para permitir a comparação entre o desenvolvimento utilizando o Homero e o desenvolvimento sem o uso do Homero as equipes foram dividas em dois grupos, onde um grupo, chamado de GH, utilizou o Homero no desenvolvimento e o outro, chamado GnH, não o utilizou. A formação das equipes e dos grupos foi feita de forma com que cada equipe em cada grupo tivesse conhecimento, ao menos, razoável das tecnologias utilizadas na abordagem, garantindo um nivelamento dos grupos. Sendo assim, no grupo GH cada equipe deveria ter conhecimento ao menos razoável em orientação a objetos e PHP e no grupo GnH cada equipe deveria ter conhecimento ao menos razoável em HTML. Nos Quadros 5.4 e 5.5 podem-se analisar as equipes formadas e o nível de conhecimento de cada equipe, considerando o nível do conhecimento da equipe o de maior granularidade dos componentes.

Quadro 5.4 - Nível de Conhecimento das Equipes do Grupo GH.

Equipe	Conhecimento em Orientação a Objetos	Conhecimento teórico em PHP	Conhecimento prático em PHP
GH1	Alto	Alto	Usado em vários projetos na indústria
GH2	Razoável	Alto	Usado em vários projetos na indústria
GH3	Razoável	Alto	Usado em vários projetos na indústria
GH4	Razoável	Razoável	Usado em vários projetos na indústria
GH5	Alto	Alto	Usado em vários projetos na indústria
GH6	Alto	Razoável	Usado em vários projetos na indústria

[9] Forma de desenvolver *sites* que não utiliza tabelas para disposição de conteúdo na página, defendendo que os códigos HTML devem ser usados para o propósito que foram criados. Para a disposição do *layout* o recomendado é usar CSS.

Quadro 5.5 - Nível de Conhecimento das Equipes do Grupo GnH.

Equipe	Conhecimento teórico em HTML	Conhecimento prático em HTML
GnH1	Razoável	Praticado em um projeto de classe
GnH2	Razoável	Usado em vários projetos na indústria
GnH3	Razoável	Praticado em um projeto de classe
GnH4	Razoável	Estudado em aula, ou a partir de um livro
GnH5	Razoável	Usado em um projeto ou na indústria

Para permitir a comparação entre o desenvolvimento utilizando ou não o Homero, foi criado um modelo de interface a ser desenvolvido. Tal modelo é constituído por:

- Um *mockup*[10] de uma aplicação semanticamente acessível, usado como modelo da interface desenvolvida (Apêndice D);
- Uma imagem da aparência esperada da interface, com as divisões de *layout* propostas (Apêndice E);
- Um arquivo CSS, contendo as características de estilo da interface; e
- Uma pasta contendo cinco imagens utilizadas na interface.

O resultado esperado é que as equipes do grupo GH consigam obter uma interface acessível, com um maior número de elementos implementados e com o mais alto nível de acessibilidade (AAA) da WCAG 2.0 obtido no validador automático utilizado.

O plano de execução do estudo é composto por uma única fase, na qual as equipes dos dois grupos tiveram que desenvolver a interface proposta. No início do estudo foram fornecidas para as equipes o modelo de interface a ser desenvolvido, um formulário de consentimento de uso das informações (Apêndice F) e um formulário de execução (Apêndice G para o grupo GH e Apêndice H para o grupo GnH).

O formulário de consentimento tem como objetivo solicitar a permissão dos participantes do estudo quanto ao uso das informações e o formulário de execução acompanhar com exatidão os tempos decorridos no estudo e coletar métricas, opiniões e sugestões referentes às abordagens utilizadas. Foram solicitados às equipes que somente o campo de horário de início do formulário de execução fosse preenchido no começo do estudo, e os demais campos e o formulário de consentimento deveriam ser preenchidos no final, após

[10] Modelo em escala, ou tamanho real, de um *design* ou dispositivo.

o registro do horário de término do desenvolvimento da interface, para evitar que o tempo de preenchimento influenciasse os dados do estudo. O formulário de execução é composto por questões quantitativas e qualitativas, sendo estas complementares ao estudo, não participando assim de seus resultados.

Instrumentação: O estudo foi planejado para que os grupos executassem o estudo em ambiente controlado e em momentos diferentes. O tempo estipulado para o estudo foi de 60 minutos. O responsável pelo estudo acompanhou as equipes de cada grupo no momento da execução do estudo. Antes do início da execução do estudo, os participantes tiveram apresentações conforme a abordagem utilizada. Estas duraram 15 minutos. Logo na sequência o estudo foi aplicado, onde de início as equipes receberam o modelo a ser desenvolvido, o formulário de consentimento e os formulários de execução correspondentes. As equipes do grupo GH receberam ainda o diagrama de classes do Homero, para facilitar as buscas pelas classes que seriam utilizadas, e as equipes do grupo GnH receberam o endereço de uma aplicação *web* que valida a acessibilidade.

Após essas entregas os participantes das equipes tiveram um tempo de 15 minutos para esclarecer dúvidas referentes à interface a ser desenvolvida. Finalizado esse tempo de entendimento os participantes foram instruídos a iniciar o desenvolvimento.

No Quadro 5.6 são apresentados os documentos utilizados durante a condução do estudo, indicando o momento do uso de cada um deles e o grupo que o utilizou.

Quadro 5.6 - Documentos do estudo empírico.

Documento	**Descrição**	**Utilização**	**Grupo**
Questionário de perfil	Identificação, pelos participantes, do nível de conhecimento e experiência prática com desenvolvimento *web*.	Antes do início da formação dos grupos.	Ambos
Formulário de consentimento	Autorização, pelos participantes, do uso dos dados coletados no estudo.	Após a execução.	Ambos
Formulário de execução	Acompanhamento quantitativo e qualitativo do estudo.	Durante a execução.	Ambos
Mockup	Modelo da interface	Durante a execução.	Ambos
Imagem da aparência esperada com as divisões de *layout*	Uma imagem de como a interface deve ficar depois de pronta com as divisões contempladas no arquivo CSS fornecido.	Durante a execução.	Ambos
Diagrama de classes	Diagrama de classes do Homero	Durante a execução.	Grupo GH

Ameaças à validade: o tratamento de ameaças à validade tem o propósito de garantir que os resultados produzidos são válidos. O agrupamento dos tipos de validade foi baseado na classificação descrita por Travassos *et al*. (2002), considerando dessa classificação três tipos de validade: a validade interna que determina que o resultado não é influenciado por outro fator não controlado ou não medido; a validade de construção considera os relacionamentos entre a teoria e a observação; e a validade externa que determina o quanto é possível generalizar os resultados de um estudo empírico para a população real que se deseja investigar.

Validade interna:

- **Interferência no desempenho:** Como o número de elementos implementados em determinado período de tempo foi um indicador selecionado para analisar a eficiência, pode-se argumentar que a medida de eficiência pode ser influenciada pelo tempo gasto no entendimento dos elementos do modelo de interface a ser desenvolvido. Para eliminar essa possibilidade, os participantes tiveram um tempo de 15 minutos para analisar o modelo e esclarecer possíveis dúvidas sobre o mesmo.
- **Interferência na medição:** Como existem vários validadores de acessibilidade, inclusive alguns em versões *Beta*, o uso de validadores distintos pode levar a um equívoco no nível de acessibilidade alcançado. Para evitar essa interferência o validador utilizado em todas as análises será o *Total Validador*(2013).

Validade de construção:

- **Crença nas hipóteses:** para que os resultados não fossem influenciados pelo prévio conhecimento das hipóteses do estudo pelos participantes, as hipóteses não foram reveladas aos mesmos;
- **Expectativas dos participantes:** para evitar que os participantes efetuassem o desenvolvimento com a expectativa de que com uma ou outra abordagem seria mais fácil, as equipes de um grupo não puderam acompanhar a execução do estudo com o outro grupo.
- **Favorecimento da ferramenta:** Para evitar que alguma das equipes utilizasse ferramentas que geram código de forma automática a ferramenta padrão utilizada no estudo foi o Gedit (2013).

- **Favorecimento na apresentação:** para evitar que a apresentação fornecida para uma abordagem tivesse um tempo maior que a da outra abordagem, ambas as apresentações foram cronometradas, no tempo de 15 minutos.
- **Favorecimento na construção das equipes:** para evitar que equipes com níveis de conhecimento muito discrepante fossem criadas, o perfil de cada participante foi analisado previamente e as equipes montadas de maneira a equilibrar os níveis de conhecimento.
- **Desinteresse dos participantes:** para evitar que os participantes perdessem o interesse no estudo, este foi tratado como trabalho de sala da disciplina de Engenharia de *Software* do curso onde os participantes estavam matriculados e juntamente com outras formas de avaliação compõe a nota da disciplina.

Validade externa:

- **Participantes *versus* população real:** o público alvo do *framework* Homero são pessoas com todos os níveis experiência em desenvolvimento *web* utilizando a linguagem PHP e HTML. Dado que o desejo é que a amostra do estudo represente tal público, se dentre os participantes não tiverem ao menos um participante em cada nível de experiência em desenvolvimento *web* e PHP pode comprometer a representatividade da amostra. Dessa forma, foi previamente verificado entre os participantes que existia pelo menos um que pertencesse a cada nível de conhecimento de PHP e HTML, conforme apresentado no Quadro 5.2;

5.3. Execução do Estudo Empírico

Os participantes executaram as atividades do estudo conforme o planejamento descrito na Seção 5.3. No Quadro 5.7 é descrito o procedimento de desenvolvimento de uma interface *web* acessível adotado no estudo por cada abordagem.

Tabela 5.7 – Procedimentos Adotados na Execução do Estudo.

Abordagem	Procedimento Adotado
Utilizando o Homero	1. Participar da apresentação das tecnologias envolvidas (15 minutos) 2. Analisar o modelo a ser desenvolvido (15 minutos) 3. Desenvolver a interface (60 minutos)
Não utilizando o Homero	1. Participar da apresentação das tecnologias envolvidas (15 minutos) 2. Analisar o modelo a ser desenvolvido (15 minutos) 3. Desenvolver a interface (até 60 minutos)

5.4. Apresentação e Análise dos Dados Coletados

Nesta Seção são apresentados os dados coletados durante a condução do estudo e são discutidos os resultados obtidos em relação às hipóteses e ao objetivo do mesmo.

a) Número de Elementos Desenvolvidos em um Tempo Determinado

O tempo de execução do estudo foi de 60 minutos. Nesse tempo as equipes de cada grupo desenvolveram os componentes da interface. Devido ao número limitado de participantes no estudo, a eliminação de *outliers*[11] foi desconsiderada. Ao concluir esse tempo, os arquivos fontes foram imediatamente recolhidos para evitar a possível inclusão de novos componentes. O *mockup* da interface com os componentes numerados para contagem pode ser analisado no Apêndice I. As equipes do grupo GnH não tiveram grandes problemas, visto que o nível de conhecimento das equipes era no mínimo razoável na utilização da linguagem HTML, como pode ser visto no Quadro 5.5. Já o grupo GH precisava além de ter conhecimento razoável em PHP e orientação a objetos, requisitos que foram cumpridos segundo análise do Quadro 5.4, entender o funcionamento do *framework*. Para isso foi investigado se os participantes acharam complicado o uso do Homero para criação de interfaces acessíveis. Como pode ser visto no Gráfico 5.1 quase todas as equipes do grupo GH consideraram simples a abordagem proposta pelo Homero, somente 1 equipe achou complicado, justificando que o tempo foi insuficiente para melhor entendimento da ferramenta e que a abordagem proposta diferia muito da comumente utilizada.

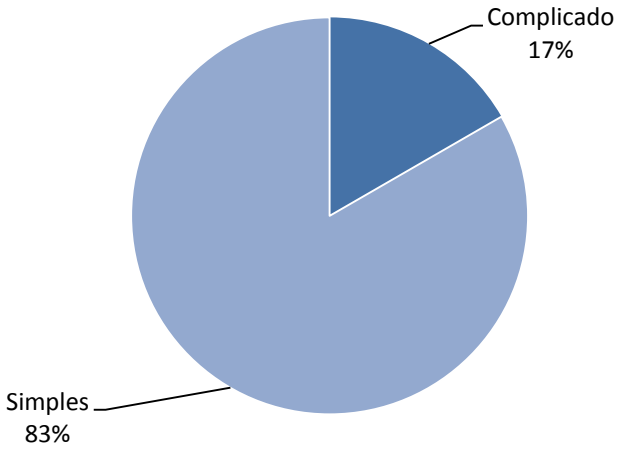


Gráfico 5.1 – Consideração das equipes do grupo GH sobre o uso do Homero.

[11] Outlier, ou valor atípico, é uma observação que apresenta um grande afastamento das demais da série.

Mesmo as equipes do Grupo GH tendo considerado simples a forma de utilização do Homero, o número médio de elementos implementados pelas equipes do grupo GnH foi maior, como pode ser analisado no Gráfico 5.2 e detalhado no Gráfico 5.3. Tal fato nos propõe que a abordagem utilizada no desenvolvimento impacta diretamente na quantidade de elementos implementados.

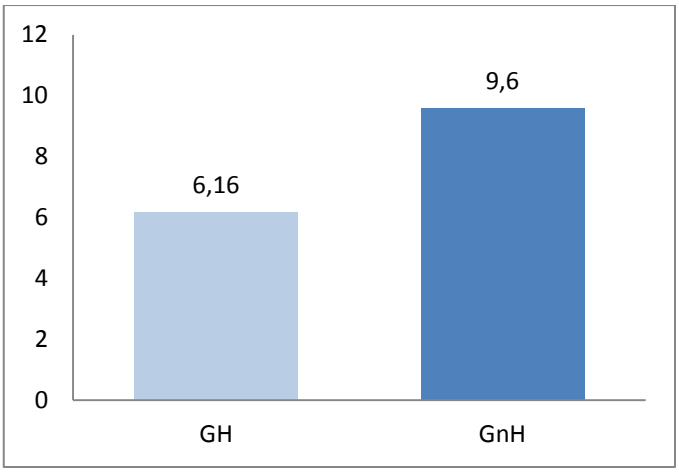


Gráfico 5.2 – Média de elementos implementados por grupo.

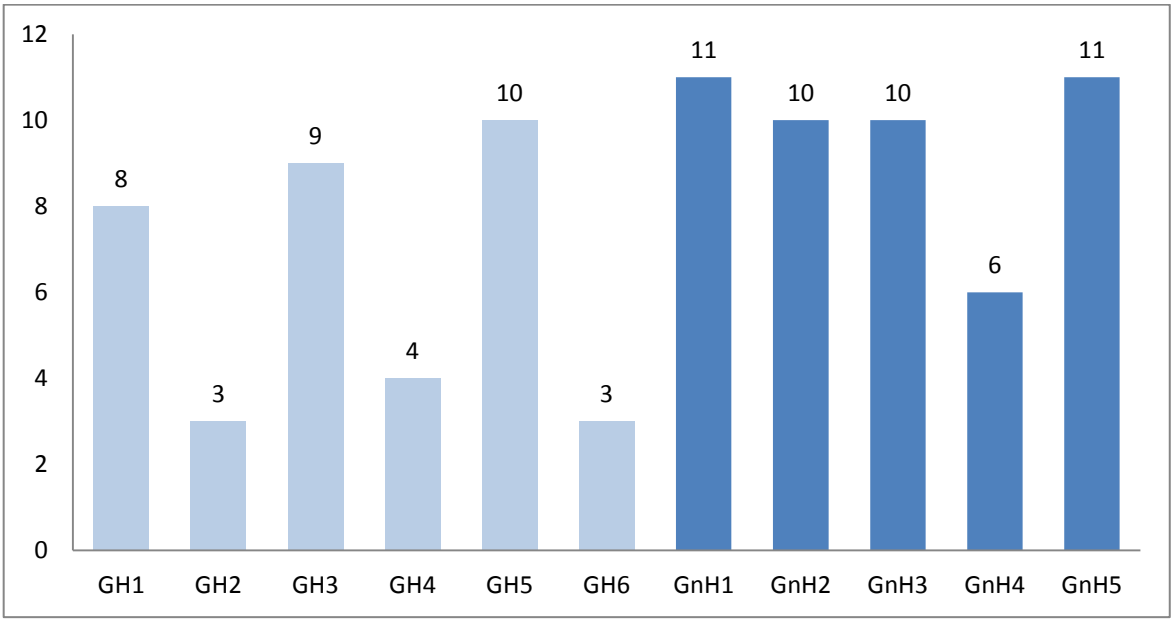


Gráfico 5.3 – Número de elementos implementados por equipe.

Essa discrepância acredita-se que pode ter sido acarretada pelo desconhecimento da ferramenta, visto que no formulário de execução para o grupo GH foi levantada qual a principal dificuldade das equipes em desenvolver utilizando o Homero, com 50% das equipes terem apontado esse motivo. Essa análise pode ser constatada no Gráfico 5.4.

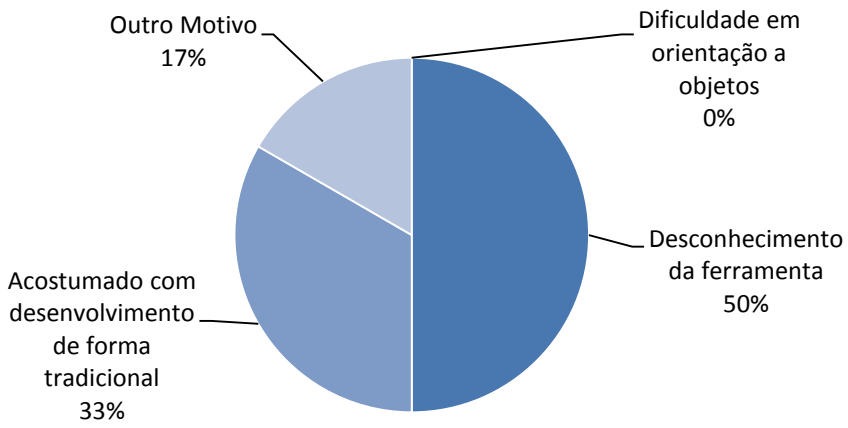


Gráfico 5.4 – Dificuldade de utilização do Homero.

Tal fato pode ser contornado com uma familiaridade dos participantes com a ferramenta, o que depende de uso da ferramenta e da documentação da mesma. Buscando entender se esse contorno é possível foi investigada também a opinião dos participantes sobre a documentação do Homero, tendo sido considerada razoável ou ótima por grande parte da população, como pode ser visto no Gráfico 5.5.

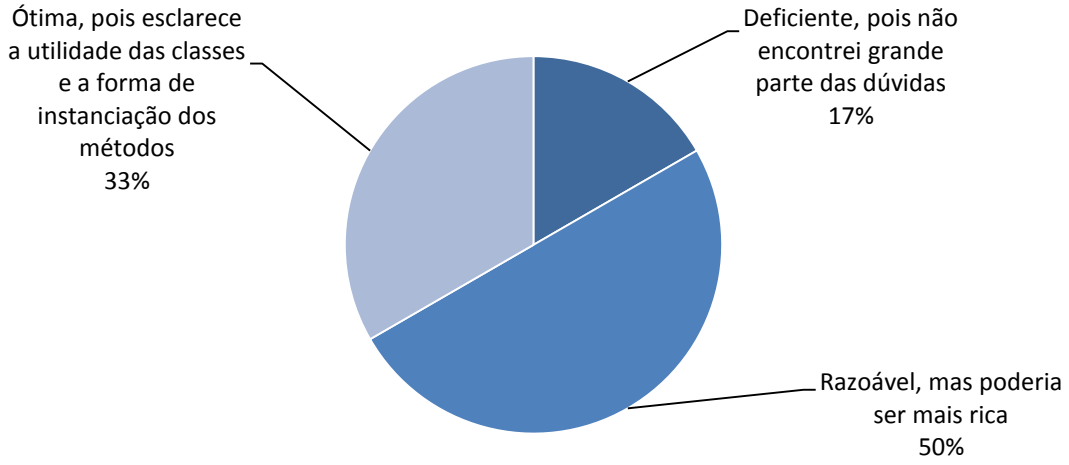


Gráfico 5.5 – Opinião sobre a documentação do Homero.

b) Nível de Acessibilidade da WCAG 2.0 atingido em testes automatizados

Com os arquivos fontes recolhidos, estes foram submetidos à validação da ferramenta *Total Validator* (2013). Como resultado dessa validação tem-se o número de erros de acessibilidade que cada interface possui e o nível de acessibilidade da WCAG 2.0 atingido. No Gráfico 5.6 pode-se analisar o nível da WCAG 2.0 atingido por cada equipe e no 5.7 o número de erros de acessibilidade de cada equipe.

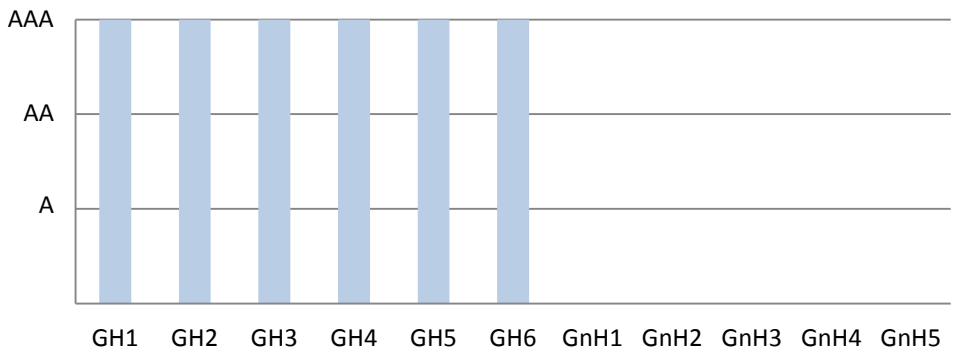


Gráfico 5.6 – Nível de acessibilidade da WCAG 2.0 alcançado por equipe.

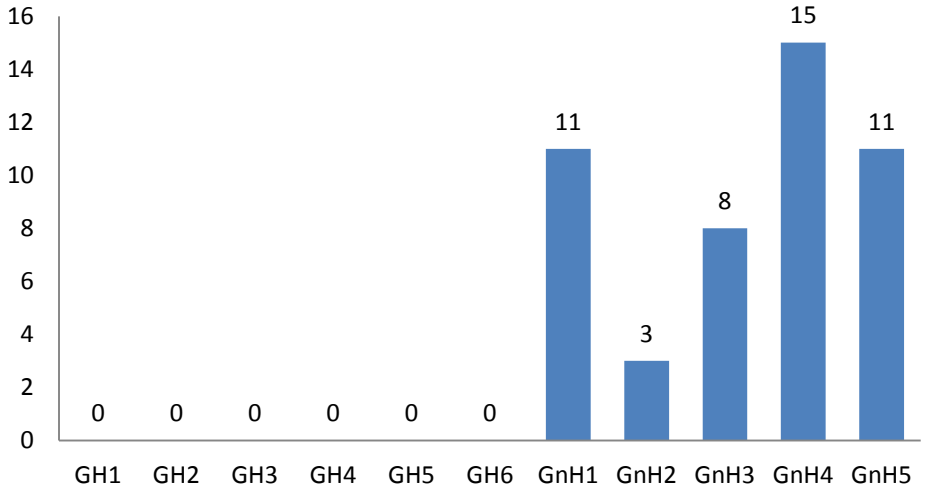


Gráfico 5.7 – Número de erros de acessibilidade por equipe.

Com a análise desses gráficos, podemos concluir que o Homero atingiu o objetivo proposto que era de garantir a aplicação das diretrizes automatizáveis nos componentes desenvolvidos, pois nenhum erro de acessibilidade foi encontrado nas interfaces do grupo GH, atingindo estas o nível AAA da WCAG 2.0. Diferentemente as interfaces desenvolvidas pelas equipes do grupo GnH, mesmo com maior quantidade de elementos implementados segundo os Gráficos 5.2 e 5.3, não obtiveram sucesso em alcançar nenhum nível da WCAG 2.0. Isso se deu pela grande dificuldade que os participantes tiveram em utilizar a documentação fornecida pela WCAG 2.0, pois mesmo que grande parte dos participantes das equipes do grupo GnH terem considerado a documentação fornecida razoável ou ótima, segundo Gráfico 5.8, ao serem questionados se acharam complicado o uso da Documentação da WCAG 2.0 para criação de interfaces *web* acessíveis, mais da metade considerou que sim, conforme pode ser analisado no Gráfico 5.9.

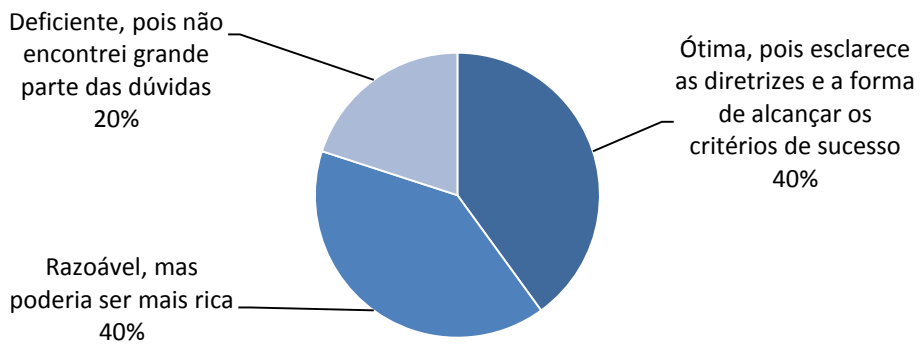


Gráfico 5.8 – Opinião sobre a documentação da WCAG 2.0.

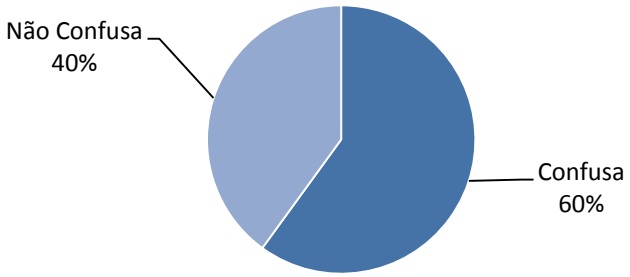


Gráfico 5.9 – Opinião sobre a utilização da documentação da WCAG 2.0.

Um dado qualitativo de deve ser considerado nessa análise é que todos que consideraram a documentação confusa enfatizaram os mesmos pontos, que foi a falta de objetividade e clareza da documentação.

Mesmo com 40% das equipes do grupo GnH afirmando que a documentação da WCAG 2.0 não era confusa, estes não a utilizaram para fazer análise de acessibilidade, preferindo desenvolver sem se fazer análise prévia de acessibilidade e validar em ferramentas auxiliares, como pode ser analisado no Gráfico 5.10.

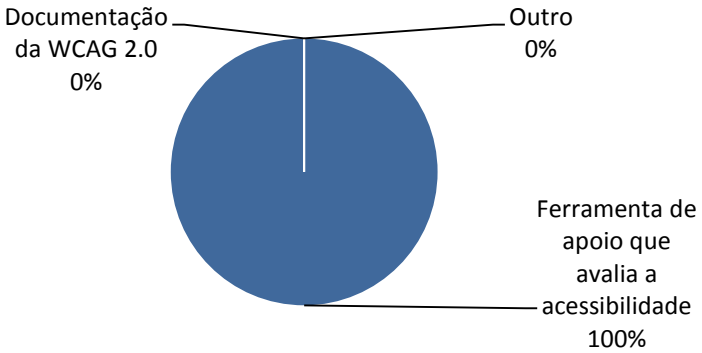


Gráfico 5.10 – Metodologia utilizada para buscar acessibilidade.

Tal abordagem utilizada pelas equipes entrou no “mito da qualidade” apresentado por Pressman (2006), que propõe que “enquanto não tiver o programa funcionando, eu não terei realmente nenhuma maneira de avaliar sua qualidade”. Isso porque todas as equipes só se preocuparam com a acessibilidade na fase final do desenvolvimento, mesmo os participantes tendo sido informados que acessibilidade era um requisito não funcional e deveriam priorizar durante todo o desenvolvimento, como pode ser visto no Gráfico 5.11.

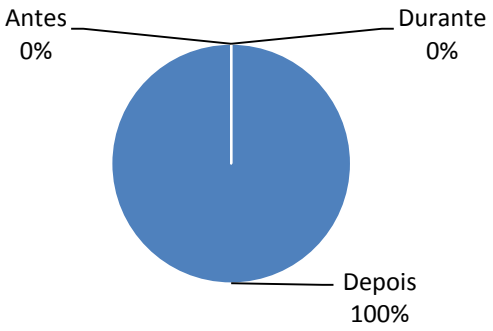


Gráfico 5.11 – Momento de análise da acessibilidade.

Em contrapartida, o Homero força a análise de acessibilidade durante o processo de desenvolvimento, garantindo que os elementos desenvolvidos possuam os elementos de sintaxe obrigatórios para prover acessibilidade, atendendo assim o requisito não funcional da acessibilidade. O Homero além de garantir a sintaxe correta dos elementos para que estes sejam acessíveis propõe uma nova forma de desenvolver interfaces *web*, sendo necessário que os programadores conheçam somente PHP, como tecnologia, e orientação a objetos, para instanciar os métodos. Perguntado aos participantes se eles acharam interessante a forma proposta pelo *framework* de desenvolver interfaces, a grande maioria das equipes achou interessante, como pode ser visto no Gráfico 5.12. Somente uma equipe não achou interessante a forma proposta, alegando que a mesma ficou inflexível, porém essa “inflexibilidade” no processo de desenvolvimento é o que garante a acessibilidade.

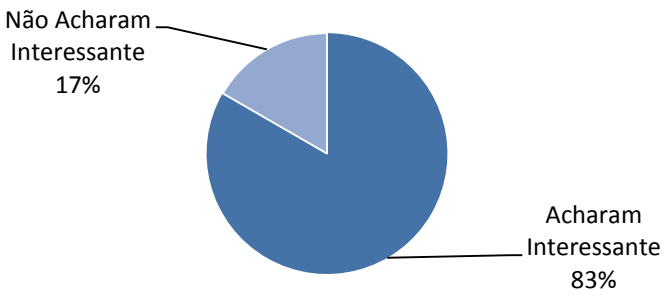


Gráfico 5.12 – Nível de interesse dos participantes na forma de utilização do Homero.

5.5. Considerações Finais

O estudo empírico conduzido foi um pouco limitado quanto a amostra e tempo, pois participaram somente 11 equipes durante um tempo de 60 minutos. Mesmo com essa limitação, o estudo realizado apresentou pontos importantes sobre a utilização do Homero e os benefícios trazidos por este.

Em primeiro lugar, observou-se que a utilização do Homero atingiu o objetivo proposto, garantindo os elementos sintáticos obrigatórios para propiciar acessibilidade aos componentes desenvolvidos, mesmo nas equipes que não conseguiram desenvolver grande quantidade de elementos. Por outro lado as equipes que utilizaram a abordagem sem o uso do Homero conseguiram um número maior de elementos, porém não obtiverem sucesso em desenvolver a aplicação de forma acessível. O número reduzido de componentes desenvolvidos pelas equipes do grupo GH, acredita-se ter sido causado pelo pouco conhecimento da ferramenta, visto que a forma de desenvolvimento muda bastante da forma tradicional. Pode-se tentar contornar essa questão aplicando um treinamento mais efetivo na ferramenta.

O estudo demonstrou ainda a importância de difundir o conhecimento dos desenvolvedores quanto aos conceitos de acessibilidade, quanto de ferramentas que propiciam esta. Como visto, nas análises dos gráficos, tais conceitos não são amplamente difundidos. Adicionalmente, o estudo conduzido colaborou indiretamente para avaliar o processo de desenvolvimento de *software* e verificar que conceitos de qualidade ainda são deixados para o final do desenvolvimento, o que acaba, muitas vezes, não tendo tempo de tratá-los, deixando-os de lado, como aconteceu com a acessibilidade no grupo GnH.

6. CONCLUSÃO

Neste capítulo são apresentadas as conclusões desse trabalho. Primeiramente são discutidas as contribuições observadas com a realização deste trabalho, seguido das limitações do mesmo. Por fim, são elencadas as possibilidade de trabalhos futuros.

6.1. Contribuições

Em primeiro lugar, foi proposto o *framework* Homero para facilitar o desenvolvimento de interfaces de aplicações *web* acessíveis, diferenciando-se pela simplicidade de uso quando comparado à forma tradicional de utilização das diretrizes da WCAG 2.0. Tal fato pode ser constatado com o resultado obtido no estudo empírico realizado. O Homero pode ser utilizado para criação de interfaces de aplicações *web* independente do domínio, podendo ser utilizado para criação desde *sites* simples, até grandes aplicações.

Com relação à abordagem proposta pelo Homero, este veio a contribuir de duas formas: 1) garantindo que a acessibilidade fosse preocupação constante durante toda a fase de desenvolvimento, e não somente no final como pode ser visto no estudo empírico feito com as equipes do Grupo GnH; e 2) contribuindo com a familiarização dos desenvolvedores com os conceitos de acessibilidade e das diretrizes internacionais propostas para alcançar a mesma. Por meio da revisão da literatura, foi possível identificar a grande quantidade de iniciativas nesse aspecto, porém nenhuma trouxe uma abordagem como a proposta neste trabalho.

Como o Homero pertence à fase de desenvolvimento de *software*, o trabalho contribuiu disseminando os modelos existentes e analisando, de forma superficial, os pontos fortes e fracos de cada modelo apresentando. O trabalho também contribuiu com a disseminação das boas práticas de programação, visto a organização estrutural do código do *framework*, além de apresentar um processo de construção que pode se utilizado para a construção do *framework* em outras linguagens de programação.

6.2. Limitações

As seguintes limitações foram identificadas em relação ao *framework* Homero e consequentemente a este trabalho de mestrado:

- O Homero suporta somente a linguagem de programação PHP.

- O Homero cobre somente as diretrizes da WCAG 2.0 que podem ser automatizadas.
- O Homero depende fortemente do nível de conhecimento dos usuários quanto à linguagem PHP e orientação a objetos;
- O desenvolvimento de interfaces utilizando o Homero deve ser feito de forma manual, pois ainda não existem ferramentas CASE que suportem seus componentes;
- A forma de desenvolvimento é diferente dos padrões de programação PHP/HTML difundidos.
- O Homero presta suporte somente a fase de implementação do *software*, tendo que receber como entrada um projeto acessível.

6.3. Trabalhos Futuros

A área de pesquisa desenvolvimento acessível é relativamente recente e pouco explorada, por isso, com o embasamento teórico adquirido e documentado por meio deste trabalho de mestrado, vários ramos de pesquisa podem ser explorados de maneira a compor um estudo abrangente da área. A seguir estão elencados sugestões de trabalhos futuros decorrentes do desenvolvimento deste trabalho:

- Estruturar um processo completo de desenvolvimento de *software* acessível, desde a análise de requisitos até a manutenção do *software*.
- Estender o *framework* para este possa atender o e-MAG (Brasil, 2007).
- Desenvolver tal *framework* em outras linguagens de programação.
- Criar uma ferramenta CASE que suporte os componentes do Homero, facilitando seu uso.
- Viabilizar possibilidades de se utilizar o Homero em cenários reais, na indústria de *software*, para aumentar a maturidade.
- Investigar a possibilidade de se utilizar uma linguagem intermediária, como o XML, para instanciação de componentes, diminuindo a necessidade de o usuário conhecer a linguagem PHP.
- Investigar a possibilidade de integração do Homero com outros *framework*s de *software*, como o Zend (Zend Framework, 2013) ou o CakePHP (CakePHP, 2013).
- Efetuar um estudo com testes e análises manuais, utilizando especialistas em acessibilidade *web*, ou ainda, com usuários com deficiência de vários tipos e em vários níveis.

REFERÊNCIA

ABNT – Associação Brasileira de Normas Técnicas. **NBR 9050, 2o edição, de 31 de maio de 2004. Acessibilidade a edificações, mobiliário, espaços e equipamento urbanos.** Disponível em http://www.pessoacomdeficiencia.gov.br/app/sites/default/files/arquivos/[field_generico_imagens-filefield-description]_24.pdf. Acesso em 27 de junho de 2013.

Abril, Editora. **E-commerce brasileiro deve crescer 24% neste ano diz e-bit.** Disponível em: http://info.abril.com.br/noticias/mercado/e-commerce-brasileiro-deve-crescer-24-neste-ano-diz-e-bit-20032013-35.shl. Acesso em: 16 de maio de 2013.

Appleton, B. **Patterns and software: Essential concepts and terminology.** 1997. http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html. Acesso em 28 de junho de 2013.

Boehm, B.W. **A spiral model of software development and enhancement.** ACM software Engineering Notes, v.11, 1988.

Booch G. Rumbaugh J; Jacobson I. **The Unified Modeling Language User Guide**, Second Edition. Addison-Wesley, 2005

Bosch, J., Molin, P., Mattsson, M., Bengtsson, P., Fayad, M. Framework problem and experiences in M. Fayad, R. Johnson, D. Schmidt. **Building Application Frameworks: Object-Oriented Foundations of Framework Design**, John Willy and Sons, p.55-82,1999.

Braga, R**. Um Processo para Construção e Instanciação de Frameworks baseados em uma Linguagem de Padrões para um Domínio Específico.** Tese de Doutoramento, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2003.

Brasil. **Decreto-Lei N. 3298, de 20 de dezembro de 1999. Regulamenta a lei no 7.853, de 24 de outubro de 1989, dispõe sobre a Política Nacional pata a Integração da Pessoa Portadora de Deficiência, consolida as normas de proteção, e dá outras providências.** Disponível em http://www.planalto.gov.br/ccivil_03/decreto/d3298.htm. Acesso em 27 de junho de 2013.

Brasil. **Decreto-Lei N. 5296, de 2 dezembro de 2004. Regulamenta as Leis nos 10.048, de 8 de novembro de 2000, que dá prioridade de atendimento às pessoas que especifica, e 10.098, de 19 de dezembro de 2000, que estabelece normas gerais e critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência ou com mobilidade reduzida, e dá outras providências.** Disponível em http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/decreto/d5296.htm. Acesso em 27 de junho de 2013.

Brasil. **Lei Complementar N. 131, de 27 de maio de 2009. Alterou a redação da Lei de Responsabilidade Fiscal no que se refere à transparência da gestão fiscal, inovando ao determinar a disponibilização, em tempo real, de informações pormenorizadas sobre a execução orçamentária e financeira da União, dos Estados, do Distrito Federal e dos Municípios.** Disponível em http://www.portaldatransparencia.gov.br/faleConosco/perguntas-tema-transparencia-lei-complementar.asp#1. Acesso em 17 de maio de 2013.

Brasil. **Ministério do Planejamento, Orçamento e Gestão. Secretaria de Logística e Tecnologia da Informação e-MAG Modelo de Acessibilidade em Governo Eletrônico.** Brasília. 2011. Disponível em http://www.governoeletronico.gov.br/biblioteca/arquivos/e-mag-3.0/download. Acesso em 27 de junho de 2013.

Brasil. **Portaria N. 3, de 7 de maio de 2007. Institucionaliza o Modelo de Acessibilidade em Governo Eletrônico – e-MAG no âmbito do Sistema de Administração dos Recursos da Informação e Informática – SISP.** Disponível em http://www.governoeletronico.gov.br/anexos/portaria-no-03-e-mag/view. Acesso em 27 de junho de 2013.

CakePHP. **CakePHP Framework**. Disponível em: http://cakephp.org/. Acesso em 29 de junho de 2013.

Camargo, Gy. **IRPF - mudanças nas regras para 2011 - imposto de renda.** http://www.impostoderenda.org/2011/02/16/irpf-mudancas-nas-regras-para-2011-imposto-de-renda/. Acesso em: 16 de maio de 2013.

CEUD – Centre for Excellence in Universal Design. **Irish National IT Accessibility Guidelines**. Disponível em http://universaldesign.ie/useandapply/ict/irishnationalit accessibilityguidelines/. Acesso em 18 de junho de 2013.

CMMI, CMMI for Development, Version 1.2, CMU/SEI-2006-TR-008. **Development Methods and Tools, In: Information and Software Technology**, vol. 38, fourth edition, 2006.

Dinesh, T.B., Uskudarli, S., Sastry, S. **Alipi: A framework for renarrating web pages**. W4A: Proceedings of the 2012 international cross-disciplinary conference on Web accessibility, 2012.

Dijkstra, E. W. **The humble programmer**, Communications of the ACM, v.15 n.10, p.859-866, Oct. 1972.

Fayad, M. E., Johnson, R. E. **Domain-specific application frameworks: Frameworks experience by industry.** Ed. John Wiley & Sons, Inc. New York, NY, USA. 2000.

Fayad, M.E., Schmidt, D.C. **Object-Oriented Application Frameworks**. Communications of the ACM, V. 40, nº 10, p. 32-38, 1997.

Fowler, M. **The New Methodology**. 2005. Disponível em http://martinfowler.com/articles/newMethodology.html. Acesso em 03 de junho de 2013.

Freedom Scientific. **JAWS Headquarters**, 2013 Disponível em: http://www.freedomscientific.com/jaws-hq.asp. Acesso em: 16 de maio de 2013.

G1. **Saiba como registrar um boletim de ocorrência pela internet em MS**. Disponível em: http://g1.globo.com/mato-grosso-do-sul/noticia/2013/04/saiba-como-registrar-um-boletim-de-ocorrencia-pela-internet-em-ms.html/. Acesso em: 16 de maio de 2013.

Gedit. **Gedit text editor**. Disponível em http://projects.gnome.org/gedit/. Acesso em 18 de junho de 2013.

IBGE – Instituto Brasileiro de Geografia e Estatística. **Censo Demográfico**. 2010. Disponível em: http://www.ibge.gov.br/censo2010. Acesso: 27 de junho de 2013.

iMasters. **Boas práticas para programação em php**. 2013. Disponível em: http://imasters.com.br/artigo/5082/php/boas-praticas-para-programacao-em-php/. Acesso em: 04 de junho de 2013.

IRPF – Imposto de Renda de Pessoa Física. **Imposto de renda 2011 só poderá ser declarado pela internet**. Disponível em: http://irpf.estacaobr.net/imposto-de-renda-2011-so-podera-ser-declarado-pela-internet/. Acesso em: 16 de maio de 2013.

ISO 9241-171. **Ergonomics of Human-System Interaction - Part 171: Guidance on Software Accessibility**, 2008.

Júnior, L. M. C. M. **História do Movimento Político das Pessoas com Deficiência no Brasil. - Brasília: Secretaria de Direitos Humanos. Secretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência**, 2010.

Maia, L. **Um processo para o desenvolvimento de aplicações web acessíveis.** Dissertação de Mestrado. Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, 2010, 103 p.

Mangiatordi, A., Sareen, H. S. **Farfalla project: browser-based accessibility solutions**. W4A: Proceedings of the 2011 international cross-disciplinary conference on Web accessibility, 2011.

Martin, A., Cechich, A., Rossi, G. **Contextual Web Accessibility – Accessibility at Early Stages: Insights from the Designer Perspective**. W4A: Proceedings of the 2011 international cross-disciplinary conference on Web accessibility, 2011.

NAIS – Núcleo de acessibilidade e inclusão. **Menos de 5% dos sites de órgãos públicos são acessíveis a pessoas com deficiência**. Disponível em: http://www.prograd.uff.br/sensibiliza/menos-de-5-dos-sites-de-órgãos-públicos-são-acessíveis-pessoas-com-deficiência. Acesso em: 16 de maio de 2013.

Neo Luzz Brasil. **Mouse teclado especial para deficientes físicos**. Disponível em: http://www.neoluzzbrasil.com.br/mouse_teclado_10.html. Acesso em: 16 de maio de 2013.

Oikonomou, T., Kaklanis, N., Votis, K., Kastori, G., Partarakis, N., Tzovaras, D. **WaaT: Personalised Web Accessibility Evaluation Tool**. W4A: Proceedings of the 2011 international cross-disciplinary conference on Web accessibility, 2011.

Oliveira, R.C. **Homero: Um Framework de Apoio ao Desenvolvimento de Aplicações Web Acessíveis**. Monografia de conclusão de curso. Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, 2010.

Pfleeger, S.L. **Engenharia de Software: Teoria e Prática**. 2º Edição. São Paulo, Prentice Hall, 2001.

PHP. **PHP: Hypertext Preprocessor.** Disponível em http://www.php.net. Acesso em 24 de junho de 2013.

PHPDoc. **PHPDocumentor**. Disponível em: http://www.phpdoc.org. Acesso em 06 de junho de 2013.

Pingdom. **Internet 2012 in number**. Disponível em: http://royal.pingdom.com/2013/01/16/ internet-2012-in-numbers/. Acesso em: 16 de maio de 2013.

PMBOK. **Project Management Body of Knowledg**. Project Management Institute, 4ª edição, 2008.

Pree, W., Pomberger, G., Schappert, A., Sommerlad, P. **Active guidance of framework development**. Software- Concepts and Tool, v.16, n.3, p.136, 1995.

Pressman, Roger. S. **Engenharia de Software**, 6ª edição McGraw-Hill, 2006.

Royve, W. W. **Managing the Development of Large Software Systems: concepts and techniques**, Los Angeles, Proceedings, IEE Wescon, 1970.

Sandim, H,C., Turine, M.A.S., Vieira C.C.A., Arakaki, A.A. **Pantaneiro: Um Gerados de Aplicações Web no Contexto e-Gov**. Congresso da SBC – Sociedade Brasileira de Software. 2006.

Schmid, H. A. **Systematic Frameworks design by generalization**, Communications of the ACM, V. 40, nº 10, p. 48-51, 1997.

Schwaber, K., Beedle, M. **Agile Software Development with SCRUM**. Prentice Hall, 2002.

Section 508. **Section508.gov Opening Doors to IT**. Disponível em: http://www.section508. gov/. Acesso em 29 de junho de 2013.

Sloan, D., Kelly, B., Heath, A., Petrie, H., Fraser, H. and Phipps, L. **Contextual Web Accessibility - Maximizing the Benefit of Accessibility Guidelines**. W4A: Proceedings of the 2006 international cross-disciplinary conference on Web accessibility, 2006.

Sommerville, I. **Engenharia de Software**, 8ª edição, Pearson Addison-Wesley, 2007.

Tarr, A. **Wai-aria roles in accessible admin template**. 2011. http://community.joomla.org/ blogs/community/963-wai-aria-roles-inaccessible-admin-template.html.

Thiessen, P. **WAI-ARIA Live Regions and HTML5**. W4A: Proceedings of the 2011 international cross-disciplinary conference on Web accessibility, 2011.

Tiobe Software. **TIOBE Programming Community Index for July 2013**. Disponível em http://www.tiobe.com/index.php/content/ paperinfo/tpci/index.html. Acesso em 06 de junho de 2013.

Total Validator. **Total Validator – Validation Tool**. Disponível em http://www.totalvalidator.com. Acesso em 01 de junho de 2013.

Travassos, H. G.; Gurov, D.; Amaral E. A. G. **Introdução à Engenharia de Software Experimental**. Relatório Técnico, COPPE/UFRJ, 2002.

Villain, J., Nourry, O. **MIPAW – Modele of a Progressive Implementation of Web Accessibility**. W4A: Proceedings of the 2012 international cross-disciplinary conference on Web accessibility, 2012.

W3C. World Wide Web Consortium. **HTML 5 Techniques for WCAG 2.0 Task Force**. 2011. Disponível em http://www.w3.org/WAI/GL/wcag-pf-html5-task-force. Acesso em 09 de junho de 2013.

W3C. World Wide Web Consortium Escritório Brasil, 2009. **Apresentações e palestras da equipe W3C**. Disponível em http://www.w3c.br/palestras/2009/W3C_Acessibilidade_Secop. pdf. Acesso em 17 de maio de 2013.

W3C. World Wide Web Consortium Escritório Brasil, 2012. **Grupo de Acessibilidade**. Disponível em http://www.w3c.br/GT/GrupoAcessibilidade. Acesso em 17 de maio de 2013.

WAI-ARIA. **Accessible Rich Internet Applications**. 2011. Disponível em http://www.w3.org/ WAI/intro/aria/. Acesso em 20 de junho de 2013.

Wald, M., Draffan, E. A., Skuse, S., Newman, R., Phethean, C. **Southampton Accessibility Tools**. W4A: Proceedings of the 2011 international cross-disciplinary conference on Web accessibility, 2011.

WCAG. **Web Content Acessibility Guidelines (WCAG) 1.0**, 1999. Disponível em http://www.w3c.org/TR/WCAG10/. Acesso em 20 de junho de 2013.

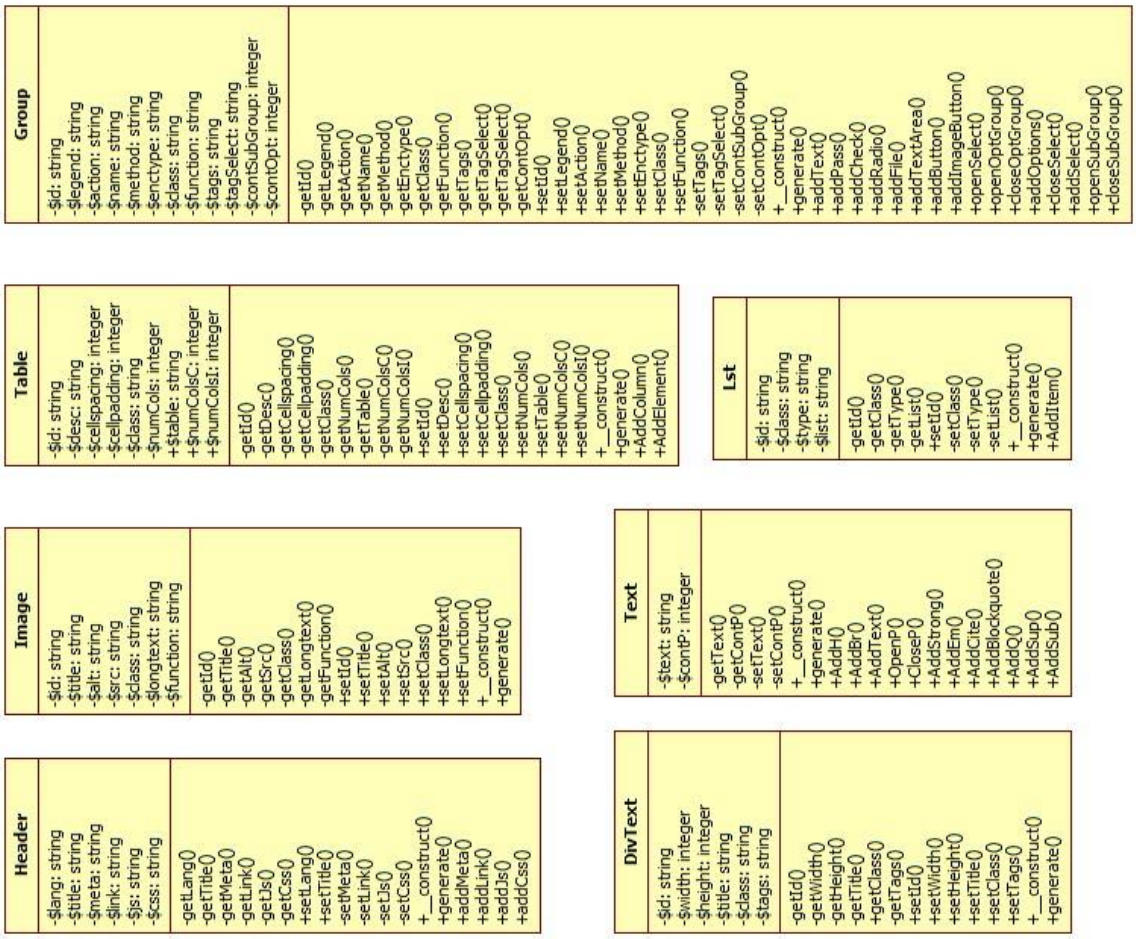
WCAG. **Web Content Acessibility Guidelines (WCAG) 2.0**, 2008. Disponível em http://www.w3.org/TR/WCAG20/. Acesso em 20 de junho de 2013.

Wholin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B., and Wesslen, A. **Experimentation in Software Engineering: An Introduction**. Kluwer Academic Publishers. 2000.

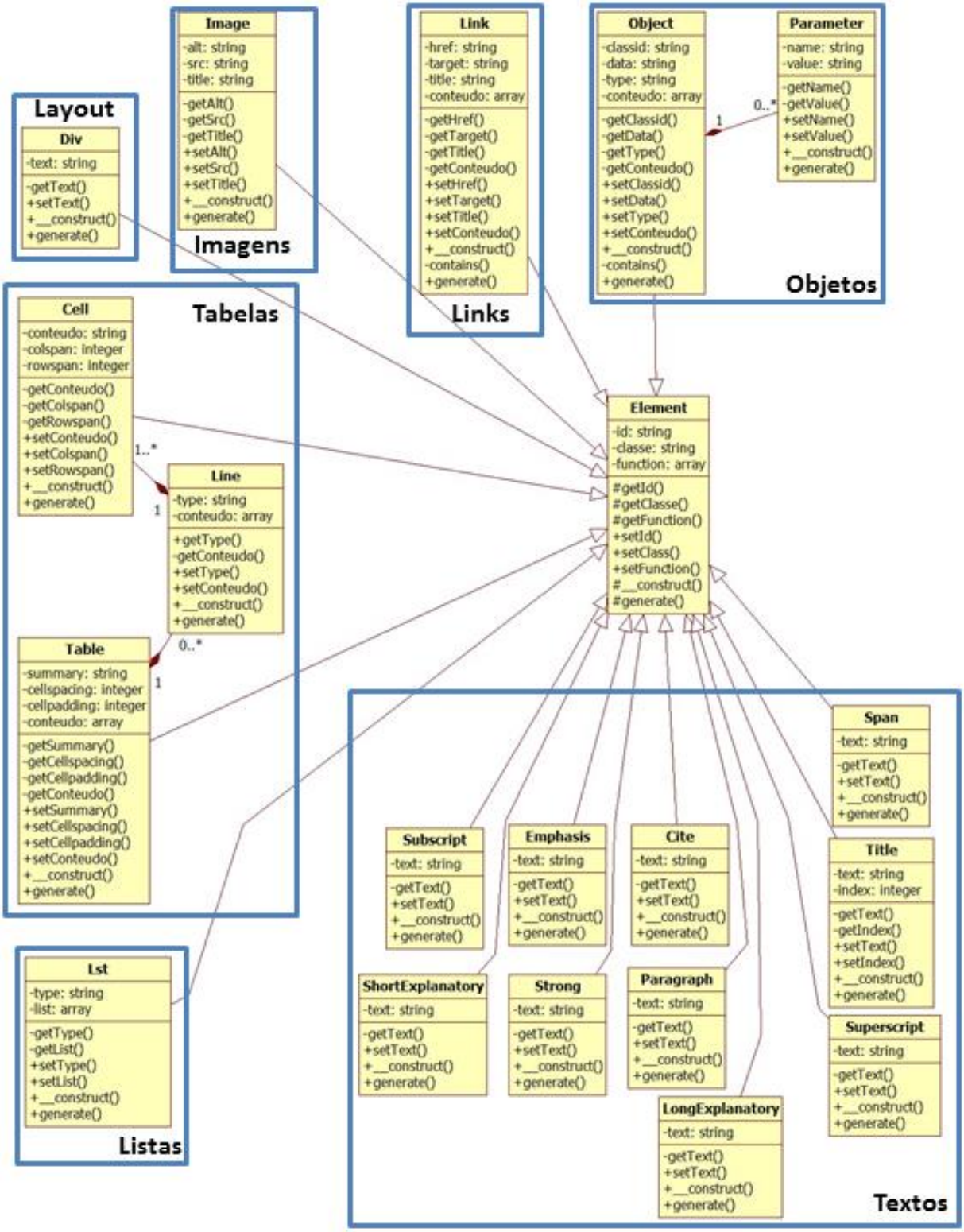
Zend Framework. **Zend Framework for PHP**. Disponível em: http://framework.zend.com/. Acesso em 29 de junho de 2013.

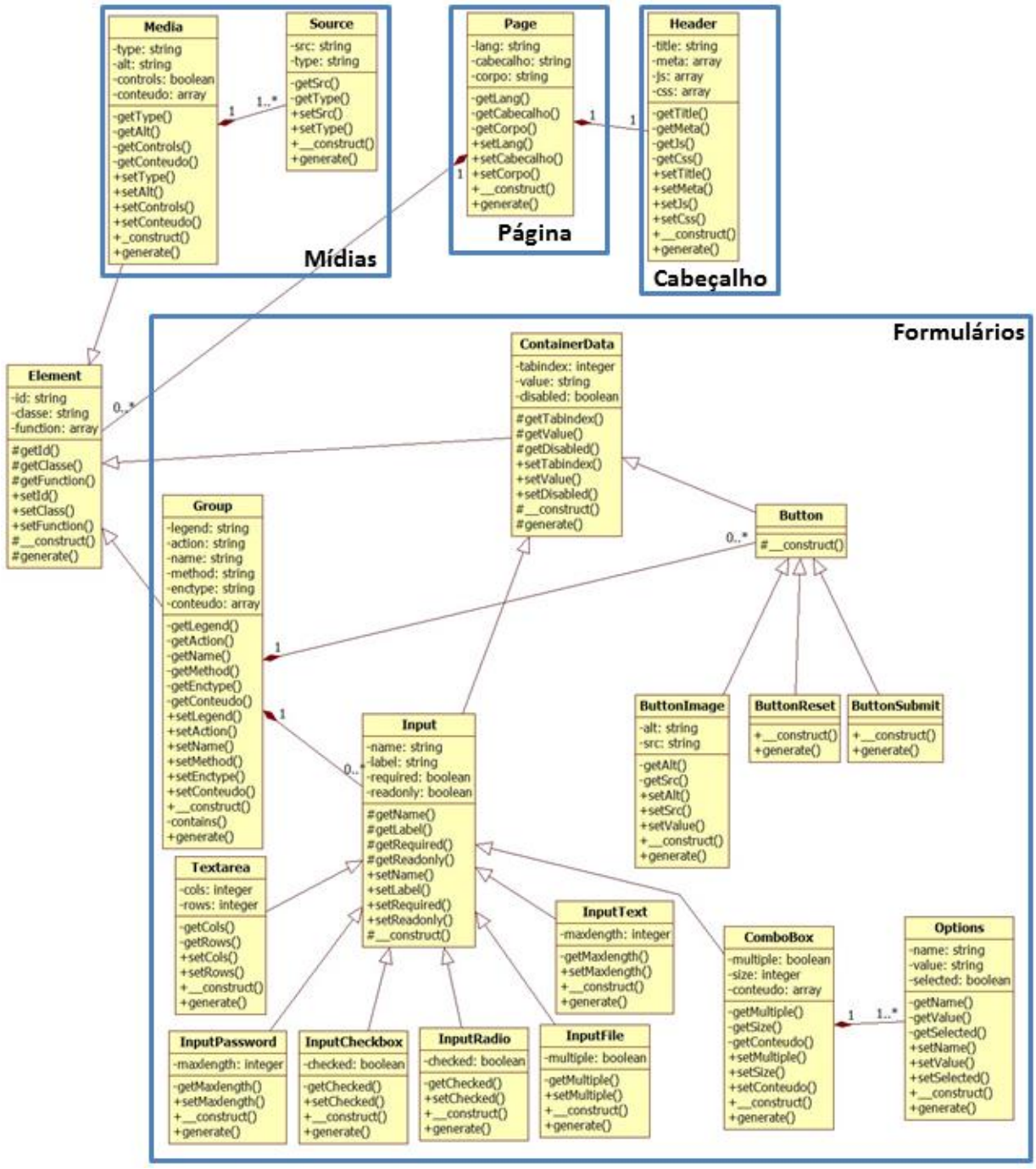
Zend Studio. **Zend Studio – The Leading PHP IDE**. Disponível em: http://www.zend.com/en/products/studio. Acesso em 29 de junho de 2013.

APÊNDICE A – Diagrama de classes completo do Homero – versão Beta



APÊNDICE B – Diagrama de classes completo do Homero




Media
-type: string
-alt: string
-controls: boolean
-conteudo: array
-getType()
-getAlt()
-getControls()
-getConteudo()
+setType()
+setAlt()
+setControls()
+setConteudo()
+_construct()
+generate()
Source
-src: string
-type: string
-getSrc()
-getType()
+setSrc()
+setType()
+__construct()
+generate()
Mídias
Page
-lang: string
-cabecalho: string
-corpo: string
-getLang()
-getCabecalho()
-getCorpo()
+setLang()
+setCabecalho()
+setCorpo()
+__construct()
+generate()
Página
Header
-title: string
-meta: array
-js: array
-css: array
-getTitle()
-getMeta()
-getJs()
-getCss()
+setTitle()
+setMeta()
+setJs()
+setCss()
+__construct()
+generate()
Cabeçalho
Formulários
Element
-id: string
-classe: string
-function: array
#getId()
#getClasse()
#getFunction()
+setId()
+setClass()
+setFunction()
#__construct()
#generate()
ContainerData
-tabindex: integer
-value: string
-disabled: boolean
#getTabindex()
#getValue()
#getDisabled()
+setTabindex()
+setValue()
+setDisabled()
#__construct()
#generate()
Group
-legend: string
-action: string
-name: string
-method: string
-enctype: string
-conteudo: array
-getLegend()
-getAction()
-getName()
-getMethod()
-getEnctype()
-getConteudo()
+setLegend()
+setAction()
+setName()
+setMethod()
+setEnctype()
+setConteudo()
+__construct()
-contains()
+generate()
Button
#__construct()
Input
-name: string
-label: string
-required: boolean
-readonly: boolean
#getName()
#getLabel()
#getRequired()
#getReadonly()
+setName()
+setLabel()
+setRequired()
+setReadonly()
#__construct()
ButtonImage
-alt: string
-src: string
-getAlt()
-getSrc()
+setAlt()
+setSrc()
+setValue()
+__construct()
+generate()
ButtonReset
+__construct()
+generate()
ButtonSubmit
+__construct()
+generate()
Textarea
-cols: integer
-rows: integer
-getCols()
-getRows()
+setCols()
+setRows()
+__construct()
+generate()
InputText
-maxlength: integer
-getMaxlength()
+setMaxlength()
+__construct()
+generate()
ComboBox
-multiple: boolean
-size: integer
-conteudo: array
-getMultiple()
-getSize()
-getConteudo()
+setMultiple()
+setSize()
+setConteudo()
+__construct()
+generate()
Options
-name: string
-value: string
-selected: boolean
-getName()
-getValue()
-getSelected()
+setName()
+setValue()
+setSelected()
+__construct()
+generate()
InputPassword
-maxlength: integer
-getMaxlength()
+setMaxlength()
+__construct()
+generate()
InputCheckbox
-checked: boolean
-getChecked()
+setChecked()
+__construct()
+generate()
InputRadio
-checked: boolean
-getChecked()
+setChecked()
+__construct()
+generate()
InputFile
-multiple: boolean
-getMultiple()
+setMultiple()
+__construct()
+generate()
0..*
1
1..*

APÊNDICE C – Formulário de levantamento do perfil dos participantes do estudo empírico

Questionário de Perfil

Este formulário deve ser respondido por todos os participantes do estudo empírico sobre o framework Homero.

Nome do participante: _____

Conhecimento específico

1. Qual é o seu nível de conhecimento em Orientação a Objetos?
a) Nenhum
b) Pouco
c) Razoável
d) Alto

2. Qual é o seu nível de conhecimento na linguagem PHP?
a) Nenhum
b) Pouco
c) Razoável
d) Alto

3. Qual é o seu nível de conhecimento na linguagem HTML?
a) Nenhum
b) Pouco
c) Razoável
d) Alto

4. Qual é o seu nível de conhecimento em acessibilidade web?
a) Nenhum
b) Pouco
c) Razoável
d) Alto

5. Qual é o seu nível de conhecimento nas diretrizes da WCAG Web Content Accessibility Guidelines 2.0?
a) Nenhum
b) Pouco
c) Razoável
d) Alto

Conhecimento prático

6. Qual é sua experiência prática na linguagem PHP?

Nenhuma	Estudado em aula ou a partir de um livro	Praticado em um projeto de classe	Usado em um projeto ou na indústria	Usado em vários projetos na indústria

Quantos meses/anos de experiência você tem nesta prática? _____

7. Qual é sua experiência prática na linguagem HTML?

Nenhuma	Estudado em aula ou a partir de um livro	Praticado em um projeto de classe	Usado em um projeto ou na indústria	Usado em vários projetos na indústria

Quantos meses/anos de experiência você tem nesta prática? _____

8. Qual é sua experiência prática no desenvolvimento de aplicações web acessíveis?

Nenhuma	Estudado em aula ou a partir de um livro	Praticado em um projeto de classe	Usado em um projeto ou na indústria	Usado em vários projetos na indústria

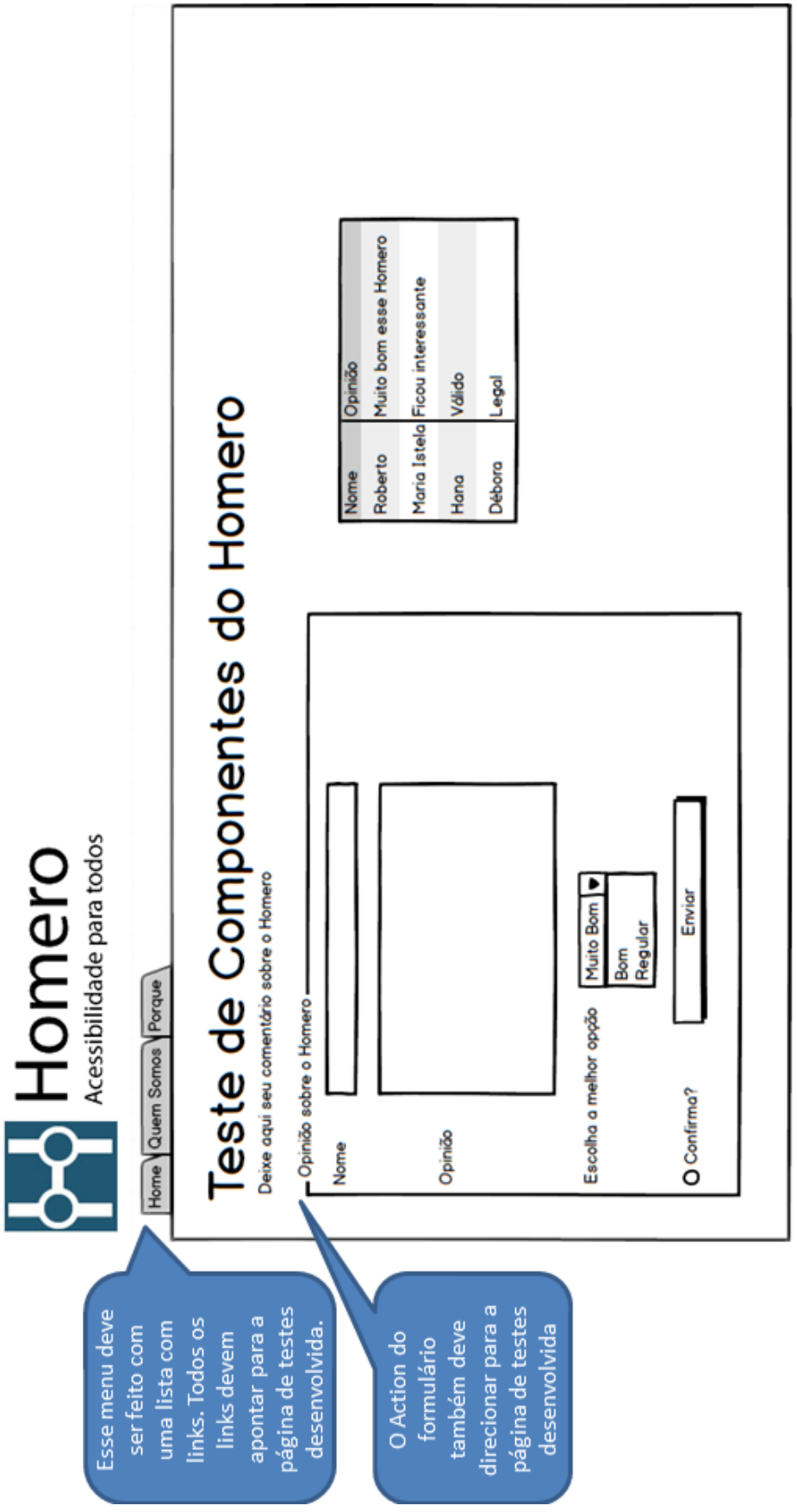
Quantos meses/anos de experiência você tem nesta prática? _____

9. Qual é sua experiência prática no uso da WCAG 2.0 durante o desenvolvimento de aplicações web?

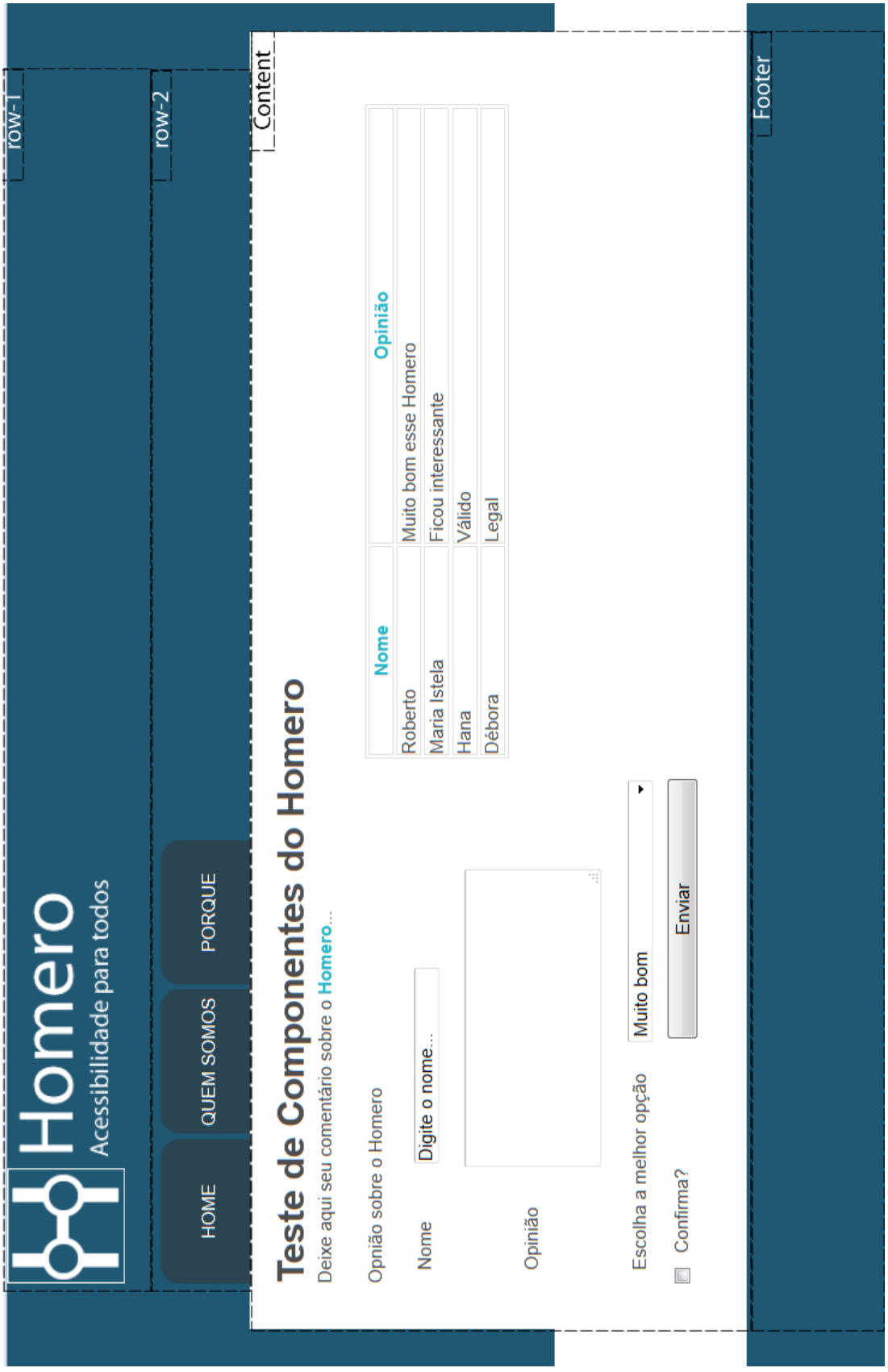
Nenhuma	Estudado em aula ou a partir de um livro	Praticado em um projeto de classe	Usado em um projeto ou na indústria	Usado em vários projetos na indústria

Quantos meses/anos de experiência você tem nesta prática? _____

APÊNDICE D – Mockup da aplicação a ser desenvolvida pelos participantes



APÊNDICE E – Imagem esperada da interface com as divisões de *layout*



APÊNDICE F - Formulário de consentimento de uso das informações

Formulário de Consentimento

Título do projeto
Homero: Um Framework de Apoio ao Desenvolvimento de Camadas de Interface de Aplicações Web Acessível

Declaração
Eu declaro que quero participar de um estudo conduzido pelo aluno de mestrado Roberto Cícero de Oliveira, cuja orientadora é a Profa. Dra. Hana Karina S. Rubstejn.

Procedimento
O estudo será conduzido em _____/_____/_____ e os horários de início e fim serão anotados em um formulário fornecido.

Confidência
Toda informação coletada no estudo é confidencial e meu nome não será identificado.

Nome do participante: _____

Assinatura do participante: _____

Nome do participante: _____

Assinatura do participante: _____

Nome do participante: _____

Assinatura do participante: _____

Data: _____/_____/_____

Formulário de Execução – Usando o Homero

Formulário de execução do estudo sobre o desenvolvimento de interfaces de aplicações web acessível utilizando o framework Homero.

Grupo: _____

Horários
Início: _____
Final: _____

Opiniões
1. Achou complicado o uso do Framework Homero para ciração de interfaces web acessíveis?
a) Não.
b) Sim. Porquê?_____
_____
_____

2. Qual sua opinião sobre a documentação do Framework?
a) Deficiente, pois não encontrei grande parte das dúvidas.
b) Razoável, mas poderia ser mais rica.
c) Ótima, pois esclarece a utilidade das classes e a forma de instanciação dos métodos.

3. Qual sua maior dificuldade de utilização do Homero?
a) Dificuldade em orientação a objetos.
b) Desconhecimento da ferramenta.
c) Acostumado com desenvolvimento de forma tradicional.
d) Outro. Qual_____

4. Achou interessante a forma como o Framework Homero prove acessibilidade e permite que o desenvolvedor utilize somente PHP?
a) Sim.
b) Não. Porquê?_____
_____
_____

5. Se tem alguma sugestão referente ao Framework deixe abaixo:
_____
_____
_____
_____
_____
_____
_____
_____

Formulário de Execução – Usando a Documentação da WCAG 2.0

Formulário de execução do estudo sobre o desenvolvimento de interfaces de aplicações web acessível utilizando a documentação da WCAG 2.0.

Grupo: _____

Horários
Início: _____
Final: _____

Opiniões
1. Achou complicado o uso da Documentação do WCAG 2.0 para criação de interfaces web acessível?
a) Não.
b) Sim. Porquê?_____
_____
_____

2. Qual sua opinião sobre a documentação da WCAG 2.0?
a) Deficiente, pois não encontrei grande parte das dúvidas.
b) Razoável, mas poderia ser mais rica.
c) Ótima, pois esclarece as diretrizes e a forma de alcançar os critérios de sucesso.

3. Momento de análise de acessibilidade?
a) Antes de iniciar o desenvolvimento, verificando quais itens precisariam de atenção especial.
b) Durante o desenvolvimento, onde a cada item implementado era analisado se o mesmo estava acessível.
c) Depois do desenvolvimento, quando a aplicação estava com a aparência sugerida começamos a analisar se estava acessível .

4. Forma utilizada para buscar acessibilidade?
a) Documentação da WCAG 2.0.
b) Ferramenta de apoio que avalia a acessibilidade.
c) Outro .

5. Se tem alguma sugestão referente a prática de desenvolvimento acessível utilizando o a documentação da WCAG 2.0:
_____
_____
_____
_____
_____
_____
_____
_____

APÊNDICE I – Mockup da interface com os componentes numerados