UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL - CENTRO DE CIÊNCIAS TECNOLÓGICAS**

CIÊNCIA DA COMPUTAÇÃO

TALES PINHEIRO FERNANDES

PROPOSTA INICIAL DE UM PROCESSO DE DESENVOLVIMENTO WEB FRONT-END UTILIZANDO *WEB COMPONENTS*

Bandeirantes

2017

TALES PINHEIRO FERNANDES

PROPOSTA INICIAL DE UM PROCESSO DE DESENVOLVIMENTO WEB FRONT-END UTILIZANDO *WEB COMPONENTS*

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Wellington A. Della Mura

Bandeirantes

2017

TALES PINHEIRO FERNANDES

PROPOSTA INICIAL DE UM PROCESSO DE DESENVOLVIMENTO WEB FRONT-END UTILIZANDO *WEB COMPONENTS*

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

COMISSÃO EXAMINADORA

_____
Prof. Me. Wellington A. Della Mura
UENP – *Campus* Luiz Meneghel

_____
Prof. Dr. André Luis A. Menolli
UENP – *Campus* Luiz Meneghel

_____
Prof. Dr. Mauricio M. Arimoto
Nome da Instituição

Bandeirantes, 16 de novembro de 2017

Dedico este trabalho à minha mãe Rita de Cássia e a minha tia Aurea Caetano, as quais me deram forças para realizá-lo.

AGRADECIMENTOS

Agradeço a todos os meus professores da UENP, especialmente ao meu orientador, Della Mura, que contribuiu para que eu realizasse este trabalho; aos companheiros da I e II turma de ciência da computação da UENP, que sempre colaboraram comigo; aos amigos da extinta República Talaricos, que foram grandes companheiros ao longos desses anos de faculdade; ao meu amigo Denílson, que para mim é como irmão e que sempre esteve presente para me ajudar quando precisei; aos irmãos que Bandeirantes me concedeu, Marcus Dorta e Fernando Moribe, que sempre me deram todo apoio possível; a toda minha família, em especial minha mãe Rita e minha tia Aurea, que reuniram esforços, centavos e orações para que eu pudesse estar aqui hoje; e a Deus acima de tudo, que me tornou apto a esta batalha.

RESUMO

O reuso na área de desenvolvimento de softwares é utilizado a um bom tempo, e com o passar dos anos passou a se tornar cada vez mais modular graças a ideia de componentização. Por sua vez, a componentização atingiu os diversos campos da área de desenvolvimento de softwares até chegar ao desenvolvimento web front-end através de frameworks e bibliotecas. Atualmente graças a quatro especificações da W3C surgiram recentemente os *web components*, que provê ainda mais granularidade para o desenvolvimento front-end. A ideia presente neste trabalho é prover um processo de desenvolvimento web front-end que faça uso dos *web components*, sendo uma tecnologia com potencial forte porém ainda um pouco inexplorada pelos desenvolvedores front-end. Para tal, foi realizado estudos e uma análise com as abordagens de desenvolvimento que possuíam afinidade com o que se buscava propor e lidar neste trabalho. Ao final, foram elaboradas 7 etapas que norteiam o desenvolvimento front-end e realiza o tratamento de componentes web.

Palavras-chave: Processo de desenvolvimento. Desenvolvimento web. front-end. *Web components*.

ABSTRACT

The reuse in the area of software development has been used for a long time, and over the years it has become increasingly modular thanks to the idea of componentization. In turn, the componentization reached the various fields of the software development area until reaching the front-end web development through frameworks and libraries. Currently, thanks to four W3C specifications, web components have recently emerged, providing even more granularity for front-end development. The idea in this work is to provide a front-end web development process that makes use of web components, being a technology with strong potential but still a little unexplored by the front-end developers. To that end, studies and an analysis were carried out with development approaches that had affinity with what was sought to propose and deal with in this work. At the end, 7 steps were developed that guide the front-end development and perform the treatment of web components.

LISTA DE FIGURAS

Figura 1 - Camadas da Engenharia de Software [fonte: PRESSMAN, 2011] 16
Figura 2 - Evolução dos requisitos [fonte: PAULA FILHO, 2000 (adaptado)].....20
Figura 3 - Atividades ESBC.....24
Figura 4 - Modelo Espiral [fonte: PRESSMAN, 2011] 25
Figura 5 - Cliente-Servidor 29
Figura 6- Front-end vs Back-end [Fonte: comic.browserling.com] 30
Figura 7- Tecnologias Front-end e Back-end 31
Figura 8- código componente de diálogo jQuery UI.. 32
Figura 9 - Componente Web de diálogo [fonte: OVERSON e STRIMPEL, 2015] 32
Figura 10 - Suporte dos Browser 20/06/2017 [fonte: webcomponents.org].....33
Figura 11 - PDFWC.....43
Figura 12 - Exemplo de planta baixa.....46
Figura 13 - Exemplo de Wireframe.....47
Figura 14 - Exemplo de modelo navegacional 49
Figura 15 - webcomponents.org.....50
Figura 16 - Processo de Seleção de *web components* 51
Figura 17 - Exemplo de Catalogação 52
Figura 18 - Stars em webcomponents.org 53
Figura 19 - Licença de um componente 53
Figura 20 - Informações do componente.....54
Figura 21 - Issues no webcomponents.org 55
Figura 22 - Exemplo de layout.....57
Figura 23 - Exemplo de layout 2.....57

LISTA DE SIGLAS

ESBC	Engenharia de Software baseada em componentes
DBC	Desenvolvimento baseado em componentes
ED	Engenharia de Domínio
UML	*Unified Modeling Language*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
DNS	*Domain Name System*
HTTP	*HyperText Transfer Protocol*
FTP	*File Transfer Protocol*
SMTP	*Simple Mail Transfer Protocol*
URL	*Uniform Resource Locator*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
OOHDM	*Object-Oriented Hypermedia Design Method*
DADI	*Definition, Architecture, Design, Implementation*
PDFWC	Processo de Desenvolvimento web utilizando *Web Components*
API	*Application Programming Interface*

SUMÁRIO

1. Introdução 13
   1.1 Justificativa 14
   1.2 Objetivos 14
      1.2.1 Objetivo Geral 14
      1.2.2 Objetivos Específicos 15
   1.3 Organização do Trabalho 15
2. Fundamentação teórica 16
   2.1 Engenharia de Software 16
      2.1.1 Objetivos da Engenharia de Software 17
      2.1.2 Categorias de Software 17
      2.1.3 Projeto 18
      2.1.4 Processos 18
      2.1.5 Requisitos 20
   2.2 Engenharia de Software Baseada em Componentes 22
      2.2.1 Visão Geral 22
      2.2.2 Fundamentos da ESBC 23
      2.2.3 Processos ESBC 24
      2.2.4 Desenvolvimento Baseado em Componentes 25
      2.2.5 Engenharia de Domínio 26
   2.3 Desenvolvimento Web 27
      2.3.1 Visão Geral sobre o funcionamento básico da Web 27
      2.3.2 Front-end e Back-end 29
   2.4 *Web Components* 31
   2.5 Engenharia Web 34
      2.5.1 OOHDM 35
      2.5.2 DADI 37
3. Metodologia 40
4. Processo de Desenvolvimento Web Front-end utilizando *Web Components* 43
   4.1 Briefing 44
   4.2 Projeto de Interface Abstrata 46
   4.3 Projeto Navegacional 48

4.4 Seleção de *Web Components* 50
4.4.1 Distinção 51
4.4.2 Busca 52
4.4.3 Ordenação 52
4.5 Qualificação 52
4.6 Adaptação de *Web Components* 55
4.7 Implementação 56
5. Conclusões 58
5.1 Discussões e Resultados 58
5.2 Trabalhos Futuros 58
Referências 60

1. INTRODUÇÃO

A *World Wide Web* cresceu demasiadamente nos últimos anos, a mesma iniciou-se de maneira simples no início da década de 90 e atualmente é altamente desenvolvida e poderosa. O lugar onde havia somente websites estáticos foi dando espaço para websites mais dinâmicos e logo foi tomada por aplicações, ou seja, softwares construídos e dedicados para a web, o que a tornou em uma plataforma de desenvolvimento.

Na área de desenvolvimento web, envolve-se diversos assuntos para tal, como exemplos engenharia web, banco de dados online, usabilidade e mais uma infinidade que varia de acordo com o que se procura desenvolver. Cada assunto segmenta-se em diversos subassuntos, o que torna algo altamente complexo abranger tudo que se existe sobre desenvolvimento web. Por conta disto, as preocupações foram divididas, como é o caso do front-end e back-end. Cada uma trata um conjunto de preocupações diferentes, a fim de tornar mais fácil o desenvolvimento web.

Outro ponto que favorece o aumento da dimensão do desenvolvimento web é o surgimento de diversos recursos e tecnologias para o mesmo. Adotar recursos novos nem sempre é o melhor caminho, para tal cabe um estudo e análise sobre isso. Porém, a “componentização” é algo prezado dentro da área de desenvolvimento de software e que nunca deixa de ser praticada. O uso de componentes já é algo existente no desenvolvimento web através da implementação de *frameworks* e bibliotecas, porém os mesmos contam com certas adversidades nos quais os *web components* buscam resolver. *Web components* são que uma composição de tecnologias que permitem que os desenvolvedores descrevam, de modo eficiente, as implementações dos elementos HTML que já existem (OVERSON e STRIMPEL, 2015).

O uso de *Web components* têm se mostrado promissor. Por conta disto, adequar-se com sua utilização é um bom passo a ser dado. Uma forma encontrada de se efetuar isto, é com a elaboração de um processo dedicado ao front-end, sendo o qual não se tem conhecimento de similares. Tendo isto em vista, a premissa deste trabalho é a de elaborar a especificação de um processo para o desenvolvimento front-end onde seja possível a reutilização de elementos das interfaces utilizando este conceito recém introduzido na comunidade de desenvolvimento web.

1.1 JUSTIFICATIVA

A criação de um novo recurso dentro de uma determinada área específica tem como por objetivo a melhoria em um determinado aspecto, sendo assim a adoção desse novo recurso é a chave para um caminho de aperfeiçoamento.

Assim como no desenvolvimento desktop, a ideia de se reutilizar código e modularizar tornou-se essencial para auxiliar no desenvolvimento web, para tal, surgiram diversas bibliotecas e *frameworks* como Bootstrap, jQuery UI e entre outros, porém o alto uso destas tecnologias pode trazer desafios tais como o aumento da complexidade do código-fonte e a impossibilidade do reuso de diferentes componentes de diferentes *frameworks* e bibliotecas dentro da mesma aplicação. Então para alcançar as vantagens da “componentização” no desenvolvimento web, surgiu um novo recurso denominado *Web Components*, que pode-se dizer que sua serventia é a de importar interfaces de maneira uniforme, ou seja, todo o aglomerado de HTML, CSS, JavaScript e imagens (OVERSON e STRIMPEL, 2015).

Relatado seu valor no ambiente de desenvolvimento web, este presente trabalho irá apurar a introdução de *Web Components* em um processo de desenvolvimento front-end a fim de viabilizar ainda mais a construção de interfaces. A motivação para tal se dá por conta de considerar *Web Components* altamente vantajoso, porém ainda muito inexplorado para a fabricação de aplicações web, eis aqui uma forma de disseminar este prospero recurso e enriquecer ainda mais a comunidade dos *Web Components*.

1.2 OBJETIVOS

Nesta seção serão apresentados o objetivo geral e os objetivos específicos do trabalho.

1.2.1 Objetivo Geral

Formular um processo de desenvolvimento web *front-end* que trate e utilize a coleção de tecnologias denominada *Web Components.*

1.2.2 Objetivos Específicos

Para a realização do que se tem como objetivo geral foi necessário atingir os seguintes objetivos específicos:

- Determinação do levantamento de requisitos necessários e específicos ao front-end;
- Definição da forma de modelagem para o projeto de interface;
- Determinação da modelagem para a navegação;
- Especificação de um tratamento de componentes web;
- Formulação de tal processo em notação BPMN.

1.3 ORGANIZAÇÃO DO TRABALHO

O restante do trabalho está organizado da seguinte forma: A Seção 2 contém a fundamentação teórica. A metodologia deste trabalho encontra-se na Seção 3. A Seção 4 contém a descrição do processo proposto. E as conclusões sobre estre trabalho estão situadas na Seção 5.

.

2. FUNDAMENTAÇÃO TEÓRICA

Nesta seção será apresentado uma revisão bibliográfica dos conteúdos de diversas áreas que serão utilizados na realização deste presente trabalho, tais eles como engenharia de software, engenharia de software baseada em componentes, desenvolvimento web, *web components* e engenharia web.

2.1 ENGENHARIA DE SOFTWARE

O termo Engenharia pode ser compreendido como a aplicação de conhecimentos (sejam eles científicos e/ou empíricos) e a utilização de recursos a fim de construir algo que seja de benefício humano. O software, no entanto, está cada vez mais se tornando algo essencial para a humanidade e a sua qualidade depende da forma em que ele foi construído, eis então que emerge a Engenharia de Software.

Pressman (2011) argumenta que a engenharia de software foi definida inicialmente em 1969 em uma conferência dedicada ao assunto. A definição foi proposta por Fritz Bauer e sua constatação sobre engenharia de software é “O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais”. Ainda que sucinta, esta é uma das definições para tal, para melhor entendimento Pressman (2011) sugere as seguintes camadas as quais a engenharia de software é constituída e aborda em seu conteúdo. Tais estão apresentadas na Figura 1.

Figura 1 - Camadas da Engenharia de Software [fonte: PRESSMAN, 2011]

Estas camadas apresentadas na Figura 1 abordam conteúdos como a gestão de qualidade, metodologias a serem aplicadas, especificações técnicas e também ferramentas de suporte ao desenvolvimento.

Estabelecido que resumidamente software é a parte lógica de um sistema computacional, e assim como qualquer outro produto industrial ele possui diversas características sobre a sua produção, ou seja, ele também possui um ciclo de vida. Então, a partir de uma necessidade ele é concebido, desenvolvido, posto em operação, reparado se necessário e por último é finalizado sua produção.

2.1.1 Objetivos da Engenharia de Software

A engenharia de software tem um importante papel no desenvolvimento de softwares, seu objetivo central é tornar que os produtos de software atinjam maiores índices de qualidade com os menores custos possíveis. Pode-se dizer que há uma contenda entre dois fatores importantes na fabricação de softwares, Qualidade x Produtividade. Deixa de ser interessante mesmo um software que possua altos índices de qualidade, porém possui alto custo de produção, o que faz dele se tornar inviável, o mesmo acontece se ocorrer o contrário, baixos custos, porém baixíssimos níveis de qualidade.

Mesmo tendo em vista estes importantes fatores, o prazo também é um fator significativo e que é crucial para que não haja insatisfações por parte do cliente. Mesmo produzindo um software de qualidade sem custos elevados, é necessário que os prazos estabelecidos sejam cumpridos ou ao menos desrespeitados o mínimo possível, tendo em vista que estimativas de prazo são frequentemente imprecisas. Em geral, a engenharia de software preza por produzir soluções mais rápidas, melhores e mais baratas.

2.1.2 Categorias de Software

Nesta seção serão apresentadas algumas das principais categorias de softwares que se existem. São elas:

- Software de Sistema: responsáveis por oferecer recursos básicos disponíveis no computador, são uma coleção de programas escritos para apoiar outros programas e possuem uma forte interação com o hardware. Exemplos: Sistemas operacionais, drivers, compiladores, etc.

- Software Embutido: são utilizados para controlar produtos e sistemas para os mercados industriais e de consumo, empregados dentro de máquinas que não são computadores de uso geral e possuem, comumente, um fim muito especifico. Exemplos: controle de microondas, de ar condicionado, de geladeiras, etc.
- Software de aplicação – são softwares desenvolvidos para cumprimento de tarefas especificas. Estes podem ser subdividos da seguinte maneira:
    - Softwares para desktop: são softwares desenvolvidos para computadores ou notebooks com finalidades bastante específicas.
    - Softwares para dispositivos móveis: comumente chamados de apps, podem ser executados em smartphones e/ou tablets.
    - Softwares para web: tratam-se de todos os aplicativos nos quais o seu acesso é feito por meio da Internet, através de algum navegador. É esta categoria na qual o processo proposto abordará.

2.1.3 Projeto

Dado a necessidade de um software e iniciado seu desenvolvimento, o mesmo é feito dentro de um projeto, sendo estabelecido uma data de início, uma data fim, uma equipe e entre outros recursos (PAULA FILHO, 2000).

Quem contrata a execução de um projeto é denominado Cliente na área de engenharia de software, seja ele uma pessoa física ou uma pessoa jurídica, ou até mesmo podendo ser um representante legal. Já o indivíduo que usará o software efetivamente é denominado como Usuário, podendo ele ser ou não o cliente (PAULA FILHO, 2000).

Paula Filho (2000) aponta um projeto como a representação da execução de um processo.

2.1.4 Processos

Um processo pode ser considerado como um conjunto de passos bem definidos que tem por objetivo alcançar algum resultado, podendo ele ser um

produto ou uma parte de um produto. Processos podem ou não serem ricos em detalhes, mas basicamente são descritos como uma receita na qual algum determinado projeto possa seguir.

Pressman (2011) aponta que um processo dentro da área de engenharia de software não é uma prescrição rígida de como se desenvolver um software. A ideia é de que um processo seja uma abordagem adaptável, sendo possível que a equipe de software escolha o que seja apropriado para o desenvolvimento. A finalidade deve ser a de sempre entregar o software dentro do prazo e com qualidade suficiente para satisfazer o cliente e os usuários.

De acordo com Pressman (2011) o processo da engenharia de software forma a base para o controle do gerenciamento de projetos de software e estabelece o contexto no qual:

- Os métodos técnicos são aplicados;
- São produzidos produtos de trabalho;
- Os marcos são estabelecidos;
- A qualidade é assegurada;
- As modificações são adequadamente geridas.

A escolha de um modelo de processo de desenvolvimento de software é o ponto de partida para iniciar um projeto e existem diversos modelos. Não se existe um modelo que seja ideal para todos os casos, cabe a equipe de desenvolvimento adotar para o projeto aquele que mais se adequa com o que se busca produzir. Os diversos modelos existentes possuem prós e contras e em conta disso pode ser que uma combinação de mais de um modelo possa ser uma boa solução para um determinado projeto.

Mesmo sendo variado o número de modelos existentes, Sommerville (2011) indica que existem quatro atividades genéricas que são comuns e fundamentais entre os processos, são elas:

- Especificação: As funcionalidades e as restrições devem ser definidas;
- Projeto: Produzir o software de acordo com as especificações;
- Validação: O software deve ser validado para assegurar de que as necessidades do cliente sejam satisfeitas;

- Evolução: O software pode evoluir para atender as necessidades mutáveis do cliente.

As principais diferenças entres os diversos modelos existentes são a organização destas atividades e o nível de detalhes.

2.1.5 Requisitos

Quando se inicia um projeto de software, pode-se dizer de que compreender todas as necessidades e exigências do cliente não é uma tarefa trivial. Obter com clareza as premissas do cliente e entender a natureza dos problemas que precisam ser solucionados é um dos primeiros passos fundamentais para se construir um software de qualidade. As divergências de pensamento e de compreensão entre todos os agentes envolvidos é algo relativamente comum e prejudicial no desenvolvimento de software. A Figura 2 contém ilustrações destas divergências.

Figura 2 - Evolução dos requisitos [fonte: PAULA FILHO, 2000 (adaptado)]

Eis então que para se desenvolver um software, inicialmente é necessário realizar o levantamento de requisitos. Sommerville (2011) define requisitos para sistemas como a descrição das funções e das restrições. Já Paula Filho (2000) descreve como características do sistema, sendo elas que definem os critérios de aceitação de um produto. Os requisitos se distinguem nos seguintes tipos:

- Funcionais: estes requisitos representam como as funções do sistema devem ser feitas, a reação que o sistema deve apresentar de acordo com certas ações dos usuários.
- Não funcionais: estes abordam as restrições e os aspectos de qualidade.

De maneira sucinta, pode-se dizer que os funcionais indicam “O que fazer” e os não funcionais indicam “Como fazer”. Por exemplo, em um terminal de caixa eletrônico de um banco, todos os tipos de transações que podem ser realizadas são considerados como requisitos funcionais. Já aspectos como a facilidade de uso, o tempo de resposta são considerados como requisitos não funcionais.

Para separar os níveis de especificação dos requisitos de um projeto, Sommerville (2011) divide em três níveis, utilizando os termos requisitos do usuário, requisitos do sistema e especificação de projeto de software. Estes três termos podem ser definidos da seguinte maneira:

- Requisitos do usuário: são declarações sobre as funções que o sistema deve fornecer e as restrições sob as quais deve operar, sendo todos eles descritos em uma linguagem natural e também com o auxílio de diagramas.
- Requisitos do sistema: estes são responsáveis por estabelecer de forma detalhada as funções e as restrições do sistema. Estas especificações devem ser precisas e podem servir como um contrato entre o cliente e o fabricante de software.
- Especificação do projeto de software: responsável pela descrição abstrata do projeto de software, sendo ele uma base para o projeto e a implementação de forma mais detalhada. Sendo que esta especificação acrescenta mais detalhes a especificação de requisitos do sistema.

Existem inúmeras formas e técnicas para se realizar o levantamento, a documentação e a análise dos requisitos, Paula Filho (2000) aponta que este conjunto de técnicas constitui a engenharia de requisitos, sendo ela uma das disciplinas da engenharia de software.

Paula Filho (2000) ressalta que é de responsabilidade do engenheiro de software insistir e garantir uma boa especificação de requisitos. Sendo parte do trabalho deles convencer os clientes e usuários de que:

- É fundamental e indispensável uma boa especificação dos requisitos;
- As especificações não são custos supérfluos e sim investimentos necessários, que se pagam com altos juros;

- A participação e colaboração de todos os que irão utilizar o sistema é fundamental na engenharia de requisitos para que as necessidades de todos os usuários sejam atendidas corretamente pelo produto de software;
- Uma boa especificação de requisitos custa tempo e dinheiro (e que a ausência da mesma custa muito mais).

2.2 Engenharia de Software Baseada em Componentes

Dentro desta seção será explanado sobre a área da engenharia de software responsável por abordar a construção de “sistemas modulares” e de componentes para estes sistemas. O intuito é transmitir a ideia da “componentização” na qual é influxo para este presente trabalho.

2.2.1 Visão Geral

A engenharia de software veio para beneficiar o desenvolvimento de softwares. Reunindo diversos conjuntos, sendo eles de conhecimentos, de técnicas, de métodos e de recursos a prol de organizar e produzir um produto de software que seja de qualidade. Com a engenharia de software baseada em componentes não é diferente, sendo ela uma disciplina da engenharia de software, tem por vantagem acelerar ainda mais o desenvolvimento com base em sua lógica de construção de software. A ESBC (engenharia de software baseada em componentes) surgiu como uma abordagem para softwares de desenvolvimento de sistemas tendo como base o reuso de componentes de software (SOMMERVILLE, 2011). Pressman (2011) utiliza a definição de componente da Especificação da Linguagem de Modelagem Unificada da OMG (*Object Management Group*) que caracteriza como “...Uma parte modular, possível de ser implementada e substituível de um sistema que encapsula implementação e expõe um conjunto de interfaces”.

O surgimento da ESBC aconteceu por volta da década de 90 e veio por conta da frustração dos projetistas em relação ao desenvolvimento orientado a objetos não ter atingido o amplo reuso que haviam dito na época (SOMMERVILLE, 2011). A ESBC é considerada o passo seguinte da Orientado a Objetos e sua motivação se dá por conta de quebrar os blocos monolíticos em componentes interoperáveis, tendo assim facilidade na manutenção, pois a possibilidade de remoção e adaptação

por outro componente independente viabiliza a correção de problemas no software sem ter que alterar as demais partes do sistema. E a construção de um novo produto de software, que seja baseado em componentes, pode ser construído utilizando componentes já produzidos o que faz com que a produtividade e agilidade seja significativamente aumentada.

Nitidamente o número de softwares só vem aumentando com o passar do tempo e junto o tamanho e a complexidade dos softwares também vêm aumentando. E está cada vez mais sendo exigido maiores níveis de qualidade e mais agilidade no desenvolvimento, a “componentização” é uma saída para suportar tais demandas.

2.2.2 Fundamentos da ESBC

Segundo Sommerville (2011) existem fundamentos os quais são pertencentes a ESBC, sendo eles:

- As interfaces especificam completamente os componentes independentes. Sendo necessário haver separação nítida entre a interface e a implementação. Ou seja, a implementação de um componente pode ser substituída por outra sem a necessidade de alteração em outras partes do sistema.
- Para a facilidade de integração há padrões de componentes. Sendo essas normas incorporadas a um modelo de componentes. No mínimo eles definem como interfaces de componentes devem ser especificadas e como será a comunicação entres os componentes. Há modelos que vão além e definem as interfaces que devem ser implementadas por todos os componentes. Se há conformidade dos componentes com os padrões, sua operação é independente de sua linguagem de programação.
- O middleware é responsável por prover suporte de software para a integração de componentes.
- Um processo de desenvolvimento que seja voltado para ESBC. É necessário que haja um processo de desenvolvimento que possibilite que os requisitos evoluam.

2.2.3 Processos ESBC

Sommerville (2011) aponta que em nível mais alto existem dois tipos de processos da ESBC, sendo eles:

- Desenvolvimento para reuso: trata de desenvolver componentes que posteriormente serão reusados por aplicações.
- Desenvolvimento com reuso: este trata desenvolver sistemas utilizando os componentes já existentes.

Ambos processos são distintos e então possuem atividades diferentes, o desenvolvimento para reuso é designado a criar componentes, desde a especificação (analise e documentação) até a implementação. A documentação de um componente é o que possibilita a reutilização daquele componente em outros sistemas, caso haja necessidades de alterações. Neste processo o desenvolvedor claramente possui acesso ao código-fonte do componente. Já no desenvolvimento com reuso é efetivamente a criação de sistemas com um conjunto de componentes interconectados. Mas para isso não se sabe quais componentes estão aptos e disponíveis para reutilização, sendo assim necessário pela busca por componentes para o teu projeto. Neste processo não há acesso para o código-fonte do componente.

Portanto pode-se dizer que existem duas atividades nas quais constituem a ESBC, sendo ela o Desenvolvimento Baseado em Componentes (DBC) e a Engenharia de Domínio. A Figura 3 traz uma ilustração com os conjuntos que compõem a ESBC.

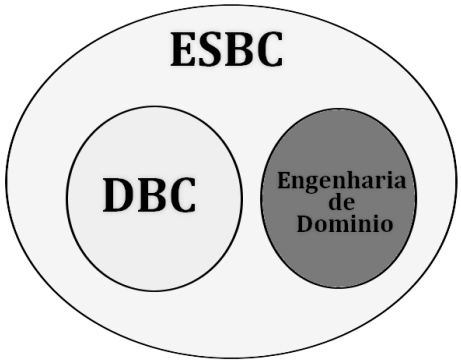

Figura 3 - Atividades ESBC

2.2.4 Desenvolvimento Baseado em Componentes

É a construção em si de sistemas utilizando um conjunto de componentes já existentes, podendo os componentes terem sido produzidos internamente ou comprados de terceiros. O sistema desenvolvido desta forma deve suportar alterações apenas em partes específicas e independentes sem a necessidade de ter que alterar grandes partes do seu sistema. O DBC é uma adaptação do Modelo Espiral (vide Figura 4). Tal modelo que consiste em obter um produto de software no qual é desenvolvido em uma série de iterações, onde cada iteração equivale a uma volta no espiral. Sendo que nas primeiras iterações são gerados protótipos simples e no decorrer deste espiral, são construídas versões cada vez mais completas do software (PRESSMAN, 2011).

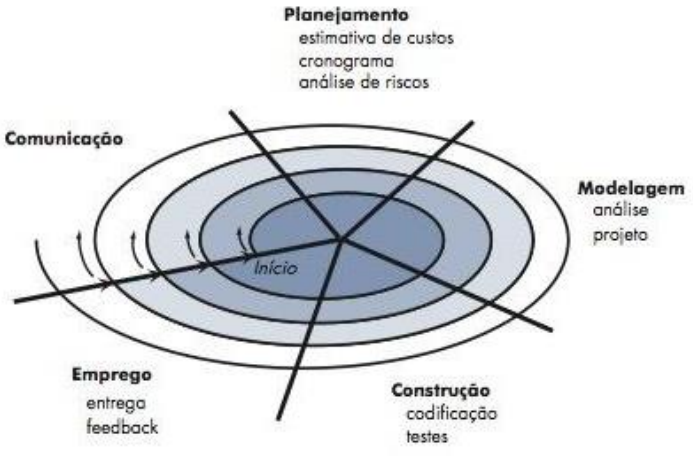

Figura 4 - Modelo Espiral [fonte: PRESSMAN, 2011]

O DBC é descrito por diversos autores e para Brown e Short (1997), há cinco fases dentro do DBC, sendo elas:

- Seleção: nesta fase é feita a busca por componentes e são reunidos aqueles que têm chances de ser interessantes para o projeto.
- Qualificação: neste estágio são examinados os componentes nos quais foram reunidos na fase anterior averiguando quais se adequam a arquitetura do sistema a ser desenvolvido. Fatores importantes usados na avaliação dos componentes são a performance, confiabilidade e usabilidade.
- Adaptação: nesta fase serão feitas as adaptações necessárias para poder se utilizar os componentes já selecionados anteriormente.

- Composição: é feito a integração dos componentes no sistema.
- Atualização: é a última fase e ela realiza a troca de componentes obsoletos por versões mais novas e sofisticadas, com o comportamento e interfaces similares.

Basicamente se componentes reutilizáveis existem para uma determinada ocasião, eles precisarão ser qualificados e adaptados. Porém caso não existam componentes necessários, então terá de ser desenvolvido os mesmos seguindo modelos de componentes. Os componentes desenvolvidos devem ser organizados junto com aqueles componentes que sejam de mesmo domínio de aplicação para que possam ser facilmente reutilizados em diversos projetos. A atividade dentro da ESBC na qual é responsável por isso é denominada Engenharia de domínio.

Ao longo dos anos foram surgindo métodos, frameworks, tecnologias, ambos baseados na ideia de se construir sistemas fazendo uso de componentes. Método de desenvolvimento desta área que pode-se destacar além de Brown e Short (1997) é o Catalysis que faz uso da UML para definir seu conjunto de técnicas e métodos, tanto para análise de negócios quanto para o desenvolvimento de sistemas (MIRANDA e LAGES, 2003). O mesmo serviu como base para os outros métodos que surgiram logo após, sendo um deles que também pode-se destacar é o próprio UML components (MIRANDA e LAGES, 2003). O processo proposto neste trabalho de conclusão de curso se baseará nas fases da abordagem de Brown e Short (1997) por abstraírem princípios compatíveis com o que se trabalhará aqui neste processo.

2.2.5 Engenharia de Domínio

Segundo Pressman (2011), a ED (Engenharia de Domínio) tem como por objetivo identificar, construir, catalogar e disseminar um conjunto de componentes de software que tenham aplicabilidade em algum software existente. É ela quem dispõe a biblioteca de componentes reutilizáveis necessários para a ESBC. Um domínio é um conjunto de áreas funcionais que possuem funcionalidades similares. Segundo Dabhade, Suryawanshi e Manjula (2016), a ED é dividida principalmente em três fases, sendo elas a de análise, projeto e implementação. Sendo que estas fases basicamente irão identificar os domínios, identificar soluções cabíveis para determinados domínios e implementar estas soluções.

2.3 Desenvolvimento Web

Desde o início da web, seu crescimento se comporta de forma exponencial, a décadas passadas a web apenas provia de sites simples, estáticos, construídos apenas com a linguagem de marcação denominada HTML. Tais sites apenas dispunham de texto e imagens. Com o passar dos anos, a web foi crescendo e junto foram surgindo outras tecnologias para a contribuição no desenvolvimento de websites.

Nos últimos anos a web deixou de ser apenas um ambiente para páginas simples e passou a se tornar uma plataforma de aplicações. E pode-se dizer que são variados os tipos de sistemas que se existem na web. Conforme o crescimento da web e com o surgimento de novas tecnologias, o desenvolvimento web passou a se tornar mais complexo. Com um grande aumento no número de usuários e no número de serviços disponíveis, a exigência por aplicações cada vez melhores e mais robustas se fez necessário. Consequentemente o número de tecnologias envolvidas também foram só aumentando, exigindo cada vez mais profissionais no desenvolvimento de uma aplicação web.

2.3.1 Visão Geral sobre o funcionamento básico da Web

A *World Wide Web*, popularmente conhecida apenas como web, é disposta pela Internet, a rede mundial de computadores, a web nada mais é do que um sistema de documentos que permite acesso a conteúdos, apresentados no formato de hipertexto. A internet segue o modelo TCP/IP e a web faz uso de alguns de seus recursos.

A web é baseada na arquitetura cliente-servidor (vide Figura 5), sendo a máquina que deseja acessar uma página através de browser é denominada cliente e a máquina na qual está hospedado esta página é caracterizada como servidor.

- **Endereço IP e portas**

Todas as máquinas nas quais estejam conectadas a rede mundial de computadores possuem um endereço numérico único denominado endereço IP (*Internet Protocol*) e além do endereço, também há um grande número de portas nas quais as aplicações e processos realizam a comunicação. As portas denominam algum tipo de serviço, alguns exemplos são o de aplicações que realizam serviços de entrega de e-mail que é realizado através da porta 25 e utiliza o protocolo SMTP

(*Simple Mail Transfer Protocol*), ou então serviço de transferência de arquivos que utiliza a porta 21 e faz uso do protocolo FTP (*File Transfer Protocol*). Já o serviço de entrega de páginas web se encontra na porta 80.

- **DNS**

As máquinas que estejam conectadas em rede, são configuradas para acessarem um servidor denominado servidor DNS (*Domain Name System*), é ele quem é responsável por associar a URL (*Uniform Resource Locator*) a um endereço IP especifico que seja da máquina que está localizada os documentos do site requerido pela URL.

- **Comunicação entre computadores**

Para que um servidor possa servir os documentos requeridos pelo cliente, antes se deve estabelecer uma comunicação entre as máquinas. A máquina cliente solicita abrir uma conexão com o servidor e quando aceito é realizado a comunicação entre elas. Tal tipo de conexão é realizado através do protocolo TCP, que é um protocolo que possui garantia de entrega de pacotes de maneira organizada, sem alterações de ordem e conteúdo.

- **Protocolo HTTP e o HTML**

A comunicação entre os servidores e os browsers (clientes) é padronizada e é realizada através do HTTP (*HyperText Transfer Protocol*). É ele o protocolo responsável por especificar as mensagens que os clientes podem enviar a servidores e que respostas eles irão receber. Sendo que todos os clientes e servidores devem obedecer este protocolo (TANENBAUM, 2003). É denominado como requisição a solicitação do cliente de uma página web, e denominada resposta quando o servidor retorna o que o cliente solicitou. Ambos os dois possuem em seu conteúdo informações necessárias para que aconteça a comunicação entre as máquinas. A Figura 5 ilustra o modelo Cliente-Servidor.

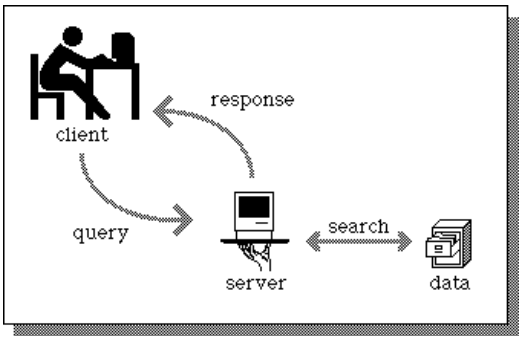

Figura 5 - Cliente-Servidor

O HTML (*HyperText Markup Language*) é a linguagem utilizada para a marcação de websites. É importante ressaltar de que HTML não é uma linguagem de programação e sim de marcação. Depois de ter obtido a resposta do servidor através de uma requisição feita pelo cliente, é disponibilizado ao cliente um documento HTML. O browser é responsável por analisar o conteúdo do documento para realizar novas requisições, sendo assim possível a aquisição de outros recursos que constituem a página web, por exemplo, folhas de estilo, scripts, imagens, etc. Adquirido todos recursos necessários, o browser começa a renderizar a página web.

O HTML faz parte de um conjunto de tecnologias atualmente essenciais no desenvolvimento de websites. Este conjunto é formado pela linguagem de estilos denominada CSS e também pela linguagem de scripts denominada JavaScript. O CSS é usado para formatar a exibição dos conteúdos, através dele é possível estilizar as páginas web, dando mais estética ao site. Já a linguagem JavaScript é utilizada principalmente para dinamizar e prover interações nas páginas web.

Estas três tecnologias são suficientes para a criação de websites, porém são insuficientes para a criação de um sistema web, onde se pode realizar transações. Para o desenvolvimento de sistemas web é necessário o uso de linguagens de programação voltadas para a web, o que faz com que se divida em duas partes a sua construção, sendo elas o Front-end e o Back-end (vide Seção 2.3.2).

2.3.2 Front-end e Back-end

Quando se trata de desenvolvimento web, o desenvolvedor front-end e o desenvolvedor back-end se completam, sendo que cada um deles trabalham com assuntos específicos.

- Front-End: Como o próprio nome diz, é responsável por trabalhar com a parte da “frente” do sistema. Trabalha-se com interfaces gráficas e é designada a exibir e coletar informações. Aspectos como usabilidade por exemplo são de suma importância nesta área. Dentro desta área trabalha-se principalmente com HTML, CSS, JavaScript, entre outras tecnologias para a composição de interfaces gráficas (páginas web).
- Back-end: Como o nome sugere, trabalha-se nesta área com a parte de “trás” do sistema, tudo aquilo que acontece internamente no qual não fica aos olhos do usuário. Área responsável por tratar das regras de negócio, é ela quem processa os dados captados pelo front-end. Dentro desta área trabalha-se principalmente com bancos de dados e linguagens de programação, como por exemplo PHP, Java, mySQL e entre outras diversas tecnologias que podem ser utilizadas.

A Figura 6 e a Figura 7 contém ilustrações destas áreas envolvidas no desenvolvimento web.

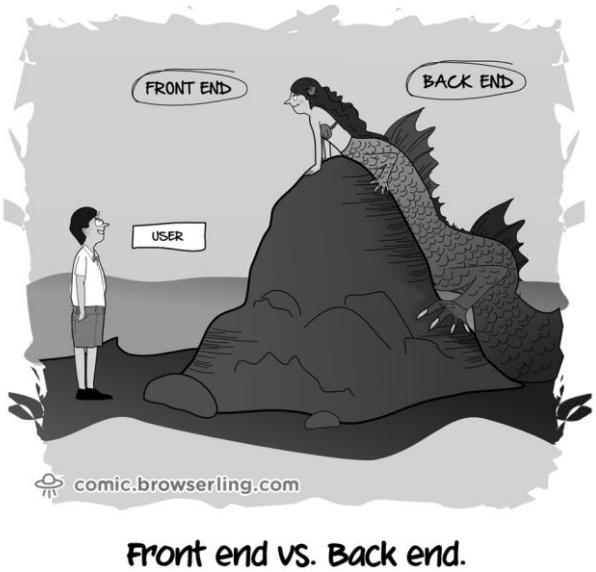

Figura 6- Front-end vs Back-end [Fonte: comic.browserling.com]

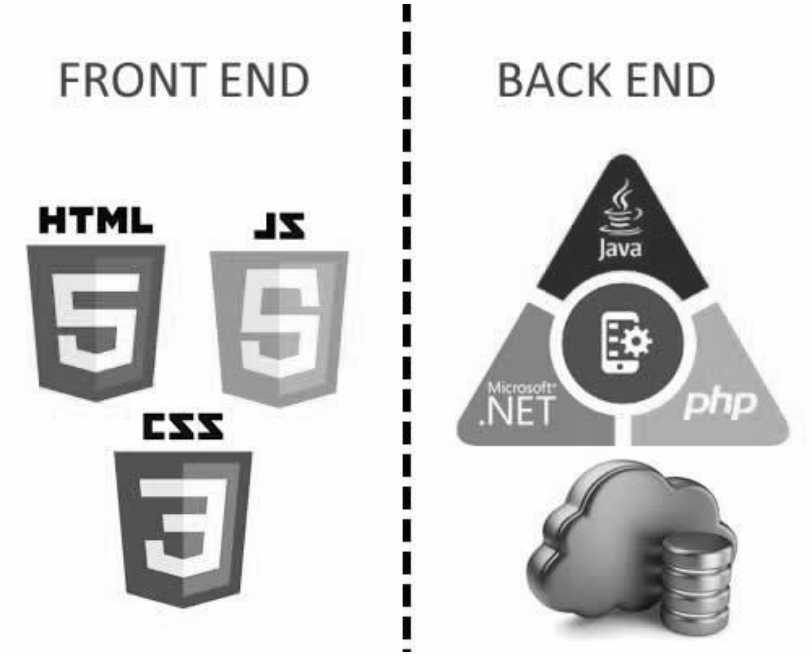

Figura 7- Tecnologias Front-end e Back-end

Com a divisão destas áreas, as preocupações também são divididas, em resumo o front-end deve se preocupar mais em dispor a melhor experiência possível ao usuário, enquanto o back-end deve se preocupar com a persistência dos dados, regras de negócio, performance, segurança da informação e entre outros quesitos.

2.4 Web Components

A ideia de se construir utilizando componentes para interfaces do usuário já é existente a um bom tempo e largamente utilizada pelos desenvolvedores. Diversas bibliotecas e frameworks dispõe destes recursos, porém ainda há limitações no uso destes componentes existentes. O mais famoso *framework* front-end é o Bootstrap, que é amplamente utilizado pelos desenvolvedores e repleto de diversos componentes feitos a partir de HTML, CSS e JavaScript.

Porém ao se utilizar um *framework* pode-se notar certas adversidades, por exemplo, limitações ao se utilizar outros componentes de outros *frameworks* dentro da mesma aplicação, importação de código desnecessário, cópia de várias linhas de código e com isso também o aumento significativo da complexidade no código-fonte e entre outros contratempos. Os *web components* vêm com a promessa de otimizar e revolucionar o modo de se trabalhar com componentes na web, no qual os componentes podem ser facilmente importados e de maneira uniforme, incluindo

todos os seus artefatos como JavaScript, CSS e imagens (OVERSON e STRIMPEL, 2015).

A Figura 8 contém um trecho de código para se utilizar um componente de diálogo usando a biblioteca jQuery UI.

```
<!-- based on http://jqueryui.com/dialog/ -->
<head>
    <link
     rel="stylesheet"
     href="//code.jquery.com/ui/1.11.0/themes/smoothness/jquery-ui.css">
    <script src="//code.jquery.com/jquery-1.10.2.js"></script>
    <script src="//code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
    <script>
        $(function () {
            $( "#dialog" ).dialog();
        });
    </script>
</head>
<body>
    <div id="dialog" title="After Ford">
        Ending is better than mending. <br />
        The more stitches, the less riches.
    </div>
</body>
```

Figura 8- código componente de diálogo jQuery UI [fonte: OVERSON e STRIMPEL, 2015]

E como seria utilizar um componente de diálogo feito a partir de *web components*? A Figura 9 exibe um trecho de código necessário para a utilização do mesmo.

```
<head>
    <link rel="import" href="/imports/dialog/index.html">
</head>
<body>
    <dialog-component title="After Ford">
        Ending is better than mending. <br />
        The more stitches, the less riches.
    </dialog-component>
</body>
```

Figura 9 - Componente Web de diálogo [fonte: OVERSON e STRIMPEL, 2015]

Neste caso seria necessário apenas realizar a importação do recurso e declarar a marcação. *Web components* nada mais é do que um aglomerado de especificações que torna tudo isso possível. São quatro especificações da W3C e são elas:

- *Custom Elements*: esta especificação permite que você crie sua própria tag HTML e também permite que ela possa operar ao teu gosto. Por exemplo <meu-componente>.
- *Templates*: permite definir blocos de códigos que são reutilizáveis podendo inserir elementos dinâmicos.
- *Shadow* DOM: oferece uma maneira de encapsular o HTML e o CSS em uma árvore separada do DOM.
- HTML *imports*: permite importar componentes HTML da mesma maneira que CSS e JavaScript separados.

Segundo Overson e Strimpel (2015), *web Components* estão na vanguarda absoluta. Uma prova disso é o suporte dos navegadores diante as quatro especificações que constituem *web components*, apenas o Chrome e o Opera têm suporte a todas elas, sendo que outros navegadores fazem uso de Polyfills (arquivos JavaScript que simulam as funcionalidades que o browser deveria prover) para possibilitar a execução destas especificações. O suporte atualmente pode ser visto pela Figura 10.

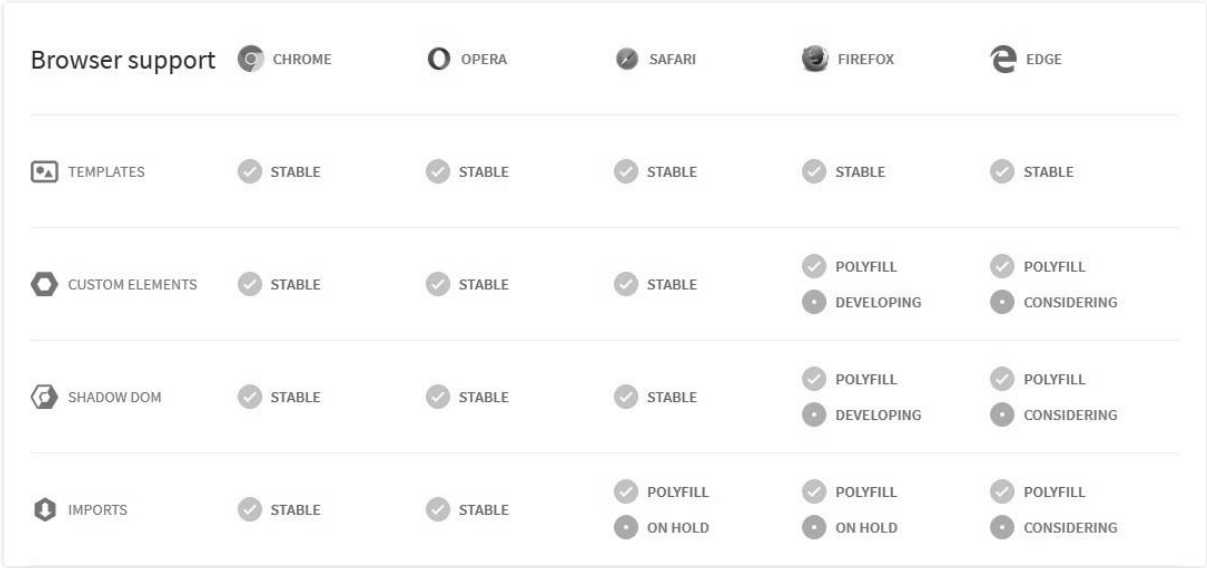

Figura 10 - Suporte dos Browser nov/2017 [fonte: webcomponents.org]

Existem bibliotecas para auxiliar no desenvolvimento a partir do conceito de *Web Components*, sendo importante o destaque de duas, uma delas é a X-TAG, que foi criada por um desenvolvedor da Mozilla que posteriormente migrou e foi trabalhar na Microsoft e levou consigo o projeto, e a biblioteca mais popular que é o Polymer,

sendo um projeto da Google e que possui forte investimento, onde já possuem aplicações rodando com ela, um exemplo é o próprio Google Music.

Ao que tudo indica, *Web Components* tem todo potencial para revolucionar a web, podendo transformar a plataforma web extensível, onde qualquer elemento possa ser estendido, e novos elementos possam ser definidos para criarem facilmente ricas interfaces do usuário (OVERSON e STRIMPEL, 2015).

2.5 Engenharia Web

No início da *World Wide Web*, popularmente conhecida apenas como *Web*, os *websites* eram desenvolvidos de maneira *ad-hoc*, ou seja, não haviam metodologias, processos e padrões para o desenvolvimento do mesmo. Porém ao longo dos anos a complexidade no desenvolvimento web foi só aumentando, tornando *websites* em verdadeiras aplicações, transformando a *web* em uma plataforma de implementação. Eis então que emergiu a necessidade de se aplicar métodos disciplinados de Engenharia de Software adaptados ao contexto *web* (SOUZA, 2007) e evidentemente também foram surgindo novos métodos totalmente específicos para a mesma.

Com as características do ambiente web, como estética, concorrência, carga imprevisível, disponibilidade, sensibilidade ao conteúdo, evolução dinâmica, imediatismo e segurança (PRESSMAN, 2005) se fez necessário uma nova dinâmica aos processos já existentes, surgindo então o que pode-se dizer subárea da Engenharia de Software denominada Engenharia Web.

A engenharia web é multidisciplinar, e, portanto, para o desenvolvimento de um sistema web, se faz necessário do conhecimento de profissionais de diversas áreas, Ginige e Murugesan (2001) indica alguns como:

- engenharia de software,
- análise de sistemas e projetos,
- engenharia de hipermídia e hipertexto,
- engenharia de requisitos,
- interação humano-computador,
- desenvolvimento de interface de usuário,
- engenharia de informação,
- indexação e recuperação de informações,

- teste,
- modelagem e simulação,
- gerenciamento de projetos,
- projeto gráfico e apresentação.

Existem diversas metodologias e processos para o desenvolvimento de aplicações web, e aqui neste presente trabalho serão apresentadas algumas, nos quais mais se tem afinidade com o que se procurar elaborar dentro deste presente trabalho. As próximas seções apresentarão algumas metodologias propostas para a Engenharia Web. Foram consideradas metodologias que listem as atividades a serem executadas e artefatos a serem produzidos e apresentando ou sugerindo a notação a ser utilizada na construção de tais artefatos. São elas apenas as quais serviram como norteio para este trabalho, sendo: OOHDM (*Object-Oriented Hypermedia Design Method* e DADI (*Definition, Architecture, Design, Implementation*).

2.5.1 OOHDM

O *Object-Oriented Hypermedia Design Method* (OOHDM) é uma metodologia proposta por Schwabe, Rossi e Barbosa (1996) e é utilizada para o desenvolvimento de aplicações do tipo hipermídia. Tal metodologia é baseada em modelos e utiliza técnicas de orientação a objetos.

Conforme Nunes (2005), o OOHDM provê cinco principais fases, sendo elas: Levantamento de requisitos, projeto conceitual, projeto navegacional, projeto de interface abstrata e implementação. Sendo que em cada fase um modelo é produzido e refinado, gerando um conjunto de artefatos relacionados. Vale ressaltar de que tais fases não precisam necessariamente serem trabalhadas de maneira sequencial e de mesma ordem, estas podem ser executadas de maneira iterativa e cíclica.

- **1 - Levantamento de Requisitos**:

Tal etapa, segundo Nunes (2005), tem como objetivo principal a identificação de atores e tarefas a serem apoiadas pela aplicação. Para tal, devem ser formulados descrições dos cenários e casos de uso, tais são apresentados através de narrativas em formato textual. Nesta etapa também se produz diagramas de interação que

servirão de base para uma validação através de representação gráfica de comportamento da aplicação (NUNES, 2005).

- **2 – Projeto Conceitual**:

Nesta etapa é produzido um modelo conceitual da aplicação, sem levar em conta ainda aspectos navegacionais ou então relativos a plataforma de implementação (NUNES, 2005). Pois então é nesta fase que será modelado a semântica do domínio da aplicação. Segundo Nunes (2005), os modelos nesta fase produzidos podem ser representados através de primitivas de orientação a objetos, como classes e atributos, utilizando a notação da UML.

- **3 – Projeto Navegacional**:

Nesta etapa será criado um modelo de navegação, mapeado a partir do modelo conceitual, dando ênfase aos aspectos cognitivos. Segundo Nunes (2005), o modelo utilizado nesta etapa permite definir primitivas que irão representar o projeto de navegação, sendo que as classes conceituais serão mapeadas em classes navegacionais, relações entre classes são mapeados em elos, que por sua vez permitem definir novos tipos de atributos navegacionais como ancoras e índices. Tal modelo pode ser representado visualmente através de uma “UML customizada”.

De acordo com Nunes (2005), também é necessário ser produzido nesta etapa, um esquema de contextos navegacionais, utilizando uma própria notação, que irá permitir a representação de primitivas navegacionais como landmarks (ponto de referência no qual o usuário pode querer acessa-lo rapidamente), estruturas de acesso e contextos de navegação. Este esquema permite a representação precisa de todos os caminhos de navegação oferecidos pela aplicação (NUNES, 2005).

- **4 – Projeto de Interface Abstrata**:

Será nesta etapa a qual será produzido um modelo abstrato da interface do usuário, onde serão definidas as visões que o usuário poderá obter sobre as instâncias de objetos navegacionais, os quais são chamados de nós (NUNES, 2005).

Tal modelo projetará a estrutura arquitetural da interface, no qual a próxima etapa, a de implementação, usará como mapeamento para implementar os elementos de apresentação (NUNES, 2005).

- **5 – Implementação**:

Será nesta etapa onde será produzido uma aplicação executável com base em todos os modelos gerados anteriormente. Para tal, são utilizadas linguagens de programação, *frameworks*, ambientes de desenvolvimento e entre outros recursos que ajudem e minimizem o esforço necessário para o desenvolvimento desta etapa (NUNES, 2005).

2.5.2 DADI

O nome desta metodologia é o acrônimo para as suas quatro principais etapas: *Definition, Architecture, Design* e *Implementation*. Sendo que no português significa: Definição, Arquitetura, Design e Implementação.

Segundo Schwartzman (1998), esta metodologia foi elaborada por Clement Mok, no ano de 1996, na qual foi adaptada e é utilizada para o desenvolvimento de websites. Assim como toda metodologia esta também está sujeita a alterações e constante evolução. A mesma serve como base no processo de desenvolvimento de websites em algumas empresas brasileiras, sendo tal metodologia adaptada de acordo com a realidade e as necessidades da empresa em si. A seguir veremos suas quatro etapas e o que cada uma representa:

- **1 – Definição**

Segundo Vicentini e Mileck (2000), esta etapa tem início logo na primeira reunião. Tal etapa inicia-se com *Briefing* para o levantamento de diversas informações nas quais serão analisadas para a definição do escopo do projeto.

Para Vicentini e Mileck (2000), os pontos importantes nesta etapa são:

- Objetivos x Orçamentos: objetivo do site, resultados para o cliente, prazo de desenvolvimento.
- Levantamento de fontes: coleta de textos (impressos ou digitalizados), imagens, logotipos que poderão ser utilizados
- Análise do conteúdo: após coleta do material, selecionar quais são relevantes e devem constar no website.
- Análise do contexto: observação do cenário no qual o site que está sendo desenvolvido se encaixa. É recomendável que se visite alguns outros sites de mesma área.

- Público alvo e tecnologias empregadas: a análise do perfil do público alvo a fim de tentar identificar qual é o equipamento utilizado e situar o público no contexto sociocultural e econômico.
- Protótipo e aprovação: apresentação de um protótipo do site proposto contendo alguns elementos de design e um primeiro nível de navegação.

- **2 – Arquitetura**

Para Vicentini e Mileck (2000), esta etapa é a que a equipe do projeto analisa as informações obtidas na primeira etapa, sendo determinado a relevância do material recolhido, a estrutura da informação e a definição das prioridades (hierarquia) que estas informações deverão ser apresentadas. São utilizados *wireframes* para a representação da estrutura do website a ser desenvolvido. Sendo que para esta fase os pontos importantes são:

- Definição da “Mensagem do Site”: é o que foi considerado como objetivo do site, e a forma de apresentação;
- Estrutura da informação: agrupar as informações, identificando-as e separando em blocos organizados por seções ou assuntos principais;
- Recursos de interface: sendo que para cada bloco de informações definido, aproveitar os recursos que a interface nos oferece, determinando a melhor maneira de apresentar estas informações considerando diferentes mídias;
- Interatividade: levar em consideração a interatividade no website, encaixando-a perfeitamente no contexto do mesmo.
- Navegabilidade: definir um processo de navegação pela interface, como o usuário poderá navegar, podendo ser um trajeto exploratório aonde ao interagir com o site o usuário vai descobrindo aos poucos suas diversas informações, funções, serviços e produtos disponíveis.

- **3 – Design**

Após a definição da estrutura do website e a sua funcionalidade, inicia-se a transição dos *wireframes* para propostas gráficas produzidas pelo *webdesigner,* antes de iniciar o desenvolvimento *front-end*. Após a aprovação destas propostas,

inicia-se então a produção com uso de linguagem de marcação HTML com folhas de estilo CSS e entre outras tecnologias utilizadas para o desenvolvimento do *front-end*.

Para Vicentini e Mileck (2000), diversos aspectos são considerados nesta etapa como tipografia, forma de apresentação dos textos, padronização sobre as imagens criadas, o tratamento das diversas mídias que podem ser utilizadas. Todas as informações captadas do início do projeto até a presente etapa são relevantes e servem para conduzir a forma de desenvolvimento aqui realizada, sendo de grande importância levar em consideração os aspectos relacionados a usabilidade.

- **4 – Implementação**

Nesta fase serão implementadas e testadas as funcionalidades nas quais o website deve possuir e a integração de todas as páginas web. Para tal são utilizadas linguagens de programação e entre outros recursos.

As principais atividades nesta fase, segundo Vicentini e Mileck (2000), são a programação, os testes de interface, a definição do endereço URL e do servidor, o *upload* dos arquivos para o servidor e o lançamento do website, que consiste em realizar um trabalho de divulgação do website desenvolvido.

- **Conclusão**:

Em resumo, a primeira etapa desta metodologia consiste em levantar e analisar os requisitos. Já a segunda etapa baseia-se em estruturar os elementos que constituirão o website e a terceira etapa projetar o que foi proposto na segunda etapa. E a quarta etapa desempenha o papel de fazer funcionar por completo o website proposto. A metodologia DADI pode servir como base para processos de desenvolvimento web, apenas incrementando alguns outros métodos, fases e recursos, de acordo com as necessidades identificadas pela equipe de produção.

3. METODOLOGIA

O presente trabalho é a formulação de um processo de desenvolvimento web front-end utilizando *web components*. Foram realizadas pesquisas nas quais orientaram a formulação do processo aqui proposto. Esta abordagem reuni métodos, boas práticas, padrões, conhecimentos e entre outros conteúdos para atingir o objetivo geral especificado.

A pesquisa deste trabalho seguiu diversos conceitos contidos em diferentes áreas, como: desenvolvimento web, engenharia de software, engenharia de software baseada em componentes, *web components* e Engenharia Web.

Conteúdo essencial presente neste trabalho sendo ele o desenvolvimento web, a partir do mesmo, buscou-se explicitar sobre o uso de linguagens como HTML, CSS e JavaScript, que atuam em determinada parte no desenvolvimento de uma aplicação web, tais aspectos que são peças chaves para este processo.

Dentro da área de engenharia de software foi buscado explicitações de conceitos que relatam sobre boas práticas, o que é um processo, passos iniciais para o desenvolvimento, distinções entre tipos de sistemas e entre outros quesitos. Já a engenharia web indica pontos a mais para se preocupar em relação a engenharia de software tradicional, pois trata-se de um ambiente de desenvolvimento que envolve diversos outros fatores a mais que devem ser avaliados para a construção de uma aplicação.

A engenharia de software baseada em componentes ilustra como a componentização é algo que está presente na construção de aplicações já há um tempo. Dentro da mesma, está contida assuntos como o Desenvolvimento Baseado em Componentes, que foi o influxo para este presente trabalho, onde não se entrou em níveis muito técnicos e sim superficiais, pois mesmo tratando-se de “componentização”, DBC trata de tecnologias distintas das que se encontra neste trabalho.

Futuramente, pretende-se também trabalhar com o desenvolvimento de novos componentes como um subprocesso dentro deste processo. Ao início desta abordagem, devem ser captados o conjunto de quesitos para o projeto e como produto, a geração de um conjunto de interfaces de usuário construídas a partir de *web components*, tais que estes componentes poderão ser reutilizados no próximo desenvolvimento.

Para elaboração desta abordagem aqui proposta, foi realizado uma análise comparativa com as abordagens de desenvolvimento já citadas anteriormente na fundamentação teórica deste trabalho (Vide Seção 2). Diversas abordagens foram estudadas e foram selecionadas as quais haviam mais afinidade com o que se procura dispor neste projeto a fim de propor algo presumivelmente melhor. O Quadro 1 dispõe as características, de cada abordagem, consideradas relevantes para a elaboração deste processo.

Quadro 1 - Análise

	OOHDM	DBC (Brown e Short)	DADI	Abordagem proposta
Briefing e análise de requisitos	sim, porém inconcluso para o caso.	não	Sim, atende boa parte do que se busca propor.	sim
Projeto Navegacional	sim, atende bem ao mapeamento das interfaces do projeto da aplicação.	não	sim, indica a produzir um mapeamento das interfaces.	sim
Projeto de Interface	sim, possui um modelo de apresentação que apresenta como as informações serão dispostas ao usuário.	não	sim, preocupa-se em produzir um modelo estrutural das informações que serão dispostas ao usuário.	sim
Tratamento de Componentes	não	sim, mesmo sendo componentes de software, há análise de componentes.	não	sim
Desenvolvimento Front-end	não	não	sim, metodologia na qual dá boa ênfase ao desenvolvimento front-end.	sim

Feito os estudos e a análise, pôde-se sugerir as etapas necessárias para o desenvolvimento de front-end utilizando componentes web. No total serão dispostas 7 etapas onde cada uma delas descreverá uma série de atividades necessárias e algumas delas deverão prover modelos.

Este processo é uma descrição que servirá como apoio a equipe de desenvolvedores front-end, sendo a qual não necessariamente é uma forma absoluta de se criar interfaces gráficas para a web, dessa forma poderão ser acrescidas as atividades que forem de acordo com a política e de vezo de cada equipe de desenvolvimento, assim como afirma Pressman (2011) sobre processos na área de engenharia de software, onde a ideia é de que um processo deva ser uma abordagem adaptável, sendo possível que a equipe de software escolha o que seja apropriado para o desenvolvimento, onde a sua finalidade seja a de sempre entregar o produto dentro do prazo e com qualidade suficiente para satisfazer o cliente e os usuários.

As etapas do processo são: Briefing, Projeto de Interface Abstrata, Projeto Navegacional, Seleção de Componentes, Qualificação de Componentes, Adaptação de Componentes e Implementação. As três primeiras etapas, Briefing, Projeto de Interface Abstrata e Projeto Navegacional são baseadas de acordo com as metodologias web OOHDM e DADI, sendo acrescidas apenas de alguns aspectos importantes dentro deste contexto. As demais são acrescidas com o tratamento dos componentes. A última etapa, Implementação, é a responsável por implementar o front-end.

Um *web component* pode ser construído com outros *web components*, sendo possível até mesmo utilizar um *layout* completo do front-end através de um único *web component*. O uso desse tipo de recurso pode ser utilizado caso se enquadre com as exigências obtidas através da primeira fase, a do Briefing.

4. PROCESSO DE DESENVOLVIMENTO WEB FRONT-END UTILIZANDO *WEB COMPONENTS*

Nesta seção, apresenta-se um processo de desenvolvimento front-end, que é a contribuição deste presente trabalho. Este processo, chamado de “Processo de Desenvolvimento web Front-End utilizando *Web Components*” (PDFWC), especifica sete etapas, sendo desde a especificação do front-end até a implementação final das interfaces gráficas. Com diversas etapas, o processo pretende auxiliar a equipe de desenvolvimento front-end e lhes familiarizar com um conceito novo e promissor de componente web de uma maneira metódica. O processo inicia-se no Briefing e encerra na Implementação. A Figura 11 contém um diagrama com notação BPMN (*Business Process Model and Notation*) que mostra uma visão geral do processo proposto.

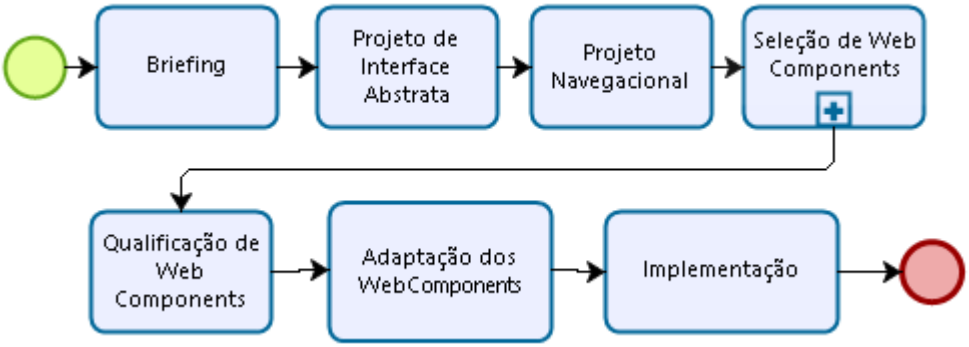

Figura 11 - PDFWC

O processo também estará disponível através do link: talespinheiroo.github.io. As sete etapas que constituem o PDFWC são:

- **Briefing:** etapa responsável pelo levantamento dos quesitos para o desenvolvimento do front-end.
- **Projeto de Interface Abstrata:** tem como princípio abstrair a estrutura de como a página web deverá ser através de modelos como *wireframes*.
- **Projeto Navegacional:** consiste em mapear a navegação entre essas páginas.

- **Seleção de webcomponents:** ficará responsável por fazer o levantamento de todos os componentes que possuem potencial para o projeto, de acordo com os elementos necessários das páginas web.
- **Qualificação de webcomponents:** seleciona de fato quais serão os que realmente vão ser utilizados no front-end.
- **Adaptação de webcomponents:** faz os ajustes necessários para que os componentes fiquem de acordo com o que se busca desenvolver.
- **Implementação:** consiste em implementar o front-end com as linguagens necessárias e fazendo uso de componentes web.

O PDFWC poderá sugerir algumas maneiras de se realizar testes, porém não especifica detalhadamente uma fase estritamente para isto. Neste processo, trabalhará concentradamente no sistema de navegação e então podendo até executar o procedimento de Implementar-Testar dentro de etapas como a de Adaptação de *web components* e a de Implementação. A seguir será explanado as sete etapas deste processo.

4.1 Briefing

O significado da palavra Briefing denota uma reunião para se obter um conjunto de informações e instruções concisas e objetivas para o desenvolvimento de uma determinada tarefa. Assim como na primeira etapa do PDFWC, Briefing é a reunião entre cliente e membros das demais equipes do projeto, não apenas a equipe de front-end, a fim de realizar a análise de requisitos e assim poder obter um conjunto de informações e materiais, tudo a fim de alcançar o domínio e especificar como as interfaces gráficas deverão ser. O projeto do front-end deve estar em conformidade com as premissas do projeto do software web como um todo.

Coletado todos os quesitos necessários para a criação das interfaces nesta etapa, os mesmos serão analisados e então serão realizadas estimativas e a definição do escopo das interfaces, o número de contratados necessários e o Prazo.

Todas informações e materiais obtidos nesta etapa deverão ser classificados e armazenados. Ao final desta etapa, deve-se ter uma ideia inicial da estrutura do site, sendo o qual deverá ser feito um esboço simples para o cliente, apenas para se ter certeza se está de acordo com as necessidades dele.

No início do Briefing deverá ser produzido uma descrição. Nela será feito o relato descritivo, através de linguagem natural, de como deverá ser o front-end da aplicação web. A partir disso, deve-se preocupar com uma gama de pontos importantes que aconselhavelmente devem ser ressaltados e apurados dentro desta etapa. Alguns deles são:

- **Conjunto de conteúdos**: são todos os assuntos nos quais poderão ser expostos nas interfaces. É realizado o apuramento de imagens, textos, logotipos, vídeos, cores e entre outros conteúdos.
- **Conjunto de elementos**: É o levantamento de possíveis elementos que deverão conter nas interfaces, tais como campo de login, formulários, menus e entre outros.
- **Análise de conteúdo**: deverá ser refinado os conteúdos, onde deverá ser apontado o que de fato irá para as interfaces e o que não irá, ou seja, todo material coletado é analisado e selecionado considerando o escopo do projeto. Também deverá se atentar com o que deverá ser refeito, como por exemplo uma logomarca ou *folder*.
- **Análise de contexto**: terá de situar em que contexto aquela aplicação está contida. Sendo assim, pode-se analisar sites dentro do mesmo contexto a fim de procurar estruturas comuns, a possibilidade de aperfeiçoamento e de construção de algo que forneça um diferencial inovador.
- **Detecção de Público-alvo**: Na web, muitas vezes é incerto a definição exata do tipo de público o qual o site terá. Mas se o contexto permitir, apurar o possível público alvo. Tal informação pode colaborar na escolha de linguagem e estilo que o site poderá utilizar.

Com as questões anteriormente citadas já abordadas, poderá ser feito uma análise de todas essas informações e construir uma nova descrição do projeto de front-end, sendo esta mais refinada e técnica. Todas as informações obtidas anteriormente devem ser classificadas e documentadas juntas, pois servirão como base para as próximas etapas. Como já citado anteriormente no início desta subseção, ao final da etapa do Briefing um esboço simples que possa ser compreendido pelo cliente deverá ser produzido, a fim de compreender se o projeto está em conformidade com suas imposições e se necessita de alternâncias.

4.2 Projeto de Interface Abstrata

Assim como uma construção civil necessita de uma planta baixa (vide Figura 12) para representar de maneira esquemática o que deverá ser construído, a construção do front-end também necessita de uma “planta” para conduzir o seu desenvolvimento. Nesta subseção será apresentado a segunda etapa do PDFWC que consiste em elaborar um modelo que abstraia a estruturação de uma interface gráfica.

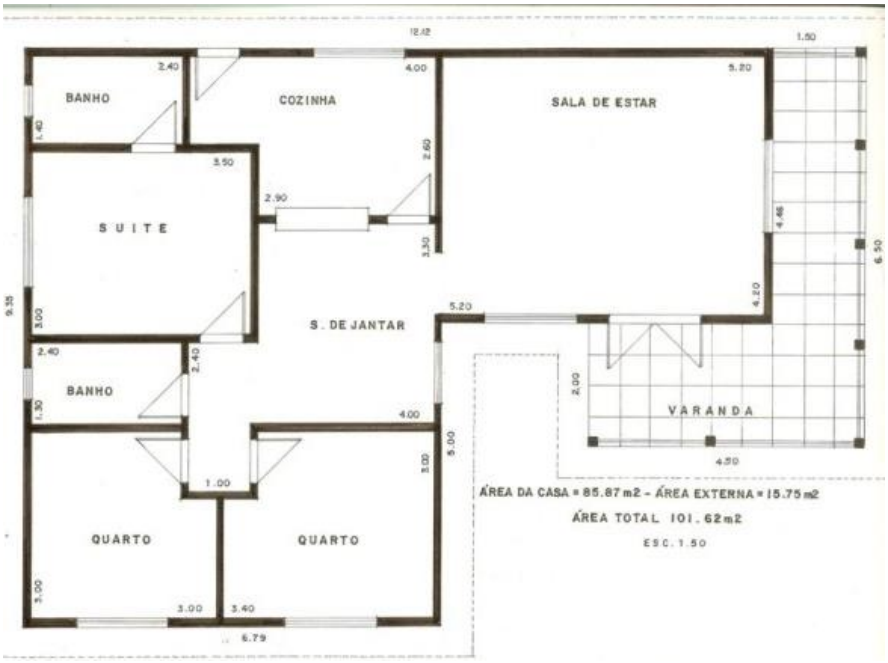

Figura 12 - Exemplo de planta baixa

A geração de código é algo relativamente complexo e requer profissionais qualificados, portanto desenvolver algo concisamente o mais próximo possível do desejado é a solução para que esta tarefa seja o menos custosa possível. Para obter este êxito, é necessário um delineamento íntegro do que será desenvolvido, por conta disso, percebe-se a necessidade da modelagem do front-end. Com base nas informações obtidas na primeira etapa do processo, deverá ser construído um modelo que servirá como guia na construção do front-end. A maneira mais apropriada de se criar os modelos das interfaces é através do uso de wireframes, que são um desenho básico, como um esqueleto, que representa de forma direta a arquitetura de uma interface gráfica de acordo com as especificações relatadas. Os wireframes devem ser feitos de maneira simples, sem se preocupar com cores e

estética, apenas para indicar o mapeamento dos elementos necessários dentro da interface. A Figura 13 contém um exemplo de wireframe.

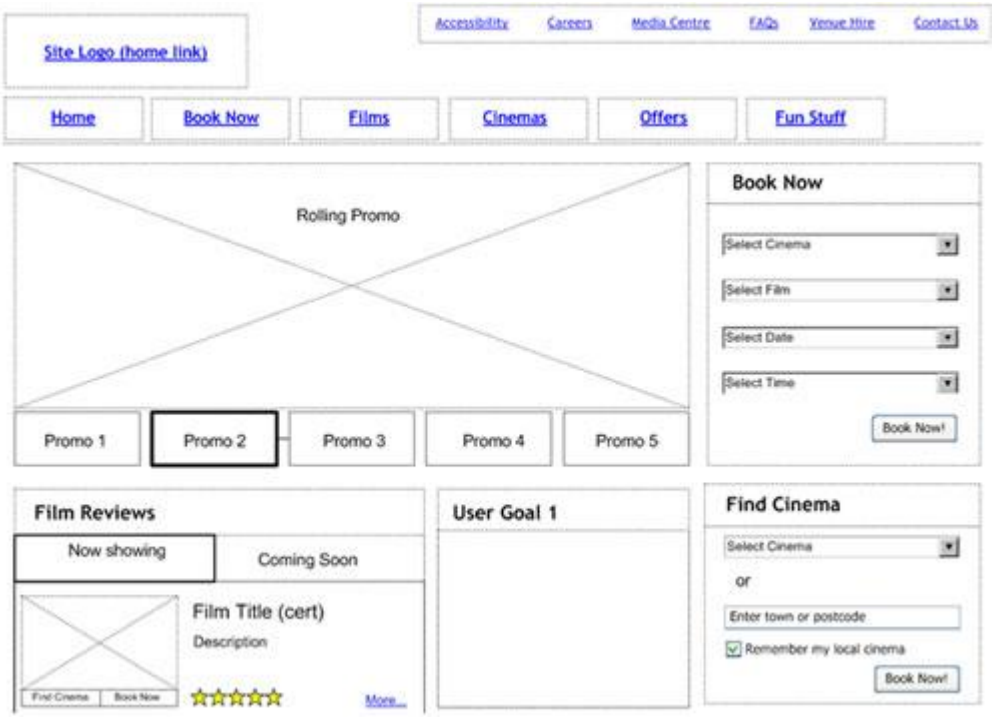

Figura 13 - Exemplo de *Wireframe* [fonte: experienceux.co.uk]

Para este caso deverão ser utilizados wireframes web, e para a construção do mesmo, existem diversas ferramentas que auxiliam a criação destes modelos, sendo essencial o uso destes tipos de ferramentas a fim de acelerar o processo.

Aspectos de suma importância que devem ser levados em consideração na hora de modelar as páginas web que constituíram o front-end da aplicação, estes pontos importantes são:

- **Organização:** identificar e organizar em blocos as informações, separando-as em seções.
- **Categorização:** estabelecer prioridades da informação.
- **Definição de recursos:** Juntamente com o *wireframe*, indicar, determinar e descrever a melhor maneira de se apresentar as informações podendo variar de diversas formas, um exemplo é: podendo ser estática, dinâmica, com uso de recursos multimídias ou entre outras maneiras.

Se faz necessário o uso de boas práticas e padrões web para a estruturação do front-end de maneira que, posteriormente, sejam construídas interfaces ricas e agradáveis. Alguns exemplos adequados para isto é o uso das heurísticas de Nielsen (1994), padrões elaborados pela W3C (2017) ou então elementos de interface gráfica racionalizados por Toxboe (2017). O importante é de sempre entregar interfaces de qualidade, que maximizem os níveis de satisfação do usuário.

Ao final desta etapa, todas as interfaces gráficas que deverão constituir o front-end da aplicação deverão ter seus respectivos *wireframes,* semelhantes ao da Figura 13 juntamente com as descrições relacionadas aos modelos. O cliente deverá ser consultado para avaliar o projeto de interface abstrata, sendo que, se necessário, serão realizadas alterações nos modelos para que fiquem de acordo com as exigências do cliente.

4.3 Projeto Navegacional

Para assimilar como poderá ser a navegação entres as interfaces, se faz necessário um mapeamento de toda possível navegação dentro da aplicação a ser desenvolvida, ou seja, deverá ser construído um Projeto Navegacional que representará e descreverá toda essa navegação. E esta etapa é a responsável pela criação deste projeto navegacional.

O Projeto Navegacional pode ser um modelo visual ou textualmente organizado. Este modelo a ser elaborado deve ser demonstrado de maneira compreensiva, o mesmo pode ser representado de diversas maneiras, sendo através de diagramas, mapas mentais, *storyboards* e entre outros. Todos estes modelos podem ser criados de maneira automatizada com o auxílio de ferramentas, sendo essencial o uso destas por contribuírem com a agilidade no processo. Para a execução desta etapa do processo, os pontos importantes são:

- **Definição da estrutura**: Definir a estrutura navegacional em todos os pontos da aplicação, onde deverá ser linear, hierárquica e/ou em rede.
- **Definição da navegabilidade**: como o usuário navega pela estrutura, como ele pode ir de um canto a outro dentro da aplicação.
- **Definição de ações**: definir as ações necessárias através de *storyboards.*

Um exemplo de artefato gerado nesta etapa do processo que represente um projeto navegacional está contido na Figura 14.

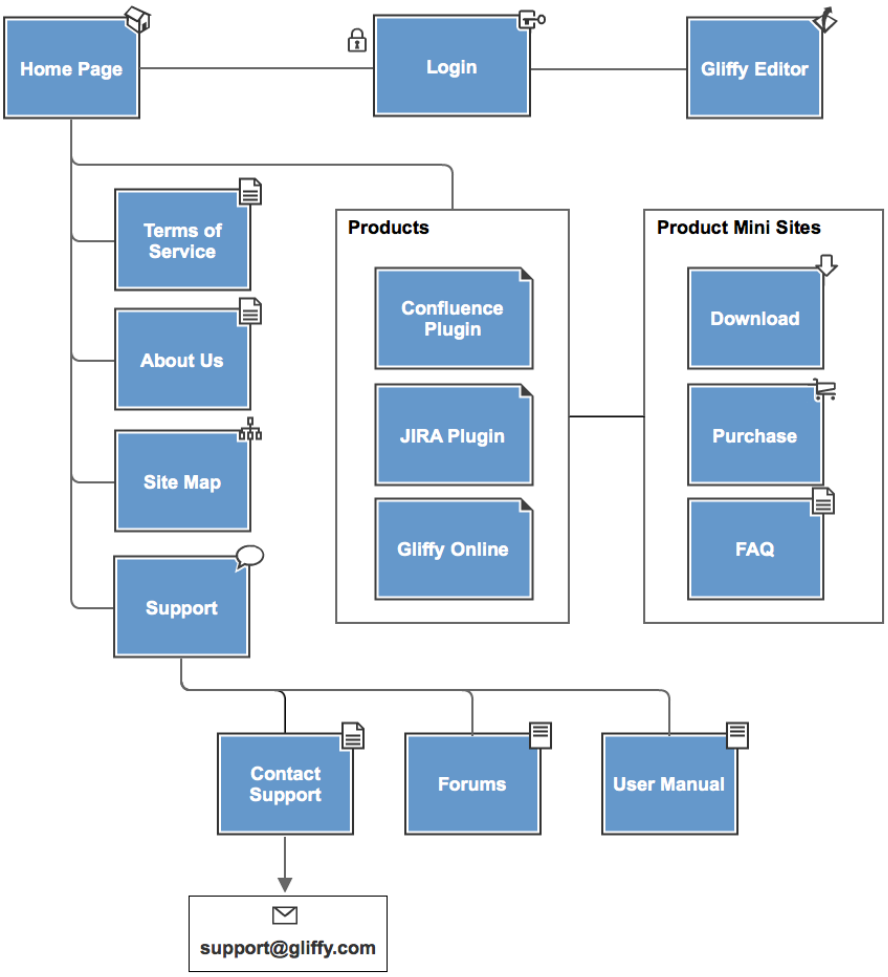

Figura 14 - Exemplo de modelo navegacional [fonte: sitemaps.org]

O projeto navegacional deve proporcionar um panorama geral de como funcionará a navegação pela sua aplicação, sendo assim mais fácil de enxergar certas necessidades de melhorias navegacionais, podendo até gerar alterações nos *wireframes* construídos na etapa anterior para que se adequem as essas melhorias. O projeto navegacional também poderá auxiliar na formulação de um possível *Sitemap*, que nada mais é do que um mapa da aplicação web para indicar ao robô de busca da Google (2017), por exemplo, quais páginas serão indexadas no servidor, facilitando assim o entendimento do buscador e tornando mais eficaz o vasculhamento de conteúdo no site.

Ao final desta etapa, é obtido mais um material que contribuirá para o entendimento e desenvolvimento da aplicação web. Tudo que foi levantado nas etapas anteriores e ao fim desta etapa serão juntamente indexados, o que proverá em uma documentação detalhada de como deverá ser o front-end da aplicação web a ser desenvolvida. As próximas etapas irão realizar o tratamento dos componentes.

4.4 Seleção de *Web Components*

Assim como no DBC descrito por Brown e Short (1997), neste processo haverá etapas para o tratamento de componentes, sendo esta etapa, a Seleção de *web components*, a primeira a discutir sobre componentes. Por tratar de tecnologias distintas, o PDFWC usará indicadores diferentes dos do DBC, sendo os quais são restritos aos webcomponents. Até o presente momento deste trabalho, webcomponents.org (2017), conforme Figura 15, encontra-se como o maior repositório aberto de webcomponents, sendo este o qual o PDFWC irá explorar.

Nesta seção será explicitado sobre a etapa que consiste em identificar e reunir os componentes como potenciais interessantes para a equipe de desenvolvedores front-end do projeto. Os componentes web podem vir de uma variedade de fornecedores, portanto, será necessária uma investigação sobre as propriedades e qualidades do componente, no qual tal tarefa caberá a próxima etapa do PDFWC. Nesta etapa, sabe-se pouco ainda sobre os componentes, apenas deverão ser reunidos de acordo com suas funcionalidades.

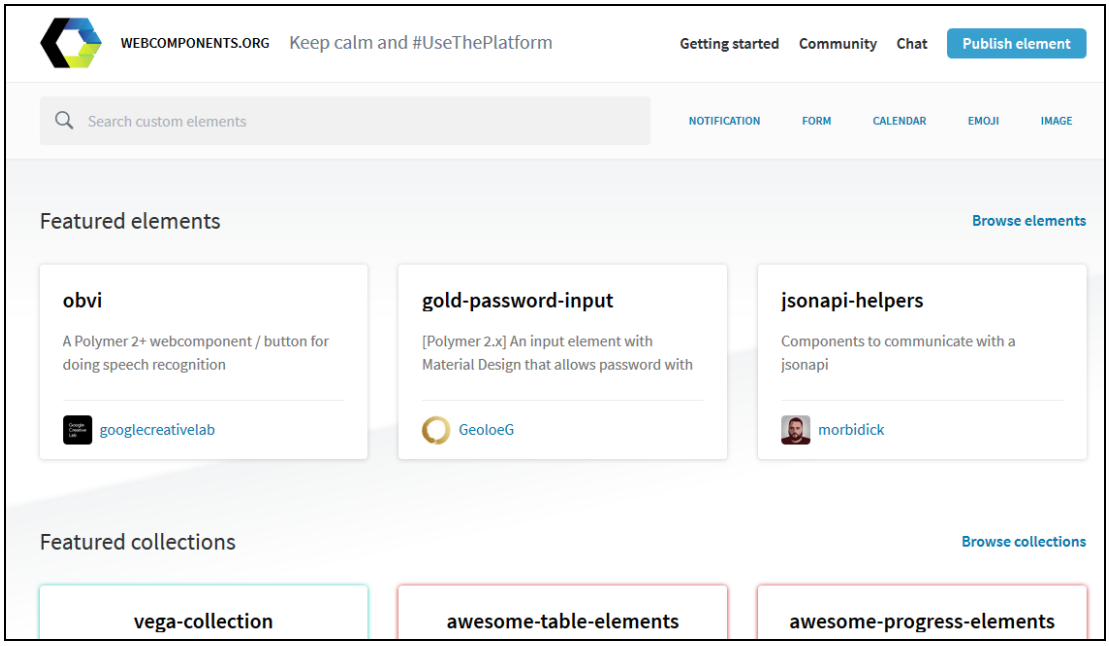

Figura 15 - webcomponents.org

A determinação para se escolher um componente deverá estar de acordo com as necessidades do projeto, por isso esta etapa de seleção é um subprocesso do PDFWC constituída por até 4 etapas, porém até o presente momento, o trabalho

discutirá apenas 3 das 4 etapas. A Figura 16 traz um diagrama com a notação BPMN no qual apresenta um panorama deste subprocesso.

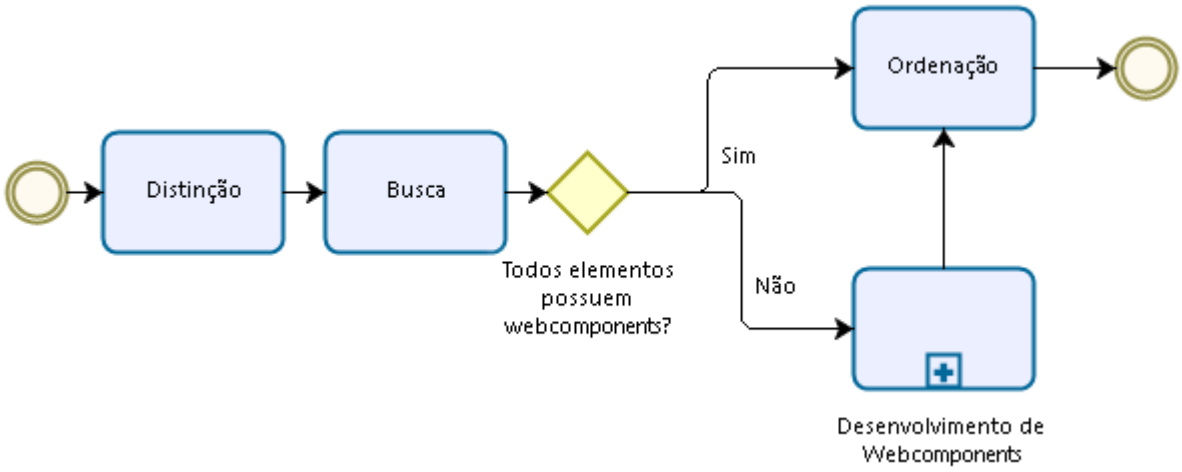

Figura 16 - Processo de Seleção de *web components*

As etapas que constituem e serão apresentadas do subprocesso de seleção de webcomponents são:

- Distinção;
- Busca;
- Ordenação.

A etapa de Desenvolvimento de *Web Components*, como consta na Figura 16, não será explanada no PDFWC por ser um outro subprocesso no qual não se encontra no escopo deste trabalho por conta de sua complexidade. Porém a ressalva desta etapa neste trabalho se dá por conta de ser um dos próximos trabalhos a serem elaborados. A seguir serão apresentadas as etapas de subprocesso de seleção.

4.4.1 Distinção

Os wireframes construídos na segunda etapa do PDFWC serão utilizados nesta etapa do subprocesso de seleção. Serão distinguidos todos os elementos, que constituem o modelo, possíveis de serem *web components*. Feito isto, deve ser construído uma relação de elementos que serão buscados a partir de *web components*. Exemplo: *toolbars, forms, calendars, buttons, footers,* etc.

4.4.2 Busca

Após a realização da etapa de Distinção, será utilizada a relação elaborada na etapa anterior para iniciar a busca por estes elementos no repositório webcomponents.org (2017). Todos os *web components* que possuírem algum potencial de serem utilizados no projeto, deverão ser devidamente relacionados.

4.4.3 Ordenação

Todos os *web components* buscados e selecionados como potenciais interessantes para o projeto deverão ser catalogados de maneira sistemática, descrevendo por exemplo o seu nome, tipo, para que servem, para qual página possivelmente será destinado e anexar a documentação provida pelo *web component*. Um exemplo simples é apresentado pela Figura 17.

```
PAGINA:  index.php

#1 COMPONENT
nome: ibm-toolbar
tipo: toolbar
documentação: https://www.webcomponents.org/element/IBMResearch/ibm-toolbar

#2 COMPONENT
nome: google-map
tipo: Mapa
documentação: https://www.webcomponents.org/element/GoogleWebComponents/google-map

...
```

Figura 17 - Exemplo de Catalogação

É importante que todos esses *web components* estejam devidamente organizados, porém sua catalogação pode ser feita da maneira em que a equipe presumir melhor. Note que a Figura 17 apenas exemplifica e não necessariamente rege como essa organização deva ser.

4.5 Qualificação

A etapa anterior, a de seleção de *web components*, proveu uma relação de possíveis componentes e, dentro desta etapa, a de qualificação, serão selecionados os componentes que de fato serão utilizados na construção do front-end da

aplicação. Para isto, esta etapa descreverá alguns indicadores que possam auxiliar na escolha efetiva desses componentes. Ressaltando de que serão apenas indicadores e não um regimento inviolável de como eleger componentes web. A seguir serão apresentados estes indicadores.

Stars: Uma Star tem o mesmo princípio de “gostei” ou “like” de uma rede social por exemplo. É este indicador que define o número de pessoas que utilizaram aquele determinado componente e aprovam/recomendam. A Figura 18 apresenta alguns *web components* do repositório webcomponents.org com os seus respectivos indicadores Stars destacados em vermelho.

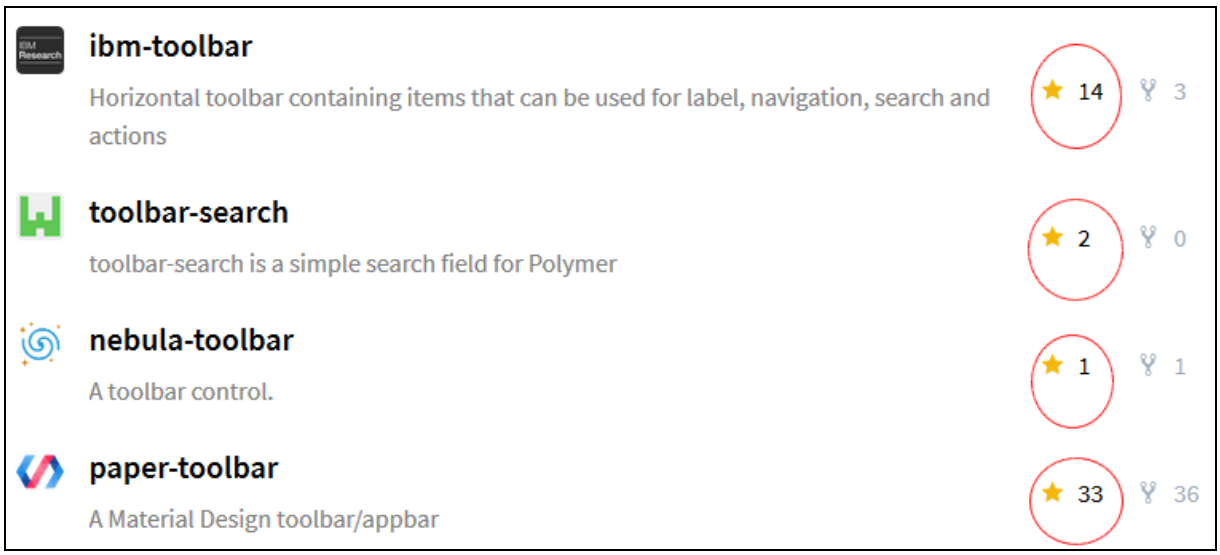

Figura 18 - *Stars* em webcomponents.org

Licença: é importante atentar-se a licença a qual o componente se encontra, para averiguar se a mesma está em conformidade com as especificações legais do projeto da aplicação web. Dentro do repositório, ao entrar na documentação do componente, possui a descrição de qual licença aquele componente está, assim como o destaque em vermelho na Figura 19 indica.

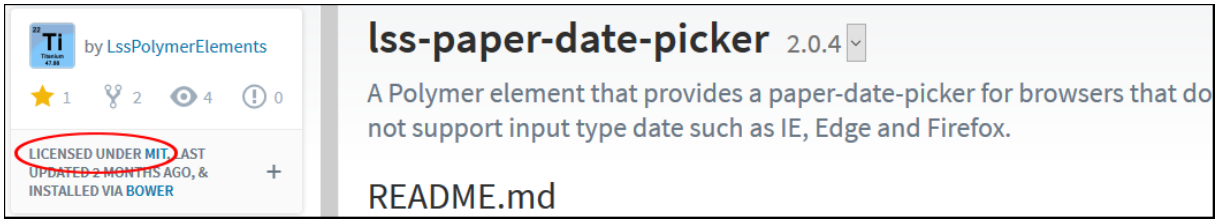

Figura 19 - Licença de um componente

Documentação: todo componente possui uma documentação, sendo o qual possui um README, que nada mais é uma descrição que contém as informações daquele componente, tais como instalar o componente, a API, e todos os possíveis subcomponentes que o compõe, demonstrações de como é o componente, a licença no qual se encontra e entre outras informações. O quão descritivo a documentação do componente é, pode ser um dos critérios de escolha, um componente pobre de documentação pode encarecer o projeto por conta da implementação e manutenção complexa daquele componente. A Figura 20 contém a documentação de um componente como exemplo, em que se encontra no repositório.

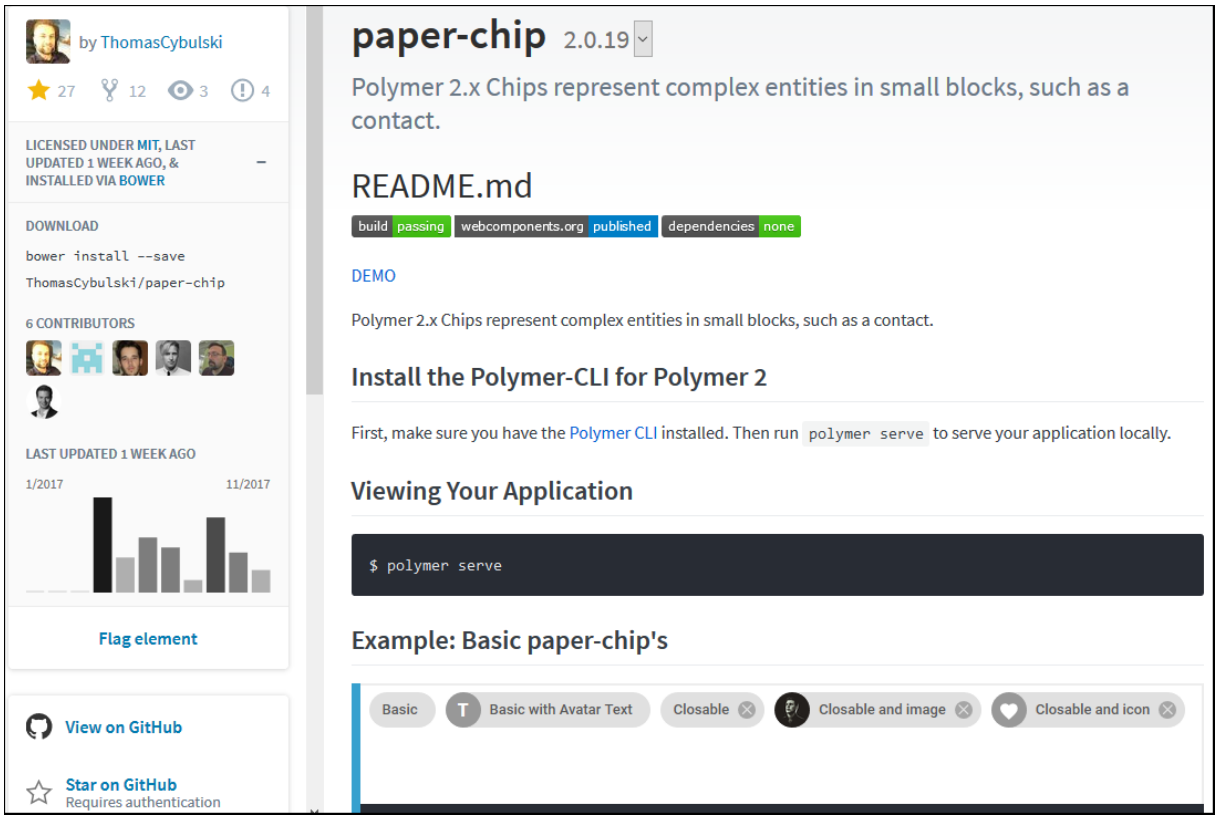

Figura 20 - Informações do componente

Issues: este indicador aponta o número de problemas no qual o componente possui e que necessita corrigir. Um valor muito elevado de *issues* pode indicar que o desenvolvimento daquele componente conteve falhas. A Figura 21 destaca em vermelho o indicador que aponta o número de *issues* de um determinado componente.

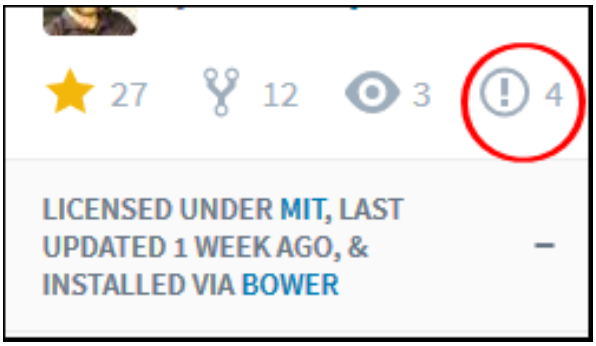

Figura 21 - *Issues* no webcomponents.org

Operações: verificar e apontar se as operações que os componentes possuem estão em conformidade com as exigências do projeto.

Design: Examinar se o visual do componente está de acordo com o que se vai construir de front-end.

4.6 Adaptação de *Web Components*

Esta etapa é a responsável por tratar de todas as alterações necessárias para que os componentes fiquem de acordo com as necessidades do projeto. A necessidade de alterações no componente varia de acordo com o tipo de componente, um formulário por exemplo que precisa ser acrescido de mais campos e ter o seus *labels* alterados.

Então, para as modificações nos *web components*, deverá ser organizado uma relação de componentes que necessitam de adaptações. Nesta relação com estes componentes, conterá:

- O componente a ser alterado;
- Que tipo de modificação necessita;
- E a descrição detalhada da modificação.

Feito isto, inicia-se as modificações no qual serão necessários conhecimentos prévios em linguagens como HTML, CSS e JavaScript, e dependendo do componente poderão ser necessários também conhecimentos em determinadas bibliotecas, por exemplo jQuery (2017). Mas primordialmente deverá ser realizado o download do componente, tal ação é realizada através de um gerenciador de dependências. Até o presente momento o gerenciador usual dentro do repositório Webcomponents.org é o Bower (2017). Segundo Overson e Strimpel (2015) a

realização de testes em *web components* é algo tanto quanto crítico pois as ferramentas que auxiliam em testes na web, ainda não são perfeitamente adequadas para o teste de *web components*. Porém, felizmente ainda há uma escolha de alto nível para o teste de unidade para *web components*, o Karma. Mas ainda há ressalvas, pois ainda existem poucos casos de vanguarda que não são muito bem suportados, mas mesmo assim continua sendo uma opção sólida, que provavelmente se expandirá até cumprir melhor seu papel no futuro.

Ao final desta etapa, todos os componentes devem estar devidamente adaptados e prontos para a Implementação. Se faz necessário que todas as modificações realizadas sejam documentadas e acondicionadas.

4.7 Implementação

Até a presente instância do PDFWC, devem ter sido elaborados diversos artefatos como documentos, modelos, componentes apropriados e entre outros. Todos estes materiais e recursos serão auxiliadores para a Implementação das interfaces gráficas da aplicação web. Nesta seção será relatado sobre a última etapa do PDFWC que é o desenvolvimento do front-end fazendo uso de *web components*.

Esta etapa fica principalmente delegada aos desenvolvedores front-end que deverão criar as páginas web que constituirão o front-end da aplicação de acordo com todos os artefatos obtidos até a presente instância do processo. Nesta etapa dará vida aos modelos gerados anteriormente. A equipe de desenvolvedores sintetizará e converterá as informações elaboradas nas etapas anteriores no front-end propriamente dito da aplicação web que será desenvolvida.

Dentro do projeto, deverão ser incluídos todos os *web components* necessários. Os que foram adaptados na etapa anterior, serão acrescidos no projeto e os quais não foram necessitados de modificações serão instalados através de um gerenciador de dependências, conforme consta em sua documentação no webcomponents.org. Feito isto, os componentes estão prontos para serem utilizados nas páginas web.

Inicia-se então a codificação e geração de *layouts* através de linguagens como HTML, CSS e JavaScript. Já devidamente instalados os *web components*, dentro do código serão feitos as importações dos componentes que pertencerão a aquela determinada página. A partir disso, poderá então utilizar os componentes através de suas *tags* customizadas. Para poder trabalhar com os *web components*,

cada um deles possui uma API, a mesma é descrita em sua documentação no Webcomponents.org.

Ao final desta etapa, deverão ser produzidas todas as interfaces que constituirão o front-end da aplicação. A Figura 22 e a Figura 23 exemplificam páginas web que devem ser construídas ao final desta etapa. A navegação deverá respeitar o projeto navegacional já elaborado. E construído dos as páginas web, as mesmas devem ser documentadas de acordo com os webcomponents que foram utilizados nela. Encerrado esta etapa, inicia-se então a integração do sistema, o qual está fora do escopo deste processo.

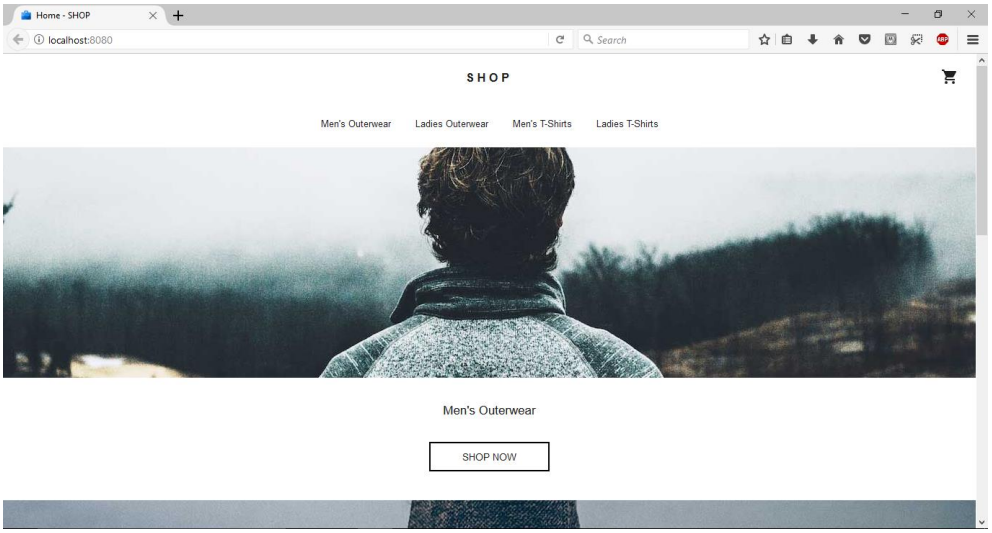

Figura 22 - Exemplo de *layout*

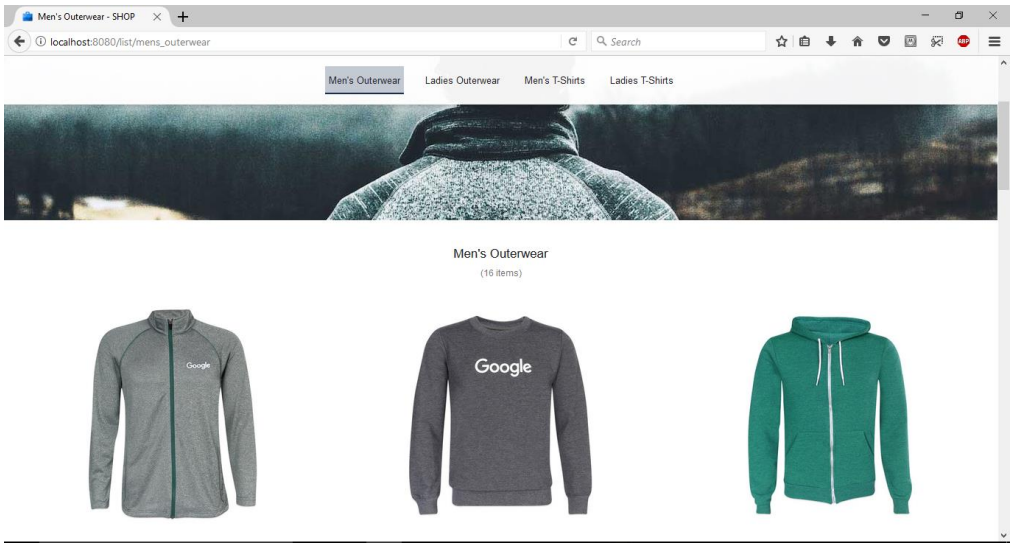

Figura 23 - Exemplo de layout 2

5. CONCLUSÕES

Web components estão na vanguarda absoluta e prometem modularizar a web que conhecemos. Tal recurso demonstrou-se promissor e está sendo altamente investido por grandes companhias da web. Desta forma, o que se pretendeu com este trabalho foi elaborar e mostrar um processo de desenvolvimento front-end no qual fizesse uso desta nova tendência de maneira metódica. Não se tem conhecimento de abordagens semelhantes nas quais disciplinem o desenvolvimento front-end estritamente. Nesta Seção serão apontados discussões, resultados e trabalhos futuros.

A finalidade deste processo é nortear a equipe de front-end através de suas sete etapas. Incrementada da modularização que os *web components* nos permitem, escassear o uso de *frameworks* que são relativamente limitados e implicitamente volumosos. A primeira etapa deste processo é genérica, como a de qualquer outra abordagem de desenvolvimento de software, apenas focando em pontos importantes para as interfaces. A segunda e terceira são indispensáveis para se construir interfaces e as demais etapas são norteio para o uso de *web components*.

5.1 Discussões e Resultados

De acordo com a pesquisa efetuada neste trabalho, a concepção é de que todas as necessidades essenciais encontradas ao se trabalhar com o desenvolvimento front-end são supridas neste processo. A ressalva é de que além das concepções aqui descritas, ainda não há, até o presente momento, resultados obtidos que provem efetivamente tal relato. Isso se dá por conta de haver limitações para o teste e validação do PDFWC neste momento. Tal adversidade será um dos trabalhos futuros, no qual será descrita a seguir.

5.2 Trabalhos Futuros

A formulação deste trabalho foi embasada em um *benchmarking* entre as abordagens de desenvolvimento nas quais possuíam maior afinidade com o que se procurava propor e lidar neste trabalho. E ao formular o processo, surgiu a necessidade de elaboração de outros trabalhos que completem e/ou que deem continuidade a este trabalho. Os trabalhos futuros são:

- **Validação do processo**: Arquitetar uma validação para este processo ainda segue como um dos trabalhos futuros do PDFWC. Para tal seria

necessário recursos humanos, tais como equipes de desenvolvimento para realizar experiências e obter um *feedback* concreto sobre o PDFWC.

- **Desenvolvimento de *Web Components***: A elaboração de um subprocesso que promovesse a construção de diversos *web components* faltantes, aumentando assim ainda mais o número de componentes a disposição do desenvolvimento front-end. Overson e Strimpel (2015) descrevem o desenvolvimento de um *web component*, mas a intenção para este trabalho futuro seria a de disciplinar este desenvolvimento com o PDFWC. Uma proposta para a elaboração deste trabalho futuro poderia ser a de abstrair e embasar em metodologias ágeis, unir o desenvolvimento de *web components* descritos por Overson e Strimpel (2015) com alguns padrões de UI.

REFERÊNCIAS

BROWN, Alan W.; SHORT, Keith. On Components and Objects: The Foundations of Component-Based Development. 1997. Disponível em: <http://ieeexplore.ieee.org/document/599921/>. Acesso em: 10 jun. 2017.

BOWER. **Bower.** 2017. Disponível em: <https://bower.io/>. Acesso em: 16 out. 2017.

CHEESMAN, John; DANIELS, John. **UML Components: A Simple Process for Specifying Component-based Software.** Disponível em: <https://books.google.com.br/books/about/UML_components.html?id=jq9QAAAAMAAJ&hl=pt-BR>. Acesso em: 10 set. 2017.

DABHADE, Megha; SURYAWANSHI, Shivam; MANJULA, R. A Systematic Review of Software Reuse using Domain Engineering Paradigms. 2016. Disponível em: <http://ieeexplore.ieee.org/document/7916646/>. Acesso em: 10 jun. 2017.

ERIKSSON, Hans-erik; PENKER, Magnus. **UML toolkit.** 1998. Disponível em: <https://books.google.com.br/books?id=a8lXAAAAYAAJ&hl=pt-BR&source=gbs_similarbooks>. Acesso em: 10 set. 2017.

GIMENES, Itana Maria de Souza; HUZITA, Elisa Hatsue Moriya. Desenvolvimento baseado em componentes: conceitos e técnicas. 2005. Disponível em: <https://books.google.com.br/books/about/Desenvolvimento_baseado_em_componentes.html?id=2_jvXwAACAAJ&redir_esc=y>. Acesso em: 28 maio 2017.

GINIGE, Athula; MURUGESAN, San. **Web Engineering: An Introduction.** 2001. Disponível em: <https://www.computer.org/csdl/mags/mu/2001/01/u1014.pdf&usg=AFQjCNHqIm6AXSt3A5Mvh1U_dak_MVpLMQ>. Acesso em: 10 set. 2017.

GOOGLE. **Google.** 2017. Disponível em: <https://www.google.com.br/>. Acesso em: 15 nov. 2017.

JQUERY. **JQuery.** 2017. Disponível em: <https://jquery.com/>. Acesso em: 16 out. 2017.

KRAUSE, Jörg. Introducing Web Development. 2016. Disponível em: <https://books.google.com.br/books/about/Introducing_Web_Development.html?id=EKAcvgAACAAJ&redir_esc=y>. Acesso em: 18 jun. 2017.

KOCH, Nora et al. **Extending UML to Model Navigation and Presentation in Web Applications.** 2000. Disponível em: <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwjCwPKE_LTWAhVJUJAKHRTBDdYQFggpMAA&url=https://pdfs.semanticscholar.org/b5c0/a3adf887d2828d114b7c625b8e432c2576ba.pdf&usg=AFQjCNFfudZkdIHfFIiHdCpjTgHLtEqD4w>. Acesso em: 10 set. 2017.

KOCH, Nora; KRAUS, Andreas; HENNICKER, Rolf. **The Authoring Process of the UML-based Web Engineering Approach.** 2001. Disponível em: <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjyzdS3_LTWAhVKf5AKHRTODVMQFggnMAA&url=http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.9417&rep=rep1&type=pdf&usg=AFQjCNEarDq5TekZuZqQLlAdxooIQQLGNQ>. Acesso em: 10 set. 2017.

KOCH, Nora; KRAUS, Andreas. **The Expressive Power of UML-based Web Engineering.** 2002. Disponível em: <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwimgJz3_LTWAhVIgpAKHZvbCRMQFggqMAA&url=http://www.dsic.upv.es/~west/iwwost02/papers/koch.pdf&usg=AFQjCNEZttCNq48dJP6FrGs6SdNrpbbjNw>. Acesso em: 10 set. 2017.

MIRANDA, Bernardo Faria de; LAGES, Diego Delgado. Uma comparação de dois métodos de desenvolvimento de software baseado em componentes: Catalysis e UML Components. 2003. Disponível em: <http://reuse.cos.ufrj.br/files/publicacoes/graduacao/PF_MirandaLages.pdf>. Acesso em: 10 jun. 2017.

NIELSEN, Jakob. **Usability Inspection Methods.** 1994. Disponível em: <https://dl.acm.org/citation.cfm?id=260531&CFID=1006400217&CFTOKEN=14234561>. Acesso em: 16 out. 2017.

NUNES, Demetrius Arraes. **HyperD - um Framework e Ambiente de Desenvolvimento dirigido por Ontologias para Aplicações Hipermídia.** 2005. Disponível em: <https://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=7617@1>. Acesso em: 10 set. 2017.

OVERSON, Jarrod; STRIMPEL, Jason. Developing Web Components. 2015. Disponível em: <https://books.google.com.br/books?id=7mvIBgAAQBAJ&printsec=frontcover&dq=developing+web+components&hl=pt-BR&sa=X&ved=0ahUKEwiG8qrVsNzUAhXHxpAKHX-3Bf4Q6AEIKjAA#v=onepage&q=developing web components&f=false>. Acesso em: 26 jul. 2017.

PAULA FILHO, Wilson de Padua. Engenharia de software: fundamentos, métodos e padrões. 2000. Disponível em: <https://books.google.com.br/books/about/Engenharia_de_software.html?id=7c_vAAAACAAJ&redir_esc=y>. Acesso em: 10 maio 2017.

PRESSMAN, Roger. Engenharia de Software - 7.ed. 2011. Disponível em: <https://books.google.com.br/books?id=y0rH9wuXe68C&dq=Pressman+engenharia&hl=pt-BR&sa=X&ved=0ahUKEwiQnOXNsdzUAhXGHpAKHav2B5gQ6AEILTAB>. Acesso em: 10 maio 2017.

PRESSMAN, Roger S.. **Software Engineering: A Practitioner's Approach.** 2005. Disponível em: <https://books.google.com.br/books?id=bL7QZHtWvaUC&hl=pt-BR&source=gbs_book_other_versions>. Acesso em: 10 set. 2017.

SCHWABE, Daniel; ROSSI, Gustavo; BARBOSA, Simone D.j.. **Systematic hypermedia application design with OOHDM.** 1996. Disponível em: <http://dl.acm.org/citation.cfm?id=234840>. Acesso em: 10 set. 2017.

SCHWARTZMAN, Michel Lent. **Projetando um site em etapas.** 1998. Disponível em: <https://www.uol.com.br/publish/37/webdesign/dadi_37.html>. Acesso em: 10 set. 2017.

SOMMERVILLE, Ian. Engenharia de software. 2011. Disponível em: <https://books.google.com.br/books?id=H4u5ygAACAAJ>. Acesso em: 10 maio 2017.

SOUZA, Vitor Estevão Silva. **FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web.** 2007. Disponível em: <https://nemo.inf.ufes.br/wp-content/papercite-data/pdf/frameweb__um_metodo_baseado_em_frameworks_para_o_projeto_de_sistemas_de_informacao_web_2007.pdf&usg=AFQjCNG5A50iN_ESSwIiPXCklJAXQH02XA>. Acesso em: 10 set. 2017.

TANENBAUM, Andrew S.. Redes de computadores. 2003. Disponível em: <https://books.google.com.br/books?id=0tjB8FbV590C&printsec=frontcover&dq=tanenbaum+redes+4+ed&hl=pt-BR&sa=X&ved=0ahUKEwi_2Kn3stzUAhVKjpAKHT9PAiMQ6AEIJzAA#v=onepage&q=tanenbaum redes 4 ed&f=false>. Acesso em: 22 jun. 2017.

TOXBOE, Anders. **UI patterns.** 2017. Disponível em: <http://ui-patterns.com/>. Acesso em: 16 out. 2017.

VICENTINI, Luiz Atilio; MILECK, Luciângela Slemer. **DESENVOLVIMENTO DE SITES NA WEB EM UNIDADES DE INFORMAÇÃO: METODOLOGIAS, PADRÕES E FERRAMENTAS.** 2000. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?down=3&usg=AFQjCNEq1lbu1pwKsa1Qjhsmi6Vchh-ifA>. Acesso em: 10 set. 2017.

W3C. **Padrões.** 2017. Disponível em: <http://www.w3c.br/Padroes/>. Acesso em: 16 out. 2017.

WEBCOMPONENTS.ORG. **Web Components.** 2017. Disponível em: <https://www.webcomponents.org/>. Acesso em: 16 out. 2017.