

UNIVERSIDADE DE CAXIAS DO SUL
Centro de Computação e Tecnologia da Informação
Curso de Sistemas de Informação

Diego Troitiño

Aplicações Ricas na *Web*

Caxias do Sul

2012

Diego Troitiño

Aplicações Ricas na *Web*

Trabalho de Conclusão de Curso
para obtenção do Grau de
Bacharel em Sistemas de
Informação da Universidade de
Caxias do Sul.

Alexandre Erasmo Krohn Nascimento
Orientador

Caxias do Sul

2012

AGRADECIMENTOS

Agradeço a todos envolvidos direta e indiretamente, que me apoiaram e incentivaram tanto nos estudos quanto na vida pessoal. Especialmente para minha família, pela criação, aos meus amigos e namorada pela compreensão, ao orientador pela dedicação e ao café, pelas horas a mais acordado.

RESUMO

Este trabalho apresenta um comparativo entre ferramentas para desenvolvimento de aplicações ricas na *web*. Nele são abordadas características pertinentes a aplicações ricas. Juntamente com as características, são descritas tecnologias envolvidas no desenvolvimento de aplicações para a internet. É apresentado um modelo de comparação das ferramentas de desenvolvimento das aplicações ricas baseado no Método Analítico Hierárquico (MAH). Este modelo visa avaliar as seguintes ferramentas e tecnologias: *HTML5*, *Flex*, *Silverlight* e *JavaFX*. Este comparativo possui o intuito de analisar o uso efetivo de cada tecnologia no desenvolvimento de aplicações ricas, apontando vantagens e desvantagens do uso de cada uma.

Palavras-chaves: Método Analítico Hierárquico, *HTML5*, *Flex*, *Silverlight*, *JavaFX*, Aplicações Ricas, Aplicação de Referência.

ABSTRACT

This study presents a comparison between tools for rich internet applications development. It covers relevant rich applications features. Along with features, technologies involved in developing applications for internet are described. The study present a comparison model between tools of rich applications development based on Analytic Hierarchy Process (AHP). This model aims to evaluate the following tools and technologies: HTML5, Flex, Silverlight, and JavaFX. The objective of this comparison is to analyze the effective use of each technology in development of rich applications, pointing out advantages and disadvantages of each one.

Keywords: Analytic Hierarchy Process, HTML5, Flex, Silverlight, JavaFX, Rich Applications, Reference Application.

LISTA DE FIGURAS

Figura 1 - Crescimento da internet medido pelo número de computadores conectados a cada ano, de 1981 a 2003.	6
Figura 2 - Representação temporal do modelo clássico de uma aplicação <i>web</i>	8
Figura 3 - Representação temporal do modelo <i>Ajax</i> de uma aplicação <i>web</i>	9
Figura 4 - O modelo tradicional para aplicações <i>web</i> (esquerda) em comparação ao modelo <i>AJAX</i> (direita).	10
Figura 5 - Mercado das ferramentas <i>RIA</i>	13
Figura 6 - Exemplo de elementos do <i>Forms 2.0</i>	19
Figura 7 - Exemplo de <i>inputs</i> em <i>Forms 2.0</i> no navegador <i>Chrome</i> versão 15.0.874.121.	19
Figura 8 - Definição da <i>tag</i> de vídeo no <i>HTML5</i>	22
Figura 9 - Exibição do player de vídeo do <i>HTML5</i>	22
Figura 10 - Definição da área a ser exibida a imagem em <i>canvas</i>	23
Figura 11 - Código <i>JavaScript</i> exemplificando o uso do <i>canvas</i>	24
Figura 12 - Renderização de <i>canvas</i> obtida pelos comandos exibidos na figura 11.	24
Figura 13 - Exemplo de comandos em <i>SVG</i> no <i>HTML5</i>	25
Figura 14 - Renderização de <i>SVG</i> obtida pelos comandos exibidos na Figura 13.	26
Figura 15 - Exemplo de novos seletores do <i>CSS3</i>	28
Figura 16 - Aplicação dos estilos descritos na figura 15.	28
Figura 17 - Exemplo da aplicação de <i>media queries</i>	29
Figura 18 - Exemplo de uso do <i>@font-face</i>	29
Figura 19 - Exemplo do uso do estilo <i>transform</i> e <i>transition</i>	30
Figura 20 - Exemplo de hierarquia com mais de um nível de critérios (Corso e Löbler, 2010).	33
Figura 21- Exemplo de hierarquia somente com um nível de critérios.	33
Figura 22 - Hierarquia dos critérios de avaliação do método <i>AHP</i>	41
Figura 23 - Tela de informações gerais do pedido.	44
Figura 24 - Tela de itens do pedido.	45
Figura 25 - Navegação entre as telas do pedido.	46
Figura 26 - Filtro dos produtos disponíveis.	48
Figura 27 - Adição dos itens no pedido.	50
Figura 28 - Validação da quantidade solicitada no item.	51
Figura 29 - Validação da obrigatoriedade dos campos.	52
Figura 30 - Trajeto até o distribuidor mais próximo.	54
Figura 31 - Gráfico da representatividade de clientes.	56
Figura 32 - Transição entre as telas do pedido.	57
Figura 33 - Acesso da <i>webcam</i> para obter descontos no pedido através de um <i>Qrcode</i> promocional.	60
Figura 34 - Controles dinâmicos na listagem de produtos do <i>Flex</i>	71
Figura 35 - Implementação dos itens do pedido - <i>C2</i> em <i>Silverlight</i>	74
Figura 36 - Implementação dos itens do pedido - <i>C2</i> em <i>HTML5</i>	77
Figura 37 - Efeitos em <i>CSS</i> para a característica de transição de telas (<i>C10</i>).	79
Figura 38 - Gráficos estatísticos utilizando <i>JavaFX</i>	82
Figura 39 - Código para realizar a transição entre as telas em <i>JavaFX</i>	82

Figura 40 - Resultados obtidos em relação ao critério de performance.	115
Figura 41 - Resultados obtidos em relação ao critério de compatibilidade.....	115
Figura 42 - Resultados obtidos em relação ao critério de facilidade de desenvolvimento.....	116
Figura 43 - Resultados obtidos em relação ao critério de semelhança com aplicações <i>desktop</i>	116
Figura 44 - Resultados gerais da avaliação.	117
Figura 45 - Desempenho de cada tecnologia para os critérios avaliados.	119
Figura 46 - Código fonte das informações gerais do pedido em <i>Flex</i>	127
Figura 47 - Resultado da implementação de informações gerais do pedido em <i>Flex</i>	127
Figura 48 - Código fonte das informações gerais do pedido em <i>Silverlight</i>	128
Figura 49 - Resultado da implementação de informações gerais do pedido em <i>Silverlight</i> ...	129
Figura 50 - Código fonte das informações gerais do pedido em HTML5 Pate 1.....	130
Figura 51 - Código fonte das informações gerais do pedido em HTML5 Pate 2.....	131
Figura 52 - Resultado da implementação de informações gerais do pedido em HTML5.....	131
Figura 53 - Código fonte das informações gerais do pedido em <i>JavaFX</i>	132
Figura 54 - Resultado da implementação de informações gerais do pedido em <i>JavaFX</i>	133
Figura 55 - Código fonte dos itens do pedido em <i>Flex</i>	134
Figura 56 - Código <i>Action Script</i> para exibir os produtos disponíveis a venda.	135
Figura 57 - Resultado da implementação de itens do pedido em <i>Flex</i>	136
Figura 58 - Código fonte dos itens do pedido em <i>Silverlight</i> parte 1.....	137
Figura 59 - Código fonte dos itens do pedido em <i>Silverlight</i> parte 2.....	138
Figura 60 - Resultado da implementação de itens do pedido em <i>Silverlight</i>	139
Figura 61 - Código fonte dos itens do pedido em HTML.	140
Figura 62 - Código para montar a listagem de produtos em <i>JavaScript</i>	141
Figura 63 - Resultado da implementação de itens do pedido em HTML5.....	142
Figura 64 - Código fonte dos itens do pedido em <i>JavaFX</i>	143
Figura 65 - Código para montar a listagem de produtos em <i>Java</i>	144
Figura 66 - Resultado da implementação de itens do pedido em <i>JavaFX</i>	145

LISTA DE TABELAS

Tabela 1 - Escala Fundamental de Saaty (Silva e Belderrain, 2005).....	34
Tabela 2 - Matriz de comparação (Vargas, 2010).....	35
Tabela 3 - Normalização dos dados de uma matriz de comparação.....	35
Tabela 4 - Média dos critérios.....	36
Tabela 5 - Comparativo dos critérios de avaliação.....	63
Tabela 6 - Listagem das características da aplicação de referência.....	64
Tabela 7 - Relevância de cada característica para o critério de performance.....	65
Tabela 8 - Relevância de cada característica para o critério de compatibilidade.....	66
Tabela 9 - Relevância de cada característica para o critério de facilidade de desenvolvimento.....	68
Tabela 10 - Relevância de cada característica para o critério de semelhança com aplicações <i>desktop</i>	69
Tabela 11- Exemplo hipotético de um resultado de comparação do critério de facilidade de desenvolvimento.....	84
Tabela 12 - Medições da performance das informações gerais do pedido.....	85
Tabela 13 - Resultados do critério de performance para a característica C1.....	85
Tabela 14 - Resultados do critério de compatibilidade para a característica C1.....	86
Tabela 15 - Resultados do critério de facilidade de desenvolvimento da característica C1.....	87
Tabela 16 - Resultados do critério de semelhança com aplicações <i>desktop</i> da característica C1.....	87
Tabela 17 - Medições da performance dos itens do pedido.....	88
Tabela 18 - Resultados do critério de performance para a característica C2.....	89
Tabela 19 - Resultados do critério de compatibilidade para a característica C2.....	89
Tabela 20 - Resultados do critério de facilidade de desenvolvimento da característica C2.....	90
Tabela 21 - Resultados do critério de semelhança com aplicações <i>desktop</i> da característica C2.....	91
Tabela 22 - Medições da performance da busca de itens no pedido.....	92
Tabela 23 - Resultados do critério de performance para a característica C4.....	92
Tabela 24 - Resultados do critério de compatibilidade para a característica C4.....	93
Tabela 25 - Medições da performance da adição de itens no pedido.....	94
Tabela 26 - Resultados do critério de performance para a característica C5.....	94
Tabela 27 - Resultados do critério de compatibilidade para a característica C5.....	95
Tabela 28 - Resultados do critério de facilidade de desenvolvimento da característica C5.....	95
Tabela 29 - Resultados do critério de semelhança com aplicações <i>desktop</i> da característica C5.....	96
Tabela 30 - Medições da performance de alterar a quantidade solicitada dos itens.....	97
Tabela 31 - Resultados do critério de performance para a característica C6.....	97
Tabela 32 - Resultados do critério de compatibilidade para a característica C6.....	98
Tabela 33 - Resultados do critério de facilidade de desenvolvimento da característica C6.....	98
Tabela 34 - Resultados do critério de semelhança com aplicações <i>desktop</i> da característica C6.....	99
Tabela 35 - Medições da performance das validações das informações gerais.....	100
Tabela 36 - Resultados do critério de performance para a característica C7.....	100

Tabela 37 - Resultados do critério de compatibilidade para a característica C7.	101
Tabela 38 - Resultados do critério de facilidade de desenvolvimento da característica C7. ..	101
Tabela 39 - Resultados do critério de semelhança com aplicações <i>desktop</i> da característica C7.	102
Tabela 40 - Medições da performance na logística na emissão do pedido.	103
Tabela 41 - Resultados do critério de performance para a característica C8.	103
Tabela 42 - Resultados do critério de compatibilidade para a característica C8.	104
Tabela 43 - Resultados do critério de facilidade de desenvolvimento da característica C8. ..	105
Tabela 44 - Resultados do critério de semelhança com aplicações <i>desktop</i> da característica C8.	105
Tabela 45 - Medições da performance na logística na emissão do pedido.	106
Tabela 46 - Resultados do critério de performance para a característica C9.	107
Tabela 47 - Resultados do critério de compatibilidade para a característica C9.	107
Tabela 48 - Resultados do critério de facilidade de desenvolvimento da característica C9. ..	108
Tabela 49 - Resultados do critério de semelhança com aplicações <i>desktop</i> da característica C9.	108
Tabela 50 - Medições da performance da transição de telas no pedido.	109
Tabela 51 - Resultados do critério de performance para a característica C10.	110
Tabela 52 - Resultados do critério de compatibilidade para a característica C10.	110
Tabela 53 - Resultados do critério de facilidade de desenvolvimento da característica C10.	111
Tabela 54 - Resultados do critério de semelhança com aplicações <i>desktop</i> da característica C10.	111
Tabela 55 - Medições da performance do uso da <i>webcam</i> para obter promoções no pedido.	112
Tabela 56 - Resultados do critério de performance para a característica C12.	112
Tabela 57 - Resultados do critério de compatibilidade para a característica C12.	113
Tabela 58 - Resultados do critério de facilidade de desenvolvimento da característica C12.	113
Tabela 59 - Resultados do critério de semelhança com aplicações <i>desktop</i> da característica C12.	114
Tabela 60 - Resultados finais das tecnologias para com os critérios avaliados.	118

LISTA DE ABREVIATURAS E SIGLAS

Sigla	Significado em Português	Significado em Inglês
AHP	Processo hierárquico analítico	<i>Analytic Hierarchy Process</i>
AJAX	JavaScript e XML assíncronos	<i>Asynchronous JavaScript and XML</i>
API	Interface de Programação de Aplicativos	<i>Application Programming Interface</i>
CERN		<i>European Council for Nuclear Research</i>
CORS	Compartilhamento de recursos de origem cruzada	<i>Cross-Origin Resource Sharing</i>
CPU	Unidade central de processamento	<i>Central Processing Unit</i>
CSS	Folhas de estilo em cascata	<i>Cascading Style Sheets</i>
DNS	Nome do domínio do sistema	<i>Domain Name System</i>
DOM	Modelo de objeto de documentos	<i>Document Object Model</i>
DRM	Gerenciamento de direitos digitais	<i>Digital Rights Management</i>
ECMA	Associação europeia de fabricantes de computadores	<i>European Computer Manufacturers Association</i>
EDGE		<i>Enhanced Data rates for GSM Evolution</i>
FXML		
GPS	Sistema de posicionamento global	<i>Global Position System</i>
HTML	Linguagem de marcação de hipertexto	<i>Hypertext Markup Language</i>
HTTP	Protocolo de transferência de Hipertexto	<i>Hypertext Transfer Protocol</i>
IBM		<i>International Business Machines</i>
IP	Protocolo de internet	<i>Internet Protocol</i>
JSON	Notação de objetos em JavaScript	<i>JavaScript Object Notation</i>
MAC	Controle de Acesso ao Meio	<i>Media Access Control</i>

MXML	Linguagem de marcação extensível da Macromedia	<i>Macromedia Extensible Markup Language</i>
RFID	Identificação de rádio frequência	<i>Radio Frequency Identification</i>
RIA	Aplicações de internet ricas	<i>Rich Internet Application</i>
SVG	Gráficos vetoriais escaláveis	<i>Scalable Vector Graphics</i>
SWF		<i>Shockwave Flash</i>
TCP	Protocolo de controle de transmissão	<i>Transmission Control Protocol</i>
W3C		<i>World Wide Web Consortium</i>
WHATWG	Grupo de trabalho de tecnologias web aplicações de hipertexto	<i>Web Hypertext Application Technology Working Group</i>
WPF/E		<i>Windows Presentation Foundation / Everywhere</i>
WWW		<i>World Wide Web</i>
XAML	Linguagem de marcação de aplicação extensível	<i>Extensible Application Markup Language</i>
XML	Linguagem de marcação extensível	<i>Extensible Markup Language</i>

SUMÁRIO

Agradecimentos	III
Resumo	IV
Abstract.....	V
Lista de Figuras	VI
Lista de Tabelas	VIII
Lista de Abreviaturas e Siglas	X
Sumário.....	XII
1 Introdução	1
1.1 Objetivos	2
1.2 Metodologia	2
1.3 Estrutura do Texto	3
2 Referencial Teórico	4
2.1 A Internet	4
2.2 <i>Web 2.0</i>	7
2.3 Aplicações ricas	8
2.4 Tecnologias	10
2.4.1 Ferramentas RIA	12
2.4.1.1 <i>Flex</i>	14
2.4.1.2 <i>Silverlight</i>	15
2.4.1.3 <i>JavaFX</i>	16
2.4.2 HTML5	17
2.4.2.1 <i>Web Semântica</i>	18
2.4.2.2 <i>Offline e Storage</i>	19
2.4.2.3 Dispositivos Móveis.....	20
2.4.2.4 Conectividade.....	21
2.4.2.5 Multimídia.....	21
2.4.2.6 Gráficos e Efeitos	23
2.4.2.7 Performance e Integração.....	26
2.4.2.8 CSS3.....	27
2.5 Considerações	30
3 Método de Avaliação	31
3.1 <i>Analytic Hierarchic Process</i>	32
3.2 Considerações	36
4 Modelo de Avaliação	38
4.1 Alternativas	38
4.2 Critérios de Avaliação	39
4.2.1 Performance.....	39
4.2.2 Compatibilidade	39
4.2.3 Facilidade de Desenvolvimento.....	40
4.2.4 Semelhança na aparência e usabilidade de aplicações <i>desktop</i>	40
4.3 Hierarquia	40
4.4 Considerações	41
5 Aplicação de Referência	43

5.1	Usabilidade	43
5.1.1	Característica 1 – Informações Gerais do Pedido.....	44
5.1.2	Característica 2 – Itens do Pedido	45
5.1.3	Característica 3 – Navegação do Pedido	46
5.2	Interatividade	47
5.2.1	Característica 4 – Busca de Itens no Pedido.....	48
5.2.2	Característica 5 – Adição de Itens no Pedido	49
5.2.3	Característica 6 – Quantidade Solicitada dos Itens	50
5.2.4	Característica 7 – Validações das Informações Gerais do Pedido	51
5.3	Geolocalização	53
5.3.1	Característica 8 – Logística na Emissão do Pedido.....	53
5.4	Gráficos e Animações	55
5.4.1	Característica 9 – Gráficos Estatísticos	55
5.4.2	Característica 10 – Transição de Telas no Pedido	56
5.5	Aplicações <i>offline</i> e Armazenamento Local	58
5.5.1	Característica 11 – Suporte a Armazenamento Local.....	58
5.6	Interação com Hardware Local.....	59
5.6.1	Característica 12 – Uso de <i>Webcam</i> para Obter Promoções no Pedido.....	59
5.7	Aplicações Mobile	60
5.7.1	Característica 13 – Suporte a Dispositivos Móveis.....	61
5.8	Considerações	62
6	Avaliação.....	63
6.1	Relevância das características de avaliação.....	64
6.1.1	Critério de performance.....	65
6.1.2	Critério de compatibilidade	66
6.1.3	Critério de facilidade de desenvolvimento	67
6.1.4	Critério de semelhança com aplicações <i>desktop</i>	68
6.2	Implementações da aplicação de referência.....	69
6.2.1	Implementação utilizando <i>Adobe FLEX</i>	70
6.2.2	Implementação utilizando <i>Microsoft Silverlight</i>	73
6.2.3	Implementação utilizando HTML5	76
6.2.4	Implementação utilizando <i>Oracle JavaFX</i>	79
6.3	Avaliação das Características.....	83
6.3.1	C1 - Informações gerais do pedido.....	84
6.3.2	C2 - Itens do pedido	88
6.3.3	C3 - Navegação do pedido.....	91
6.3.4	C4 - Busca de itens no pedido	91
6.3.5	C5 - Adição de itens no pedido.....	93
6.3.6	C6 - Quantidade solicitada dos itens	96
6.3.7	C7 - Validações das informações gerais	99
6.3.8	C8 - Logística na emissão do pedido.....	102
6.3.9	C9 - Gráficos estatísticos.....	106
6.3.10	C10 - Transição de telas no pedido	109
6.3.11	C12 - Uso de <i>webcam</i> para obter promoções no pedido.....	112
6.4	Resultados obtidos	114
7	Considerações Finais.....	120
7.1	Trabalhos Futuros	121
8	Referências.....	122

1 INTRODUÇÃO

Segundo Comer (2007, p. 38) a internet teve seu início na década de 70, tendo o auge do seu crescimento em 1998, quando a quantidade de computadores, conectados à internet, ultrapassou a média de um por segundo. Antes disso, em 1993, houve a padronização da *web* e o surgimento da linguagem de marcação HTML (*Hypertext Markup Language*) (David, 2010). No início, a necessidade era apenas manipular páginas de texto puro (Berners-Lee, 1999). Com a expansão da internet, houve evoluções sobre o padrão HTML.

Em 1995 foi lançada a versão 2.0 e a partir de 1996 as especificações do HTML estão de responsabilidade da W3C (*World Wide Web Consortium*) (Raggett, 1997). Novas funcionalidades surgiram com a disseminação do HTML na rede de computadores. As principais foram a inclusão de imagens nas páginas e suporte a folhas de estilos. Com o tempo várias mudanças ocorreram nas aplicações *web*, a interatividade da *web* aumentou com o surgimento de tecnologias sofisticadas de interface na *web*, não concorrendo diretamente, mas sim funcionando como um complemento ao HTML.

Tecnologias como *Flash* (Adobe) e posteriormente *Silverlight* (Microsoft) e *JavaFX* (Oracle) surgiram para mudar a aparência das páginas na internet. Com elas é possível desenvolver aplicações *web* com interatividade similar ao do *desktop* (Deitel, 2009, p. 5). Aplicações ricas possuem semelhança na aparência, comportamento e usabilidade das aplicações *desktop*. Com isso são criadas aplicações mais interativas e intuitivas, tornando-se assim menor a necessidade de treinamentos ao usuário final e simplificando seu uso. Outras características de aplicações ricas citadas por Allaire (2002, p. 2) são o nível de integração, uso de conteúdo *offline*, performance e suporte a múltiplas plataformas e dispositivos.

Tendo em vista a evolução das aplicações *web*, em 2007 iniciou-se a especificação do HTML5. A meta era possibilitar o desenvolvimento de aplicações ricas diretamente com HTML (David, 2010 p. xiii). Atualmente as especificações do HTML5 não estão concluídas, porém a maioria dos navegadores no mercado suportam as funcionalidades já documentadas. Desde o surgimento do HTML5 muitas pessoas apostam que esta nova versão veio para revolucionar a *web*. Houve declarações polêmicas, como a do CEO da *Apple*, Steve Jobs (2010), que mencionou que com o HTML5, o *Flash* não seria mais necessário.

1.1 OBJETIVOS

Comparar as tecnologias *HTML5*, *Flash*, *Silverlight* e *JavaFX*, verificando o ganho real do uso de cada uma no que diz respeito ao desenvolvimento de aplicações corporativas ricas.

Para alcançar o objetivo geral proposto, serão levantadas e avaliadas algumas características das aplicações de negócio. Especialmente no tocante à interfaces gráficas *web*. As seguintes áreas serão focadas para o levantamento das características:

- Usabilidade: análise de elementos que facilitam a interação com as aplicações *web* incluindo características de dispositivos móveis;
- Portabilidade: padronização das tecnologias em relação aos diversos navegadores existentes focando nas aplicações clientes, além da compatibilidade com dispositivos móveis;
- Performance: capacidade de renderizar aplicações em navegadores distintos de forma mais rápida;
- Escalabilidade: facilidade de expansão das aplicações remotamente de forma escalonar;
- Custo: relação entre os pontos positivos encontrados com o custo de licença e manutenção das mesmas;

Para mensurarmos as vantagens de cada tecnologia deverá haver um método de comparação. O mesmo avaliará as características citadas anteriormente.

1.2 METODOLOGIA

Para o desenvolvimento deste trabalho, serão estudadas e descritas as seguintes tecnologias: *HTML5*, *Flash*, *Silverlight*, *JavaFX*. Outras tecnologias complementares às mesmas terão suas devidas explicações teóricas conforme a necessidade. Com isso espera-se que haja uma compreensão geral dos assuntos envolvidos neste trabalho.

O Método Analítico Hierárquico (MAH) será abordado a fim de compreender o conceito de sua aplicação. Com base neste, será definido um modelo de avaliação a ser aplicado às quatro tecnologias propostas. Tomando como escopo as características relevantes às aplicações ricas na *web*, será especificada uma aplicação de referência abrangendo-as. Ou seja, nesta aplicação de referência, serão implementadas funcionalidades que irão demonstrar as características das aplicações ricas apontadas pelo referencial bibliográfico. Serão

desenvolvidas apenas as funcionalidades que forem pertinentes à problemática em questão. A finalidade é comprovar na prática as diferenças entre as tecnologias.

Nos objetivos específicos que não forem passíveis de implementação, serão realizadas pesquisas das características pertinentes aos mesmos. Cada aspecto a ser comparado, com desenvolvimento ou não, possuirá tabelas comparativas. Nelas serão discriminados os resultados dos testes e/ou pesquisas bibliográficas. Também será detalhado o cenário ao qual foram submetidos os testes, bem como as versões das tecnologias envolvidas e os esforços necessários para alcançar os objetivos propostos.

1.3 ESTRUTURA DO TEXTO

No Capítulo 2 é apresentada uma contextualização das aplicações ricas, relatando seu histórico e as tecnologias envolvidas. No Capítulo 3 é abordado o Método Analítico Hierárquico que serve de base para a definição do modelo de avaliação, apresentado no Capítulo 4, aplicado às tecnologias estudadas. No Capítulo 5, uma aplicação de referência é especificada para ser implementada a fim de aplicar o modelo de avaliação proposto. A avaliação realizada está descrita no Capítulo 6, neste são apresentados os resultados obtidos na avaliação efetuada. No capítulo 7, são apresentadas as conclusões e contribuições obtidas bem como a avaliação de futuros estudos em relação à problemática.

2 REFERENCIAL TEÓRICO

Neste capítulo será explicada a evolução das tecnologias relativas às aplicações ricas na internet. Cada tecnologia relevante ao trabalho terá sua devida explicação. O capítulo inicia discorrendo sobre a internet e sua evolução até a *Web 2.0*. Em seguida são detalhadas as características das aplicações ricas bem como as tecnologias existentes atualmente para suprir as necessidades das aplicações *online*.

2.1 A INTERNET

Em 1945, o engenheiro Vannevar Bush (1945) escreveu um livro chamado “*As We May Think*” (Como Podemos Pensar). Nele é descrito um mecanismo capaz de armazenar inúmeros livros com a possibilidade de acesso rápido e flexível. Bush o descreve como uma sequência de conteúdos relacionados a fim de aumentar a capacidade da mente humana. Este dispositivo, denominado de *Memex*, serviu de inspiração para que em 1965, Theodor Nelson (1987) definisse o hipertexto. Segundo ele o hipertexto é:

“...uma escrita não sequencial, um texto ramificado que permite ao leitor realizar escolhas, algo que pode ser melhor lido na frente de uma tela interativa. [...] Com o hipertexto, nós podemos criar novas formas de escrita que refletem a estrutura do que escrevemos; e os leitores podem realizar escolhas diferentes conforme suas atitudes e seus fluxos de pensamento...”

O que Nelson definiu é um conjunto de textos relacionados entre si. Por exemplo, em um texto pode haver a necessidade de explicações detalhadas de um determinado termo ou assunto. Em um hipertexto, se o leitor sentir necessidade, pode acessar este texto secundário de forma rápida e prática. O hipertexto é muito difundido atualmente em enciclopédias, manuais, jogos e na internet a qual possui como base a linguagem HTML (Cicconi, 1999).

O HTML (*Hypertext Markup Language*), como o próprio nome diz, é uma linguagem de marcação baseada em hipertexto. Nela é possível conectar textos distintos de forma bidirecional através de *links*. Os elementos de marcação servem para formatação do texto. Esta linguagem foi criada por Tim Berners-Lee no final da década de 1980, atuando no CERN

(*European Council for Nuclear Research*) (CERN, 2008). O grande triunfo de Tim Berners-Lee não foi criar uma linguagem de marcação, mas sim juntar o HTML aos protocolos TCP (*Transmission Control Protocol*) e DNS (*Domain Name System*) (Berners-Lee, 2011). Em um servidor ficam armazenadas as páginas escritas em HTML, computadores remotos fazem requisições via HTTP (*Hypertext Transfer Protocol*) (Wong, 2000) de determinadas páginas que são então copiadas localmente, que por sua vez, são interpretadas pelos navegadores (Freeman, 2005).

Este mecanismo definiu a *World Wide Web* ou simplesmente WWW. O maior benefício de Berners-Lee não foi apenas essa integração entre as tecnologias, mas sim fazer com que essa forma de comunicação fosse pública. Isto fez com que a internet se difundisse. Sua intenção era criar um ambiente com informações colaborativas interligadas, onde todas as questões pudessem ser respondidas, atualmente isto é representado através de páginas *Wiki* dentre as mais famosas, a Wikipédia (Berners-Lee, 1999).

Divergindo da ideia inicial de seu criador, a internet foi além de redes de hipertexto colaborativas. Ela se tornou um meio de comunicação. As páginas eram denominadas de “folhetos *online*” (Deitel, 2009 p.4). Apenas páginas estáticas, que exibiam informações unidirecionais. Os proprietários dos sites elaboravam os conteúdos, os publicavam nos servidores da rede mundial e lá permaneciam possibilitando apenas a sua leitura.

Na Figura 1 há um gráfico representando o crescimento da internet, onde o eixo y é representado em milhões de computadores. Nele pode-se verificar que durante a concepção da internet o crescimento foi relativamente baixo. Somente a partir de 1994 houve um aumento de computadores com acesso à internet. Foi uma evolução exponencial devido à popularização da rede mundial e ao fácil acesso aos microcomputadores (Comer, 2007 p. 39).

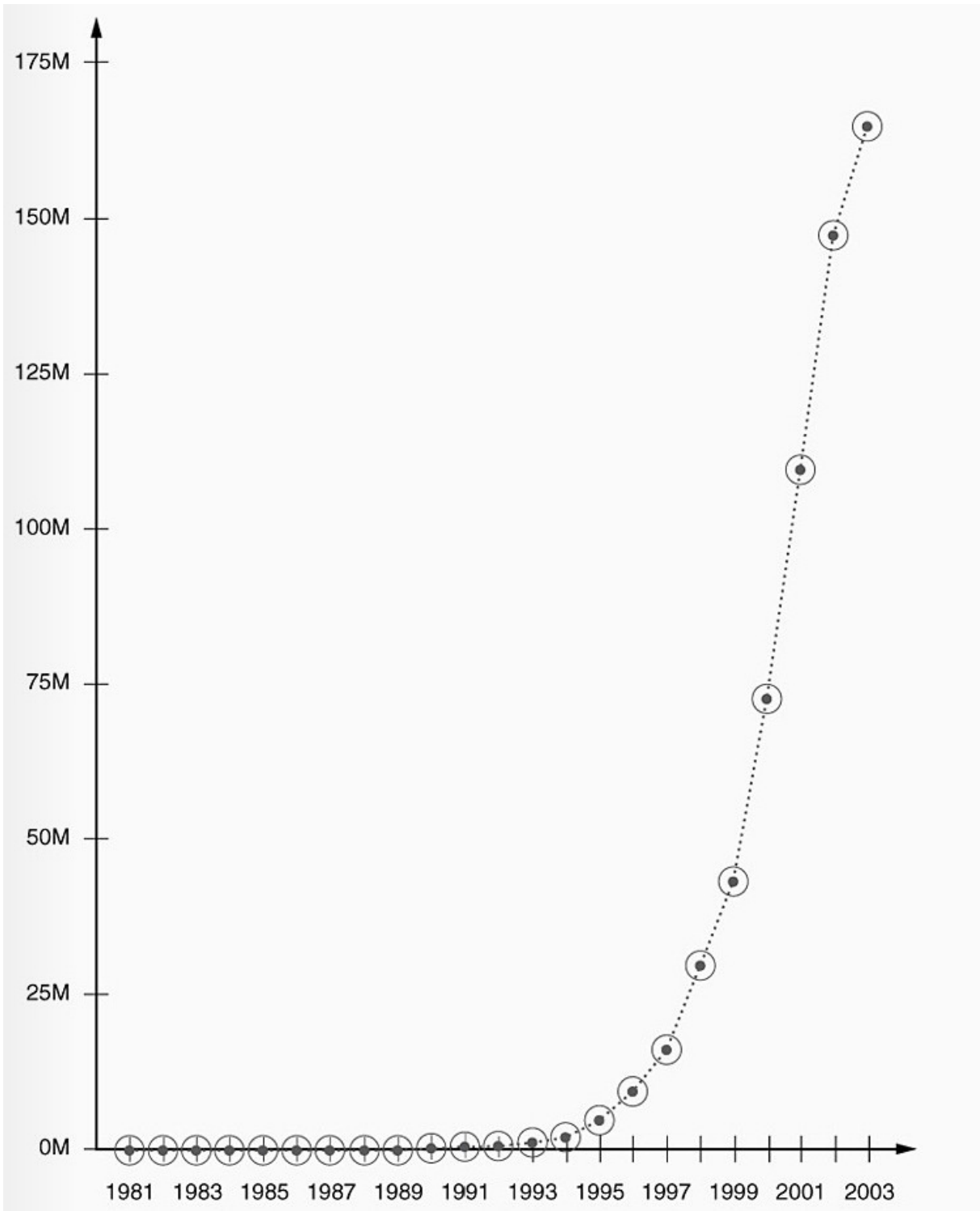


Figura 1 - Crescimento da internet medido pelo número de computadores conectados a cada ano, de 1981 a 2003.

Durante o final da década de 1990 houve um grande aumento no número de usuários na *web*. Este crescimento foi chamado de “bolha ponto-com”. Muitas empresas cresceram rapidamente por estarem presentes na *web*. Investidores deixaram de lado as tendências e os

padrões dos gráficos apostando em empresas *online*, mas isto não durou por muito tempo. Por volta de 2001 esta realidade começou a mudar, várias empresas que cresceram por especulação faliram. Em 2003, as empresas presentes na *web*, mudaram sua abordagem de negócio, surgindo assim a “*Web 2.0*” (Deitel, 2009 p.3).

2.2 WEB 2.0

O termo *Web 2.0* foi criado por Dale Dougherty em 2003 e foi introduzido pelo evento¹ de mesmo nome da *O’Reilly Media Inc* e *Internacional MediaLive* em 2004. Segundo O’Reilly (2005) a *Web 2.0* é uma característica adotada pelas empresas que sobreviveram à “bolha ponto-com”. As características definidas por ele são, em sua maior parte, de um novo modelo de negócio da *web*. Entretanto algumas definições de aplicações ricas são também atribuídas ao termo *Web 2.0*.

A definição de páginas interativas é uma delas. A grande presença de interatividade na *Web 2.0* muda o papel do usuário de espectador para colaborador. Esta colaboração enriquece o conteúdo das aplicações, fazendo com que elas evoluam constantemente baseadas no conhecimento coletivo agregado. Esta liberdade do usuário acaba envolvendo quem colabora, criando assim um vínculo com o mesmo. De certa forma faz com que este retorne à sua aplicação para acompanhar a repercussão de sua colaboração (O’Reilly, 2005).

Nem todos que acessam uma página que permite colaboração irão auxiliar em seu conteúdo. A grande maioria não investe seu tempo para agregar conhecimento em aplicações de terceiros. Para incentivar o envolvimento dos usuários, as aplicações *Web 2.0* devem ser de fácil acesso e manuseio. Esta é outra definição pertinente às aplicações ricas propostas pela *Web 2.0*. Uma página *Web 2.0* deve ser similar a uma aplicação *desktop*. Tornando-se familiar e facilitando assim seu uso e interação. Para que isso aconteça, é fundamental o uso do conjunto de tecnologias denominado AJAX (*Asynchronous JavaScript and XML*). Explicado na Seção 2.4, ele possibilita a atualização parcial das páginas deixando contínua a interação do usuário, sem a necessidade de esperar recarregar todo conteúdo após uma ação (O’Reilly, 2005).

Entretanto, não há uma definição exata sobre a *Web 2.0* até mesmo porque a *web* não é um software ou algo que possa sofrer versionamento. A *Web 2.0* em si, é o conjunto de

¹ <http://www.web2summit.com/web2011/public/content/about> Disponível em 20/09/2011

características e práticas adotadas pelas empresas da internet. Isso torna muito vago definir se uma página é ou não “2.0”, não existe um molde que caracterize isso, já que ela pode possuir apenas algumas características da *Web 2.0*.

Em uma entrevista² para a IBM, Tim Berners-Lee menciona sua posição em relação à *Web 2.0*. Ele a trata como apenas uma evolução natural da *web*. As funcionalidades agregadas a *Web 2.0* sempre existiram, porém somente agora elas estão sendo utilizadas. Características como páginas colaborativas e a visão da *web* como uma plataforma já estavam no planejamento inicial da *World Wide Web*.

2.3 APLICAÇÕES RICAS

Aplicações ricas na *web* são sistemas *online* que possuem semelhança na aparência, comportamento e usabilidade das aplicações *desktop* (Deitel, 2009, p. 313). Para isso as páginas *web* devem resolver alguns empecilhos referentes a arquitetura da internet. O fato de a *web* usar uma comunicação assíncrona, impede que as páginas atualizem seus dados dinamicamente. As aplicações necessitavam o reenvio de toda a página do servidor para o cliente mesmo que seja necessário atualizar apenas um campo da tela.

Modelo clássico de uma aplicação web

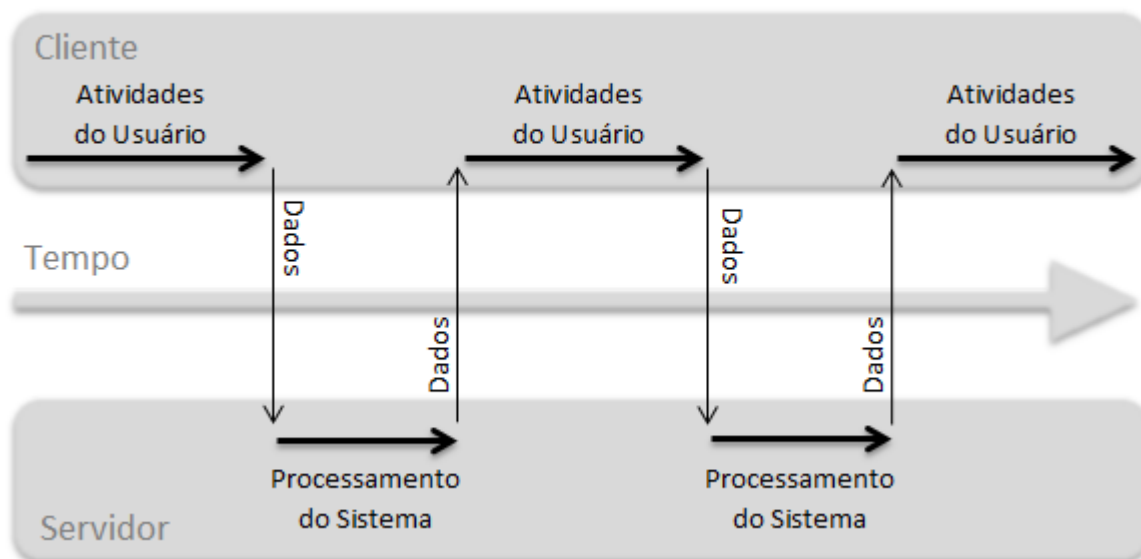


Figura 2 - Representação temporal do modelo clássico de uma aplicação web.

Todo esse processo de atualização gera transtornos ao usuário. Como pode ser visto na

² <http://www.ibm.com/developerworks/podcast/dwi/cm-int082206txt.html> Disponível em 20/09/2011

Figura 2, no modelo clássico da *web*, há uma quebra das atividades do usuário a cada requisição ao servidor. Este tempo de espera é ocasionado pelo tráfego dos dados e pelo tempo de processamento do servidor. Após o final da requisição, toda a página, incluindo suas funcionalidades, é redesenhada para o usuário. Todo esse ciclo resulta em um *refresh* na página (Garrett, 2005).

Por volta de 2005, difundiu-se o uso de um conjunto de tecnologias para resolver este problema. Conhecido como AJAX, sua principal característica é atualizar parcialmente as páginas de uma aplicação *web*. Isso deixa o usuário livre para interagir com o restante da aplicação enquanto a comunicação com o servidor é feita. Esta interação constante com a aplicação pode ser vista na Figura 3, onde a linha de atividades do usuário não sofre interrupções (Garrett, 2005).

Modelo Ajax de uma aplicação web

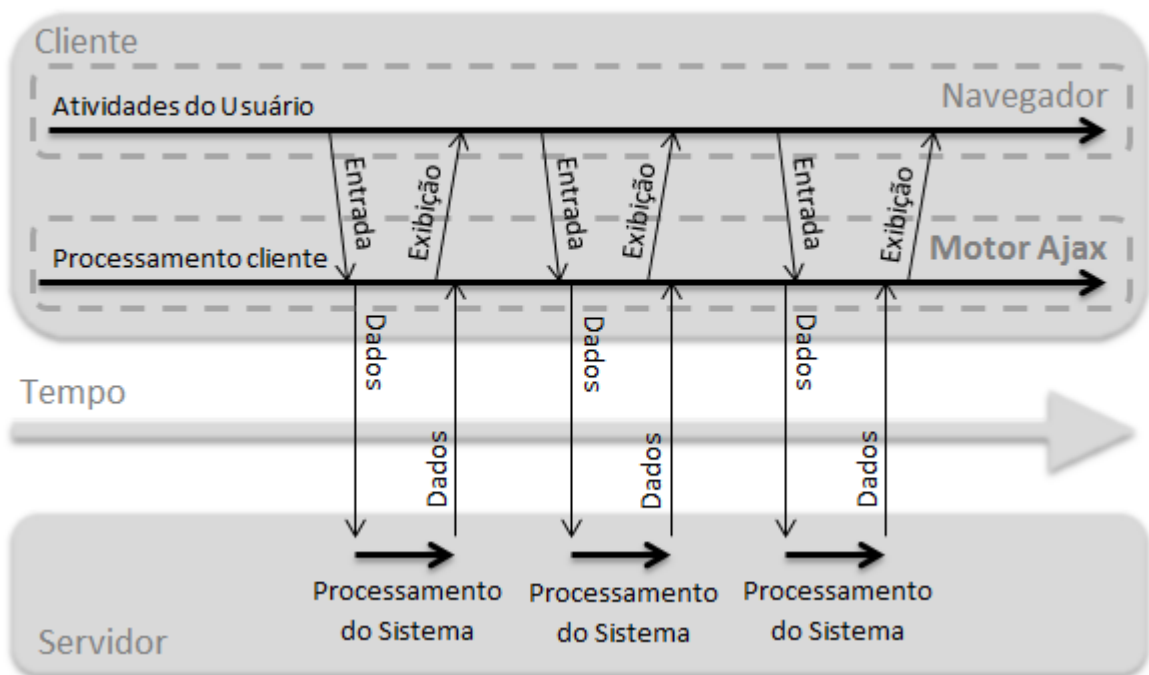


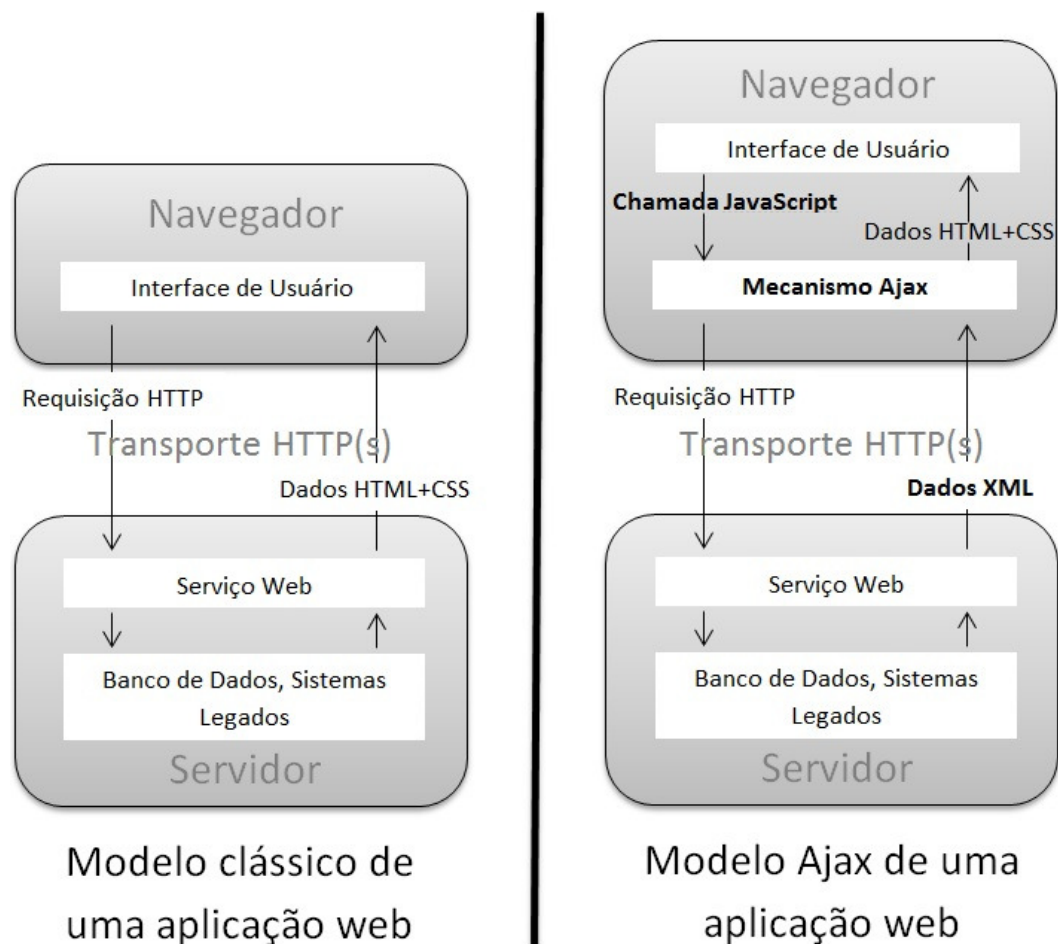
Figura 3 - Representação temporal do modelo Ajax de uma aplicação web.

Além da linha constante das atividades do usuário, na Figura 3, é possível ver que a comunicação ocorre diretamente entre o motor AJAX e o servidor. Tornando assim o AJAX responsável para atualizar as informações na tela por meio dos objetos DOM (*Document Object Model*) conforme será visto na Seção 2.4 (Gallo, 2008).

2.4 TECNOLOGIAS

O AJAX é um padrão de desenvolvimento *web* que comporta tecnologias já existentes antes da *Web 2.0*. Ele possibilita atualizações mais suaves sem a necessidade de ocorrer um *refresh* na página. Com isso a aplicação se torna mais agradável ao usuário, sem a necessidade de espera para atualizar a página após cada ação, o que simula ao usuário um uso contínuo sem interrupções (Gallo, 2008 p. 4).

A Figura 4 representa o fluxo das informações com AJAX, comparando com o modelo tradicional das aplicações *web* (Garrett, 2005 e Gallo, 2008 p.5). Nela pode-se observar como o AJAX interage entre o servidor e a interface de usuário, servindo como mecanismo intermediário, atualizando parcialmente o conteúdo. O contrário ocorre no modelo convencional, onde para cada requisição HTTP é retornada uma página HTML inteira com os dados requeridos.



Jesse James Garrett / adaptivepath.com

Figura 4 - O modelo tradicional para aplicações *web* (esquerda) em comparação ao modelo *AJAX* (direita).

O termo AJAX foi concebido por Jessé James Garret e é o acrônimo de *Asynchronous JavaScript and XML*. Como o próprio nome define, esta é uma forma de comunicação assíncrona entre a página e o servidor. Ele permite que a página *web* transfira apenas o necessário entre o cliente e o servidor, diminuindo assim o tráfego da rede e o tempo de espera. Para que isso seja possível, o AJAX acessa objetos DOM (*Document Object Model*)³ da página, se comunicando com o servidor por meio de requisições em XML (*Extensible Markup Language*) e HTTP (*XMLHttpRequest*) e atualizadas na página em HTML com CSS. Toda essa interação é controlada por um mecanismo escrito em *JavaScript*.

O DOM é a forma de representação de objetos em uma página. É possível acessar estes objetos via *JavaScript* e montar a página dinamicamente através dele. O DOM é uma estrutura em XML especificada pela W3C para facilitar o acesso aos componentes da página a fim de remover, adicionar e alterar seu conteúdo, seus atributos e seus estilos.

O *JavaScript* é uma linguagem executada no navegador *web* que foi criada pela *Netscape* e pela *Sun Microsystems* para o navegador *Netscape 2.0* em meados dos anos 90 (*Netscape*, 1995). Posteriormente a ECMA (*European Computer Manufacturers Association*) especificou os padrões das linguagens de *script* criando a definição de linguagens *ECMAScript*⁴ que se tornou base para o *JavaScript* (*Netscape*, 1996). Resumindo, o *JavaScript* é executado no navegador para interagir com os objetos DOM das páginas HTML. Além disso, é a linguagem de *script* mais popular, sendo compatível com a maioria dos navegadores do mercado (Gallo, 2008 p.5).

Para a comunicação entre o cliente e o servidor no modelo AJAX, os dados são transmitidos com o objeto *XMLHttpRequest* ou simplesmente XHR, que é especificado pela W3C⁵. Este objeto foi introduzido pela Microsoft em 1995 (Deitel, 2009, p.313). Nele é possível realizar requisições assíncronas no servidor. Os resultados da requisição podem ser retornados no formato XML ou JSON (*JavaScript Object Notation*). Este segundo, possui um formato mais compacto, tornando o retorno mais leve que uma representação em XML (Gallo, 2008 p.86). Segundo Deitel (2009):

“Cada objeto em JSON é representado como uma lista de nomes de propriedade e valores, contidos entre chaves, no seguinte formato:

{ “nomePropriedade1” : valor1, “nomePropriedade2” : valor2 }

³ <http://www.w3.org/DOM/> Disponível em 03/08/2011

⁴ <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf> Disponível em 03/08/2011

⁵ <http://www.w3.org/TR/XMLHttpRequest/> Disponível em 03/08/2011

Os arrays são representados em JSON com colchetes no seguinte modo:

[valor1, valor2, valor3]

Cada valor pode ser uma string, um número, uma representação JSON de um objeto, true, false ou null.”

Conforme descrição de Deitel, o JSON é representado em uma *string*. Seus valores são separados por vírgulas e agrupados entre chaves. Além disso, ele pode conter outros objetos JSON, trabalhando assim, como uma árvore de dados. Isto o torna mais fácil de ser manipulado e ocupa menos *bytes* que o XML, já que a estrutura do objeto não está discriminada (Deitel, 2009 p. 313).

O CSS (*Cascading Style Sheets*) é uma linguagem de marcação usada para definir a aparência visual dos elementos de uma página (Gallo, 2008 p.5). Ela é uma linguagem independente do HTML, o que facilita na manutenção das aplicações *web*, já que mantém separado o estilo do conteúdo apresentado (W3C, 2011).

2.4.1 FERRAMENTAS RIA

Acostumados com o comportamento e usabilidade das aplicações *desktop*, os usuários sentiam falta das mesmas características nas páginas da *web*. A necessidade era acessar conteúdos *online* de forma mais rica e interativa. A visualização de vídeo e áudio são exemplos disso. Em resposta a este problema, tecnologias denominadas RIA (*Rich Internet Application*) foram introduzidas no mercado (ALLAIRE, 2002 p. 1).

As RIAs são executadas no lado cliente da aplicação utilizando o navegador como plataforma. Para isso ocorrer é necessário o uso de *plug-ins* que comportam o motor de execução⁶ das linguagens RIA. Este mecanismo possibilita a execução e renderização de aplicações que não são baseadas em HTML pelo navegador. Para isso, o código a ser executado na aplicação deve ser transferido do servidor ao cliente.

Segundo ALLAIRE (2002, p.2), aplicações RIA proveem uma alta performance de execução da aplicação por evitar a constante comunicação com o servidor. Além disso, elas devem disponibilizar ferramentas para a criação de aplicações interativas. Bem como a possibilidade de execução *offline* e execução em diversas plataformas e dispositivos. Com isso aplicações RIA se tornam ferramentas poderosas por serem semelhantes às aplicações

⁶ Software de processamento que transforma a linguagem de marcação em conteúdo.

desktop e ao mesmo tempo possuir as vantagens da escalabilidade por estarem sendo disponibilizadas na web.

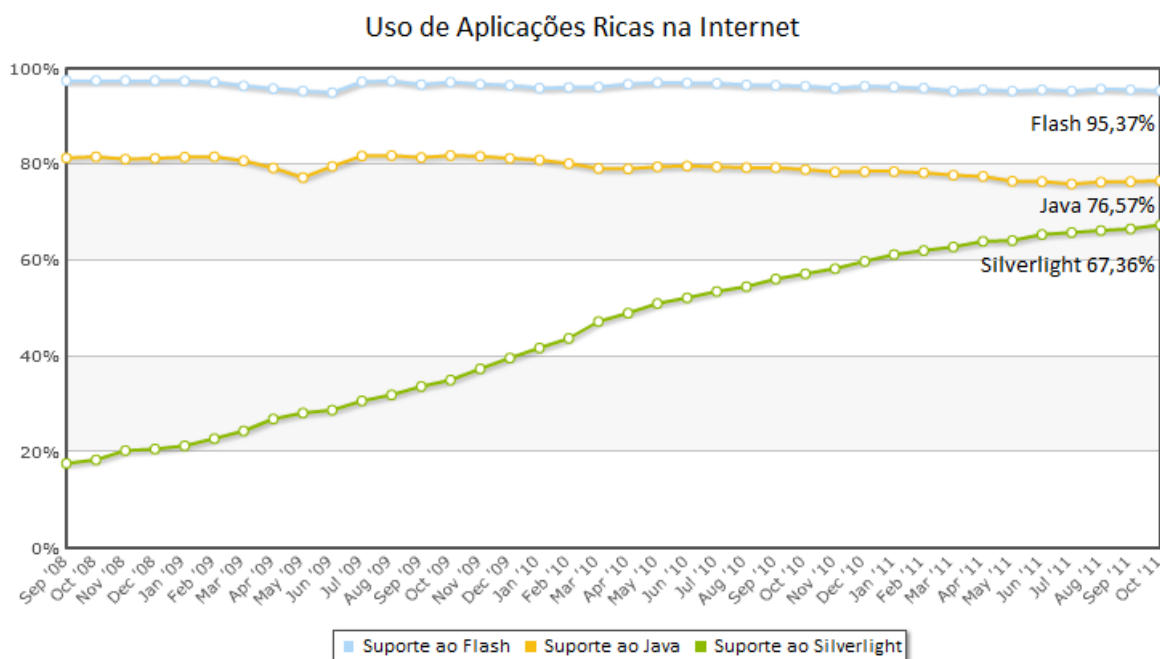


Figura 5 - Mercado das ferramentas RIA.

O site *Stat Owl*⁷, especializado em análise de comportamento *web*, realiza pesquisas estatísticas do uso de ferramentas RIA na internet. Na Figura 5, pode-se perceber a grande evolução dos últimos anos da tecnologia *Silverlight*. Chegando ao valor de 67,36% de uso da tecnologia. O *Silverlight* ainda está abaixo do *Java* que encontra-se com 76,57% e do *Flash* com 95,37%. Deve-se levar em consideração que a pesquisa analisa apenas a presença das tecnologias nos computadores e os acessam determinadas páginas na internet. Tendo uma participação de 80% de páginas americanas, é possível que haja margens de erro em uma amostra mais ampla.

As ferramentas de desenvolvimento de aplicações RIA não possuem uma especificação de um órgão regulamentador como a W3C. Isso faz com que cada uma possua características distintas. Cada qual adota implementações diferentes para suprir a necessidade das aplicações ricas. Nas seções seguintes, será analisada cada tecnologia RIA a ser explorada neste trabalho.

⁷ http://www.statowl.com/custom_ria_market_penetration.php Disponível em 14/10/2011

2.4.1.1 FLEX

O *Flash* é uma linguagem de renderização gráfica interativa. Nela é possível desenhar gráficos, reproduzir multimídia, criar sites interativos, *banners* publicitários, jogos entre outros (Deitel, 2009, p.344). Mantido pela empresa *Adobe Systems*, o *Flash* possui foco na reprodução de áudio, vídeo e animações (Busch e Koch, 2009).

Baseado na plataforma *Flash*, foi criado o *framework* de desenvolvimento de aplicações RIA chamada *Flex*. Lançada em 2004, possui o apoio da linguagem *ActionScript*, implementação da *Adobe* do padrão *ECMAScript*. O *Flex* aproveitou o poder gráfico do *Flash* e a presença multiplataforma de seu motor de execução para se difundir no desenvolvimento de aplicações ricas (Deitel, 2009, p.394).

Para a definição do *layout*, comportamentos e aparências da interface do usuário, o *Flex* utiliza a linguagem de marcação *MXML* (*Macromedia Extensible Markup Language*⁸). O *MXML* possui estrutura muito semelhante ao *HTML* e possui integração direta com o *ActionScript*. A compilação do *ActionScript* juntamente com o *MXML* gera um arquivo chamado *SWF* (*Shockwave Flash*). Este arquivo, por sua vez é executado pela plataforma *Flash* (*Adobe*, 2011).

Segundo a *Adobe* (2011) e Deitel (2009, p.394) o *Flex* possui uma biblioteca de componentes avançados. Isto facilita o desenvolvimento de aplicações ricas. Esses componentes abrangem características como arrastar-e-soltar (*drag-and-drop*), conteúdo dinâmico, multimídia, gerenciamento de *layout* entre outros. Com o *ActionScript*, ainda é possível aumentar o poder da biblioteca de interface, codificando e adaptando as características conforme necessidade.

A plataforma *Flash*, segundo o site da *Adobe*⁹, encontra-se em 99% dos computadores conectados à internet. Este número parece um pouco exagerado, pois como vimos anteriormente, segundo o site *Stat Owl*, sua presença alcança 95,37%. Mesmo podendo haver distorções na pesquisa realizada pela própria fabricante, a popularidade do *Flash* é indiscutível. Isto auxiliou enormemente no crescimento da tecnologia *Flex*.

Por outro lado, um ponto negativo da plataforma *Flash* é a aversão das empresas desenvolvedoras de sistemas operacionais. Como citado anteriormente, Steve Jobs (2010)

⁸ Segundo Gassner (2009), *MXML* também pode ser um acrônimo para “*Multidimensional XML*” ou ainda “*Maximum eXperience Markup Language*”.

⁹ <http://www.adobe.com/products/flashplatformruntimes/statistics.html> Disponível em 22/11/2011.

chegou a escrever uma carta sobre seus problemas, tais como a falta de segurança, a fraca confiabilidade e o baixo desempenho. Por esses motivos, a *Apple* não disponibilizou suporte ao *Flash* em seus dispositivos móveis. Já *Microsoft*, fabricante do sistema operacional *Windows*, possui sua própria linguagem RIA chamada *Silverlight*, sendo assim concorrente direto ao *Flex*.

2.4.1.2 SILVERLIGHT

Inicialmente chamado de WPF/E (*Windows Presentation Foundation / Everywhere*), por ser baseado nas características do WPF¹⁰ (*Windows Presentation Foundation*), o *Silverlight* teve sua primeira versão lançada em setembro de 2007. Desenvolvido pela *Microsoft*, sua evolução foi relativamente rápida. Suas versões seguintes 2, 3 e 4 foram lançadas em outubro de 2008, setembro de 2009, e abril de 2010 (Ghoda, 2010). No site da *Microsoft*¹¹, já está disponível a *Silverlight 5 RC (Release Candidate)*¹².

Inicialmente, o *Silverlight* era visto como um concorrente do *Flash* para a exibição de mídia *online* e animação de sites. Porém, houve um grande investimento da *Microsoft* na evolução dessa tecnologia. Como pode-se ver anteriormente, seu versionamento foi anual. Isto fez com que novas funcionalidades fossem agregadas, tornando o *Silverlight* uma poderosa ferramenta de desenvolvimento de aplicações RIA (Lair, 2010).

O *Silverlight* é baseado em XAML (*Extensible Application Markup Language*). Linguagem de marcação desenvolvida primeiramente para o WPF. Assim como o MXML do *Flex*, a grande diferença para o HTML é o poder de especificar linhas do tempo, transformações e animações. Ou seja, não apenas para definir *layout* (Lair, 2010).

Diferente do *Flex*, a linguagem de interação com linguagem de marcação não é baseada em *ECMAScript*. Para o *Silverlight* pode-se utilizar tanto *Visual Basic* quanto *C#*. Porém, é necessário observar que o motor de execução, executado no navegador, não é o *Framework .NET* completo. Por este motivo, pode haver funcionalidades das linguagens não suportadas pelo *Silverlight* (Ghoda, 2010).

Como visto anteriormente no site *Stat Owl*, o *Silverlight* possuiu um forte crescimento

¹⁰ Linguagem de desenvolvimento desktop desenvolvida pela *Microsoft* (<http://msdn.microsoft.com/en-us/netframework/aa663321.aspx>).

¹¹ <http://www.silverlight.net/downloads> Disponível em 22/11/2011

¹² Versão candidata disponível para que os desenvolvedores a testem.

desde 2008. Porém, ele ainda é a tecnologia com menor presença nos computadores com acesso à internet. Mesmo assim é possível ver os esforços da *Microsoft* para que este cenário mude. Recentemente foi anunciado o uso do *Silverlight* como linguagem de desenvolvimento predominante no *Windows Phone 7* (Cameron, 2011).

Em Junho de 2011, foi anunciada a nova versão do sistema operacional da *Microsoft*, chamado de *Windows 8*. Juntamente foi divulgado que os aplicativos para a nova interface o *Windows* serão baseadas no uso do *HTML5* e *JavaScript*. Em postagem no *blog* oficial de Steven Sinofsky¹³ (2011) explicou o motivo do uso destas tecnologias. Segundo ele, o uso de *plug-ins* (como o *Silverlight* e o *Flash*) para a execução de aplicativos em navegadores, aumenta o consumo de energia e reduz a duração da bateria em dispositivos móveis. Além de que o uso do *HTML5* melhora a segurança, confiabilidade e privacidade dos consumidores.

Esse movimento da *Microsoft* deixa transparecer uma mudança de prioridades para a companhia. Priorizando o uso do *Silverlight* apenas como linguagem nativa de desenvolvimento do *Windows Phone* e investindo no uso do *HTML5*. Esta mudança preocupa os desenvolvedores *Silverlight* sem uma resposta concreta sobre o futuro da tecnologia.

2.4.1.3 JAVAFX

Lançado pela *Sun Microsystems* na conferência *JavaOne* de 2007, o *JavaFX* surgiu com o intuito de auxiliar o desenvolvimento de conteúdos e aplicações ricas. Seu objetivo era estar presente em dispositivos móveis, *desktops*, televisores e outros dispositivos de consumo. Diversas versões do *JavaFX* foram lançadas desde então, tendo disponível para *download* a versão 2.0.1 no site da *Oracle*¹⁴.

Inicialmente, o *JavaFX* baseava seu desenvolvimento em *JavaFX Script*. Esta era uma linguagem declarativa e orientada a objetos, que foi descontinuada após a aquisição da *Sun* pela *Oracle*. A partir da versão 2.0, a linguagem base do *JavaFX* se tornou o próprio *Java*. Esta mudança colaborou para que os desenvolvedores não necessitem aprender uma nova linguagem de programação (Fain, 2011).

Outra característica alterada nesta última versão é o uso da linguagem de marcação *FXML*. Utilizada para organização de *layout* da aplicação, o *FXML* é muito semelhante ao

¹³ Presidente da divisão *Windows* na *Microsoft* desde 2008.

¹⁴ <http://www.oracle.com/technetwork/java/javafx/downloads/index.html> Disponível em 22/11/2011.

XAML (*Silverlight*) e ao MXML (*Flex*). Isso facilita a aceitação de desenvolvedores de aplicações ricas de outras plataformas. Além disso, melhora a organização e compreensão do da disposição dos elementos na tela (*Oracle*, 2011).

No início havia certa desconfiança se o *Oracle* iria continuar a tecnologia. Porém com o lançamento da versão 2.0 pode-se ver uma grande evolução nas funcionalidades comportadas pelo *JavaFX*. Além das mudanças citadas anteriormente, foi melhorada a aceleração gráfica pelo CPU, criados cerca de 50 componentes como controles de formulários, gráficos e gerenciamento de *layout*. As funcionalidades agregadas podem ser encontradas no próprio site do *JavaFX*¹⁵.

A vantagem da máquina virtual *Java* estar presente na maior parte dos computadores representa o maior diferencial desta tecnologia. Além disso, o *JavaFX* possui acesso total às funcionalidades disponíveis no *framework Java*, ao contrário de seu concorrente *Silverlight*. Isto a torna compatível com aplicações já existentes escritas em *Java*, o que aumenta potencialmente o uso desta tecnologia.

Por outro lado, esta mudança de tecnologias utilizadas pelo *JavaFX* gera outras complicações. A adaptação do desenvolvedor é uma delas, já que o referencial teórico existente até então se torna obsoleto. Sendo assim, o *JavaFX* se torna de certa forma imaturo inibindo seu uso em ambientes corporativos. As empresas não costumam adotar tecnologias não difundidas no mercado por haver receio de sua consistência. Outro problema causado por esta evolução é que por haver uma melhora na renderização gráfica das aplicações, ainda não há suporte para o *JavaFX* em sistemas operacionais baseados em *Linux* e *MacOS*. A promessa é que essa compatibilidade seja resolvida somente em meados de 2012 (*Oracle*, 2011).

2.4.2 HTML5

O HTML5 promete ser a base de desenvolvimento da maioria das aplicações *web* nos próximos 10 anos (David, 2010). Esta não é apenas uma nova especificação da linguagem de marcação HTML. Mas é sim o conjunto de novas especificações baseadas nas tecnologias HTML, CSS, *JavaScript* e DOM. Elas focam na redução do uso de *plug-ins* como o do *Flash*. Estas especificações são mantidas pela parceria entre a W3C e a WHATWG¹⁶ (*Web Hypertext*

¹⁵ <http://javafx.com/features/> Disponível em 22/11/2011.

¹⁶ WHATWG é um grupo de pessoas interessadas na evolução da web fundada em 2004 pela Apple, Mozilla

Application Technology Working Group) (W3C, 2011).

Todas as especificações novas podem ser encontradas no próprio site da W3C¹⁷. Segundo a W3C¹⁸, pode-se dividir as novas características em 8 classes, que são: *web* semântica, *offline* e *storage*, dispositivos móveis, conectividade, multimídia, gráficos e efeitos, performance e integração e CSS3. A seguir cada uma dessas classes será explicada e exemplificada conforme a necessidade.

2.4.2.1 WEB SEMÂNTICA

A *web* semântica é a evolução da “*Web* de Documentos” para uma visão de “*Web* dos Dados”, ou seja, as marcações da linguagem HTML5 não servirão apenas para organizar os dados visualmente, mas sim conforme seus significados. O objetivo é fazer com que não só as pessoas entendam o conteúdo de uma página, mas também os computadores identifiquem o assunto tratado. Isto facilitará a busca pelas informações na internet, relacionando de forma mais fácil outras páginas semelhantes.

Para que isso aconteça novas *tags* foram criadas na linguagem HTML. As principais citadas por David (2010) são o *header*, o *footer*, o *section*, o *article*, o *aside* e o *nav*. O nome das *tags* representa o conteúdo nelas contido, significando respectivamente o cabeçalho, o rodapé, a seção, o artigo, ao lado (do conteúdo) e navegação. Cada uma pode ser adicionada em uma página conforme a necessidade. Por exemplo, diferente de um documento que há apenas um cabeçalho e um rodapé, as páginas HTML podem conter diversas marcações de *footer* e *header*. Seja para representar o cabeçalho de uma seção, de um bloco ou da página inteira (David, 2010).

Além das *tags*, menciona-se aqui também o *Forms* 2.0. Este é composto por novidades nos elementos de interface das aplicações. Segundo David (2010), a maior mudança do *Forms* é extensão do *input*¹⁹ com o novo atributo *type*. Isto possibilita uma variação maior de tipos de entradas de dados na aplicação. É possível especificar se uma entrada de dados será somente números, uma representação de data, um *e-mail*, telefone, uma cor, entre outros.

Foundation, e Opera Software (<http://www.whatwg.org/>)

¹⁷ <http://www.w3.org/TR/> Disponível em 14/08/2011.

¹⁸ <http://www.w3.org/html/logo/#the-technology> Disponível em 14/08/2011.

¹⁹ *Inputs* são controles de entrada de dados. Uma caixa de texto é um exemplo de *input*.

```
<FORM>
  <input type="date" required>
  <input type="URL" required>
  <input type="submit" />
</FORM>
```

Figura 6 - Exemplo de elementos do *Forms 2.0*.

No exemplo da Figura 6, é possível ver a representação de dois campos. O primeiro representando uma data e o segundo um endereço de site *web*. Outra característica exemplificada é que ambos os campos são requeridos. Estes podem ser vistos na Figura 7 onde o campo de data possui um auxílio para a seleção do conteúdo e site *web* exhibe que o conteúdo é necessário.

Figura 7 - Exemplo de *inputs* em *Forms 2.0* no navegador *Chrome* versão 15.0.874.121.

2.4.2.2 OFFLINE E STORAGE

Usuários apenas podem interagir com aplicações da *web* quando estão conectados à internet. Quando estiverem *offline* eles não podem checar seus *e-mails*, consultar seu calendário ou trabalhar com ferramentas de edição *online*. Além disso, para obter as funcionalidades de uma aplicação *web*, o usuário depende de conexão HTTP para efetuar requisições ao servidor. Para isso, o HTML5 especifica duas soluções. A primeira é o uso de persistência de dados das aplicações e a segunda é a criação de um cache HTTP de aplicações *offline* (W3C, 2011).

Segundo MacDonald (2011), até a especificação do HTML5, a única forma de armazenar dados temporariamente no navegador era através de *cookies*. Além de suportar pequenas quantidades de dados, havia como inconveniente a data de expiração e a necessidade de enviar os dados de volta a cada requisição. Para suprir a persistência de dados *offline*, foram especificados 2 mecanismos, que são: *SessionStorage* e *LocalStorage*.

O *SessionStorage* também possui persistência temporária, porém não necessita o reenvio a cada requisição. Os dados são armazenados na sessão da aplicação, logo as

informações armazenadas serão excluídas quando a sessão for fechada. Já o *LocalStorage* armazena os dados no navegador tendo sua persistência por tempo indeterminado. Isso significa que se quando o usuário retornar ao site, no mesmo navegador, as informações armazenadas continuarão persistidas. Porém, assim como os *cookies*, os navegadores disponibilizam uma funcionalidade possibilitando excluir todos dados do navegador (MacDonald, 2011).

Ambas a formas de persistência possuem mecanismo semelhante. O acesso se dá por meio do *JavaScript*. Deve-se definir a chave para as informações a serem armazenadas, caso a mesma exista, apenas os dados serão substituídos (MacDonald, 2011). Há ainda a segurança de que somente o domínio da aplicação que persistiu uma informação terá o direito de acessá-la (W3C, 2011).

Para disponibilizar o uso de funcionalidades de uma aplicação de forma *offline*, o HTML5 especifica o uso de *cache* das páginas. Este mecanismo baseia-se em um arquivo onde são definidas todas as páginas que o navegador deve armazenar localmente. Neste arquivo devem ser listados todos os arquivos necessários para que a página execute de forma *offline*. Isso inclui a necessidade de especificar no arquivo, também os arquivos de estilo (CSS) e *script (JavaScript)* (W3C, 2011)²⁰.

2.4.2.3 DISPOSITIVOS MÓVEIS

A junção de novas especificações possibilita o melhor aproveitamento de aplicações *web* em dispositivos móveis. Algumas características facilitam a interação com os recursos de um celular ou um *tablet*. São elas, acesso a geolocalização, reconhecimento da resolução do dispositivo e acesso ao microfone e câmeras (W3C, 2011)²¹.

Explicada na Seção 5.3, a geolocalização é o poder de identificar sua posição no globo terrestre por meios como GPS (*Global Position System*), endereço de IP (*Internet Protocol*), RFID²² (*Radio Frequency Identification*), *Bluetooth*, *Wi-Fi*, endereço MAC (*Media Access Control*) ou até mesmo pela rede de telefonia. O HTML5 especifica que os navegadores deverão disponibilizar a localização por meio de uma API acessada por *JavaScript*. Se o dispositivo, ou computador possuir GPS, o mesmo deve ser utilizado. Caso contrário a

²⁰ <http://www.w3.org/TR/html5/offline.html> Disponível em 24/11/2011

²¹ <http://www.w3.org/html/logo/> Disponível em 24/11/2011

²² http://www.aimglobal.org/technologies/rfid/what_is_rfid.asp Disponível em 13/10/2011.

localização deve se dar por meio do endereço de IP (W3C, 2010).

APIs semelhantes são usadas para o acesso ao microfone e à câmeras. Estas estão fase de especificação pela W3C23 e não há uma definição concreta sobre como serão implementadas estas funcionalidades.

2.4.2.4 CONECTIVIDADE

A eficiência na conectividade significa conversas em tempo real trazendo uma melhora na comunicação. Para isso a tecnologia de *Web Sockets* está presente nas especificações do HTML5. Esta diminuem as requisições para o servidor, diminuindo o tráfego na rede (W3C, 2011)²⁴. Segundo Casario et al. (2011), *Web Sockets* são conexões bi-direcionais de comunicação entre canais dispostos pelo protocolo TCP. Esta tecnologia é útil para aplicações de *chat* e exibição de conteúdo com atualização ao vivo, como os comentários de uma partida de futebol.

O modelo convencional de aplicações *web*, como descrito na Seção 2.3, trabalha respondendo às requisições. Para que uma página seja atualizada com informações do servidor, primeiramente é necessário uma requisição. Somente tendo esta requisição feita é que é possível o envio de dados do servidor para o navegador. Com os *Web Sockets* a disponibilização de informações “empurradas” se torna possível. Ele abre uma comunicação direta do servidor com o navegador possibilitando o envio de informações conforme a necessidade sem o envio prévio de uma requisição.

2.4.2.5 MULTIMÍDIA

O grande diferencial das tecnologias de desenvolvimento *web* que utilizam *plug-ins* consiste na reprodução de arquivos de áudio e vídeo. O uso do *Flash*, por exemplo, possibilita a exibição de vídeos. Para adicionar conteúdo de mídia em uma página utilizando *Flash* em HTML4, era necessário usar ferramentas da *Adobe* para importar o vídeo, converter, publicar no formato suportado e por fim realizar o *upload* para o seu servidor. Além disso, havia a necessidade de escrever códigos complexos em HTML para exibir o conteúdo em *Flash*

²³ <http://www.w3.org/TR/html-media-capture/#introduction> Disponível em 24/11/2011.

²⁴ <http://www.w3.org/html/logo/> Disponível em 24/11/2011.

(David, 2010).

No HTML5 foram introduzidas as *tags* de vídeo e áudio. O uso destas foi simplificado. Basta realizar o *upload* do arquivo para o servidor e adicionar a linha de código demonstrada na Figura 8. Isto basta para que o navegador reconheça o arquivo e exiba todos os controles de mídia necessários para sua reprodução (David, 2010).

```
<video src="Video.mp4" controls="controls"></video>
```

Figura 8 - Definição da *tag* de vídeo no HTML5.

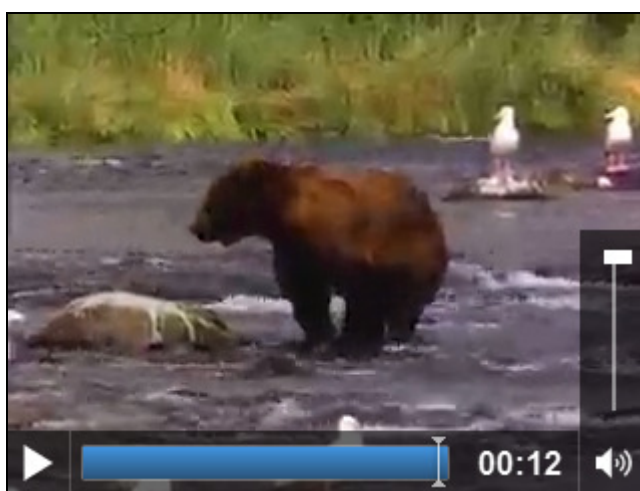


Figura 9 - Exibição do player de vídeo do HTML5.

Na Figura 9 pode-se ver um vídeo sendo executado em HTML5. Nela é possível observar os controles de execução de vídeo. Este foi gerado a partir do código descrito na Figura 8. Para que estes controles apareçam bastou apenas adicionar a propriedade “*controls='controls'*”.

O ponto negativo em relação à multimídia por parte do HTML5 é a falta de suporte a direitos autorais. O uso de DRM (*Digital Rights Management*) não está planejado inicialmente. Em nota nas perguntas frequentes²⁵, o W3C anuncia que só haverá esforços para o uso de DRM caso haja interesse de todas as partes envolvidas. Enquanto esse esforço não se concretizar e não houver implementação de terceiros, poderá haver certa resistência na adoção do HTML5 em massa.

²⁵http://www.w3.org/html/wiki/FAQs#Is_there_support_for_digital_rights_management_.28DRM.29_in_HTML_5_video.3F Disponível em 24/11/2011.

2.4.2.6 GRÁFICOS E EFEITOS

Conforme comentado neste trabalho, a *web* foi criada para a exibição de páginas de texto puro. Logo foi introduzido o suporte a imagens. Com isso, se você quiser publicar um texto ou uma imagem basta usar HTML e CSS. Eles darão todo suporte necessário. Se houver a necessidade de exibir arquivos de áudio ou vídeo, conforme descrito anteriormente, torna-se necessário o uso de tecnologias baseadas em *plug-ins* (Keith, 2011).

Além das *tags* novas de vídeo e áudio, com o HTML5 surgiram duas novas formas de renderização de gráficos em 2D: o *canvas* e SVG (*Scalable Vector Graphic*). Segundo Pilgrim (2011), o HTML5 define *canvas* como “imagens *bitmap*²⁶ dependentes de resolução, utilizados para renderização de gráficos, desenhos de jogos e outras imagens de forma dinâmica”. O *canvas* é um espaço definido na página onde, através de *JavaScript*, poderá ser desenhado o que for necessário.

```
<canvas id="meuCanvas" width="200" height="100"></canvas>
```

Figura 10 - Definição da área a ser exibida a imagem em *canvas*.

Na Figura 10, pode se ver como é definido o local onde serão renderizado os gráficos em *canvas*. É possível verificar que é necessário apenas definir o identificador, a altura e a largura da região da região. A definição do que será exibido é controlado por códigos em *JavaScript* conforme exemplificado na Figura 11. Neste é possível observar comandos predefinidos para desenhar quadrados e círculos.

²⁶ *Bitmap*, ou mapa de bits, são imagens não vetoriais que contém a descrição de como cada *pixel* será renderizado.

```

<script type="text/javascript">

//Recupera elemento canvas da página
var c=document.getElementById("meuCanvas");

//define o contexto
var cxt=c.getContext("2d");

//Desenha retângulo azul
cxt.fillStyle="rgb(0, 0, 555)";
cxt.fillRect(15,5,100,75);
cxt.closePath();

//Desenha círculo amarelo transparente
cxt.fillStyle = "rgba(999, 999, 0, 0.5)";
cxt.arc(110,70,35,0,Math.PI*2,true);
cxt.closePath();
cxt.fill();

//Desenha linhas sobre a imagem
cxt.moveTo(15,5);
cxt.lineTo(180,80);
cxt.lineTo(5,80);
cxt.lineTo(160,5);
cxt.stroke();

</script>

```

Figura 11 - Código *JavaScript* exemplificando o uso do *canvas*.

Pode-se ver na Figura 11 os comandos comuns no uso do *canvas*. Primeiro é necessário localizar o elemento *canvas* na página através do comando “*getElementById*”. Para este define-se o contexto a ser desenhado que será em 2D. No contexto então é possível aplicar formas geométricas como exemplificados os comandos “*fillRect*” e “*arc*”, nestes são desenhados um quadrado e um círculo, respectivamente. Ainda pode-se realizar desenhos livres, por meio do “*moveTo*” o qual define onde o desenho será iniciado e o “*lineTo*”, que cria uma linha do ponto atual até as coordenadas indicadas. O resultado obtido pode ser visto na Figura 12.

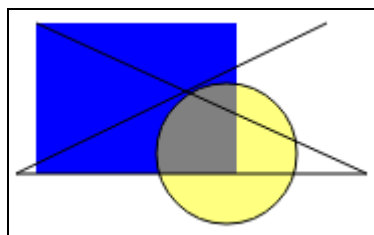


Figura 12 - Renderização de *canvas* obtida pelos comandos exibidos na Figura 11.

Existem diversos outros comandos para desenhar imagens em *canvas*, tais como *quadraticCurveTo* e *bezierCurveTo* que servem para desenhar curvas. Além disso, é possível também carregar imagens prontas em uma área de *canvas*.

Porém, como as imagens são “desenhadas” em tempo de execução não há controle sobre o que foi renderizado. Logo, surgem desvantagens tais como não ser possível efetuar eventos de interação com a imagem por meios convencionais, com um *clique* em um elemento exibido em *canvas*. Para suprir essa necessidade existe o SVG (Keith, 2011).

Segundo a W3C (2011), o SVG é uma linguagem gráfica de duas dimensões baseada em XML. Com ele é possível efetuar renderização de vetores gráficos, imagens e textos. Sendo manipulada por *JavaScript*, é possível alterá-la em tempo de execução. Por ser descrita em XML existem objetos relacionados às imagens (Sanders, 2011). Ou seja, se for renderizado um círculo, este pode sofrer interação via *JavaScript*.

Na Figura 13 está exemplificado um trecho de código escrito em SVG diretamente no HTML. Nota-se que não há necessidade de códigos em *JavaScript* para renderizar. Porém pode-se ver que o elemento *circle* (círculo) e *rect* (retângulo) possuem eventos de Script. Isto possibilita a interação do usuário nos elementos do SVG ao contrário do *canvas*.

```
<svg>
  <circle id="meuCircle" cx="100" cy="100" r="50" fill="RGBA
    (0,0,999,0.5)" onmousedown="alert('Círculo');"/>
  <rect id="meuQuadrado" x="5" y="10" width="100" height="80"
    fill="RGBA(0,555,33,0.5)" onmousedown="alert('Retângulo');"/>
  <line x1="0" y1="0" x2="120" y2="120" style="stroke:rgb
    (255,0,0);stroke-width:2" />
  <path id="curva" d="M 10 120 q 80 -200 140 0"
    stroke="blue" stroke-width="5" fill="none" />
</svg>
```

Figura 13 - Exemplo de comandos em SVG no HTML5.

A renderização do código de imagens em SVG, demonstrado na Figura 13, pode ser vista na Figura 14. Pode-se observar que assim como no exemplo do *canvas*, neste, também foi aplicado transparência aos elementos. Neste também é apresentando o elemento *path*. Este representa uma imagem sem uma forma geométrica definida, como um círculo ou um retângulo. Segundo David (2010), o elemento *path* pode ser desenhado em ferramentas como *Google Docs* (Google) e o *Adobe Illustrator CS5* (Adobe) (Sanders, 2011). Ambos possuem exportação para o formato “*.svg”.

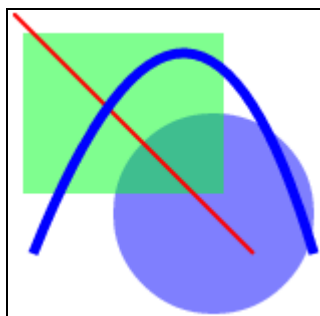


Figura 14 - Renderização de SVG obtida pelos comandos exibidos na Figura 13.

Um ponto positivo do SVG é o fato de ser uma forma de renderização vetorial. Sendo assim, independente da resolução, as imagens são exibidas da mesma forma sem perder qualidade, ao contrário do *canvas*. Segundo Keith (2011) isso faz com que seja um dos grandes problemas do *canvas*, já que, por ser baseado em *pixels* não há suporte à acessibilidade.

Cada uma das tecnologias possui vantagens e desvantagens. Enquanto o SVG mantém a fidelidade por ser vetorial, o *canvas* possui melhor performance por trabalhar com *pixels*. Logo ambas possuem uso em aplicações web. O SVG mais focado para gráficos interativos, multiplataforma e acessibilidade por não depender de resolução (Keith, 2011) e o *canvas* para animações como em jogos que necessitam melhor performance (Fulton, 2011).

2.4.2.7 PERFORMANCE E INTEGRAÇÃO

A especificação do HTML5 possui como uma das premissas aumentar a performance e a integração das aplicações *Web*. Para isso está sendo especificada a nova versão do *XMLHttpRequest* nível 2 que se refere maior integração. Já na questão da performance surgiu uma API de execução paralela denominada *Web Workers* (W3C, 2011).

O *XMLHttpRequest*, descrito na Seção 2.4 deste trabalho, havia sido introduzido pela *Microsoft* em 1995 (Deitel, 2009, p. 313). Posteriormente se tornou base da tecnologia AJAX. No HTML5, foi especificada uma nova versão do *XMLHttpRequest* agregando novas funcionalidades na questão da integração. Exemplificada por Casario et al. (2011), uma das funcionalidade novas é a denominada CORS²⁷ (*Cross-Origin Resource Sharing*).

CORS significa compartilhamento de conteúdo de origem cruzada. Ela permite o

²⁷ <http://www.w3.org/TR/cors/> Disponível em 28/11/2011

envio de dados entre domínios distintos. Isso facilita a integração entre aplicações *web*, sendo possível assim acessar serviços de terceiros facilmente. Outra funcionalidade nova do *XMLHttpRequest* é a possibilidade de *upload* de arquivos binários de forma assíncrona. Ou seja, sem interromper a atividade do usuário, como observado no padrão AJAX, e retorna se o *upload* foi concluído com sucesso ou não (Casario et al. 2011).

Já na questão da performance, os *Web Workers* não são parte fundamental do HTML5. Porém, são funcionalidades chaves para a criação de melhores aplicações *web*. Eles possibilitam a execução de blocos escritos em *JavaScript* de forma simultânea. Também conhecida como *threads*, esta funcionalidade agrega performance às aplicações, já que os processos executam paralelamente sem afetar o processo do navegador (Lawson e Sharp, 2011).

Segundo o próprio site da WHATWG²⁸, os *Web Workers* executam rotinas de forma paralela. O uso mais simples é executar processamentos de alto custo computacional, por meio de *JavaScript*, sem interromper a interface do usuário. Além disso, a WHATWG propõe o uso de *Web Workers* em conjunto ao *Web Socket*. Ou seja, é criado um processo paralelo que aguarda o recebimento de mensagens do *Web Socket* e em seguida armazena as informações localmente para posterior uso.

2.4.2.8 CSS3

A especificação do HTML5 comporta também novas funcionalidades ao CSS. Conforme visto na Seção 2.4, as folhas de estilo são fundamentais para a internet rica, onde sem as quais não seria possível criar páginas semelhantes visualmente aos aplicativos *desktop*. Entre as novidades do CSS3 podem-se destacar os seletores, *Media Queries*, tipografia flexível, transições e transformações. A seguir, estes serão conceituados e se possível exemplificados.

- Seletores: determinam quais elementos da página receberão um estilo. Na versão 3 do CSS foram introduzidos novos seletores que podem ser encontrados na própria especificação da W3C²⁹. Na Figura 15 está exemplificado o uso do seletor *”:nth-child(n)“*. Este considera a ordem dos elementos irmãos na árvore da página,

²⁸ <http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html#examples-4> Disponível em 28/11/2011.

²⁹ <http://www.w3.org/TR/selectors/#selectors> Disponível em 29/11/2011.

onde n representa a regra a ser considerada para aplicar o estilo. Estas regras podem ser numéricas, porém existem regras pré-definidas como *even* e *odd*, que são representadas na Figura 15, para aplicar estilos intercalados aos itens da lista (W3C, 2011).

```
.lista { width: 200px; }
.lista div {
  border-radius: 8px;
  padding: 2px;
  margin: 4px;
}
.lista div:nth-child(even) { background-color: #A7D0E8; }
.lista div:nth-child(odd) { background-color: #dde; }

<div class="lista">
  <div>Primeira linha</div>
  <div>Segunda linha</div>
  <div>Terceira linha</div>
  <div>Quarta linha</div>
  <div>Quinta</div>
  <div>Sexta</div>
</div>
```

Figura 15 - Exemplo de novos seletores do CSS3.

Pode-se ver que os elementos contidos no *div* da lista não possuem diferenciação de classes. Na Figura 16 observa-se o resultado obtido. Nele também foi aplicado o estilo *border-radius*, também introduzido no CSS3, para arredondar as bordas dos controles.

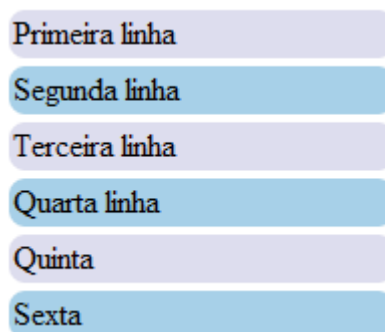


Figura 16 - Aplicação dos estilos descritos na figura 15.

- *Media Queries*: possibilita o uso de folhas de estilo distintas conforme regra definida diretamente no CSS sem intervenção de *JavaScript*. Por exemplo, podem-se ter estilos específicos para cada resolução. Na Figura 17 pode-se

observar como é utilizado a *media query* para identificação da resolução do dispositivo em que a página está sendo renderizada (W3C, 2011).

```
@media (min-width:500px) { /*estilo para desktop*/ }  
@media (max-width:500px) { /*estilo para tablet*/ }
```

Figura 17 - Exemplo da aplicação de *media queries*.

- Tipografia Flexível: com a introdução do *@font-face* possibilita ao desenvolvedor utilizar fontes que não estejam instaladas no computador do usuário. Para isso basta adicionar na folha de estilos o código descrito na Figura 18 e deixar disponível a fonte no servidor da aplicação. Os navegadores automaticamente irão efetuar o *download* da fonte ao exibir a página (W3C, 2011).

```
@font-face { font-family: MinhaFonte; src: url('MinhaFonte.otf'); }  
@font-face { font-family: MinhaFonte; font-weight: bold; src: url  
( 'MinhaFonte-Bold.otf' ); }
```

Figura 18 - Exemplo de uso do *@font-face*.

- Transformações: a partir do CSS3 surgiu a nova propriedade chamada *transform*. Com esta possível transformar a escala, a inclinação, a rotação e a posição de determinado elemento. Esta ainda não esta implementada de forma padrão entre os navegadores. Atualmente é necessário adicionar o prefixo de cada navegador a propriedade. Na Figura 19 está representado o uso deste estilo para transformar a rotação de um elemento *div*. Nele pode-se observar a necessidade do prefixo *-webkit-* para ser executado no *Chrome*. Em outros navegadores como o *Internet Explorer*, *Opera* e *Firefox*, utiliza-se *-ms-*, *-o-* e *-moz-*, respectivamente (W3C, 2011).

```

.obj
{
    margin: 100px;
    width: 100px;
    height: 100px;
    background-color: #dde;
    -webkit-transform: rotate(20deg);
    -webkit-transition: -webkit-transform 1s ease-in-out;
}
.obj:hover {
    -webkit-transform: rotate2(-360deg);
}

```

Figura 19 - Exemplo do uso do estilo *transform* e *transition*.

- Transições: são usadas para criar efeitos entre a mudança entre um estilo e outro. O atributo *transition* possui o seguinte formato usando parâmetros separados por espaço: “*transition: <estilo> <tempo> <transição>;*”. O estilo é em que mudança deverá ocorrer a transição, ou seja, se for quando alterar a largura este deverá possui o valor “*width*”. O tempo é representado por um número e o tipo, por exemplo, para 1 segundo utiliza-se “1s”. Já a transição a ser efetuada pode ser do tipo *linear*, *ease*, *ease-in*, *ease-out*, *ease-in-out* ou customizada. Estes se referem a velocidade da curva de transição entre um estilo e outro. Ainda na Figura 19, pode-se ver o uso de transição em conjunto a transformação de rotação. Neste exemplo, a transição ocorrerá quando o cursor estiver sobre o elemento da tela.

2.5 CONSIDERAÇÕES

Foram apresentadas até aqui a evolução da internet bem como as características e tecnologias que envolvem as aplicações ricas *web*. Além disso, foram abordadas as ferramentas de desenvolvimento RIA *Flex*, *Silverlight* e *JavaFX*. O HTML5 recebeu maior detalhamento de suas funcionalidades por se tratar de uma especificação e não uma tecnologia. No próximo capítulo será estudado e definido um modelo de comparação a ser aplicado entre as ferramentas estudadas.

3 MÉTODO DE AVALIAÇÃO

“Ao se deparar com uma tomada de decisão, é comum avaliar qual a melhor opção a ser seguida em determinado momento. Tratando-se de escolher uma tecnologia ou uma linguagem de desenvolvimento a problemática não é diferente. Analisar as características das opções conforme a necessidade é fundamental. Na engenharia de software, decidir a linguagem de desenvolvimento adequada é a etapa inicial para haja sucesso em um determinado projeto. Tendo em vista a possibilidade de que nem todas as premissas sejam supridas, este se torna um processo delicado” (Pressman, 1995, p. 684).

Segundo Pressman (1995), ao decidir pelo uso de uma linguagem de desenvolvimento, deve-se levar em consideração diversos critérios de avaliação. Tais como a área de aplicação, complexidade, desempenho e conhecimento da equipe. Ao analisar diversos fatores em uma tomada de decisão é possível utilizar “Métodos de Apoio Multicritério à Decisão” (Alves, Nykiel, Belderrain, 2007). Dentre os mais conhecidos estão o *Analytic Hierarchy Process* (AHP), *Elimination and Chox Traduisant la Realité* (ELECTRE) e o *Multiplicative Analytic Hierarchic Process* (MAHP).

O ELECTRE aplica critérios flexíveis, não necessitando obrigatoriamente uma comparação direta entre todos os critérios. Isto deixa a análise mais subjetiva, sendo influenciada pela afinidade do avaliador (Alves, Nykiel, Belderrain, 2007). Além disso, o ELECTRE necessita a transformação dos dados cardinais em ordinais. Já para o MAHP, derivado do AHP, é necessário transformar os dados para a escala geométrica (Guglielmetti, Marins e Salomon, 2003 *apud* Lootsma, 1993).

O AHP é considerado um dos mais respeitados métodos de tomada de decisões multicritério. Baseado na álgebra relacional, na pesquisa operacional e na psicologia, este método estabelece uma hierarquia de critérios para realizar a tomada de decisões (Guglielmetti, Marins e Salomon, 2003). Ele propõe também que cada critério a ser comparado pode possuir relevância distinta. Sendo assim, atribuído um peso que influencia diretamente no resultado final (Saaty, 1980).

A proposta do AHP é montar matrizes de comparações diretas entre os critérios analisados. Isto diminui a subjetividade do tomador de decisão. Além disso, o AHP é um

método simples e confiável, que facilita a tomada de decisões avaliando um número finito de alternativas em um conjunto de critérios com influência ponderada (Zattera, 2011) (Jordão e Pereira, 2006).

Com base nestas características, o AHP será aplicado para a análise das ferramentas e tecnologias de desenvolvimento de aplicações ricas. Na próxima seção será descrito como é aplicado o método de apoio à decisão por multicritérios AHP.

3.1 ANALYTIC HIERARCHIC PROCESS

O AHP foi desenvolvido por Thomas L. Saaty na década de 1970. Desde então este é estudado e aplicado nas mais diversas tomadas de decisões. Muitos estudos utilizando o AHP foram publicados em diversas áreas tais como a contabilidade, logística, investimentos, assistência médica, informática entre outros (Corso e Löbler, 2010).

Segundo Saaty (2008), para tomar uma decisão de forma organizada com o AHP, é necessário primeiramente dividir a decisão nas seguintes etapas: 1) Definir a meta; 2) Definir os critérios a de comparação, sua hierarquia e as alternativas possíveis; 3) Construir matrizes de avaliação; 4) Construir matrizes de prioridade. A meta dada à análise é o objetivo da comparação. Em um exemplo fictício, a meta poderia ser comprar um carro.

Tendo a meta definida, é necessário levantar os critérios a serem avaliados. Seguindo o exemplo de comprar um carro, os critérios adotados poderiam ser preço, manutenção e consumo. Os critérios podem ser organizados em mais níveis, como é possível observar na Figura 20, onde os critérios mais específicos (preço, manutenção, aceleração, etc.) estão divididos em critérios mais abrangentes (economia e qualidade). Por fim, para cada critério mais específico na hierarquia, são atribuídas as alternativas, neste caso, Carro A, Carro B e Carro C. Com isso obtêm-se a hierarquia representada na Figura 21.

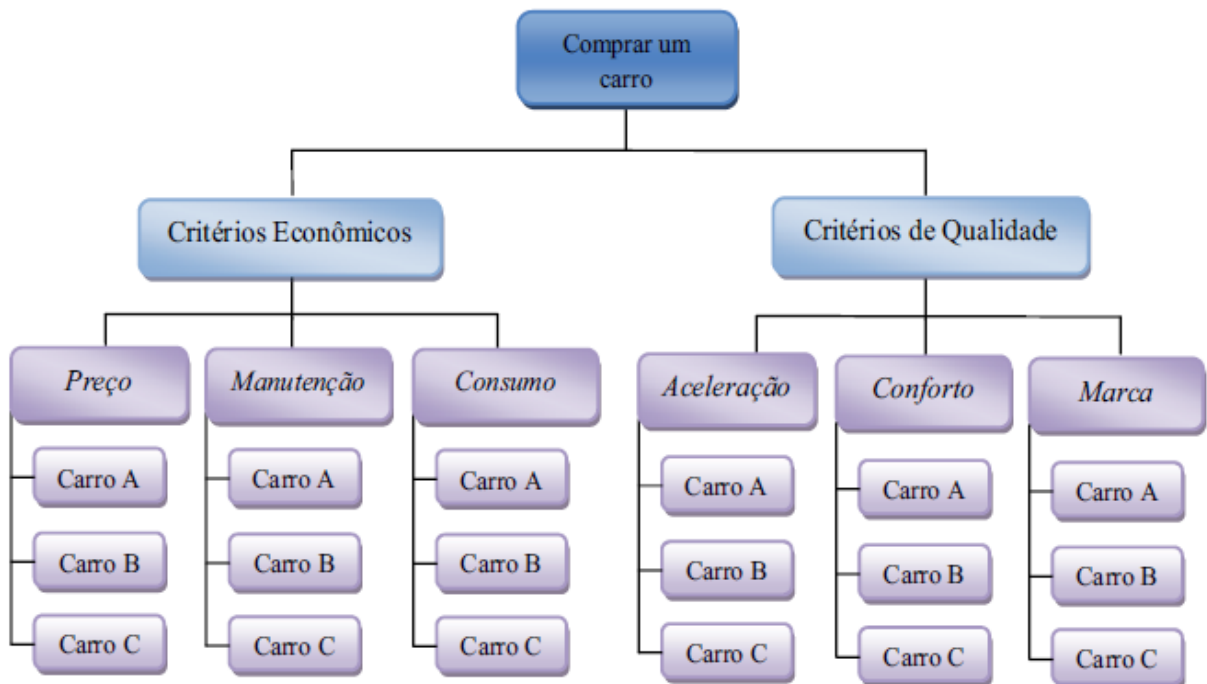


Figura 20 - Exemplo de hierarquia com mais de um nível de critérios (Corso e Löbler, 2010).

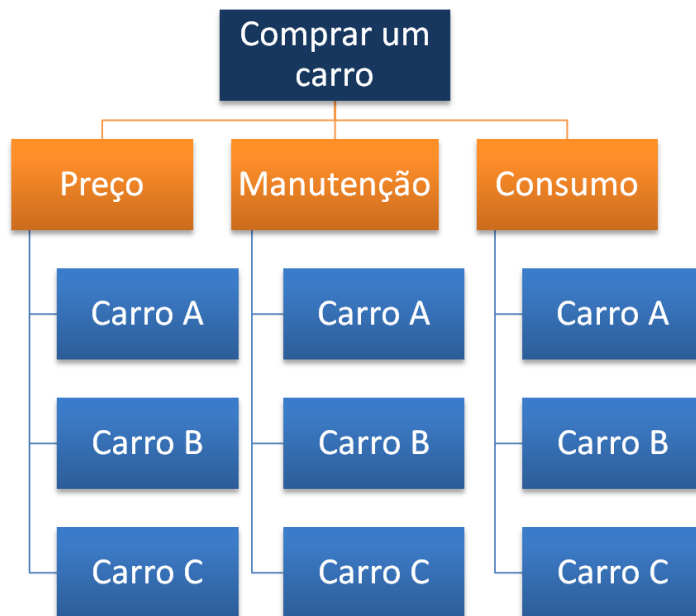


Figura 21- Exemplo de hierarquia somente com um nível de critérios.

Com a hierarquia estruturada é necessário realizar as comparações par a par de prioridades dos critérios propostos. Para isso, em cada nível de critérios e elaborada uma matriz relacionando-os a fim de definir o grau de importância de um critério em relação ao outro na tomada de decisão final. Esta comparação pode ser realizada por meio de dados reais

ou de julgamentos humanos (Saaty, 2008). Com isso é possível transformar informações empíricas em valores numéricos para que os mesmos sejam processados e comparados (Vargas, 2010). Segundo Vargas (2010), esta transformação é o maior diferencial do AHP para com os demais métodos de comparação.

A especificação do AHP propõe que as comparações realizadas baseiem-se em uma escala de 1 à 9, conhecida como Escala Fundamental de Saaty (Silva e Belderrain, 2005). Esta escala está representada na Tabela 1. Nela observa-se que somente os valores ímpares possui descrição. Isto se deve ao fato que os números pares são considerados intermediários para que possam ser utilizados em situações de dúvida.

1	Igual importância	As duas atividades contribuem igualmente para o objetivo.
3	Importância pequena de uma sobre outra	A experiência e o julgamento favorecem levemente uma atividade em relação à outra.
5	Importância grande ou essencial	A experiência e o julgamento favorecem fortemente uma atividade em relação à outra.
7	Importância muito grande ou demonstrada	Uma atividade é muito fortemente favorecida em relação à outra, sua dominação de importância é demonstrada na prática.
9	Importância absoluta	A evidência favorece uma atividade em relação à outra com o mais alto grau de certeza.
2, 4, 6, 8	Valores intermediários	Quando se procura uma condição de compromisso entre duas definições.

Tabela 1 - Escala Fundamental de Saaty (Silva e Belderrain, 2005).

Os valores da Escala Fundamental de Saaty são atribuídos na matriz de forma direta e inversa conforme demonstrado na Tabela 2. Isso significa que se o critério da minha linha possuir mais importância que o comparado da coluna, a nota atribuída será x . Caso o critério da coluna for mais importante, a nota atribuída será $1/x$. Quando um critério for comparado com ele mesmo a nota será 1. Logo, a diagonal principal da matriz sempre será 1.

	Critério 1	Critério 2
Critério 1	1	Avaliação numérica
Critério 2	1/ Avaliação numérica (inversa)	1

Tabela 2 - Matriz de comparação (Vargas, 2010).

Tendo as notas atribuídas na matriz de comparação par a par, como exibido na Tabela 23, os dados devem ser processados. Isto resultará em uma relação de percentuais totalizando 100%. Estes, por sua vez, indicam o grau de importância de determinado critério a ser adotada na comparação final. Além de comparar critérios, esta matriz também é utilizada ao analisar os valores obtidos para cada alternativa. Para cada critério mais específico (preço, manutenção, consumo) é criada uma matriz relacionando as alternativas (Carro A, Carro B e Carro C). Com isso, espera-se analisar par a par todos os dados resultantes da pesquisa.

A partir das matrizes de comparação, para alcançar o percentual de importância, o AHP necessita que a matriz seja normalizada. Para isso cada valor deve ser dividido pelo total de sua coluna. Esta normalização pode ser observada na Tabela 3 a qual exibe a somatória de cada coluna e posteriormente o resultado das divisões realizadas. É importante observar que neste momento, a soma dos valores de cada coluna sempre será igual a 1 (Vargas, 2010).

	Preço	Manutenção	Consumo
Preço	1	1/6	4
Manutenção	1/6	1	1/8
Consumo	1/4	8	1
Total	7,250	9,166	5,125
Resultados			
Preço	$1 / 7,250 = \mathbf{0,138}$	$(1/6) / 9,166 = \mathbf{0,018}$	$4 / 5,125 = \mathbf{0,780}$
Manutenção	$(1/6) / 7,250 = \mathbf{0,828}$	$1 / 9,166 = \mathbf{0,109}$	$(1/8) / 5,125 = \mathbf{0,024}$
Consumo	$(1/4) / 7,250 = \mathbf{0,034}$	$8 / 9,166 = \mathbf{0,873}$	$1 / 5,125 = \mathbf{0,195}$

Tabela 3 - Normalização dos dados de uma matriz de comparação.

Com a matriz normalizada, segundo Saaty 2005, devemos calcular a média dos valores de cada critério. Esta média, representada na Tabela 4, soma os valores de cada linha da matriz e divide pela quantidade de critérios. O resultante, multiplicado por 100, é a porcentagem da contribuição do critério para o resultado final.

Preço	$(0,138 + 0,018 + 0,780) / 3$	0,312 (31,2%)
Manutenção	$(0,828 + 0,109 + 0,024) / 3$	0,320(32%)
Consumo	$(0,034 + 0,873 + 0,195) / 3$	0,367(36,7%)

Tabela 4 - Média dos critérios.

Para cada critério mais específico, a mesma avaliação deve ser feita entre as alternativas da solução. Logo haverá uma matriz para cada critério (preço, manutenção e consumo) comparando todas as alternativas (Carro A, Carro B e Carro C).

Outra característica importante do Modelo Analítico Hierárquico é a consistência dos dados informados pelo avaliador (Vargas, 2010 e Teknomo, 2006). Em um exemplo de raciocínio lógico é possível observar que se o preço é mais relevante que a manutenção e a manutenção é mais relevante que o consumo, logo o consumo não poderá ser mais relevante que o preço. Ou seja, se $A > B$ e $B > C$ não se pode afirmar que $C > A$. Para isso, Saaty (1980) propõe um cálculo de inconsistência. Segundo Vargas (2010) uma matriz de comparação somente será válida se o grau de inconsistência não ultrapassar 10%.

Após a obtenção das porcentagens de relevância de cada critério, elabora-se uma nova matriz. Esta, chamada de Matriz de Prioridades, relaciona as alternativas para com os critérios avaliados. Os valores nela atribuídos são os resultados das matrizes das alternativas ponderados pela relevância obtida para cada critério. Nesta, o maior valor encontrado indicará a melhor escolha a ser tomada.

Para facilitar a aplicação do AHP existem diversos sistemas disponíveis no mercado. É possível utilizar ferramentas simples, como planilhas eletrônicas ou até mesmo *softwares* mais elaborados que gerem gráficos dos resultados. No artigo de Vargas (2010), a utilização do Método Analítico Hierárquico se dá utilizando o sistema *Expert Choice*³⁰ desenvolvido pelo fabricante de mesmo nome. Tendo em vista o reconhecimento de Vargas (2010) nesta ferramenta, este trabalho irá utilizá-la na avaliação das tecnologias de desenvolvimento de aplicações ricas.

3.2 CONSIDERAÇÕES

Realizar uma avaliação em uma tomada de decisão é fundamental para realizar a

³⁰ <http://www.expertchoice.com/> Disponível em 15/04/2012

melhor escolha. Para isso existem métodos que auxiliam neste processo. Neste foram citados alguns métodos além de detalhar as características do AHP. Este, por sua vez, será aplicado para realizar a comparação entre as ferramentas e tecnologias de desenvolvimento de aplicações ricas. Com isso, pretende-se obter um resultado mais imparcial, diminuindo assim, a subjetividade do avaliador. No próximo capítulo será especificado o modelo de avaliação definindo os critérios a serem avaliados.

4 MODELO DE AVALIAÇÃO

No capítulo anterior foram citados diferentes métodos de avaliação para efetuar a tomada de decisão. A fim de realizar a avaliação entre as tecnologias, o modelo proposto será apoiado pelo método AHP (*Analytic Hierarchy Process*). Conforme visto na Seção 3.1, o modelo descrito propõe primeiramente a definição da meta de avaliação. Tendo em vista a proposta do trabalho a meta adotada é a seguinte: “Melhor ferramenta/tecnologia para desenvolvimento de aplicações ricas na *web*”. Posteriormente deve-se definir as alternativas da tomada de decisão e os critérios a serem avaliados. Estes estão especificados nas seções seguintes.

4.1 ALTERNATIVAS

O AHP trabalha com a opção finita de alternativas a serem tomadas. Ao se tratar do desenvolvimento de aplicações ricas, as tecnologias escolhidas como alternativas da tomada de decisão foram o *Flex*, *Silverlight*, *JavaFX* e o *HTML5*. Ao final do processo de avaliação, estas serão elencadas conforme o atendimento dos requisitos especificados no critério de avaliação.

A escolha das ferramentas *Flex* e *Silverlight*, para essa comparação, se deu pela abrangência do mercado de ambas. O *Flex*, por seu *engine* de execução, o *flash player*, estar presente na maioria dos computadores. Já a escolha do *Silverlight* deu-se pelo seu crescimento nos últimos anos servindo inclusive como plataforma *online* de exibição das olimpíadas de inverno de 2010. O *JavaFX*, por sua vez, foi escolhido por ser baseado na linguagem *Java*, que possui uma grande abrangência de mercado e alto número de desenvolvedores.

O *HTML5* está presente neste trabalho por não ser uma ferramenta e sim uma um conjunto de especificações técnicas. Propostas pela W3C, estas tecnologias possuem a finalidade de padronizar o desenvolvimento da *web* tornando-a mais ricas. Além disso, segundo David (2010), o *HTML5* será utilizado nos próximos 10 anos para o desenvolvimento de aplicações *web*. O que, tratando-se de tecnologia, é um longo período.

4.2 CRITÉRIOS DE AVALIAÇÃO

Com a meta definida e as alternativas listadas, é necessário definir os critérios de avaliação. Os critérios terão base nas principais características de uma aplicação rica apresentadas no referencial teórico deste trabalho. Estas estão relacionadas a seguir com seu devido detalhamento sobre o que deve ser avaliado em cada critério.

4.2.1 PERFORMANCE

Esta característica deve ser validada observando a utilização do processador e memória da estação onde está sendo executada a aplicação no navegador. Eventualmente, outros processos podem interferir no resultado. Para evitar isso, esta validação deve ser realizada 6 vezes e analisada a média das mesmas. Isto se deve a vulnerabilidade da circunstância em que está sendo executado o teste. Além disso, serão descartadas as exceções, ou seja, o maior e o menor resultado serão desprezados na análise.

Em se tratando de tecnologias de interface de usuário, esta avaliação de performance também irá considerar o tempo de renderização dos gráficos. Neste será avaliado o tempo (em milissegundos), o consumo de memória (em *Kbytes*) e pelo percentual de uso do processador.

O resultado obtido em cada uma das tecnologias será comparado onde o menor resultado será a melhor performance. Para mensurar o tempo de renderização serão utilizadas as ferramentas de desenvolvimento do navegador *Google Chrome*. Já para medir a utilização de processador e memória, será utilizado o Gerenciador de Tarefas do Windows.

4.2.2 COMPATIBILIDADE

Por existirem diversos navegadores atuantes no mercado é fundamental realizar o teste em navegadores distintos. Para isso serão avaliados 4 navegadores. Estes serão escolhidos conforme a sua participação de mercado, durante o primeiro semestre de 2012. Para isso selecionou-se os seguintes sites especialistas em pesquisas de comportamento: *Stat Owl*³¹, W3

³¹ http://www.statowl.com/web_browser_market_share.php Disponível em 08/11/2011.

*Schools*³², *Clicky Web Analytics*³³ e *W3 Counter*³⁴. A pontuação atribuída será referente à quantidade de características compatíveis com os navegadores selecionados.

4.2.3 FACILIDADE DE DESENVOLVIMENTO

Este critério é referente aos esforços da implementação das características. Neste critério será avaliada a disponibilidade de documentação de cada tecnologia. Também será analisada a disponibilidade de controles e elementos que auxiliam no desenvolvimento. Quanto maior a quantidade de documentação e quando mais recursos apoiando o desenvolvimento, melhor será considerada a tecnologia.

4.2.4 SEMELHANÇA NA APARÊNCIA E USABILIDADE DE APLICAÇÕES DESKTOP

Este critério avaliará se a tecnologia satisfaz as especificações propostas, atendendo os requisitos. Estes requisitos baseiam-se nas características de uma aplicação rica. Sendo assim a tecnologia necessitará atender características como a aparência e usabilidade de uma aplicação *desktop*. A usabilidade será avaliada através de elementos como *drag-and-drop*, navegação e interação. A aparência levará em consideração a capacidade animação gráfica e outros elementos visuais. Logo, quanto maior a usabilidade e aparência mais semelhante a de uma aplicação *desktop*, melhor será considerada a tecnologia.

4.3 HIERARQUIA

Os critérios descritos anteriormente devem ser aplicados à implementações realizadas com as tecnologias proposta na Seção 4.1. A partir desta premissa, será desenvolvida uma aplicação de referência para a realização das comparações. Esta aplicação está descrita no Capítulo 5 e está dividida em características. Uma característica será composta de funcionalidades comuns de uma aplicação rica. Cada característica será avaliada nos critérios

³² http://www.w3schools.com/browsers/browsers_stats.asp Disponível em 08/11/2011.

³³ <http://getclicky.com/marketshare/global/web-browsers/> Disponível em 08/11/2011.

³⁴ <http://www.w3counter.com/globalstats.php> Disponível em 08/11/2011.

especificados na Seção 4.2.

Conforme observado na Figura 22, cada característica servirá como subcritério dos critérios propostos. Isto se deve à hierarquia proposta pelo AHP. Através desta hierarquia, ao aplicar as métricas do método de avaliação, serão definidas relevâncias distintas à cada característica da aplicação de referência para com o critério avaliado. Com isso será distribuído o peso final de cada característica conforme a finalidade em que a mesma foi proposta. Ou seja, uma característica em que uma animação gráfica foi especificada, terá maior relevância no critério de performance do que outra que foi especificado um formulário simples.

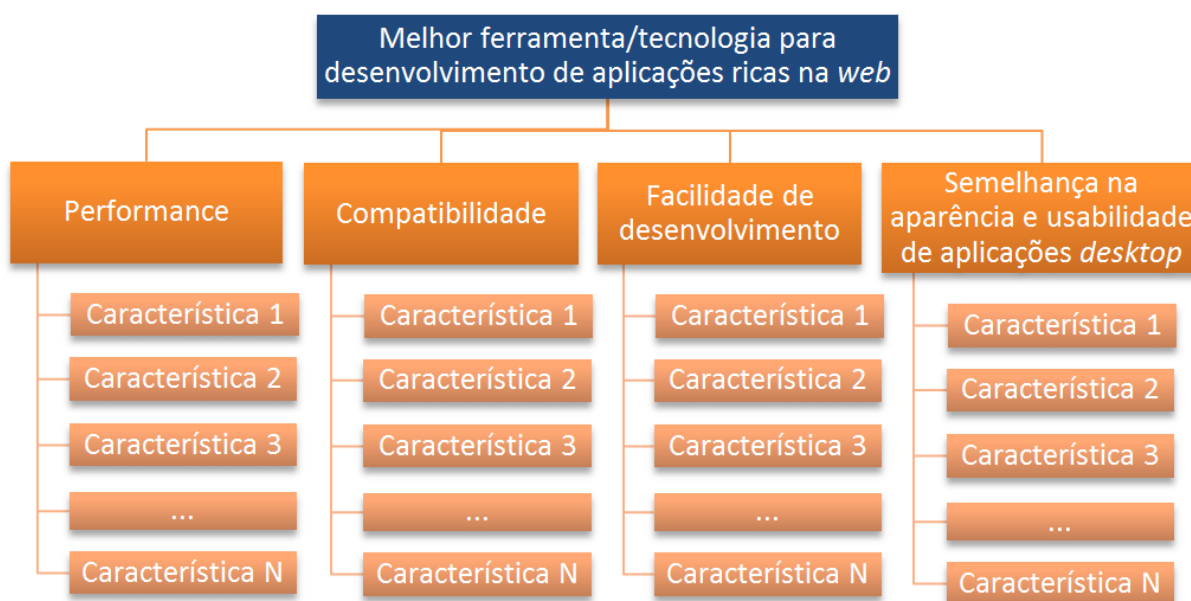


Figura 22 - Hierarquia dos critérios de avaliação do método AHP.

Na Figura 22 está representada a hierarquia que será utilizada para avaliação das tecnologias com o AHP. Nela é possível observar que há dois níveis de critérios. Para cada critério mais específico serão avaliadas as alternativas. Logo, todas as características serão avaliadas em todas as tecnologias para cada critério mais abrangente (performance, compatibilidade, etc.)

4.4 CONSIDERAÇÕES

Neste capítulo foi definido o modelo de avaliação ao qual serão submetidas as tecnologias estudadas. Foram especificados os critérios de avaliação bem como a hierarquia

adotada para a avaliação através do AHP. No próximo capítulo será definida a aplicação de referência que conterà as características a serem avaliadas em cada uma das tecnologias propostas.

5 APLICAÇÃO DE REFERÊNCIA

Uma aplicação de referência serve para exemplificar a utilização de determinada tecnologia. A escolha do desenvolvimento de uma aplicação de referência baseou-se na proposta da *Sun*, atualmente *Oracle*, que desenvolveu a aplicação *Java Pet Store*. Esta foi desenvolvida com o intuito de demonstrar o uso da tecnologia AJAX com a linguagem de programação *web Java*³⁵ (Singh, Stearns e Johnson, 2002). Já a aplicação de referência proposta, servirá para a realização de um comparativo entre as tecnologias. Nela serão avaliadas as funcionalidades aplicadas no desenvolvimento de aplicações ricas, citadas anteriormente.

A aplicação será baseada em partes de um sistema de emissão de pedidos *online*. Nela haverá a uma tela de emissão de pedidos e outra para a visualização de gráficos estatísticos das vendas. Para acesso às telas será necessário um *menu*. A aplicação também deverá se adaptar para que seja possível seu uso em dispositivos móveis.

A aplicação está dividida em características a quais servirão de avaliação para o AHP. As características estão discriminadas durante a seção e agrupadas conforme sua área. Estas áreas foram definidas conforme a necessidade da implementação de uma aplicação rica com as funcionalidades da emissão de um pedido *online*. Os grupos são: Usabilidade, Interatividade, Geolocalização, Gráficos e Animações, Aplicações *Offline*, Interação com o *Hardware* e Aplicações *Mobile*.

5.1 USABILIDADE

A usabilidade de um *software* corresponde à qualidade de uso de uma interface proporcionada ao usuário (Bevan, 1995). Para interfaces humano computador, o cuidado com a usabilidade se torna quase que obrigatória ao desenvolver a interação visual das aplicações. Se tratando de aplicações web, como visto anteriormente, muito evoluiu. Uma aplicação rica, deve se parecer com uma aplicação desktop (Deitel, 2009). Ela deve ser familiar ao usuário, facilitando o reconhecimento de padrões visuais que facilitem a sua utilização. Com vista

³⁵ <http://java.sun.com/developer/releases/petstore/> Disponível em 10/11/2011.

nesta premissa das aplicações ricas, as características a seguir possuem foco na usabilidade.

A tela de emissão de pedidos estará dividida em duas áreas. A primeira será onde o usuário irá informar os dados necessários ao pedido por meio de um formulário. Na segunda haverá a possibilidade de seleção dos itens a serem vendidos. As informações a serem selecionadas no pedido, bem como os dados dos gráficos estatísticos, serão estáticos. Ou seja, não haverá a necessidade de armazenamento em um banco de dados ou qualquer outra forma de persistência. Isto se deve ao foco deste trabalho estar relacionada à camada de apresentação da aplicação.

5.1.1 CARACTERÍSTICA 1 – INFORMAÇÕES GERAIS DO PEDIDO

Na Figura 23 está representado o protótipo da tela inicial do cadastro de pedidos. Nesta tela estão dispostos campos que exemplificam algumas informações básicas utilizadas em um cadastro de pedidos. Os campos escolhidos foram o cliente, data e emissão, representante, situação do pedido, forma de pagamento e um campo livre para as observações. Nesta tela ainda haverá um mapa exibindo o trajeto entre o local da emissão do pedido para com o distribuidor mais próximo.

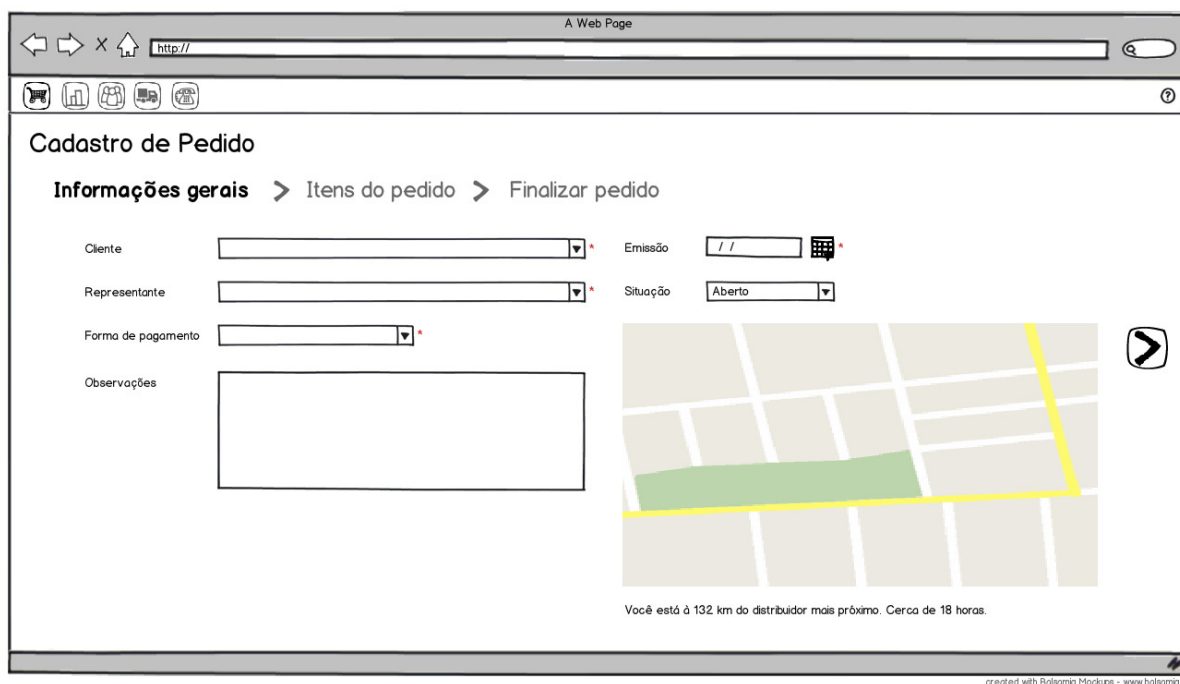


Figura 23 - Tela de informações gerais do pedido.

Na tela de informações gerais serão avaliadas as implementações das tecnologias de

um modo geral. A performance não se torna crítica nesta característica. Nesta, o mais importante é que a tela funcione em todos os navegadores. Além disso, é necessário que haja facilidade de desenvolvimento, o que representa a produtividade da tecnologia, e principalmente que possua semelhança com a usabilidade das aplicações *desktop*.

5.1.2 CARACTERÍSTICA 2 – ITENS DO PEDIDO

A tela de adição de itens está representada na Figura 24. Ao lado esquerdo da tela há uma listagem dos produtos disponíveis para o pedido. Acima dos mesmos há um campo de pesquisa para a realização de filtro pelo nome dos produtos. Os itens já adicionados ao pedido encontram-se ao lado direito da tela. Estes por sua vez possuem um campo para indicar a quantidade solicitada do item. A listagem de produtos a serem selecionados ocupa um espaço maior na tela. Isto se deve ao maior número de produtos disponíveis em relação à quantidade de produtos selecionados. Tanto a listagem de produtos quanto a de itens selecionados no pedido possuirão barras de rolagens independentes. Isto facilitará a navegação dos usuários.

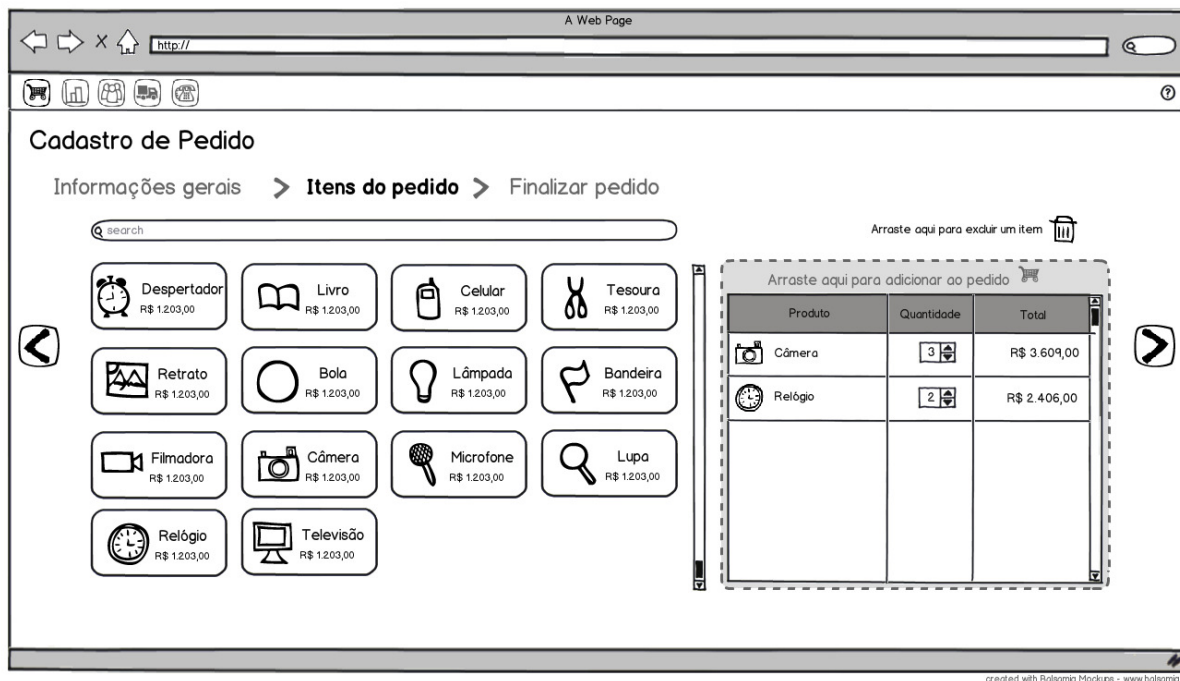


Figura 24 - Tela de itens do pedido.

A avaliação desta tela se dará da mesma forma que a tela de informações gerais, sendo

que a performance geral não é crítica. A compatibilidade se torna importante por novamente representar a produtividade da tecnologia. Além disso, se faz presente a necessidade de avaliar a facilidade e semelhança com aplicações *desktop* avaliando a implementação das duas listas, da barra de rolagem e da pesquisa.

5.1.3 CARACTERÍSTICA 3 – NAVEGAÇÃO DO PEDIDO

Para identificar as telas e percorrer entre elas, na parte superior, haverá uma sequência lógica das ações a serem tomadas. Pode-se observar em destaque na Figura 25, a navegação do cadastro de pedido. Esta representa a sequência que deve ser seguida para cadastrar um pedido. Ou seja, primeiro preenche-se as informações gerais, posteriormente adiciona-se os itens desejados e então finaliza-se o pedido. Na Figura 25, a navegação referente aos itens do pedido está em destaque representando que a tela encontra-se em posição de adição de itens. O mesmo deve ocorrer para os demais itens da navegação.

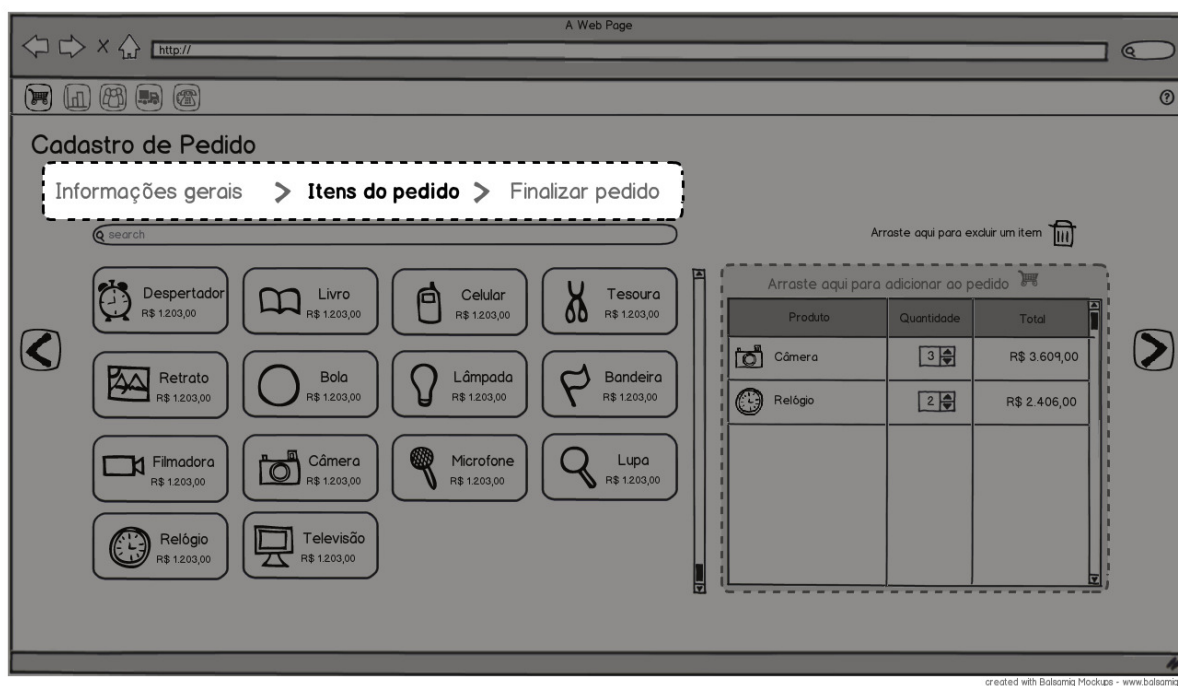


Figura 25 - Navegação entre as telas do pedido.

A avaliação da performance pouco se aplica à esta característica. Por sua vez, esta deve ser compatível com os navegadores tendo a necessidade de compatibilidade elevada. Por ser uma funcionalidade simples, os critérios de facilidade e de semelhança dos requisitos são

relevantes porem não críticos.

5.2 INTERATIVIDADE

Talvez a característica mais marcante do *desktop* seja a interatividade. As aplicações *desktop* não necessitam de uma comunicação constante entre cliente e servidor para obter as funcionalidades. Isto torna a resposta da aplicação praticamente instantânea havendo apenas a comunicação dos dados. Em uma aplicação rica esta familiaridade com aplicações *desktop* deve ser mantida. Alguns elementos facilitam atribuir essa semelhança às aplicações *web* (Niederauer, 2007). São eles:

- Drag-and-Drop³⁶: é a possibilidade de arrastar e soltar um componente para selecioná-lo. Esta funcionalidade, apesar de não ser muito explorada em aplicações de negócio, abre possibilidades de interações mais ricas que as páginas convencionais, onde o recuso não é utilizado.
- Componentes de Formulários: além dos padrões de *Combos* e *Texts*, o uso de componentes para seleção de data e integração de mídia à formulários levam uma experiência rica para o usuário. Campos de incremento numérico facilitam a alteração de um valor deixando desnecessário o uso do teclado.
- Validação Local: ao informar algum valor em um campo, em alguns momentos é necessária uma validação instantânea. Realizar uma comunicação com o servidor para isso pode ser muito custoso, deixando o usuário esperando para que a validação ocorra. A solução é executar este tipo de validação diretamente na aplicação cliente.
- Atualização Parcial da Tela: durante a digitação de um formulário, há momentos que é inevitável a comunicação com o servidor. Para isso, como explicado na Seção 2.4, existem tecnologias como o AJAX para que não haja a necessidade da atualização total da página.
- Transição de Telas: visando a aumento do uso de dispositivos sensíveis ao toque, a aplicação deverá comportar uma transição entre uma tela e outra adaptada a esta realidade. Esta funcionalidade deve ser transparente em casos em que o usuário

³⁶ <http://www.whatwg.org/specs/web-apps/current-work/multipage/dnd.html#dnd> Disponível em 14/10/2011.

interagir através do dispositivo *mouse*. O site³⁷ promocional da *Blackberry* possui comportamento semelhante ao proposto.

5.2.1 CARACTERÍSTICA 4 – BUSCA DE ITENS NO PEDIDO

Na aplicação de referência as características inerentes à interatividade estarão distribuídas na tela de cadastro de produtos e emissão de pedido. Na lista de produtos disponíveis haverá um campo possibilitando uma filtragem dos itens pelo nome parcial ou completo do produto. Este filtro deve trabalhar com processamento local, sem realizar comunicação com o servidor. Isto agregará performance à esta funcionalidade, já que o próprio navegador realiza a filtragem evitando assim, a espera excessiva do usuário. Esta funcionalidade está exemplificada na Figura 26.

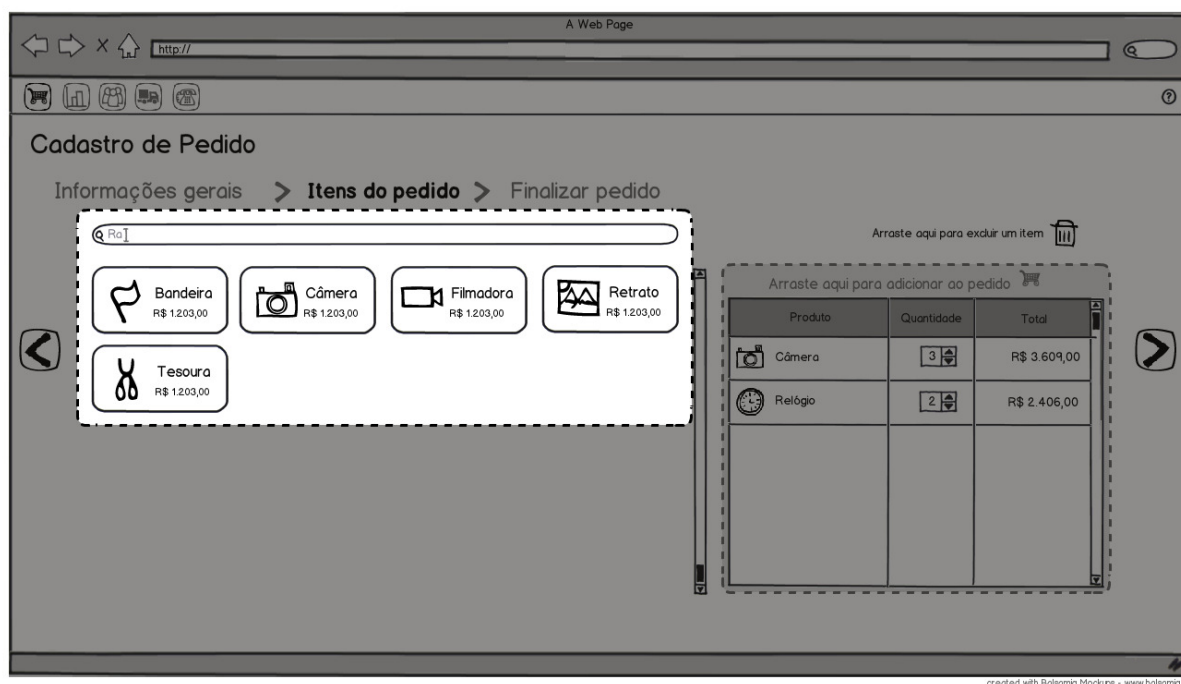


Figura 26 - Filtro dos produtos disponíveis.

É possível observar a realização da filtragem dos produtos na Figura 26. No caso, o usuário iniciou digitando “Ra” no campo e logo todos os itens foram filtrados. Os itens exibidos não precisam necessariamente iniciar com o texto digitado, eles apenas devem conter o texto usado para filtragem em seu nome.

O critério performance possuirá relevância elevada observando a memória utilizada

³⁷<http://br.blackberry.com/> Disponível em 25/10/2011.

pelo navegador, através do gerenciador de tarefas sistema operacional, e o tempo de resposta do filtro. Por ser uma funcionalidade que depende de execução diretamente no navegador a compatibilidade se torna importante. Tendo em vista a simplicidade desta implementação, a relevância do critério de facilidade de desenvolvimento é relativamente baixa. Em questões de semelhança com aplicações *desktop*, pode-se afirmar que esta característica possui impacto médio/alto.

5.2.2 CARACTERÍSTICA 5 – ADIÇÃO DE ITENS NO PEDIDO

Para adicionar um produto ao pedido bastará arrastá-lo para a região dos itens do pedido por meio de *drag-and-drop*. Na Figura 8 pode ser vista esta funcionalidade. Ao soltar o item em qualquer parte da listagem dos itens selecionados no pedido, o mesmo irá aparecer ao final da lista. Quando um item for adicionado, o foco deverá ficar no campo de quantidade solicitada.

O procedimento para remover o item será semelhante. Desta vez arrasta-se o item da lista de itens selecionado para a região superior onde está escrito “Arraste aqui para remover um item”. Ao soltar o item nesta região o mesmo não deverá mais aparecer nos itens solicitados. A adição e remoção dos itens devem ocorrer sem que a tela inteira seja atualizada. Ou seja, a tela deverá ser atualizar somente a região necessária, no caso os itens solicitados.

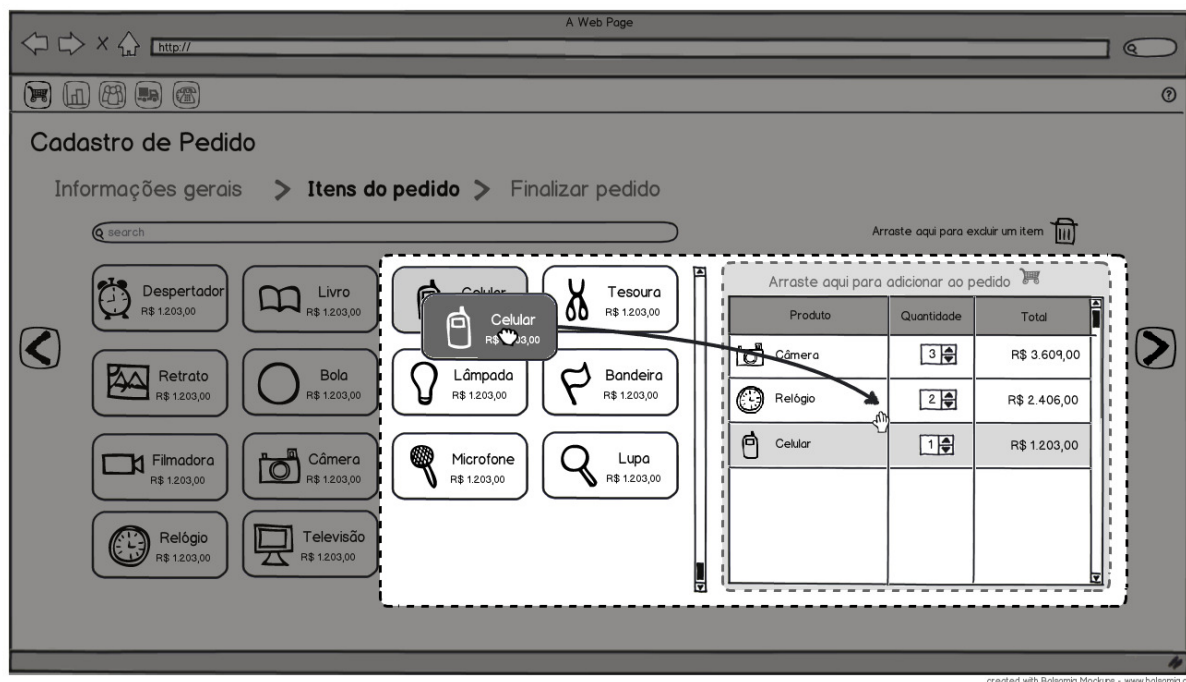


Figura 27 - Adição dos itens no pedido.

Na Figura 27 observa-se a adição dos itens no pedido. O usuário clicou no produto celular, o arrastou até a listagem de itens selecionados e então soltou. Com isso o produto apareceu ao final da lista com o foco na quantidade do item. Esta é uma funcionalidade que exige uma interação diferenciada do usuário e uma renderização dinâmica ao arrastar os itens. Com isso, a performance, analisada pela memória do navegador e pelo processamento do computador, possui impacto elevado. O mesmo ocorrerá com a compatibilidade e a facilidade de desenvolvimento por não ser uma funcionalidade tão convencional na *web*. Da mesma forma, a semelhança possui relevância alta por *drag-and-drop* ser mais utilizado em aplicações *desktop*.

5.2.3 CARACTERÍSTICA 6 – QUANTIDADE SOLICITADA DOS ITENS

Quando for adicionado um item no pedido, por padrão a quantidade solicitada será 1, sendo que essa poderá ser alterada. Ao digitar outra quantidade, o número não deverá ultrapassar a quantidade em estoque, esta validação também deverá ser feita diretamente no navegador. Se o valor digitado for inválido deverá ser exibida uma mensagem e a quantidade deve ser definida como a quantidade em estoque do produto. A mensagem está exemplificada na Figura 9. Esta quantidade estará informada fixa no produto. Quando forem usadas as setas

para incrementar a quantidade, esta simplesmente deve respeitar a condição descrita acima sem a necessidade de mensagens.

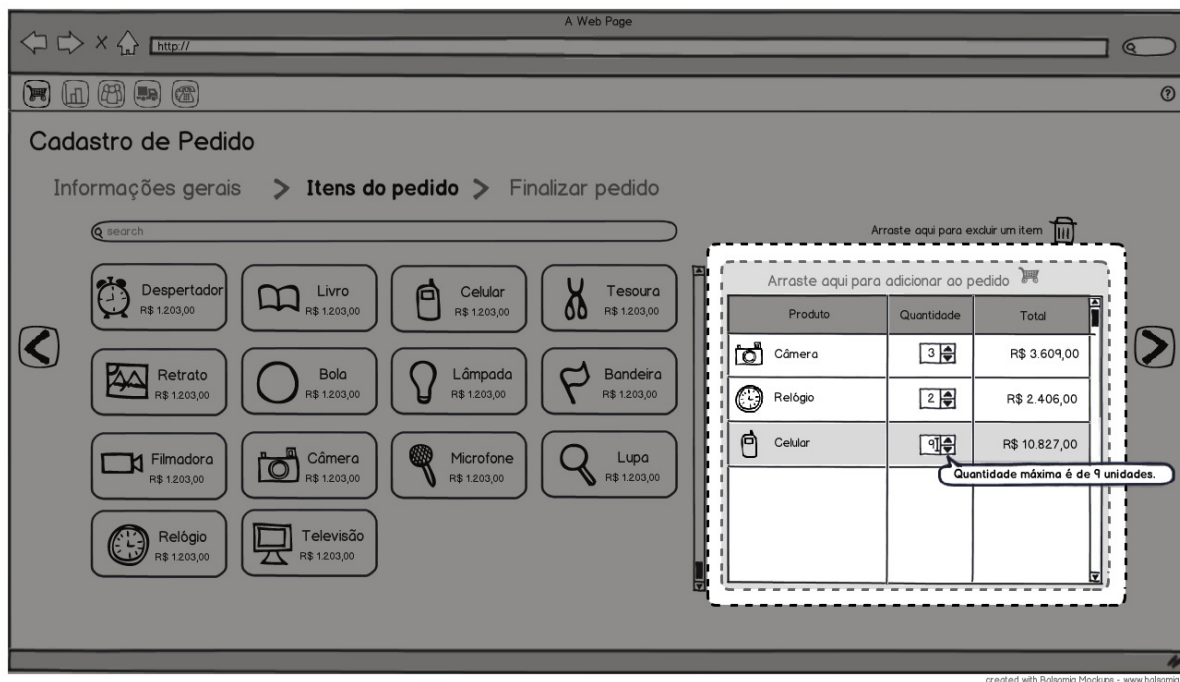


Figura 28 - Validação da quantidade solicitada no item.

Observa-se na Figura 28 a validação da quantidade do item solicitado. O produto possui estoque igual a 9. Pelo fato do usuário ter digitado um valor maior que a quantidade em estoque, o sistema exibiu uma mensagem dizendo “Quantidade máxima é de 9 unidades”. Em seguida o valor do campo foi alterado para 9.

Nesta característica o critério de performance possui relevância baixa por não ser crítica. Havendo a necessidade de exibir a mensagem na tela dinamicamente será avaliada a compatibilidade e a facilidade. Já o a semelhança com aplicações *desktop* se torna de prioridade média/alta por utilizar elementos de interatividade com o usuário

5.2.4 CARACTERÍSTICA 7 – VALIDAÇÕES DAS INFORMAÇÕES GERAIS DO PEDIDO

A aplicação irá explorar melhor a validação das informações diretamente no navegador. Para isso será validada a obrigatoriedade dos campos da tela de informações gerais. Cada campo obrigatório será exibido com um sinal de asterisco ao lado representando

que é necessário seu preenchimento. Se o usuário tentar avançar para a tela de itens sem digitar algum desses campos mensagens devem ser exibidas. Haverá uma mensagem para cada campo não preenchido. Além disso, o sinal ao lado do campo irá mudar para uma exclamação. As mensagens aparecerão logo ao lado dos campos não preenchidos. Estas mensagens não devem sobrepor umas as outras, isto deve ser controlado ao exibi-las. Os campos devem ser circundados por traços na cor vermelha chamando a atenção do usuário conforme exemplificado na Figura 29.

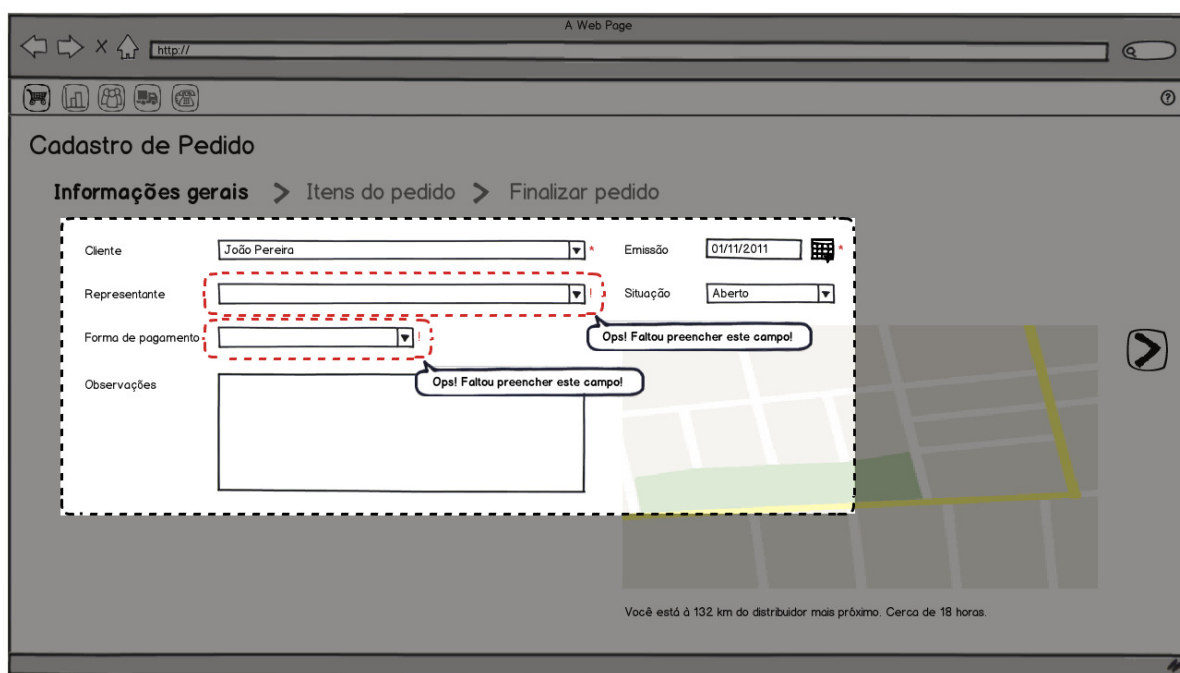


Figura 29 - Validação da obrigatoriedade dos campos.

Na Figura 29 estão sendo exibidas as mensagens de notificação de não preenchimento. A obrigatoriedade dos campos é representada pelo asterisco vermelho ao lado de cada campo. Neste caso, os campos não informados foram o de representante e o de forma de pagamento. Estes aparecem destacados por um contorno avermelhado e uma mensagem informando que faltou o preenchimento dos mesmos.

Nesta a performance se torna irrelevante por não necessitar processamento e nem memória excessiva do computador. A posição das mensagens é dinâmica e não se trata de uma característica típica, pois a imagem indicativa sobrepõe outros controles na tela. Com isso os graus de relevância dos critérios de compatibilidade e facilidade e semelhança são médio, médio e alto, respectivamente.

5.3 GEOLOCALIZAÇÃO

Na seção anterior foram apresentadas características referentes a interatividade do usuário com a aplicação. Elementos como arrastar e soltar, validações e rotinas executadas diretamente no navegador foram propostas. Também foram definidos relevâncias de avaliação dos mesmos. Nesta seção será proposta a utilização da geolocalização na aplicação de referência.

Localizar-se no globo terrestre nem sempre foi fácil. Por muitos anos utilizou-se a posição solar, das estrelas, bússolas e até mesmo sinal de fumaça para se localizar. Com a tecnologia outros métodos surgiram como a rádio frequência e o GPS (*Global Position System*). O GPS localiza a posição no globo através da triangulação de satélites. Para haver uma localização precisa, incluindo altitude, é necessário o uso de no mínimo 4 satélites. O que antes servia como localização em navegações atualmente a geolocalização agrega funcionalidade a aplicações na internet deixando o conteúdo mais rico (Holdener, 2011).

A localização do acesso de um determinado usuário à uma página na internet pode ocorrer de duas formas. Por acessos ao GPS ou localização inferida à conexão de Rede. Nesta segunda é possível obter a localização por meio do endereço de IP, RFID³⁸, *Bluetooth*, *Wi-Fi*, endereço MAC ou até mesmo pela rede de telefonia (W3C, 2010).

Em aplicações *web* existem diversas utilidades para a geolocalização. Por exemplo, é possível exibir conteúdo personalizado conforme a sua região. Mecanismos de busca podem utilizar o seu local para lhe oferecer serviços que estão próximos a você. A integração com mapas facilitam a criação de rotas necessitando apenas do endereço destino³⁹. Também é possível o uso de realidade aumentada⁴⁰ integrando com a geolocalização para, por exemplo, obter informações sobre determinado local, rua ou prédio (Holdener, 2011).

5.3.1 CARACTERÍSTICA 8 – LOGÍSTICA NA EMISSÃO DO PEDIDO

Na aplicação de referência será utilizada a localização do usuário para exibir sua distância à determinado ponto de logística. Também será exibido um mapa com o trajeto até o

³⁸ http://www.aimglobal.org/technologies/rfid/what_is_rfid.asp Disponível em 13/10/2011.

³⁹ <http://www.tecmundo.com.br/3659-o-que-e-geolocalizacao-.htm> Disponível em 13/10/2011.

⁴⁰ Realidade aumentada é uma combinação de uma visão do mundo real, através do uso de câmeras, com a sobreposição de imagens (Holdener, 2011).

mesmo. A distância e o mapa serão obtidos a partir de uma API disponível pela Google chamada de *Google Maps API*⁴¹. Para fins de estudo o endereço de origem será fixo o de São Paulo - SP. Esta informação deverá ser exibida na tela de emissão do pedido conforme exibido na Figura 30. Caso não for possível localizar o endereço do usuário uma mensagem alertando isto deverá ser exibida no local.



Figura 30 - Trajeto até o distribuidor mais próximo.

Pode-se ver na Figura 30 a exibição de um mapa com o trajeto do distribuidor até a localização do usuário. Abaixo deste mapa encontra-se a mensagem descrevendo a distância até o distribuidor e o tempo do trajeto. Ambas as informações são disponibilizadas através da API do Google especificada anteriormente.

Por se tratar de um acesso a recursos de *hardware*, a relevância da performance se torna prioritária. Pelo mesmo motivo a compatibilidade e facilidade de desenvolvimento da implementação se tornam relevantes. Já critério de semelhança com aplicações *desktop* possui relevância baixa por não ser uma característica que traz usabilidade e interatividade com o usuário.

⁴¹ <http://code.google.com/intl/pt-BR/apis/maps/> Disponível em 13/10/2011.

5.4 GRÁFICOS E ANIMAÇÕES

No HTML5 foram introduzidas novas formas de renderização de imagens utilizando canvas e SVG (David, 2010 p xiii). Estes elementos novos são apontados como a grande vantagem contra tecnologias como *Flash*. Em sua carta sobre o *Flash*, Steve Jobs (2010) aponta a criação de gráficos ricos em HTML5 um dos motivos para não utilizar o *Flash* nos dispositivos da *Apple*.

5.4.1 CARACTERÍSTICA 9 – GRÁFICOS ESTATÍSTICOS

Com o objetivo de comparar as tecnologias propostas será desenvolvido um gráfico estatístico. Este será implementado conforme detalhamento a seguir. O gráfico deverá buscar a maior interatividade possível com o usuário. O objetivo é comparar a capacidade de renderização dinâmica entre tecnologias estudadas.

Para a elaboração dos gráficos será exibida a representatividade dos clientes. Um gráfico de estilo pizza mostrará a participação de cada cliente em relação ao total de vendas do ano. Neste gráfico será possível a seleção de um cliente como demonstrado na Figura 31. Esta seleção deve ocorrer destacando a porcentagem selecionada do gráfico. Ao mesmo tempo será exibida uma caixa com informações como a porcentagem da participação, o valor das vendas e o ano da ativação do cliente. Ao lado será exibido um gráfico no estilo de barras, e neste será demonstrada a porcentagem da representatividade do cliente em cada mês do ano corrente. Ao passar o cursor nas barras deverá ser exibida a porcentagem referente.



Figura 31 - Gráfico da representatividade de clientes.

O gráfico estatístico está exemplificado na Figura 31. Nela é possível observar que o usuário selecionou um cliente no gráfico de pizza. Neste momento exibiram-se informações referentes a ele como a participação de 48%. Estas informações foram exibidas com transparência. Também foi exibido outro gráfico à direita sendo este em formato de barras. Neste será demonstrado o histórico mensal de representatividade do cliente selecionado.

Tendo em vista a criticidade de animações gráficas a performance possui grau de relevância alto. O critério de compatibilidade possui importância media/alta. Isto se deve à necessidade de renderização igual em todos os navegadores, o que hoje nem sempre acontece. Por se tratar de uma funcionalidade mais trabalhosa e que relativamente não é comum na *web*, a relevância para a facilidade e a semelhança terá sua avaliação com importância elevada.

5.4.2 CARACTERÍSTICA 10 – TRANSIÇÃO DE TELAS NO PEDIDO

Serão exploradas também as animações dos elementos visuais. Por exemplo, na troca de telas, ao navegar entre elas, as telas deverão ser exibidas de forma gradual. Estas transições deverão ser usadas sempre que possível. Isto transparece ao usuário uma interação mais

agradável com a aplicação. Deverá também ser aproveitada a aceleração gráfica⁴² do hardware quando necessário, pois as animações tem um papel crítico na performance da aplicação cliente.

As transições entre telas serão adaptadas à realidade dos dispositivos sensíveis ao toque. Por isso não será utilizada navegação por abas. Na Figura 32 é possível observar a interação da aplicação por meio do toque na tela. No caso, ao arrastar a tela para o lado esquerdo ocorre a transição para a tela de itens do pedido. O mesmo deve ocorrer quando clicado nas setas laterais ou na navegação superior.



Figura 32 - Transição entre as telas do pedido.

Observa-se na Figura 32 que a tela está dividida entre as informações gerais e os itens do pedido. Isto é para exemplificar o efeito de transição. No caso, o usuário clicou e arrastou a tela para a esquerda provocando a troca de tela.

Assim como o gráfico estatístico, nesta a animação gráfica requerer um maior peso na avaliação do critério de performance. A compatibilidade e facilidade possuirão relevância relativamente alta por ser uma característica avançada. Já a semelhança com a aparência de aplicações desktop possuirá peso mediano.

⁴² Aceleração gráfica é o uso de placas de vídeo que auxiliam o processador principal do computador para renderizar imagens (Martins, 2007 p. 83).

5.5 APLICAÇÕES *OFFLINE* E ARMAZENAMENTO LOCAL

Aplicações *desktop* possuem a vantagem de não depender da internet para que funcionem. Uma aplicação *web* rica deve ter a capacidade de continuar funcionando mesmo sem conexão com internet. Esta funcionalidade se torna importante quando estamos executando uma aplicação em um dispositivo móvel. Conectados em uma rede 3G ou EDGE (*Enhanced Data rates for GSM Evolution*), há a possibilidade de ficar sem cobertura, como em um túnel ou metrô (David, 2010 p. 33).

A fim de evitar o problema citado acima, as aplicações *web* devem disponibilizar suas funcionalidades mais importantes de forma a funcionarem *offline* temporariamente. Isto possibilita que o usuário continue utilizando a aplicação sem interrupções. A aplicação também deve sincronizar com o servidor quando retomar a conexão.

5.5.1 CARACTERÍSTICA 11 – SUPORTE A ARMAZENAMENTO LOCAL

Na aplicação de referência, esta funcionalidade será implementada no pedido. O pedido será cadastrado normalmente. No momento de finalizar o pedido, caso não haja conexão com a internet, o pedido será armazenado localmente. Assim que retomada a conexão, o sistema deve automaticamente sincronizar os pedidos pendentes com o servidor. Ao tentar finalizar o pedido, deve ser exibida uma mensagem informando que o pedido foi salvo, porém não há conexão e assim que houver o pedido será enviado.

Esta implementação será experimental. O objetivo é testar capacidade de suporte ao armazenamento *offline*. Porém é possível que não haja esta funcionalidade em todas as tecnologias. Logo, esta característica pode se tornar um diferencial. Com isso o grau de relevância do critério de facilidade de desenvolvimento será elevado. A compatibilidade, sendo parcialmente crítica terá relevância média. Já a performance, por não necessitar de uso significativo de processador e/ou memória terá relevância média/baixa.

Para a que esta característica seja atendida a aplicação deverá contemplar as seguintes funcionalidades: avaliar se há conexão com a internet, disponibilizar as páginas mesmo *offline*, armazenar o conteúdo no local temporariamente e sincronizar com o servidor assim que retomada a conexão.

5.6 INTERAÇÃO COM HARDWARE LOCAL

Por ser desenvolvida para execução local, uma aplicação *desktop* normalmente possui interação íntegra ao hardware do computador. Já uma aplicação *web* tradicional possui dificuldades de realizar tal interação. Por exemplo, a comunicação com a *webcam* e o microfone são raros sem o auxílio de *plug-ins*.

5.6.1 CARACTERÍSTICA 12 – USO DE *WEBCAM* PARA OBTER PROMOÇÕES NO PEDIDO

Com a finalidade de testar esta interação das aplicações ricas, esta será utilizada na aplicação de referência. No cadastro de pedido será possível acessar a *webcam*. No contexto da aplicação esta funcionalidade simulará uma promoção no valor total dos itens. O usuário acessará a *webcam* a partir do botão localizado acima dos itens selecionados (ver Figura 33). Com a câmera ativa, ao ser exibido um *Qrcode*⁴³ promocional, automaticamente os itens sofrerão desconto de 10%.

Em uma aplicação real este *Qrcode* poderia ser distribuído em folhetos, jornais, revistas entre outros. Ele nada mais é que um direito de desconto concedido ao usuário, ou seja, um cupom promocional. Para fins técnicos, o reconhecimento do *Qrcode* não será obrigatório. Isto se deve ao fato de estar sendo avaliando o acesso ao hardware. Nesta funcionalidade o importante será acessar a *webcam*.

⁴³ *Qrcode*: espécie de código de barras em 2D de fácil acesso por ser reconhecido através da *webcam*. Além disso, ele possibilita infinitas variações de conteúdo.

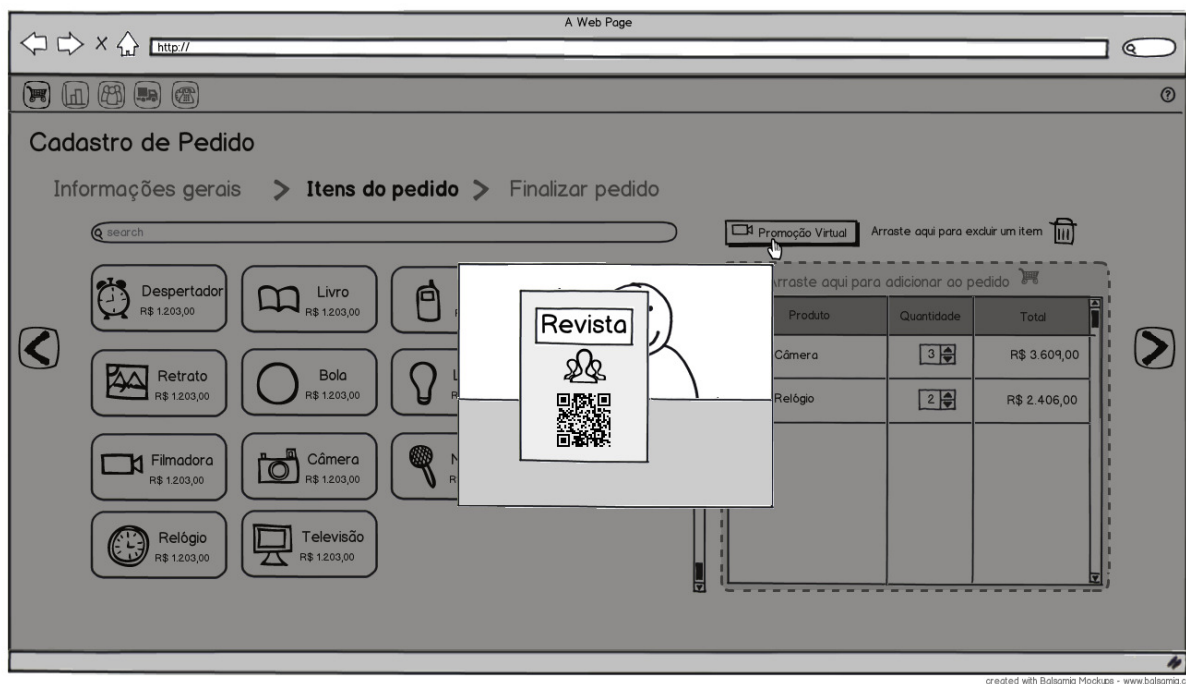


Figura 33 - Acesso da webcam para obter descontos no pedido através de um Qrcode promocional.

A Figura 33 exemplifica esta funcionalidade. Nela o usuário clicou no botão chamado de “Promoção Virtual” o que ativou o acesso a webcam. Com a webcam ativa, o usuário exibiu uma revista que possui um Qrcode promocional. Tendo o sistema reconhecido o código promocional, a webcam será desativada e a aplicará a promoção obtida nos itens do pedido.

Por cada tecnologia implementar a sua forma de acesso à webcam e no caso do HTML5 necessitar o suporte do navegador as características de performance, compatibilidade e facilidade possuirão relevância média/alta. Já a semelhança de aplicações desktop será alta já que uma tecnologia poderá simplesmente atender ou não à característica, se tornando assim um diferencial.

5.7 APLICAÇÕES MOBILE

As aplicações para smartphones estão muito populares atualmente. As ferramentas de lojas virtuais para dispositivos móveis vêm crescendo com enormes quantidades de aplicativos. Porém esses aplicativos necessitam download e são dependentes da plataforma. Isto acarreta num retrabalho por parte dos programadores necessitando a implementação de uma aplicação para plataforma como Android (Google), iOS (Apple), Windows Phone (Microsoft), entre outros (Oehlman e Blanc, 2011 p. xiii).

O desenvolvimento de aplicações *web* evita este retrabalho. A interoperabilidade da *web* possibilita a escrita de uma aplicação que funcione independente da plataforma em que é executada. Porém, os dispositivos móveis possuem características que devem ser tratadas.

A resolução da tela é uma delas. A variação nesse item é grande. As dimensões variam de aparelho para aparelho tendo ainda telas maiores como os *tablets*. Uma aplicação *web* deve ter a capacidade de se adaptar automaticamente conforme a resolução em que é executada de forma transparente ao usuário.

As aplicações *web* para *desktop* são baseadas principalmente para a interação através do dispositivo *mouse*. Nos dispositivos móveis é comum o uso de telas sensíveis ao toque. Isto gera transtornos ao usuário ao ter que acessar uma página desenvolvida para *desktop* em seu *smartphone* por exemplo. Elementos com dimensões pequenas se tornam difíceis de ser manipulados. Esta dificuldade é conhecida de síndrome de “*fat-finger*” (dedo gordo), que ocorre quando for selecionado um elemento na tela e este possuir dimensões reduzidas, acidentalmente um elemento ao lado é selecionado. Uma alternativa para isso é disponibilizar o *zoom* da aplicação e/ou exibir os controles em dimensões maiores (Voids, 2009).

5.7.1 CARACTERÍSTICA 13 – SUPORTE A DISPOSITIVOS MÓVEIS

Tendo isto em vista, a aplicação de referência deverá respeitar as características dos dispositivos móveis. Em questão de interface, os elementos devem ser exibidos de forma a facilitar a interação por telas sensíveis ao toque. Já a disposição de elementos deve se adaptar à resolução imposta pelo aparelho.

Nesta funcionalidade, a performance se torna relevante apenas em relação ao consumo da bateria. A compatibilidade é importante por haver navegadores distintos nos dispositivos móveis. O critério facilidade de desenvolvimento é crítica, pois esta é uma característica aplicada em todas as telas, o que afeta diretamente produtividade de desenvolvimento. Por fim, a semelhança com aplicações *desktop* se torna quase que irrelevante já que as mesmas não possuem tal característica.

5.8 CONSIDERAÇÕES

Neste capítulo foi descrita a aplicação de referência. Nela estão especificadas funcionalidades pertinentes às aplicações ricas na internet. Para a avaliação das implementações, foi descrito um método avaliação baseado no AHP. Este servirá para comparar as características da aplicação de referência nas tecnologias propostas. No próximo capítulo será realizada a comparação das tecnologias aplicando o modelo proposto.

6 AVALIAÇÃO

A avaliação é o momento em que é posto em prática o que foi especificado no modelo de avaliação. Com os fundamentos do AHP detalhados e com base nos critérios propostos a próxima etapa é elaborar as Matrizes de Avaliação. Nestas, todos os critérios serão confrontados a fim de definir a influência de cada um na tomada de decisão. As comparações seguiram as especificações descritas no método de avaliação (Capítulo 3). As notas atribuídas seguem o intervalo de 1 a 9 definidas na Escala Fundamental de Saaty (Tabela 1). Além disso, é aceitável graus de inconsistências de até 10% segundo Vargas (2010).

	Performance	Compatibilidade	Facilidade	Semelhança	Totais
Performance	1	2	4	1	37,7%
Compatibilidade	1/2	1	2	1/2	18,8%
Facilidade	1/4	1/2	1	1/2	11,4%
Semelhança	1	2	2	1	32,1%
				Inconsistência	2%

Tabela 5 - Comparativo dos critérios de avaliação.

Na Tabela 5 estão relacionados os critérios de avaliação. Esta comparação define qual critério será mais relevante na tomada de decisão. É possível observar que a performance se destaca com 37,7% de relevância. Isto se deve a grande necessidade de recursos que uma aplicação interativa e com alta usabilidade necessita. Usabilidade essa representada no critério de semelhança com aplicações *desktop*. Este critério ficou em segundo com 32,1% por ser uma das características mais relevantes nas aplicações ricas. A compatibilidade aparece com 18,8% que se deve a responsabilidade dos navegadores suportar as tecnologias. Por fim, a facilidade de desenvolvimento aparece menos relevante por se tratar de um critério que pode ser suprido com a realização de treinamentos e contratação de profissionais experientes.

As relevâncias encontradas podem variar conforme a necessidade da aplicação. Caso a preocupação seja com a equipe de desenvolvimento e seus conhecimentos, pode-se definir a facilidade como um critério de relevância alta. Da mesma forma, se os usuários da aplicação utilizam ambientes com navegadores diversificados, o critério de compatibilidade passa a ter

maior importância.

Conforme especificado na Seção 4.3, onde é apresentada a hierarquia dos critérios de avaliação, a seguir estão relacionadas todas as características descritas na aplicação de referência. Para facilitar os comparativos, as mesmas serão mapeada de C1 à C13 conforme apresentada na Tabela 6.

Abreviação	Descrição
C1	Informações gerais do pedido.
C2	Itens do pedido.
C3	Navegação do pedido.
C4	Busca de itens no pedido.
C5	Adição de itens no pedido.
C6	Quantidade solicitada dos itens.
C7	Validações das informações gerais.
C8	Logística na emissão do pedido.
C9	Gráficos estatísticos.
C10	Transição de telas no pedido.
C11	Suporte a armazenamento local.
C12	Uso de <i>webcam</i> para obter promoções no pedido.
C13	Suporte a dispositivos móveis.

Tabela 6 - Listagem das características da aplicação de referência.

Estas características foram implementadas nas tecnologias propostas neste trabalho. As características de suporte a armazenamento local (C11) e suporte a dispositivos móveis (C12) não puderam ser implementadas devido sua complexidade. O desenvolvimento destas necessitaria maiores recursos para serem avaliadas como a utilização de *smartphones* e *tablets*. Além disso, sua execução acarretaria na prolongação da duração deste projeto, o que impossibilitaria sua conclusão em tempo hábil.

6.1 RELEVÂNCIA DAS CARACTERÍSTICAS DE AVALIAÇÃO

Cada uma das características relacionadas na Tabela 6 possuem relevâncias distintas

ao aplicar os critérios de avaliação. Esta diferença de relevância é prevista pelo método analítico hierárquico especificado. Para defini-la é necessário confrontar diretamente cada característica a fim de encontrar o percentual de relevância para determinado critério.

As características de suporte a armazenamento local (C11) e suporte a dispositivos móveis (C12) não terão analisadas suas respectivas relevâncias. Isto se deve ao motivo que estas não foram implementadas na aplicação de referência.

6.1.1 CRITÉRIO DE PERFORMANCE

A performance a ser avaliada se refere a utilização de hardware. Definido na Seção 4.2.1, este critério será mais relevante às características que possuem animações gráficas, ou necessidade de uso de hardware como memória, CPU e *webcam*. Atenta-se para o fato da característica C3 (navegação do pedido) não estar relacionada pela sua impossibilidade de avaliação neste critério. Esta peculiaridade está descrita na Seção 6.3.3. Na Tabela 7 estão relacionados os resultados obtidos ao comparar cada característica em relação sua relevância para o critério de performance.

	Critério de performance										Totais
	C1	C2	C4	C5	C6	C7	C8	C9	C10	C12	
C1	1	1/3	1/2	1/4	1/2	1/2	1/4	1/6	1/5	1/4	2,6%
C2	3	1	1	1/3	1/2	1	1/3	1/5	1/4	1/3	4,3%
C4	2	1	1	1/4	1	1	1/3	1/5	1/4	1/4	4,1%
C5	4	3	4	1	3	3	1	1/3	1/2	1/2	10,9%
C6	2	2	1	1/3	1	1	1/3	1/4	1/4	1/3	4,8%
C7	2	1	1	1/3	1	1	1/3	1/4	1/4	1/3	4,4%
C8	4	3	3	1	3	3	1	1/3	1/2	1	11,1%
C9	6	5	5	3	4	4	3	1	3	3	26,9%
C10	5	4	4	2	4	4	2	1/3	1	3	18,5%
C12	4	3	4	2	3	3	1	1/3	1/3	1	12,3%
Inconsistência										3%	

Tabela 7 - Relevância de cada característica para o critério de performance.

Nas comparações realizadas é possível destacar alguns resultados. Na Tabela 7 é clara a

relevância elevada da característica C9 (gráficos estatísticos) com 26,9% seguida pela C10 (transição de telas no pedido) com 18,5% e a C12 (uso de webcam). Isto se deve a necessidade de renderização gráfica destas características. Como menos relevante para o critério de performance a característica C1 atingiu apenas 2,6% por não exigir demasiadamente de recursos computacionais.

6.1.2 CRITÉRIO DE COMPATIBILIDADE

O critério de compatibilidade avalia a execução das aplicações de referência implementadas nas tecnologias estudadas em cada navegador proposto. A execução de cada uma das aplicações deve ser constante, não apresentando variações de funcionalidade entre os navegadores. Este critério se torna crítico nas características que possuem funcionalidades mais complexas como aquelas que necessitam o uso de controles avançados. Quanto mais detalhes uma característica possuir, maior será a relevância do critério de compatibilidade. Os resultados das comparações realizadas estão dispostos na Tabela 8.

Critério de compatibilidade												Totais
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C12		
C1	1	1/2	6	5	3	5	3	3	2	3	3	18,6%
C2	2	1	6	5	3	6	4	3	3	3	3	23%
C3	1/6	1/6	1	1/2	1/4	1/3	1/3	1/3	1/4	1/4	1/3	2,3%
C4	1/5	1/5	2	1	1/3	1/2	1	1/3	1/4	1/3	1/3	3,1%
C5	1/3	1/3	4	3	1	3	2	1	1/2	1	1/2	7,6%
C6	1/5	1/6	3	2	1/3	1	1/2	1/3	1/3	1/2	1/2	3,9%
C7	1/3	1/4	3	1	1/2	2	1	1/3	1/4	1/3	1/2	4,4%
C8	1/3	1/3	3	3	1	3	3	1	1/2	1	1	8,1%
C9	1/2	1/3	4	4	2	3	4	2	1	3	2	13,2%
C10	1/3	1/3	4	3	1	2	3	1	1/3	1	1	7,8%
C12	1/3	1/3	3	3	2	2	2	1	1/2	1	1	8%
Inconsistência											3%	

Tabela 8 - Relevância de cada característica para o critério de compatibilidade.

Partindo dos princípios relacionados anteriormente, é possível observar que as

características com maiores funcionalidades neste critério se tornaram mais relevantes. Com 23% a característica de itens do pedido (C2) aparece em primeiro por sua especificação apontar a necessidade de desenvolvimento de duas listagens complexas, lista de produtos e itens do pedido. Em seguida a C1 (informações gerais do pedido) que define a utilização de diversos controles comuns em cadastro. Com menor relevância, neste critério, aparecem as características de navegação do pedido (C3) e validações das informações (C7), ambas implementações pontuais.

6.1.3 CRITÉRIO DE FACILIDADE DE DESENVOLVIMENTO

O critério de facilidade de desenvolvimento abrange além das dificuldades encontradas também a disponibilidade de documentação. Neste, as características mais complexas possuem maior relevância no resultado final. Porém, a complexidade abrange as funcionalidades que não sejam triviais de serem desenvolvidas. Isto inclui a renderização dinâmica de gráficos, acesso a *webcam* e ações de arrastar-e-soltar. A comparação realizada a fim de avaliar as relevâncias de cada característica está relacionada na Tabela 9.

Critério de facilidade de desenvolvimento												Totais
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C12		
C1	1	1/3	3	1	1/4	1/2	2	1/3	1/5	1/2	1/3	4,4%
C2	3	1	4	2	1/2	3	3	2	1/2	1	1	10,9%
C3	1/3	1/4	1	1/3	1/5	1/3	1/2	1/3	1/6	1/4	¼	2,3%
C4	1	1/2	3	1	1/3	1/2	1/2	1/3	1/5	1/3	¼	3,9%
C5	4	2	5	3	1	5	4	3	1	2	3	18,5%
C6	2	1/3	3	2	1/5	1	1/2	1/2	1/4	1/3	1/3	4,7%
C7	1/2	1/3	2	2	1/4	2	1	1/3	1/4	1/3	1/3	4,5%
C8	3	1/2	3	3	1/3	2	3	1	1/3	1/2	½	8,2%
C9	5	2	6	5	1	4	4	3	1	3	3	21%
C10	2	1	4	3	1/2	3	3	2	1/3	1	1	10,6%
C12	3	1	4	4	1/3	3	3	2	1/3	1	1	11%
Inconsistência											4%	

Tabela 9 - Relevância de cada característica para o critério de facilidade de desenvolvimento.

Baseado nas definições das características mais relevantes para o critério de facilidade de desenvolvimento, os resultados foram os seguintes. A característica de gráficos estatísticos (C9) apresentou-se com 21% de relevância. Isto ocorreu em função da necessidade de desenvolver gráficos dinâmicos e possuir diversos detalhes a serem implementados. Com 18,5% de relevância, a característica de adição de itens foi apontada por utilizar funcionalidades como o *drag-and-drop* que não é comum em aplicações *web*.

6.1.4 CRITÉRIO DE SEMELHANÇA COM APLICAÇÕES *DESKTOP*

Ao avaliar o critério de semelhança com aplicações *desktop* serão consideradas as implementações que possuem melhor usabilidade. Além disso, será avaliado se todas as funcionalidades propostas foram atendidas com sucesso. Neste caso as características mais relevantes serão as que possuem funcionalidades comuns em aplicações *desktop* como arrastar-e-soltar e acesso a *webcam*. Considerando estes pontos a serem avaliados, a Figura 10 relaciona as características confrontadas entre si a fim de definir a relevância de cada uma para este critério.

Critério de semelhança com aplicações <i>desktop</i>												
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C12	Totais
C1	1	1/3	2	2	1/3	2	2	2	1/3	1/2	1/2	7,1%
C2	3	1	3	3	1/2	3	3	3	1	1/2	1/2	11,9%
C3	1/2	1/3	1	1	1/4	1/2	1/2	1/2	1/3	1/3	1/3	3,5%
C4	1/2	1/3	1	1	1/3	1/2	1/3	1/2	1/4	1/3	1/3	3,4%
C5	3	2	4	3	1	3	3	2	1	2	1	15,7%
C6	1/2	1/3	2	2	1/3	1	1	3	4	3	3	4,4%
C7	1/2	1/3	2	3	1/3	1	1	3	3	3	3	4,8%
C8	1/2	1/3	2	2	1/2	1/3	1/3	1	2	2	2	7,3%
C9	3	1	3	4	1	1/4	1/3	1/2	1	2	2	12,6%
C10	2	2	3	3	1/2	1/3	1/3	1/2	1/2	1	2	13,4%
C12	2	2	3	3	1	1/3	1/3	1/2	1/2	1/2	1	15,9%
											Inconsistência	4%

Tabela 10 - Relevância de cada característica para o critério de semelhança com aplicações *desktop*.

Analisando os resultados é possível reparar que as características C12 e C5 estão praticamente empatadas obtendo 15,9% e 15,7% de relevância respectivamente. Isto se deve, pois uma especifica a utilização de *webcam* e a outra a adição de itens no pedido utilizando *drag-and-drop*. Além destas, a característica de gráficos estatísticos (C10) aparece próxima com 13,4% de relevância por não ser uma funcionalidade trivial de aplicações *web*.

6.2 IMPLEMENTAÇÕES DA APLICAÇÃO DE REFERÊNCIA

Nesta seção estão relatadas as experiências obtidas em função da utilização das tecnologias aplicadas nas implementações da aplicação de referência. As dificuldades e características encontradas durante as implementações estão organizadas conforme cada tecnologia. Os assuntos abordados procuram seguir a sequência das características especificadas da aplicação de referência, ou seja, primeiramente estão descritas as implementações de C1 - informações gerais do pedido, seguido por C2 - itens do pedido, C3 - navegação do pedido e assim por diante.

Ocasionalmente alguma característica não pôde ser implementada em determinada

tecnologia. Estes casos estão relatados e explicados os motivos de sua não implementação. As características C11 - suporte a armazenamento local e C13 - suporte a dispositivos móveis não foram implementadas em nenhuma tecnologia conforme descritos anteriormente neste capítulo.

No texto a seguir, algumas características estão exemplificadas com imagens dos resultados obtidos após as implementações. Trechos de códigos são exibidos a fim de demonstrar os esforços realizados em cada tecnologia. As implementações, na íntegra, estão disponíveis juntamente com este trabalho. Além disso, o código e as telas das soluções de algumas características mais relevantes estão discriminados nos anexos.

6.2.1 IMPLEMENTAÇÃO UTILIZANDO *ADOBE FLEX*

A implementação da aplicação de referência na tecnologia Flex se deu sobre a IDE (*Integrated Development Environment*) *Flash Builder*. A versão utilizada foi a 4.6 e a versão do *framework* Flex foi a 4.6.0. Outras informações estão disponíveis no site⁴⁴ do fabricante. Uma vantagem desta IDE é a representação gráfica da linguagem de marcação FXML. Com isso é possível “arrastar” controles a fim de estruturar e organizar o *layout* das aplicações.

Mesmo sem possuir conhecimento prévio da linguagem *ActionScript* e do FXML, Não foram encontradas grandes dificuldades em utilizar as características da tecnologia. Na internet é possível encontrar diversos exemplos de suas funcionalidades básicas. Alguns sites que podem ser citados são o fórum de *Flex* da *Adobe*⁴⁵, a documentação on-line do Flex⁴⁶ e o Centro de Desenvolvimento do Flex⁴⁷. Neste último é possível realizar o *download* de uma aplicação onde estão disponíveis diversas implementações pontuais das funcionalidades do Flex juntamente com seu código fonte. Esta aplicação chamada “*Tour De Flex*” foi importante para o desenvolvimento da aplicação de referência proposta.

A implementação da característica de informações gerais (C1) não possuiu grandes dificuldades. Foram utilizados controles prontos do Flex como *Label*, *ComboBox*, *TextArea* e *DateField*. Sua organização visual é vetorial, ou seja, a partir da definição das posições *X* e *Y* o que se assemelha com tecnologias *desktop*.

⁴⁴ <http://www.adobe.com/products/flex.html> Disponível em 04/06/2012.

⁴⁵ <http://forums.adobe.com/community/flex> Disponível em 04/06/2012.

⁴⁶ <http://www.adobe.com/devnet/flex/documentation.html> Disponível em 04/06/2012.

⁴⁷ <http://www.adobe.com/devnet/flex.html> Disponível em 04/06/2012.

Na listagem dos produtos na tela de itens do pedido (C2) foi utilizado outro controle do Flex chamado *TileGroup*. Este organizou facilmente os produtos conforme a característica especificada. Os itens foram criados dinamicamente via código *ActionScript* como pode ser visto na Figura 34. Nesta é possível observar a criação de dois *Labels*, um *Image* e um *BorderContainer*. Ao final, estes são adicionados de forma hierárquica no *TileGroup* chamado “*tgpProdutos*”.

```
//Cria controles do produto
content = new BorderContainer();
descricao = new Label();
preco = new Label();
img = new Image();

//Define propriedades do controle/borda
content.id = "bctProduto_" + prod.GetCodigo();
content.width = 144;
content.height = 60;
content.addEventListener(MouseEvent.CLICK, bordercontainer1_mouseDownHandler);
content.setStyle("cornerRadius", 10);

//Define propriedades da Descrição
descricao.text = prod.GetNome();
descricao.id = "lblDescricao_" + prod.GetCodigo();
descricao.x = 55;
descricao.y = 10;
descricao.setStyle("fontSize", 15);

//Define propriedades do preço
preco.text = cFormat.format(prod.GetValor());
preco.id = "lblPreco_" + prod.GetCodigo();
preco.x = 55;
preco.y = 37;

//Define propriedades da imagem
img.source = prod.GetUrlImagem();
img.x = 10;
img.y = 10;
img.width = 40;
img.height = 40;

//Monta hierarquia dos controles e adiciona ao TileGroup
content.addElement(descricao);
content.addElement(preco);
content.addElement(img);
tgpProdutos.addElement(content);
```

Figura 34 - Controles dinâmicos na listagem de produtos do *Flex*

O Flex possui um controle chamado de *DataGrid* para a representação de grids como nos itens do pedido. Neste, ao invés de criar por código *ActionScript*, cada coluna foi estruturada por uma espécie de *Template*. Este serve para organizar os controles dentro de

cada linha da grid.

A navegação entre as telas (C3) utilizou um controle chamado *LinkBar*. Este realiza automaticamente a troca entre as telas de “Informações gerais”, “Itens do pedido” e “Finalizar pedido”. A Busca dos itens (C4) foi implementada através do evento *change* de um controle e os itens filtrados por *ActionScript* sem grandes problemas.

O *drag-and-drop* da adição e exclusão de itens no pedido (C5) bastou definir a propriedade chamada *dragEnabled* para *true*. Com isso o controle desejado estava habilitado para ser arrastado. Para soltar os um item, o destino deveria implementar os eventos *dragEnter* e *dragDrop*. Estes eventos servem, respectivamente, para que ao arrastar um item apareça a possibilidade de soltar e ao soltar possa ser identificado o produto adicionado, por exemplo.

O *template* do *DataGrid* possibilita a inclusão de um controle do tipo *NumericStepper*, que possibilita a alteração da quantidade solicitada conforme descrito na característica C6. Para a validação das informações gerais conforme especificado na característica C7, não há uma propriedade pronta nos controles de entrada de dados. Para desenvolver esta especificação, foram criados controles do tipo *ToolTip* e aplicados estilos para que os mesmos sejam semelhantes aos propostos na aplicação de referência.

A característica C8 consiste na implementação de um mapa, baseado nas APIs do *Google*. Segundo o site da própria *Google*, atualmente a API nativa para o *Flash* está depreciada. Somente há suporte para a versão V3 desta API. A versão para *Flash* está ativa por tempo indeterminado. Para que esta API funcione, é necessária uma chave que somente é obtida através de registro junto a *Google* o qual não existe mais. Esta característica só foi implementada, pois utilizou-se uma chave obtida em um exemplo da internet.

Os gráficos estatísticos (C9), foram desenvolvidos com os controles *PieChart*, para o gráfico de pizza e o *ColumnChart*, para o gráfico de barras. Os efeitos de interação dos gráficos foram implementados com base nas animações já existentes pelos controles. Os controles dispunham de todas as funcionalidades propostas, inclusive o *Tooltip* exibido ao passar o cursor por um item do gráfico de pizza.

A transição das telas, proposta na característica C10, não foi implementada completamente, pois utilizou-se um controle chamado *ViewStack*. Este controle não possibilitou a transição exatamente como especificada. Isto ocorreu pelo motivo que as telas (Dados gerais, itens e finalizar pedido) não estavam dispostas uma ao lado da outra. Porém,

foi possível aplicar animações na navegação entre as telas. Com isso será possível realizar as medições do critério de performance do método de avaliação ao analisar o custo computacional da renderização gráfica.

O Flex possui acesso à *webcam* através dos controles *Video* e *Camera*. Ao ativar a câmera o *plug-in* do *Flash* realiza a questionamento se o usuário deseja realmente utilizar a *webcam*. Isto evita o uso indiscriminado dos recursos locais do usuário. Esta implementação supre com sucesso a característica C12.

As implementações em Flex, em geral foram fáceis. A curva de aprendizagem é bem curta e fácil de encontrar documentações e exemplos das mais diversas funcionalidades. Isto a torna uma linguagem interessante mesmo em um caso onde não haja profissionais com experiência na tecnologia.

6.2.2 IMPLEMENTAÇÃO UTILIZANDO MICROSOFT SILVERLIGHT

A versão da aplicação de referência desenvolvida em *Silverlight* utilizou-se do *Visual Studio 2010* como IDE de desenvolvimento. Esta é ferramenta proprietária da *Microsoft* e que possui recursos avançados de implementação e depuração. Assim como no *Flash Builder*, o *Visual Studio* também possui uma interface gráfica facilitando a elaboração do *layout* da aplicação. Para a implementação foram utilizadas a IDE na versão 10.0.4 com *Service Pack 1*, e o *Silverlight 5*.

Por utilizar como base a linguagem C# não foram encontradas dificuldades em relação a aprendizagem. Considera-se também que o autor possuía conhecimentos básicos em relação ao *Silverlight* e avançados em relação ao C#. Isto facilitou a implementação das características propostas. Foram utilizados diversos sites, próprios da *Microsoft*⁴⁸ e diversos *blog* pessoais com exemplos de implementações.

O *Silverlight* nativo não possui um grande número de controles de interface disponíveis. A *Microsoft* mantém em paralelo um projeto de código aberto chamado *Silverlight Toolkit*. Este disponibiliza diversos componentes que facilitam o desenvolvimento de aplicações em *Silverlight*. No site⁴⁹ deste projeto também é possível encontrar uma aplicação exemplificando a utilização de cada componente.

⁴⁸ <http://www.microsoft.com/silverlight/> Disponível em 05/06/2012.

⁴⁹ <http://silverlight.codeplex.com/> Disponível em 05/06/2012.

O *Silverlight Toolkit* foi utilizado em praticamente toda implementação da aplicação proposta. Isto se deu pela impossibilidade de implementar a maioria das características sem sua utilização. Com isso, as especificações das informações gerais do pedido (C1) foram implementadas sem nenhuma dificuldade. Nela foram utilizados controles como *Label*, *ComboBox*, *TextBox* e *DatePicker*. Para a disposição dos controles, ao invés de definir as posições *X* e *Y* definiu-se as margens de cada um em relação ao controle externo. A princípio este conceito não é muito trivial, porém com a prática torna-se fácil sua utilização.

Na tela de itens do pedido, para a disposição dos produtos disponíveis, foi utilizado um controle do tipo *ListBox* aplicado um *template* de um controle do tipo *WrapPanel*. Este *template* é utilizado para que os itens sejam dispostos conforme o especificado na aplicação de referência. Esta utilização não é algo normal de ser implementado em qualquer tecnologia o que deixa a utilização desta listagem de certa forma confusa. Para a representação dos itens adicionados no pedido utilizou-se um controle *DataGrid*. Neste, as linhas foram representadas através de *templates* para a exibição dos dados estruturados conforme a representação da característica proposta.



Figura 35 - Implementação dos itens do pedido - C2 em Silverlight

A Figura 35 exibe a implementação da característica C2 descrita. Nela observa-se a que o layout proposto foi elaborado com sucesso. Além disso, está demonstrada a implementação da navegação entre as telas. Esta foi desenvolvida utilizando o evento de clique do cursor para realizar as transições propostas na característica C3 (navegação do

pedido).

O desenvolvimento da busca de itens (C4) foi desenvolvida de forma semelhante ao *Flex*. Baseou-se filtrando a lista de produtos a partir do evento *TextChanged* de um controle do tipo *TextBox*. A funcionalidade de *drag-and-drop* definida na característica C5 (adição de itens no pedido) não possuiu o comportamento esperado.

Para arrastar e soltar um elemento foi necessária a utilização de outros controles. Estes envolvem o controle onde há elementos que possam ser arrastados ou o local onde se deseja soltar um elemento. Esta solução está presente através do *Silverlight Toolkit*. Há outra forma de implementar que consiste em controlar o clique e a posição do cursor, porém esta se torna muito trabalhosa. Mesmo usufruindo dos controles do *Toolkit* sua utilização não está estável na versão do *Silverlight* utilizada.

A implementação da quantidade solicitada (C6) utilizou do controle *NumericUpDown*. Com ele foi possível gerenciar a quantidade máxima disponível do item conforme especificação. Para a validação das informações gerais (C7) foi utilizado um padrão de exibição de aviso conforme exceção disparada da propriedade vinculada ao controle visual. Esta funcionalidade não é algo comum, porém pode ser muito útil para validações dinâmicas.

Para logística na emissão do pedido (C8) não foi possível utilizar a API da *Google*. Não há nenhuma API do *Google Maps* desenvolvida para o *Silverlight*. Além disso, segundo seus termos de serviço⁵⁰ não é possível o acesso dos serviços de mapa através de tecnologias não autorizadas pela *Google* (Seção 10.1.1 do termo). Em contra partida, a *Microsoft* disponibiliza um serviço de mapas através do *Bing Maps*⁵¹. Este foi utilizado para implementação desta característica a qual atendeu com sucesso as especificações propostas apesar de ser relativamente mais complexa que a API disponibilizada pelo *Google*.

Os gráficos estatísticos (C9) especificados na aplicação de referência foram implementados utilizando os controles *PieSeries* e *ColumnSeries*. O desenvolvimento destes foi extremamente simples não necessitando mais de duas linhas de código para cada um. A única característica não implementada foi o destaque da fatia selecionada no gráfico de pizza. O motivo disso foi por não haver nenhuma funcionalidade pronta como ocorreu no *Flex* e não foi encontrado nenhum exemplo representando isto.

A transição entre as telas do pedido (10) foi implementada de forma semelhante a do *Flex*. Neste caso utilizou-se o controle *TabControl* que não dispõe as telas de forma

⁵⁰ <https://developers.google.com/maps/terms?hl=pt-BR> Disponível em 06/06/2012.

⁵¹ <http://www.microsoft.com/maps/> Disponível em 06/06/2012.

horizontal. De qualquer forma a implementação provou a possibilidade de uso de transições entre as telas possibilitando as medições propostas no método de comparação.

Por fim, a implementação da característica de uso da *webcam* (C12) foi implementada com sucesso. Esta utilizou os controles nativos *CaptureSource* e *VideoBrush*. Da mesma forma que o *Flex*, o *Silverlight* requer a autorização do usuário para acesso aos recursos locais.

A utilização da tecnologia *Silverlight* não demonstrou falta de funcionalidades. Porém, algumas não foram completamente eficientes. Em determinadas implementações as soluções encontradas não foram as mais triviais. De qualquer forma há diversos exemplos e não faltou documentação para apoiar o desenvolvimento da aplicação de referência.

6.2.3 IMPLEMENTAÇÃO UTILIZANDO HTML5

Por ser apenas um conjunto de especificações técnicas não proprietárias, o HTML5 não possui um IDE de desenvolvimento padrão. Além disso, por não necessitar de um compilador, qualquer editor de texto pode ser utilizada para a codificação em *Javascript*, HTML e CSS. Para facilitar foi utilizado o mesmo IDE do *Silverlight*, o *Visual Studio* 10. Este foi escolhido por possuir um *plug-in* de auto completar das funcionalidades especificadas no HTML5 além de já haver familiaridade com a utilização da ferramenta. A desvantagem encontrada nesta escolha foi o fato de que a representação gráfica do *layout* não é totalmente efetiva, o contrário que ocorre com o *Flex* e *Silverlight*.

A utilização do *Javascript*, do HTML e do CSS é muito difundida no desenvolvimento *web*. Inúmeros são os sites que possuem exemplos e funcionalidades utilizando estas tecnologias. As documentações das especificações destas estão disponíveis no site da W3C⁵². Além disso, outro site muito conhecido por disponibilizar exemplos da maioria das funcionalidades especificadas pela W3C é o *W3 Schools*⁵³.

Os campos utilizados para desenvolver a característica de dados gerais (C1) foram o *input*, o *select* e o *textarea*. O *input*, utilizado para representar o campo de data de emissão, necessitou que a propriedade *type* fosse definida como *date*. Mesmo tendo sua especificação concluída, o campo de seleção de data não é implementado em todos os navegadores. Na

⁵² <http://www.w3.org/TR/> Disponível em 08/06/2012.

⁵³ <http://www.w3schools.com/html5/default.asp> Disponível em 08/06/2012.

Seção 6.3.1 será exibida a compatibilidade desta funcionalidade.

Por não possuir controles avançados especificados para o HTML5, muitas funcionalidades necessitaram ser implementadas manualmente. Um exemplo desta necessidade e não haver controles para representar a lista de produtos e itens conforme proposto pela aplicação de referência na característica de itens do pedido (C2). Neste caso tornou-se necessário a estruturação dos produtos e dos itens baseando-se em *tables* montadas dinamicamente. Na Figura 36, é possível observar o resultado obtido. Mesmo aplicando estilos avançados em CSS, a aparência fica a desejar ao comparar com as demais tecnologias. Por outro lado, tendo maiores conhecimentos de *Design* e CSS a apresentação dos controles seriam mais amigáveis.

Informações gerais > Itens do pedido > Finalizar pedido

Filtro

Promoção Arraste aqui para excluir um item

Arraste aqui para adicionar ao pedido		
Produto	Quantidade	Total
 Bola	<input type="text" value="1"/>	23
 Lupa	<input type="text" value="3"/>	64.59



The grid contains the following items:

- Despertador: 1200
- Livro: 50
- Celular: 900
- Tesoura: 23.15
- Retrato: 65
- Bola: 23
- Lâmpada: 200
- Bandeira: 154
- Filmadora: 1066
- Câmera: 3000
- Microfone: 80
- Lupa: 21.53
- Relógio: 664
- Televisão: 7200
- Cachorro: 2000
- Mesa: 5050
- Computador: 3450
- Mouse: 61

Figura 36 - Implementação dos itens do pedido - C2 em HTML5.

Ainda na Figura 36, a característica de navegação do pedido (C3) pode ser observada. Nela não houve dificuldades de desenvolvimento. O mesmo ocorreu na implementação da busca dos itens (C4). Nesta utilizou-se o evento *onkeyup* de um *input* do tipo *text* para filtrar e remontar a listagem dos produtos.

As especificações do HTML5 disponibilizam suporte total para realizar as ações de *drag-and-drop*. Para isso basta definir a propriedade *draggable* do elemento que deseja arrastar. Para identificar o elemento implementa-se o evento *ondragstart*, para aceitar e capturar este elemento é necessário implementar os eventos *ondragover* e *ondrop*. Com isso as funcionalidades descritas na característica C5 (adição de itens) foram atendidas por completo.

Assim como no controle da data, o *input* do tipo *number* não possui suporte disponível em todos os navegadores. Isto fez com que a característica de quantidade solicitada dos itens (C6) não fosse cumprida por completo. Maiores detalhes sobre os navegadores que suportam esta funcionalidade está descrito na Seção 6.3.6.

Na Seção 2.4.2.1 foi exemplificado o uso da validação de campos requeridos em HTML5. Esta funcionalidade foi utilizada para atender as necessidades da característica C7 (validações das informações gerais). Logo, a utilização da propriedade *required* dos elementos *input* e *select*, supriu os requisitos especificados.

O *Google* possui um API exclusiva para utilização de seus mapas em páginas implementadas somente com *Javascript* e HTML. Diversos exemplos⁵⁴ sobre a utilização de mapas com esta API foram encontrados. Com isso foi satisfatória a implementação da característica de logística na emissão do pedido (C8) incluindo a geolocalização especificada no HTML5.

Não há especificações de controles que representem gráficos estatísticos em HTML5. Existem bibliotecas de terceiros que fornecem estes controles. Como a intenção deste comparativo é avaliar a disponibilidade de recursos em cada tecnologia, evitou-se o uso de componentes de terceiros. Na implementação da característica de gráficos estatísticos (C9), foi necessário o desenvolvimento manual dos controles. Para isso utilizou-se a renderização gráfica do SVG, que ao trabalhar de forma vetorial, foi possível calcular as dimensões e desenhar os gráficos de forma dinâmica utilizando os elementos *path* e *rect* através de *Javascript*.

Utilizando a propriedade *transition*, especificada no CSS3, foi possível implementar a característica de transição de telas (C10). Bastou alterar as classes das telas ao navegar entre elas para aplicar a transição especificada. As classes em CSS utilizadas estão representadas na Figura 37. Nesta figura é possível observar que a propriedade *transition* é repetida 3 vezes, isto ocorre pois os navegadores ainda não suportam de forma nativa esta funcionalidade. Mais detalhes sobre quais os navegadores com suporte a esta funcionalidade na Seção 6.3.10.

⁵⁴ <https://developers.google.com/maps/documentation/javascript/examples/?hl=pt-BR> Disponível em 10/06/2012.

```
.hideToRight
{
    transition: margin-left 2s, opacity 3s;
    -moz-transition: margin-left 2s, opacity 3s;
    -webkit-transition: margin-left 2s, opacity 3s;
    margin-left: 950px;
    opacity: 0 !important;
}
```

Figura 37 - Efeitos em CSS para a característica de transição de telas (C10).

A W3C possui especificação⁵⁵ ainda não concluída para a captura recursos de mídia como áudio e vídeo. Em função disto, a implementação realizada para atender as especificações de uso da *webcam* (C12) não funcionou nos navegadores utilizados.

A implementação da aplicação de referência em HTML5 se mostrou parcialmente eficaz. Isto se deve a imaturidade de sua especificação. Por outro lado, o desenvolvimento é relativamente fácil, favorecendo quem possui experiência no desenvolvimento de aplicações *web* tradicionais. Isto facilita a adoção das características especificadas pela W3C. Na questão de controles mais avançados, como os gráficos estatísticos e até mesmo a listagem de itens, torna-se árduo o seu desenvolvimento. Neste caso, mostra-se necessária a utilização de componentes de terceiros que, mesmo implementados sobre as especificações do HTML5, abstraíam o trabalho básico de montar uma *grid* de informações como a de itens do pedido, por exemplo.

6.2.4 IMPLEMENTAÇÃO UTILIZANDO ORACLE JAVA FX

O *JavaFX*, em sua versão 2.0, foi desenvolvido utilizando a IDE *NetBeans* na versão 7.1. Esta IDE foi utilizada, pois é indicada para o desenvolvimento de aplicações *JavaFX* pela *Oracle*. Ao contrário das IDEs do *Flex* e *Silverlight*, esta não possui uma interface gráfica para a definição de *layout* da aplicação. Para suprir esta necessidade, o centro de desenvolvimento do *JavaFX* disponibiliza uma ferramenta chamada *JavaFX Scene Builder* 1.0. Esta ferramenta serve para desenhar o layout das aplicações de forma simples sem ser necessário digitar códigos em FXML. A desvantagem é que o *Scene Builder* não possui integração com o *NetBeans*.

⁵⁵ <http://dev.w3.org/2011/webRTC/editor/getusermedia.html#localmediastream> Disponível em 19/06/2012.

Por ser uma tecnologia muito nova, sofrendo muitas alterações na versão 2.0, é difícil encontrar exemplos na comunidade de desenvolvedores. Apesar disso, a *Oracle* disponibiliza uma vasta documentação⁵⁶ com o detalhamento das funcionalidades suportadas pelo *JavaFX*. Além disso, no site da *Oracle* é possível encontrar uma aplicação chamada *JavaFX Samples* onde são implementadas diversas funcionalidades com base no *JavaFX*, porém poucas utilizam a linguagem de marcação FXML para as demonstrações.

Para implementar a característica de informações gerais do pedido (C1) utilizaram-se controles do tipo *Label*, *ChoiceBox* e *TextArea*. Para a organização dos elementos foram utilizados *containers* do tipo *GridPane* que gerenciam as posições dos controles de forma semelhante a uma *table* do HTML. Mesmo possuindo diversos controles para o desenvolvimento de aplicações, não foi encontrado nenhum que atendesse a funcionalidade do campo de data de emissão.

Sobre tela de itens do pedido (C2), para a disposição dos produtos conforme especificação, foi utilizado o controle *TilePane*. Para que os produtos fossem exibidos da forma desejada, os mesmos foram adicionados via código de forma dinâmica. Já os itens adicionados ao pedido, foram renderizados com base no controle *TableView*. Este possuiu usas colunas adicionadas via código também dinâmico. Isto ocorreu em função de que as características avançadas de exibição de dados necessitaram utilizar do evento *CellFactory* que funcionou de forma semelhante a um *template*. Esta característica do *JavaFX* deixou o desenvolvimento mais complexo, porém é um conceito diferente o qual não havia familiaridade e que baseou-se em exemplos disponíveis na *web*.

A navegação entre as telas do pedido (C3) não possuiu grandes dificuldades. Esta característica foi implementada através do evento *onMouseClicked* de *Labels*. A busca de itens (C4), também não apresentou problemas para ser desenvolvida. Apenas foi necessário implementar o filtro dos produtos no evento *onKeyReleased* disparado pelo controle *TextField*.

De forma simples, para habilitar que um elemento possa ser “arrastado”, bastou implementar o evento *onDragDetected*. Para identificar e “aceitar” este elemento, a região de destino necessitou a implementação dos eventos *onDragOver* e *onDragDropped*. Esta implementação se mostrou eficiente e não apresentou dificuldades para ser desenvolvida. Com isso foi possível atender completamente a característica de adição de itens no pedido

⁵⁶ <http://docs.oracle.com/javafx/> Disponível em 19/06/2012.

(C5).

A característica de quantidade solicitada dos itens (C6) não foi possível de implementação completa. O motivo foi não haver um controle numérico conforme o especificado na aplicação de referência. Além disso, houve dificuldades em atualizar a quantidade do item após alterar o valor do campo quantidade na grade de itens do pedido.

O *JavaFX*, assim como o *Flex*, não possui a funcionalidade de validação se um campo é requerido como no HTML5 e *Silverlight*. Logo, para suprir a característica de validações das informações gerais proposta foi necessária a implementação manual das funcionalidades. Neste caso foi utilizado um controle *AnchorPane* juntamente com um *Label* possuindo comportamento semelhante ao de um *tooltip*. Esta implementação atende a característica de validações das informações gerais (C7) de forma mais complexa que as soluções encontradas nas outras tecnologias.

A especificação da característica de logística na emissão do pedido (C8) define que deverá ser utilizada a API do *Google* para sua implementação. Porém, não há uma versão desta API implementada para o *JavaFX*. Ao realizar pesquisas, a sugestão da própria *Oracle* de desenvolvimento é utilizar um bloco de código em HTML implementado de forma semelhante ao realizado no HTML5. Como o objetivo é analisar a capacidade da tecnologia suprir os requisitos, esta característica não foi implementada, pois o resultado seria igual ao do HTML5.

Os gráficos estatísticos (C9) não apresentaram dificuldades para serem implementados. O *JavaFX* fornece diversos controles para a exibição de informações em gráficos. A sua implementação foi simples apenas necessitando de implementação manual do *tooltip* do gráfico de pizza e do destaque da fatia selecionada. O resultado, que pode ser visto na Figura 38, possui aparência muito interessante sem necessitar muitos esforços para sua implementação. Por outro lado a utilização destes controles deixa o desenvolvimento pouco maleável à possíveis adaptações.

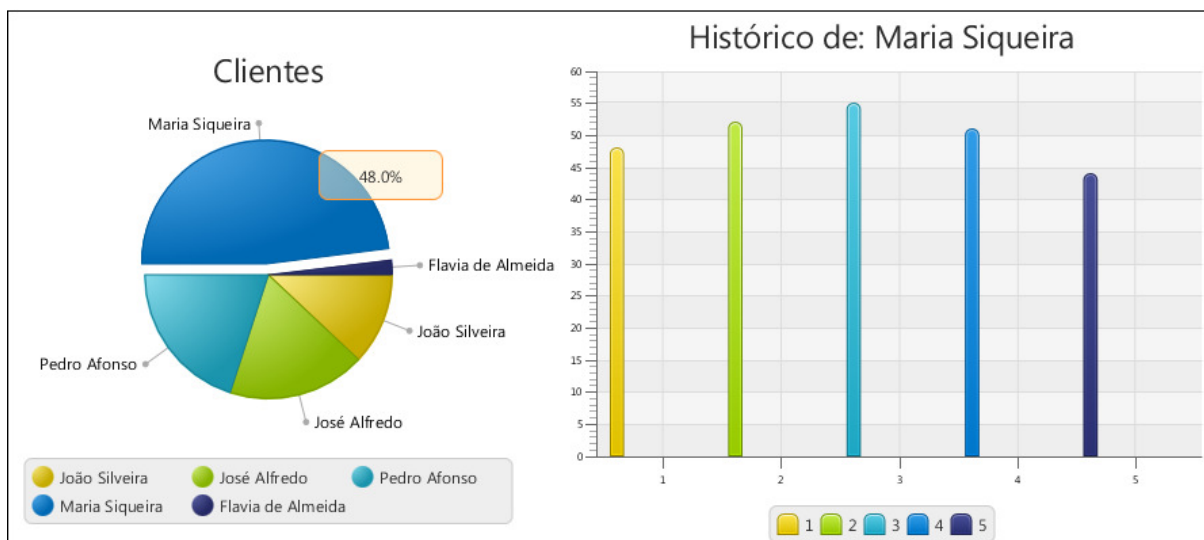


Figura 38 - Gráficos estatísticos utilizando *JavaFX*.

Para realizar o desenvolvimento da característica de transição de telas (C10) foram utilizadas as transições prontas existentes no *JavaFX*. Estas transições estão exemplificadas na Figura 39. Nela é possível observar que ao realizar o clique para trocar de tela, as transições de *TranslateTransition* e *FadeTransition* são executadas a fim de atender os requisitos especificados nesta característica.

```
@FXML protected void itensPedClicked(MouseEvent event)
{
    TranslateTransition translateTransitionViews;
    translateTransitionViews = new TranslateTransition(Duration.seconds(1), MinhasViews);
    translateTransitionViews.setToX(grpDadosPedido.getWidth() * -1);
    translateTransitionViews.play();

    FadeTransition fadeTransitionHide = FadeTransitionBuilder.create()
        .duration(Duration.seconds(1))
        .fromValue(1)
        .toValue(0)
        .build();
    fadeTransitionShow.setNode(grpItensPedido);
    fadeTransitionShow.play();
    fadeTransitionHide.setNode(grpDadosPedido);
    fadeTransitionHide.play();
    fadeTransitionHide1.setNode(grpFinalizarPedido);
    fadeTransitionHide1.play();
}
```

Figura 39 - Código para realizar a transição entre as telas em *JavaFX*.

Segundo as documentações disponíveis no site da Oracle e as pesquisas realizadas na internet, o *JavaFX* não possui acesso nativo a webcam. Em sua documentação há uma biblioteca chamada *camera*, porém ela somente é utilizada para a renderização de animações gráficas. Por este motivo não foi possível implementar a característica de uso a webcam (C12)

proposta na aplicação de referência.

O *JavaFX* é uma tecnologia que sofreu grandes mudanças em suas últimas versões. Mesmo assim, é possível observar a preocupação da *Oracle* em dispor material para apoiar os desenvolvedores. Para quem está habituado à linguagem *Java*, a curva de aprendizagem é relativamente curta. A linguagem mantém seu padrão com exceção do uso de assinaturas de eventos, deixando de lados os tradicionais *listeners*. Apesar de sua imaturidade, as implementações foram além das expectativas.

6.3 AVALIAÇÃO DAS CARACTERÍSTICAS

Conforme especificado no modelo de avaliação localizado no Capítulo 4, nesta serão apresentados os resultados obtidos. Estes resultados estão organizados conforme as características definidas na aplicação de referência. Para cada característica (C1, C2, C3, ...) serão analisados os critérios de avaliação especificados na Seção 4.2 (performance, compatibilidade, facilidade e semelhança).

Os critérios de avaliação serão aplicados conforme especificação. Para a avaliação da performance foram utilizadas as ferramentas de medição no navegador *Google Chrome* e o painel de controle do Windows. Somente a média das medições de performance serão exibidas para facilitar o estudo. O critério de compatibilidade será aplicado entre os navegadores *Google Chrome* 19.0.1, *Microsoft Internet Explorer* 9.0.8, *Mozilla Firefox* 13.0 e *Safari* 5.1.7. Estes foram escolhidos conforme classificação dos navegadores mais utilizados segundo os sites definidos na Seção 4.2.2. Todos os navegadores serão executados em um ambiente com o sistema operacional *Microsoft Windows 7 64 bits*.

As implementações em cada tecnologia serão confrontadas diretamente entre si conforme definido no processo analítico hierárquico. Supondo que uma característica C14 seja analisada e partindo de um cenário hipotético em que o *Silverlight* supriu todas as necessidades de forma fácil, o *HTML5* e o *Flex* necessitaram um pouco mais de esforços e o *JavaFX* não demonstrou suporte a esta característica. O resultado deste comparativo para o critério de facilidade de desenvolvimento resultaria em algo semelhante ao exemplificado na Tabela 11.

Critério de facilidade de desenvolvimento					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	3	1	1/3	20,7%
JavaFX	1/3	1	1/3	1/5	8%
Flex	1	3	1	1/2	22,7%
Silverlight	3	5	2	1	48,6%
				Inconsistência	1%

Tabela 11- Exemplo hipotético de um resultado de comparação do critério de facilidade de desenvolvimento.

Segundo este comparativo, na Tabela 11 é possível observar que o *Silverlight* possui vantagem de 48,6% contra 22,7% do *Flex*, 20,7% do *HTML5* e 8% do *JavaFX* em relação ao critério de facilidade de desenvolvimento. A seguir será realizado o mesmo comparativo analisando as características especificadas na aplicação de referência.

6.3.1 C1 - INFORMAÇÕES GERAIS DO PEDIDO

A avaliação da performance na característica de informações gerais deverá avaliar os recursos computacionais de memória do processo e uso da CPU utilizados para renderizar a aplicação. Além da renderização, nesta serão consideradas as medições de inicialização da aplicação. Neste momento é que ocorre a comunicação entre o servidor e o navegador para transferência dos arquivos necessários. Para realizar medições mais fiéis a esta característica, a funcionalidade de logística (C8) foi removida temporariamente da aplicação. Isso se deve ao fato de que o mapa poderia distorcer os resultados. A performance referente a característica C8 será medida separadamente na Seção 6.3.8.

A memória do processo e o uso do CPU, conforme especificado na Seção 4.2.1, foram obtidos por meio do painel de controle do Windows. As medições de transferência (KB e tempo) foram obtidos através das ferramentas de medições do *Google Chrome*. As médias finais obtidas pelas medições realizadas estão relacionadas na Tabela 12.

	KB transferidos	Tempo de transferência	Memória do processo	Uso da CPU
HTML5	132,96KB	569,25ms	22.816,16K	5,25%
JavaFX	26,85KB	567,75ms	81.280,30K	33,00%
Flex	138,25KB	1.120,00ms	33.836,80K	23,25%
Silverlight	1.080,00KB	1.200,00ms	26.468,40K	14,75%

Tabela 12 - Medições da performance das informações gerais do pedido.

Na Tabela 12 é possível observar quatro indicadores foram coletados. Ao analisar os dados, pode-se perceber que houve uma grande variação entre as tecnologias. A tecnologia *JavaFX*, por exemplo, possui uma baixa taxa de transferência, em contra partida a utilização de memória e CPU foram os mais altos obtidos. O *Silverlight*, por sua vez supera o *Flex* na questão de menor consumo de *hardware*, porém requer uma maior transferência de arquivos.

Analisando os dados obtidos e com o auxílio da ferramenta *Expert Choice*, já mencionada neste trabalho, foram obtidas as comparações relacionadas na Tabela 13. É possível observar uma grande vantagem para o HTML5 em função de seu baixo consumo de CPU e memória. Com 15,6% o *JavaFX* encontra-se em segundo, seguido de *Flex* e *Silverlight* com 9,7% e 8,7% respectivamente.

	Critério de performance				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	6	5	7	66%
JavaFX	1/6	1	2	2	15,6%
Flex	1/5	1/2	1	1	9,7%
Silverlight	1/7	1/2	1	1	8,7%
				Inconsistência	3%

Tabela 13 - Resultados do critério de performance para a característica C1.

A comparação do critério de compatibilidade analisou o comportamento da característica de informações gerais analisou o comportamento das aplicações em navegadores distintos. A comparação baseou-se no requisito de que os campos especificados apresentassem comportamentos semelhantes entre os navegadores analisados.

As aplicações implementas em *Silverlight* e *Flex* apresentaram o mesmo

comportamento em todos os navegadores. O campo da data de emissão do HTML5 não foi reconhecido em nenhum navegador, somente no *Safari* há uma opção de alterar o valor do campo. Porém esta representação não inclui um controle para selecionar a data como ocorre no *Silverlight* e no *Flex*. O *JavaFX* simplesmente não é suportado pela navegador *Safari*. Ao invés da aplicação uma mensagem dizendo que esta aplicação não é possível ser executada neste navegador é exibida.

Com base nas considerações levantadas os resultados obtidos estão exibidos na Tabela 14. Nesta é possível observar que o *Flex* e o *Silverlight* permanecem com 38,5% em função de sua invariância entre os navegadores. Já o HTML5 está com 14,3% por consequência do não suporte ao campo data e o *JavaFX* com apenas 8,7% por não funcionar no *Safari*.

	Critério de compatibilidade				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	2	1/3	1/3	14,3%
JavaFX	1/2	1	1/4	1/4	8,7%
Flex	3	4	1	1	38,5%
Silverlight	3	4	1	1	38,5%
				Inconsistência	0,7%

Tabela 14 - Resultados do critério de compatibilidade para a característica C1.

Em relação a facilidade de desenvolvimento, a característica C1 não apresentava grandes obstáculos. O que pode ser referenciado neste, são as sistemáticas de disposição dos elementos. Como já havia sido relatado na Seção 6.2, o *Flex* e *JavaFX* possibilitam a disposição vetorial como numa aplicação desktop de forma natural enquanto as outras não. Já a localização da documentação necessária para esta característica não apresentou dificuldades.

Estas conclusões estão refletidas nos resultados apresentados na Tabela 15. Neste e exibido o HTML5 com 43,5% em primeiro por seguir os conceitos de qualquer aplicação web escrita em HTML. Em seguida o *Flex* com 28,6% por apresentar grande facilidade de aprendizagem e por fim o *Silverlight* e *JavaFX* respectivamente com 18,2% e 9,7%.

Critério de facilidade de desenvolvimento					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	4	2	2	43,5%
JavaFX	1/4	1	1/3	1/2	9,7%
Flex	1/2	3	1	2	28,6%
Silverlight	1/2	2	1/2	1	18,2%
				Inconsistência	2%

Tabela 15 - Resultados do critério de facilidade de desenvolvimento da característica C1.

O critério de semelhança abrange as além da questão estética, os quesitos de usabilidade e atendimento aos requisitos. Neste caso ao analisar este critério entre as tecnologias observou-se que o *JavaFX* não possui um controle para representação de um campo data. O *HTML5* até possui este controle, porém, como já analisado no critério de compatibilidade, este não está disponível em todos os navegadores.

Além do que já foi mencionado na questão de usabilidade, nesta característica não há nenhuma deficiência relevante. Com isso é possível observar na Tabela 16 os resultados obtidos. Nela estão dispostos os valores atribuídos a cada tecnologia conforme avaliação realizada. O *Silverlight*, com 49,6%, encontra-se em primeiro seguido pelo *Flex* com 35%. Esta diferença refere-se a questões estéticas. Por fim o *HTML5* aparece com 9,4% e o *JavaFX* com 6,1% conforme as questões levantadas anteriormente.

Critério de semelhança com aplicações <i>desktop</i>					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	2	1/5	1/5	9,4%
JavaFX	1/2	1	1/6	1/6	6,1%
Flex	5	6	1	1/2	35%
Silverlight	5	6	2	1	49,6%
				Inconsistência	4,0%

Tabela 16 - Resultados do critério de semelhança com aplicações *desktop* da característica C1.

A característica de informações gerais não é considerada complexa. Suas características são de um cadastro simples. As medições apontaram vantagens para o *Silverlight* nos critérios de semelhança, e compatibilidade. No critério de compatibilidade

também se destaca o Flex. Já o HTML5 demonstrou melhor performance e maior facilidade de desenvolvimento.

6.3.2 C2 - ITENS DO PEDIDO

A característica de itens do pedido consiste na especificação geral da segunda tela. Esta não representa uma ação específica, mas sim uma definição de *layout*. As medições em relação a performance foram realizadas baseando-se no consumo de memória e CPU utilizadas em quanto a aplicação estiver renderizando a tela de itens. Neste momento não há transferência entre o servidor descartando assim as medições feitas através da ferramenta do *Google Chrome*. Isto se deve ao fato que todos os arquivos necessários para a execução da aplicação já foram carregados ao inicializar a aplicação.

Os resultados obtidos nas medições estão relacionados na Tabela 17. Nesta pode-se ver a grande vantagem do HTML5 ao consumir a menos porcentagem de CPU e o menor consumo de memória. As tecnologias *Flex* e *Silverlight* possuem uma pequena variação onde uma consome menos memória e outra consome menos CPU. Por fim o *JavaFX*, apesar de consumir pouco CPU, a memória do processo da aplicação chegou mais de duas vezes o consumo do *Flex*.

	Memória do processo	Uso da CPU
HTML5	25.968,80K	4,8%
JavaFX	87.170,40K	6%
Flex	42.535,20K	10,2%
Silverlight	31887,20K	12%

Tabela 17 - Medições da performance dos itens do pedido.

Em reflexo dos dados obtidos nas medições, foram feitas as comparações diretas definidas pelo método analítico hierárquico. A relação dos resultados está disposta na Tabela 18. Nela se confirmam as observações levantadas apontando o HTML5 com melhor performance ao atingir 51,9% seguido do *Silverlight* com 20,2%, do *Flex* com 14,3% e por fim o *JavaFX* com 6,4%. O grau de inconsistência chegou a 4% o que é aceitável por estar a baixo dos 10% tolerados definidos no método.

	Critério de performance				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	7	4	4	51,9%
JavaFX	1/7	1	1/3	1/3	6,4%
Flex	1/4	3	1	1/2	14,3%
Silverlight	1/4	3	2	1	20,2%
				Inconsistência	4%

Tabela 18 - Resultados do critério de performance para a característica C2.

Conforme citado anteriormente nesta característica não houve a necessidade de testar uma ação específica. Com isso, para o critério de compatibilidade foi analisado se a listagem de produtos e a grade de itens mantinham o padrão entre os navegadores. Isto inclui o suporte à barra de rolagem especificada de forma independente para a os itens e para os produtos.

Neste critério foi observado que a grade de itens, da aplicação desenvolvida em HTML5, não foi exibida quando testada no navegador *Safari*. Além disso, como já havia sido relatado anteriormente, o JavaFX não funcionou no navegador *Safari*. Já as tecnologias *Flex* e *Silverlight* não apresentaram divergências entre os navegadores. Com isso, na Tabela 19, estão relacionadas as comparações diretas entre cada uma tecnologia. O *Flex* e o *Silverlight* aparecem empatados com 41,7%, seguido do HTML5 com 10,8% e com apenas 5,8% o *JavaFX* aparece por último.

	Critério de compatibilidade				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	2	1/4	1/4	10,8%
JavaFX	1/2	1	1/7	1/7	5,8%
Flex	4	7	1	1	41,7%
Silverlight	4	7	1	1	41,7%
				Inconsistência	0,01%

Tabela 19 - Resultados do critério de compatibilidade para a característica C2.

Nos relatos das implementações descritos na Seção 6.2 foram apontadas questões relevantes quanto ao critério de facilidade de desenvolvimento desta característica. Nas tecnologias *Silverlight*, *Flex* e *JavaFX* foram utilizados controles nativos para organizar tanto

os itens do pedido quanto a listagem dos produtos. No HTML5, não havendo controles avançados, estes necessitaram que fossem implementados manualmente. Os *template* utilizados no *Silverlight* facilitaram a organização das informações, O *Flex* também possui *templates* semelhantes, porém estes só foram utilizados na grade de itens. Já o *JavaFX* necessitou um maior esforço pois seu conceito de *templates* funciona de forma diferente.

Baseado nos fatos citados anteriormente, a Tabela 20 exhibe os resultados das comparações entre as tecnologias. Em primeiro com 42% está o *Silverlight* e com pouca diferença, o *Flex* aparece com 38,7%. Já o *JavaFX* aparece com 13,5% considerando da falta de documentação e por fim o HTML5 em função das implementações manuais dos controles.

Critério de facilidade de desenvolvimento					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	1/3	1/6	1/6	5,9%
JavaFX	3	1	1/3	1/4	13,5%
Flex	6	3	1	1	38,7%
Silverlight	6	4	1	1	42%
				Inconsistência	2%

Tabela 20 - Resultados do critério de facilidade de desenvolvimento da característica C2.

Todas as tecnologias atenderam os requisitos especificados na aplicação de referência independentemente da dificuldade de implementação. Apesar de que nem todas as implementações estejam com o *design* igual definido no protótipo, a estética depende da aplicação de estilos aos controles. Todas as tecnologias apresentaram suporte a manipulação de estilos. Tendo isto em vista e que esta característica não contempla nenhuma ação, a avaliação do critério de semelhança com aplicações *desktop* resultou igualmente 25% para cada tecnologia. Estes resultados podem ser observados na Tabela 21.

Critério de semelhança com aplicações <i>desktop</i>					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	1	1	1	25%
JavaFX	1	1	1	1	25%
Flex	1	1	1	1	25%
Silverlight	1	1	1	1	25%
				Inconsistência	0%

Tabela 21 - Resultados do critério de semelhança com aplicações *desktop* da característica C2.

A característica de itens do pedido não abrangia ações e interação, tendo assim menor relevância para o critério de performance e questões de usabilidade. A tecnologia *Silverlight* destacou-se no critério de facilidade e compatibilidade. O *Flex* também mostrou-se eficiente na avaliação da compatibilidade entre navegadores. Por fim, o HTML5 foi o melhor no critério de desempenho ocupando menor recurso computacional.

6.3.3 C3 - NAVEGAÇÃO DO PEDIDO

A característica de navegação do pedido definia uma forma diferenciada de alterar a tela atual do pedido. Esta especificação não atende transição de telas que está especificada na característica C10. Com isso o critério de performance não se torna aplicável a esta característica. Além disso, todas as tecnologias foram eficientes para a implementação da funcionalidade especificada. De forma semelhante, as implementações utilizaram o clique de campos textos para alterar a tela visível por meio das transições.

Ao levar em consideração estes fatos, é possível excluir esta característica das avaliações. Isto se deve ao fato que todas as tecnologias receberiam a mesma avaliação. Logo estes comparativos não influenciariam na avaliação final das aplicações tornando assim irrelevante ao resultado final.

6.3.4 C4 - BUSCA DE ITENS NO PEDIDO

A busca de itens no pedido consiste em filtrar os produtos disponíveis para adição no pedido. A especificação definia que ao digitar um filtro, a listagem deveria ser atualizada

somente com os produtos que contemplassem o valor digitado. A avaliação do critério de performance avalia o memória e a porcentagem de CPU utilizada pela ação de filtrar. Nesta característica não é efetuada nenhuma transferência de arquivos entre o servidor e a aplicação. Isto descarta o uso da ferramenta de medições do *Google Chrome*. Os resultados obtidos nas medições da performance estão relacionados na Tabela 22.

	Memória do processo	Uso da CPU
HTML5	23.779,50K	2,25%
JavaFX	124.627,25K	9,25%
Flex	59.234,00K	7,5%
Silverlight	35.265,00K	5%

Tabela 22 - Medições da performance da busca de itens no pedido.

Nestes valores há uma clara sequência entre as tecnologias. Onde o HTML5 se destaca novamente por ocupar menos memória e utilizar menos o CPU. Seguido do *Silverlight*, *Flex* e *JavaFX*. Destaca-se aqui a quantidade exorbitante de memória utilizada pelo *JavaFX*. Por este motivo, nos comparativos demonstrados na Tabela 23, o *JavaFX* encontra-se em último com apenas 5,2%. E primeiro ficou o HTML5 com 59,7% seguido pelo *Silverlight* com 12,1% e o *Flex* com 12,1%.

	Critério de performance				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	8	6	3	59,7%
JavaFX	1/8	1	1/3	1/5	5,2%
Flex	1/6	3	1	1/2	12,1%
Silverlight	1/3	5	2	1	22,9%
				Inconsistência	3%

Tabela 23 - Resultados do critério de performance para a característica C4.

No critério de compatibilidade entre os navegadores, não houveram problemas. Todas as implementações de busca dos itens funcionaram nos navegadores analisados. A exceção permaneceu para o *JavaFX* que não é suportado pelo navegador *Safari*. Com o comparativo entre as tecnologias pode ser visto na Tabela 24. Nesta aparecem o HTML5, *Silverlight* e *Flex*

empatados com 30% cada um e por último o *JavaFX* com 10%.

	Critério de compatibilidade				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	2	1	1	28,6%
JavaFX	1/2	1	1/2	1/2	14,3%
Flex	1	2	1	1	28,6%
Silverlight	1	2	1	1	28,6%
				Inconsistência	0%

Tabela 24 - Resultados do critério de compatibilidade para a característica C4.

Em se tratar de facilidade, como já havia sido relatado na Seção 6.2, a implementação foi simples e semelhante em todas as tecnologias. O mesmo ocorre ao analisar o critério de semelhança com aplicações *desktop*. As características propostas foram implementadas deixando a usabilidade da característica conforme a desejada. Por esse motivo esta característica será avaliada nos critérios de facilidade e semelhança, já que, seus resultados não influenciariam nos valores finais da avaliação.

A implementação da busca dos itens não revelou valor muito relevantes. O que pode ser destacado é a variação entre a performance das tecnologias. Neste critério o HTML5 demonstrou melhor performance. Já na compatibilidade a questão de não executar no *Safari* influenciou mais uma vez no resultado negativo do *JavaFX*.

6.3.5 C5 - ADIÇÃO DE ITENS NO PEDIDO

A implementação da característica de adição de itens possui como finalidade principal explorar o uso do *drag-and-drop*. A especificação utiliza esta funcionalidade tanto para adicionar quanto para remover um item do pedido. Esta característica é avaliada em todos os critérios, na performance por razões de renderização e nos critérios de compatibilidade facilidade e semelhança por não ser uma funcionalidade comum em aplicações *web*. Os resultados obtidos no critério de performance estão listados na Tabela 25. Nesta funcionalidade também não há comunicação entre o servidor e a aplicação cliente.

	Memória do processo	Uso da CPU
HTML5	26.243,00K	4%
JavaFX	96.722,00K	8,5%
Flex	43.035,00K	7,5%
Silverlight	39.047,00K	16,25%

Tabela 25 - Medições da performance da adição de itens no pedido.

Os dados obtidos revelam uma grande variação de memória do processo e uso do CPU. O *Silverlight*, por exemplo, ocupa baixa memória em relação ao *JavaFX*, porém quase o dobro de uso de CPU. O HTML5 difere-se por possuir menor consumo de recursos em ambos os quesitos. Baseado nestas medições, a comparação da performance nas tecnologias ocorreu conforme demonstrado na Tabela 30. O HTML5 ficou em primeiro com 53,8%. Em segundo *Flex* atingiu 25,6% seguido do *Silverlight* com 16,25% por seu alto consumo de CPU e o *JavaFX* com 8,2% pela memória utilizada pelo processo.

	Critério de performance				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	5	3	4	53,8%
JavaFX	1/5	1	1/3	1/2	8,2%
Flex	1/3	3	1	3	25,6%
Silverlight	1/4	2	1/3	1	12,4%
				Inconsistência	4%

Tabela 26 - Resultados do critério de performance para a característica C5.

A avaliação da compatibilidade entre os navegadores apontou algumas deficiências no HTML5. A funcionalidade principal de arrastar-e-soltar não funcionou no navegador *Internet Explorer*. Simplesmente não é possível arrastar. Já no navegador *Safari* a ação de arrastar funciona, porém não foi possível completar o teste, pois a grade de itens não é exibida. Este problema havia sido observado na avaliação da característica de itens do pedido. Novamente o *JavaFX* possui a avaliação prejudicada por não funcionar no navegador *Safari*. Segue a Tabela 27 com os resultados das comparações.

	Critério de compatibilidade				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	1/2	1/4	1/4	8,7%
JavaFX	2	1	1/3	1/3	14,3%
Flex	4	3	1	1	38,5%
Silverlight	4	3	1	1	38,5%
				Inconsistência	1%

Tabela 27 - Resultados do critério de compatibilidade para a característica C5.

O comparativo da compatibilidade deixa clara a igualdade entre o *Flex* e o *Silverlight* ao atingirem 38,5%. Isto se deve à regularidade da funcionalidade entre os navegadores. Pelos motivos relatados o *JavaFX* encontra-se com 14,3% e o HTML5 com 8,7%.

Ao realizar o desenvolvimento desta característica encontrou-se dificuldades em relação ao *Silverlight*. Conforme descrito na Seção 6.2.2, este possui um conceito diferenciado para habilitar o arrastar-e-soltar, isto dificultou a implementação. Além disso, como esta funcionalidade só havia sido incorporada em sua última versão, poucos eram os exemplos disponíveis. Nas outras tecnologias esta implementação não apresentou dificuldades. Apenas algumas disponibilizavam maior suporte para identificar o elemento arrastado do que outras.

A Tabela 28 exibe o comparativo da avaliação de facilidade de desenvolvimento da adição de itens no pedido. O HTML5 aparece em primeiro com 41,5%, o *JavaFX* em segundo com 29,3%. Por não dispor de suporte de identificação fácil do elemento arrastado, o *Flex* ficou com 18,5% seguido do *Silverlight*.

	Critério de facilidade de desenvolvimento				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	2	2	3	41,5%
JavaFX	1/2	1	2	3	29,3%
Flex	1/2	½	1	2	18,5%
Silverlight	1/3	1/3	1/2	1	10,7%
				Inconsistência	3%

Tabela 28 - Resultados do critério de facilidade de desenvolvimento da característica C5.

No critério de avaliação referente a semelhança com aplicações *desktop*, todas as tecnologias apresentaram a usabilidade desejada. Porém, em alguns momentos, a implementação realizada em *Silverlight* apresentou certa instabilidade. Após arrastar um elemento, outros elementos que não poderiam ser arrastados ficam “presos” ao cursor, como se o usuário estivesse os arrastando. Por conta disso é possível observar na Tabela 29 que todas as tecnologias encontram-se com 30% exceto o *Silverlight*, que esta apenas com 10%.

Critério de semelhança com aplicações <i>desktop</i>					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	1	1	3	30%
JavaFX	1	1	1	3	30%
Flex	1	1	1	3	30%
Silverlight	1/3	1/3	1/3	1	10%
				Inconsistência	0%

Tabela 29 - Resultados do critério de semelhança com aplicações *desktop* da característica C5.

A característica de adição de itens no pedido apresentou deficiências em algumas tecnologias. Destaque para os problemas encontrados no *Silverlight* e na incompatibilidade do HTML5 em alguns navegadores. Já nos critérios de performance e facilidade de desenvolvimento observou-se o HTML5 como destaque. Por fim, o *Flex* e o *Silverlight* empataram novamente no critério de compatibilidade.

6.3.6 C6 - QUANTIDADE SOLICITADA DOS ITENS

A característica de quantidade solicitada regia a possibilidade de alterar o valor da quantidade diretamente na grade dos itens. Esta funcionalidade não possui muito impacto em relação a performance. Isto pode ser observado na Tabela 30, onde o consumo do CPU é relativamente baixo.

	Memória do processo	Uso da CPU
HTML5	26.471,00K	1,5%
JavaFX	81.510,00K	2%
Flex	42.803,00K	3,75%
Silverlight	40.903,25K	4,5%

Tabela 30 - Medições da performance de alterar a quantidade solicitada dos itens.

As medições confirmam a tendência do alto consumo de memória da tecnologia *JavaFX*. Porém, seu uso de CPU é baixo, perdendo apenas para o HTML5. Em questão de memória do processo, o *Flex* e o *Silverlight* estão muito próximos, o que os diferencia é o maior uso da CPU por parte do *Silverlight*. Com estas observações a avaliação ocorreu conforme a Tabela 31. Nela é possível observar a vantagem do HTML5 com 54,6%, seguido do *Flex*, *Silverlight* e *JavaFX* com 23,2%, 13,8% e 8,4% respectivamente.

	Critério de performance				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	5	3	4	54,6%
JavaFX	1/5	1	1/3	1/2	8,4%
Flex	1/3	3	1	2	23,2%
Silverlight	1/4	2	1/2	1	13,8%
				Inconsistência	2%

Tabela 31 - Resultados do critério de performance para a característica C6.

Para editar os valores diretamente na grade, a característica de quantidade solicitada de itens define a utilização de um controle que auxilie a alteração incrementando ou decrementando o valor. A especificação do HTML5 define um controle assim, porém somente no navegador *Chrome* este é renderizado. No *Firefox*, este é representado por uma caixa de texto simples sem validações. Já no *Internet Explorer*, por não ser possível a adição de itens, e no *Safari*, por não exibir a grade de itens, não foi possível realizar o teste. Com estas constatações o comparativo encontra-se na Tabela 32.

Critério de compatibilidade					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	1/2	1/3	1/3	10,9%
JavaFX	2	1	1/2	½	18,9%
Flex	3	2	1	1	35,1%
Silverlight	3	2	1	1	35,1%
				Inconsistência	0,01%

Tabela 32 - Resultados do critério de compatibilidade para a característica C6.

A avaliação aponta o *Silverlight* e o *Flex* empatados com 35,1% seguidos do *JavaFX* com 18,9%. Isto se deve ao fato da incompatibilidade o *JavaFX* com o *Safari*. Por fim o HTML5, com 10,9%, por não possuir suporte e impossibilitar a avaliação em todos os navegadores.

O controle de edição de campos numéricos citados anteriormente não possui versão implementada nativamente na tecnologia *JavaFX*. Como o objetivo é não utilizar controles de terceiros, este foi representado como um campo de texto simples. Outra dificuldade encontrada foi a de atualizar os valores dos totais ao alterar uma quantidade. A tecnologia que apresentou menor dificuldade foi o HTML5 seguido do *Silverlight*, *Flex* e *JavaFX*. Está foi a ordem encontrada ao avaliar o critério de facilidade de desenvolvimento. Nesta sequência, as avaliações resultaram em 55,3% para o HTML5 seguido de 22,6% para o *Silverlight* por fim o *Flex* e o *JavaFX* receberam 15% e 7,1% respectivamente conforme exibido na Tabela 26.

Critério de facilidade de desenvolvimento					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	6	4	3	55,3%
JavaFX	1/6	1	1/3	1/3	7,1%
Flex	1/4	3	1	1/2	15%
Silverlight	1/3	3	2	1	22,6%
				Inconsistência	3%

Tabela 33 - Resultados do critério de facilidade de desenvolvimento da característica C6.

A avaliação do critério de semelhança com as aplicações *desktop* seguem na linha das dificuldades encontradas e da falta de compatibilidade. Ou seja, pelo fato de não haver um

controle numérico conforme especificado, a usabilidade na tecnologia *JavaFX* e HTML5 ficou prejudicada. O reflexo disso pode ser visto na avaliação do critério de semelhança exposta na Tabela 34. Com isso, o *Flex* e o *Silverlight* encontram-se em primeiro com 35,1% seguidos do HTML5 com 18,9% e do *JavaFX* com 10,9%.

	Critério de semelhança com aplicações <i>desktop</i>				Totais
	HTML5	<i>JavaFX</i>	<i>Flex</i>	<i>Silverlight</i>	
HTML5	1	2	1/2	1/2	18,9%
<i>JavaFX</i>	1/2	1	1/3	1/3	10,9%
<i>Flex</i>	2	3	1	1	35,1%
<i>Silverlight</i>	2	3	1	1	35,1%
				Inconsistência	0,01%

Tabela 34 - Resultados do critério de semelhança com aplicações *desktop* da característica C6.

Apesar se ser considerada uma característica pequena, a quantidade solicitada revelou algumas deficiências nas tecnologias *JavaFX* e HTML5. Além de dificuldades de implementação principalmente em *JavaFX* e *Flex*. Os destaques ficam para facilidade e performance do HTML5 e para a semelhança com aplicações *desktop* e compatibilidade do *Flex* e *Silverlight*.

6.3.7 C7 - VALIDAÇÕES DAS INFORMAÇÕES GERAIS

A validação das informações gerais não é uma funcionalidade complexa. A exibição das mensagens ao lado dos campos não preenchidos não requer grandes recursos computacionais. Isto pode ser observado nas medições listadas na Tabela 35, onde o uso do CPU é relativamente baixo. Porém, o *Flex* demonstrou o contrário atingindo a média de 8% de uso do CPU. Como já mencionado anteriormente, a memória segue o padrão medido nas outras características. O que difere nesta é que o *Silverlight* utilizou um pouco a menos de memória que o HTML5 que vinha sempre com melhor performance.

	Memória do processo	Uso da CPU
HTML5	26.015,00K	1,5%
JavaFX	86.153,00K	1,5%
Flex	35.077,00K	8%
Silverlight	25.568,00K	1,25%

Tabela 35 - Medições da performance das validações das informações gerais.

Por meio das medições realizadas, na Tabela 36 estão listados os resultados da avaliação de performance entre as tecnologias. Neste, o *Silverlight* aparece em primeiro com 45,9% pelo seu baixo consumo de memória e uso de CPU. Em seguida o HTML5 com 30,5% e por fim o *JavaFX* com 14,3% e o *Flex* com 9,3%, ambos muito próximos em função do uso de memória e de processamento.

	Critério de performance				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	3	3	1/2	30,5%
JavaFX	1/3	1	2	1/3	14,3%
Flex	1/3	1/2	1	1/4	9,3%
Silverlight	2	3	4	1	45,9%
				Inconsistência	3%

Tabela 36 - Resultados do critério de performance para a característica C7.

No critério de compatibilidade entre os navegadores as tecnologias apresentaram poucas inconsistências. Nesta característica pode-se destacar que o HTML5 não possui suporte de campo requerido nos navegadores *Internet Explorer* e *Safari*. Além disso, o *JavaFX* possui a desvantagem de não funcionar no *Safari*. Com estas observações, o comparativo desta característica recebeu as avaliações exibidas na Tabela 40. Destaque para o *Flex* e *Silverlight*, ambos com 38,5% por não apresentarem nenhuma incompatibilidade. Em segundo o *JavaFX* com 14,3%, por não funcionar no *Safari*. Por fim, o HTML5 com apenas 8,7% devido aos problemas encontrados no *Safari* e *Internet Explorer*.

	Critério de compatibilidade				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	1/2	1/4	1/4	8,7%
JavaFX	2	1	1/3	1/3	14,3%
Flex	4	3	1	1	38,5%
Silverlight	4	3	1	1	38,5%
				Inconsistência	0,01%

Tabela 37 - Resultados do critério de compatibilidade para a característica C7.

A implementação desta característica ocorreu de forma diferente entre as tecnologias. No *Flex* e no *JavaFX*, foi necessária a implementação manual das mensagens de erro. No *Silverlight* há um conceito diferenciado conforme descrito na Seção 6.2.2. Esta forma de desenvolvimento dificulta a primeira vista, porém ao compreender o objetivo se torna muito interessante. A característica desenvolvida em HTML5 foi a que apresentou maior facilidade necessitando atribuir apenas uma propriedade.

Baseando-se nestas diferenças entre as implementações, realizou-se a avaliação exibida na Tabela 38. Em primeiro está o HTML5 com 48,1%, seguido do *Silverlight* com 29,5%. O primeiro com um desenvolvimento relativamente fácil e o outro facilitando implementações avançadas. Em função da implementação manual, o *Flex* e o *JavaFX* receberam 13,1% e 9,2% respectivamente.

	Critério de facilidade de desenvolvimento				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	4	4	2	48,1%
JavaFX	1/4	1	1/2	1/3	9,2%
Flex	1/4	2	1	1/3	13,1%
Silverlight	1/2	3	3	1	29,5%
				Inconsistência	3%

Tabela 38 - Resultados do critério de facilidade de desenvolvimento da característica C7.

A usabilidade entre as implementações realizadas não apresentam diferenças significativas. O *Silverlight*, por exemplo, somente aparece a mensagem ao selecionar o campo requerido. Além disso, no HTML5 houve diferença na exibição da mensagem de

campo requerido entre os navegadores. Já o *JavaFX* e o *Flex* não apresentaram deficiências em relação ao critério de semelhança. Com isso os resultados deste critério encontram-se dispostos na Tabela 39. O *Flex* e o *JavaFX* aparecem ambos com 35,1% seguidos pelo *Silverlight* com 18,9% e o HTML5 com 10,9%.

Critério de semelhança com aplicações <i>desktop</i>					
	HTML5	<i>JavaFX</i>	<i>Flex</i>	<i>Silverlight</i>	Totais
HTML5	1	1/3	1/3	1/2	10,9%
<i>JavaFX</i>	3	1	1	2	35,1%
<i>Flex</i>	3	1	1	2	35,1%
<i>Silverlight</i>	2	1/2	1/2	1	18,9%
Inconsistência					0,01%

Tabela 39 - Resultados do critério de semelhança com aplicações *desktop* da característica C7.

As avaliações da característica de validações das informações gerais demonstraram resultados distintos às características anteriores. Nesta, o *Silverlight* se mostrou melhor no critério de performance. No critério de compatibilidade o *Silverlight* aparece novamente em primeiro juntamente com o *Flex*. O *JavaFX* e o *Flex* se destacaram na semelhança com aplicações *desktop* por suas aparências e usabilidades. Já o critério de facilidade enfatizou a simplicidade da implementação em HTML5.

6.3.8 C8 - LOGÍSTICA NA EMISSÃO DO PEDIDO

A característica C8 consiste na exibição de um mapa utilizando a API do *Google*, além de exibir informações referentes a geolocalização de onde está sendo executada a aplicação. Pelo fato destas funcionalidades necessitarem a comunicação de arquivos, cabe a esta característica avaliar indicadores de transferência de arquivos. Para isso, conforme especificado, foi utilizada a ferramenta de medições do *Google Chrome*. Além disso, permanecem sendo consideradas as medições de memória do processo e utilização da CPU. As médias dos resultados obtidos estão relacionadas na Tabela 40.

KB	Tempo de	Memória do	Uso da CPU
-----------	-----------------	-------------------	-------------------

	transferidos	transferência	processo	
HTML5	204,73KB	22,74s	34.289,75K	11,25%
<i>JavaFX</i>	NA	NA	NA	NA
<i>Flex</i>	615,81KB	10,22s	43.599,00K	21,25%
<i>Silverlight</i>	1.230,00KB	8,60s	59.900,00K	30,5%

Tabela 40 - Medições da performance na logística na emissão do pedido.

É possível observar que não há medições realizadas para o *JavaFX*. Isto se deve ao fato de não haver uma API de mapas específica desenvolvida nativamente conforme descrito anteriormente na Seção 6.2.4. Em relação aos resultados obtidos destaca-se a grande taxa de transferência do *Silverlight* destoando do tempo de transferência, porém mesmo assim esta foi a que apresentou um maior uso de CPU e memória. Apesar do elevado tempo de transferência, o HTML5 possui os menores valores nos outros indicadores o que a deixa em vantagens neste comparativo.

Baseado nas medições relatadas os resultados dos comparativos estão expostos na Tabela 41. Neste o HTML5 possui grande vantagem por chegar a 50% seguido pelo *Flex* com 28,6% e por fim o *Silverlight* e o *JavaFX* com 17,9% e 3,4% respectivamente. Todas as tecnologias analisadas receberam nota 9 em relação ao *JavaFX*. Esta característica elevou a taxa de inconsistência, porém permaneceu abaixo dos 10% definidos como limite máximo.

	Critério de performance				Totais
	HTML5	<i>JavaFX</i>	<i>Flex</i>	<i>Silverlight</i>	
HTML5	1	9	2	4	50%
<i>JavaFX</i>	1/9	1	1/9	1/9	3,4%
<i>Flex</i>	1/2	9	1	2	28,6%
<i>Silverlight</i>	1/4	9	1/2	1	17,9%
				Inconsistência	7%

Tabela 41 - Resultados do critério de performance para a característica C8.

As funcionalidades propostas nesta característica foram avaliadas nos navegadores conforme as especificações do critério de compatibilidade. Nesta foi constatado que apesar do mapa funcionar corretamente, no navegador *Safari*, o HTML5 não conseguiu realizar a geolocalização. Nem a distância e nem a rota foram calculadas. Além disso, ao adicionar o

mapa na aplicação desenvolvida em *Silverlight*, esta não funcionou mais no navegador *Safari*. Já o *Flex* funcionou corretamente sem apresentar incompatibilidades. Por estas razões, no critério de compatibilidade, o *Flex* recebeu 54% seguido do *Silverlight* com 25,4% e por fim, o HTML5 atingindo 17,3%. Estes resultados estão demonstrados na Tabela 42.

	Critério de compatibilidade				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	9	1/4	1/2	17,3%
JavaFX	1/9	1	1/9	1/9	3,3%
Flex	4	9	1	3	54,0%
Silverlight	2	9	1/3	1	25,4%
				Inconsistência	9%

Tabela 42 - Resultados do critério de compatibilidade para a característica C8.

Desconsiderando o *JavaFX*, por não possuir suporte a esta característica nativamente, todas as outras tecnologias possuem exemplos disponíveis da implementação de mapas. Em relação a facilidade cabe destacar a alta complexidade de desenvolvimento do *Silverlight*. Neste foram necessárias diversas configurações para chegar ao resultado esperado. Em contra partida, esta complexidade possibilita maior configuração das funcionalidades. Conforme citado anteriormente o *Flex* possui o ponto negativo de sua API estar depreciada. Já o HTML5 não apresentou dificuldades, nem de desenvolvimento e nem por falta de documentação.

Ao avaliar estes fatos em relação ao critério de facilidade, os resultados dos comparativos apontaram a vantagem do HTML5 com 50%. Em segundo o *Silverlight* atingiu 28,6%. Já o *Flex*, por não haver continuidade da API utilizada, recebeu 17,9% seguido do *JavaFX* com apenas 3,4%. Estes dados estão relacionados na Tabela 43.

Critério de facilidade de desenvolvimento					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	9	4	2	50%
JavaFX	1/9	1	1/9	1/9	3,4%
Flex	1/4	9	1	1/2	17,9%
Silverlight	1/2	9	2	1	28,6%
				Inconsistência	7%

Tabela 43 - Resultados do critério de facilidade de desenvolvimento da característica C8.

O critério de avaliação de semelhança com aplicações *desktop*, nesta característica analisou a usabilidade e se a tecnologia atendeu as especificidades. Com exceção do *JavaFX*, em todas as tecnologias foi possível implementar a localização, o trajeto e mapa que havia sido proposto. Com isso, o *HTML5*, o *Flex* e o *Silverlight* receberam 32,1% contra 3,6% da tecnologia *JavaFX* na comparação realizada. Estes resultados estão dispostos na Tabela 44.

Critério de semelhança com aplicações <i>desktop</i>					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	9	1	1	32,1%
JavaFX	1/9	1	1/9	1/9	3,6%
Flex	1	9	1	1	32,1%
Silverlight	1	9	1	1	32,1%
				Inconsistência	0%

Tabela 44 - Resultados do critério de semelhança com aplicações *desktop* da característica C8.

As comparações realizadas para a característica de logística na emissão do pedido revelaram alguns pontos interessantes. Pela primeira vez o *Silverlight* apresentou uma incompatibilidade. Já na tecnologia *JavaFX* não foi possível implementar os requisitos de forma nativa. Além disso, destaca-se o *HTML5* por apresentar-se melhor nos critérios de performance e facilidade de desenvolvimento. Já no critério de compatibilidade, o *Flex* apresentou melhores resultados.

6.3.9 C9 - GRÁFICOS ESTATÍSTICOS

A característica referente aos gráficos estatísticos definia a implementação de dois gráficos interativos. Esta visava provar a capacidade de renderização gráfica de forma dinâmica. Esta característica exige um maior recurso computacional tornando o critério de performance altamente importante. Neste também foram feitas as medições de transferência, além do uso de memória e CPU. Os valores obtidos estão relacionados na Tabela 45.

	KB transferidos	Tempo de transferência	Memória do processo	Uso da CPU
HTML5	2,11KB	312,75ms	5.419,50K	2,25%
JavaFX	26,91KB	537,75ms	49.284,00K	30,25%
Flex	68,44KB	2.040,00ms	29.423,00K	21,25%
Silverlight	1.070,00KB	6.720,00ms	19.826,25K	21,5%

Tabela 45 - Medições da performance na logística na emissão do pedido.

Ao analisar os dados é possível observar a enorme diferença do HTML5 para as outras tecnologias. Todos os indicadores estão extremamente abaixo dos restantes. O *JavaFX* possui a segunda melhor taxa de transferência porém o uso de memória e CPU foram os maiores registrados. A utilização de memória do *Silverlight* apresentou-se de certa forma razoável em relação ao *JavaFX* e o *Flex*, porém a quantidade de KB transferidos e tempo de transferência foram demasiadamente altos. Com isso ao comparar as tecnologias o HTML5 se destacou atingindo 63,9%. O *JavaFX*, o *Flex* e *Silverlight* não apresentaram grandes variações obtendo 18,0%, 10,9% e 7,1%, respectivamente. Estes dados estão relacionados na Tabela 46.

Critério de performance					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	5	6	6	63,9%
JavaFX	1/5	1	2	3	18,0%
Flex	1/6	1/2	1	2	10,9%
Silverlight	1/6	1/3	1/2	1	7,1%
				Inconsistência	4%

Tabela 46 - Resultados do critério de performance para a característica C9.

Ao avaliar a compatibilidade entre os navegadores desta característica, não foram encontradas muitas inconformidades. Além do fato já conhecido do *JavaFX* não executar no *Safari*, o *HTML5* apresentou comportamento diferenciado no *Firefox*. Ao ser iniciada, a aplicação desenvolvida com o *HTML5*, os gráficos especificados aparecem parcialmente o que impossibilita sua compreensão. Avaliando estes detalhes a comparação realizada para o critério de compatibilidade resultou nos valores dispostos na Tabela 47. Nela, os resultados apontam o *Flex* e o *Silverlight* com 35,1%. Em seguida, com 18,9% aparece o *HTML5* próximo do *JavaFX* com apenas 10,9%.

Critério de compatibilidade					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	2	1/2	1/2	18,9%
JavaFX	1/2	1	1/3	1/3	10,9%
Flex	2	3	1	1	35,1%
Silverlight	2	3	1	1	35,1%
				Inconsistência	0,1%

Tabela 47 - Resultados do critério de compatibilidade para a característica C9.

Ao desenvolver esta característica cada tecnologia disponibiliza controles nativos para a implementação de gráficos estatísticos com exceção do *HTML5*. Este não possui nenhuma especificação de controles avançados. Com isso necessitaram esforços demasiados para que a aplicação desenvolvida em *HTML5* atendesse aos requisitos definidos. Ao contrário, os controles do *Silverlight*, do *Flex* e do *JavaFX* mostraram-se fáceis de serem manipulados e configurados. Destaque neste item para o *Flex* onde seus controles já possuíam as animações

necessárias. Com estas observações a comparação resultou nos valores relacionados na Tabela 48.

Critério de facilidade de desenvolvimento					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	1/3	1/4	1/4	8,2%
JavaFX	3	1	1/2	1	23,5%
Flex	4	2	1	2	42,9%
Silverlight	4	1	1/2	1	25,5%
				Inconsistência	2%

Tabela 48 - Resultados do critério de facilidade de desenvolvimento da característica C9.

A usabilidade das implementações foi atingida por todas as tecnologias. Isto supre o critério de semelhança de forma quase igualitária. O HTML5, por ser uma implementação manual, possibilita a criação de maiores detalhes, porém o trabalho é demasiado conforme relatado anteriormente. Os controles utilizados pelo *JavaFX* possuem aparência muito bem elaborada. O *Flex*, por sua vez, apresentou usabilidade mais constante durante sua execução, principalmente pela interação especificada na característica ser suportada de forma nativa. Com estas considerações as comparações resultaram nos dados relacionados na Tabela 49. Esta apresenta o *Flex* a frente com 34% seguido, sem grande variação, do *JavaFX*, *Silverlight* e HTML5 com 28,1%, 23,9% e 14%, respectivamente.

Critério de semelhança com aplicações <i>desktop</i>					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	1/2	1/2	1/2	14%
JavaFX	2	1	1	1	28,1%
Flex	2	1	1	2	34%
Silverlight	2	1	1/2	1	23,9%
				Inconsistência	2%

Tabela 49 - Resultados do critério de semelhança com aplicações *desktop* da característica C9.

Na avaliação da característica de gráficos estatísticos a tecnologia *Flex* apresentou destaque nos critérios de semelhança e facilidade. Além disso, ficou em primeira na compatibilidade juntamente com o *Silverlight*. O HTML5 mostrou-se muito eficiente no

critério de performance porem deixou a desejar nos outros critérios. Isto se deve a esta ser apenas uma especificação e que não foi utilizada nenhum componente de terceiro.

6.3.10 C10 - TRANSIÇÃO DE TELAS NO PEDIDO

A característica de transição de telas no pedido consiste em avaliar a capacidade da tecnologia executar uma animação. As medições do critério de performance avaliaram a utilização da memória e do CPU. Os resultados obtidos confirmam a alta necessidade de hardware para execução de animações. Na Tabela 50 estão relacionadas as médias dos valores obtidos. Nesta é possível observar que houve muita variação entre os indicadores. O HTML5, por exemplo, possuiu a menor utilização de memória. O menor uso de CPU ficou para o *JavaFX* que ao mesmo tempo demonstrou utilização exagerada de memória.

	Memória do processo	Uso da CPU
HTML5	25.539,00K	22,25%
JavaFX	76.998,25K	14,75%
Flex	40.995,00K	18,25%
Silverlight	32.781,00K	23%

Tabela 50 - Medições da performance da transição de telas no pedido.

Ao analisar os dados acima de forma normalizada, avaliando a representatividade dos valores de cada tecnologia, obtiveram-se os resultados encontrados na Tabela 51. Nela é possível observar que o baixo consumo de memória deixou o HTML5 com 46,7%. Entre o *Flex* e o *Silverlight* o critério decisivo foi o uso do CPU, onde com menos consumo o *Flex* atingiu 27,7% seguido do *Silverlight* com 16%. Por fim, em razão do alto consumo de memória, o *JavaFX* obteve apenas 9,5%.

	Critério de performance				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	4	2	3	46,7%
JavaFX	1/4	1	1/3	1/2	9,5%
Flex	1/2	3	1	2	27,7%
Silverlight	1/3	2	1/2	1	16%
				Inconsistência	1%

Tabela 51 - Resultados do critério de performance para a característica C10.

A compatibilidade da implementação desta característica não apresentou muitas variações. Como visto nas características anteriores o *JavaFX* não apresentou suporte no navegador *Safari*. Além disso, as transições entre as telas não foram executadas no *Internet Explorer* ao avaliar a aplicação implementada em HTML5. Com estas considerações, na Tabela 52 os resultados das comparações apontam o *Flex* e o *Silverlight* ambos com 35,1%. Em seguida, aparecem o HTML5 e o *JavaFX* respectivamente com 18,9% e 10,9%..

	Critério de compatibilidade				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	2	1/2	1/2	18,9%
JavaFX	1/2	1	1/3	1/3	10,9%
Flex	2	3	1	1	35,1%
Silverlight	2	3	1	1	35,1%
				Inconsistência	0,01%

Tabela 52 - Resultados do critério de compatibilidade para a característica C10.

Mesmo havendo soluções distintas em cada tecnologia, não houve maiores dificuldades na implementação da característica de transição de telas. Foi possível encontrar exemplos de todas as implementações necessárias. Pouca variação do critério de facilidade foi encontrada, esta variação está demonstrada na Tabela 53. Por possibilitar o desenvolvimento com menos linhas de código facilitando também a compreensão, o HTML5 recebeu 40%. Em seguida com 20%, o *JavaFX*, *Flex* e o *Silverlight* também apresentaram soluções simplificadas.

Critério de facilidade de desenvolvimento					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	2	2	2	40%
JavaFX	1/2	1	1	1	20%
Flex	1/2	1	1	1	20%
Silverlight	1/2	1	1	1	20%
				Inconsistência	0%

Tabela 53 - Resultados do critério de facilidade de desenvolvimento da característica C10.

As funcionalidades especificadas na característica de transição de tela foram desenvolvidas com sucesso em todas as implementações. Na questão visual o *Silverlight* apresentou uma pequena deficiência. Onde, dependendo da utilização, a animação nem sempre é executada. Isto pode ocorrer devido ao consumo de hardware, porém acaba afetando semelhança com aplicações *desktop*. Em função disso, os resultados dos comparativos estão apresentados na Tabela 54. Nesta, o HTML5, o *JavaFX* e o *Flex*, apresentaram avaliação de 28,6%. Por consequência da instabilidade, o *Silverlight* apresentou resultado de 14,3%.

Critério de semelhança com aplicações desktop					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	1	1	2	28,6%
JavaFX	1	1	1	2	28,6%
Flex	1	1	1	2	28,6%
Silverlight	1/2	1/2	1/2	1	14,3%
				Inconsistência	0%

Tabela 54 - Resultados do critério de semelhança com aplicações desktop da característica C10.

Os comparativos realizados, em relação a característica de transição de telas, demonstraram a efetividade da tecnologia HTML5 nos critérios de performance e facilidade de desenvolvimento. Em relação aos outros critérios, houve uma maior regularidade entre as tecnologias. O destaque negativo ficou para o *Silverlight* na semelhança com aplicações *desktop*. Em contra partida o critério de compatibilidade apontou novamente vantagem para o *Flex* e o *Silverlight*.

6.3.11 C12 - USO DE WEBCAM PARA OBTER PROMOÇÕES NO PEDIDO

Esta característica representa a utilização de recursos de hardware em aplicações *web*. Neste caso a premissa é o acesso a *webcam*. Sendo assim, a medição de performance se torna relevante à avaliação desta característica. As médias das avaliações realizadas estão dispostas na Tabela 55. Nela é possível observar que não há valores para o HTML5 e o *JavaFX*. Conforme relatado nas implementações da aplicação (Seção 6.2), estas não possibilitaram o desenvolvimento desta funcionalidade.

	Memória do processo	Uso da CPU
HTML5	NA	NA
<i>JavaFX</i>	NA	NA
<i>Flex</i>	45.634,50K	5,25%
<i>Silverlight</i>	40.302,00K	37,5%

Tabela 55 - Medições da performance do uso da *webcam* para obter promoções no pedido.

Ao analisar os valores obtidos, nota-se que o *Flex* demonstrou baixo consumo de CPU em relação ao *Silverlight*. Já o indicador de memória do que se mostrou mais eficaz foi o *Silverlight*, porém com pouca diferença. A grande variação entre os valores medidos do uso de CPU indica uma clara vantagem para o *Flex*. Por estes motivos, no comparativo representado pela Tabela 56, o *Flex* atingiu 57,9% contra 32,8% do *Silverlight*. Já o HTML5 e o *JavaFX* ficaram com apenas 4,7% por não terem suas implementações sucedidas.

	Critério de performance				Totais
	HTML5	<i>JavaFX</i>	<i>Flex</i>	<i>Silverlight</i>	
HTML5	1	1	1/9	1/9	4,7%
<i>JavaFX</i>	1	1	1/9	1/9	4,7%
<i>Flex</i>	9	9	1	3	57,9%
<i>Silverlight</i>	9	9	1/3	1	32,8%
				Inconsistência	6%

Tabela 56 - Resultados do critério de performance para a característica C12.

A implementação realizada com HTML5, conforme especificações da W3C, não se

mostrou compatível com nenhum navegador avaliado. Conforme relatado anteriormente, isto ocorreu pelo fato desta especificação não estar concluída. O *Silverlight* e o *Flex* não apresentaram incompatibilidade em nenhum navegador, por este motivo a avaliação resultou em 45% para ambos. O HTML5 obteve 5%, mesmo resultado do *JavaFX* que não dispõe de implementação para acesso a *webcam*.

	Critério de compatibilidade				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	1	1/9	1/9	5%
JavaFX	1	1	1/9	1/9	5%
Flex	9	9	1	1	45%
Silverlight	9	9	1	1	45%
				Inconsistência	0%

Tabela 57 - Resultados do critério de compatibilidade para a característica C12.

Há documentações disponíveis na *web* sobre o desenvolvimento de acesso a *webcam* tanto em *Flex* quanto em *Silverlight*. Pelo fato de ser uma especificação ainda não concluída, pouco foi encontrado a respeito do HTML5. Mesmo assim a proposta da W3C⁵⁷ é de implementação relativamente fácil, assim como as soluções fornecidas pelo *Flex* e *Silverlight*. Com estas considerações foi possível realizar a comparação exposta na Tabela 58. Nesta, o *Flex* e o *Silverlight* alcançaram 39,3% contra 18% do HTML5 e 3,4% do *JavaFX*.

	Critério de facilidade de desenvolvimento				Totais
	HTML5	JavaFX	Flex	Silverlight	
HTML5	1	9	1/3	1/3	18%
JavaFX	1/9	1	1/9	1/9	3,4%
Flex	3	9	1	1	39,3%
Silverlight	3	9	1	1	39,3%
				Inconsistência	6%

Tabela 58 - Resultados do critério de facilidade de desenvolvimento da característica C12.

Tanto o *Flex* quanto o *Silverlight* apresentaram soluções que abrangem as

⁵⁷ <http://dev.w3.org/2011/webrtc/editor/getusermedia.html#localmediastream> Disponível em 19/06/2012

especificações desta característica. Com isso ambas se mostraram eficazes no desenvolvimento proposto atingindo a usabilidade desejada. Já o *JavaFX* e o *HTML5* não possuíram êxito, ou por não possuir solução nativa ou por não sua solução não ser suportada. Analisando estas afirmações obteve-se o comparativo relacionado na Tabela 59. Nesta, o *Flex* e o *Silverlight* apresentaram resultados de 45%. Já o *HTML5* e o *JavaFX* com apenas 5% mostraram inferioridade por não ser possível a avaliação do critério de semelhança.

Critério de semelhança com aplicações <i>desktop</i>					
	HTML5	JavaFX	Flex	Silverlight	Totais
HTML5	1	1	1/9	1/9	5%
JavaFX	1	1	1/9	1/9	5%
Flex	9	9	1	1	45%
Silverlight	9	9	1	1	45%
				Inconsistência	0%

Tabela 59 - Resultados do critério de semelhança com aplicações *desktop* da característica C12.

Nesta característica atenta-se para predominância do *Flex* e do *Silverlight*. Cabe-se ainda destacar o diferencial do *Flex* na avaliação de performance. Já os resultados encontrados para o *JavaFX* e o *HTML5*, transparece a falta de recursos e incompatibilidade das tecnologias. Mesmo assim o *HTML5* demonstra perspectiva de suportar a característica proposta.

6.4 RESULTADOS OBTIDOS

Os dados das comparações realizadas foram inseridos na ferramenta *Expert Choice*⁵⁸. Esta aplica o método analítico hierárquico conforme especificado no Capítulo 3. Avaliando as relevâncias para cada característica levantadas na Seção 6.1, o método obteve primeiramente os resultados separados por critérios.

A Figura 40 representa um gráfico com os resultados das características avaliadas para o critério de performance. Neste, o *HTML5* aparecem com grande vantagem recebendo 47,9% de indicação. Em segundo o *Flex* aparece com 24,5%, seguido do *Silverlight* com 17,5% e por fim o *JavaFX* 10,1%.

⁵⁸ <http://www.expertchoice.com/> Disponível em 15/04/2012

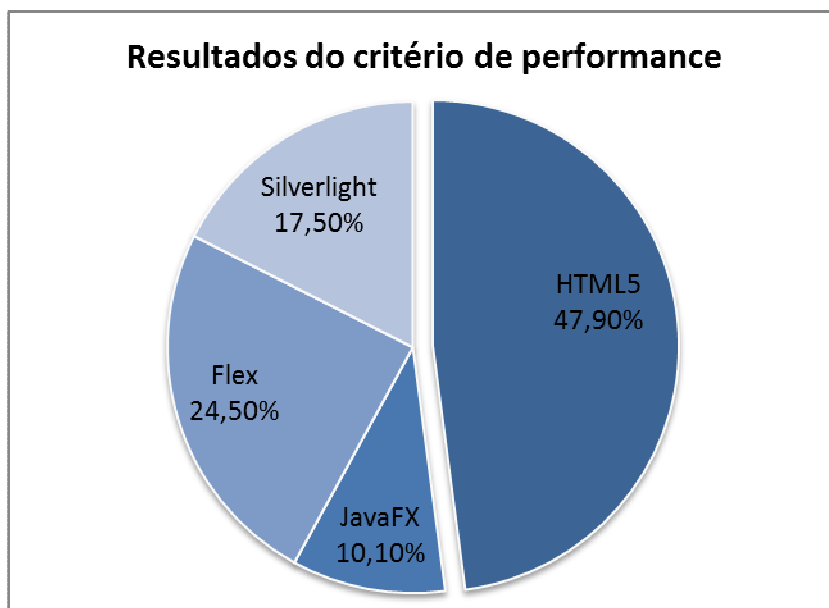


Figura 40 - Resultados obtidos em relação ao critério de performance.

Os resultados obtidos na comparação das características para o critério de compatibilidade estão expostos na Figura 41. Nesta é possível observar a pouca diferença encontrada entre o *Flex* com 38,3% e o *Silverlight* com 36,7%. Já a discrepância destas tecnologias para com o HTML5 e o *JavaFX* apontam a baixa compatibilidade de ambas. Estas por sua vez alcançaram 14,5% e 10,5% respectivamente.

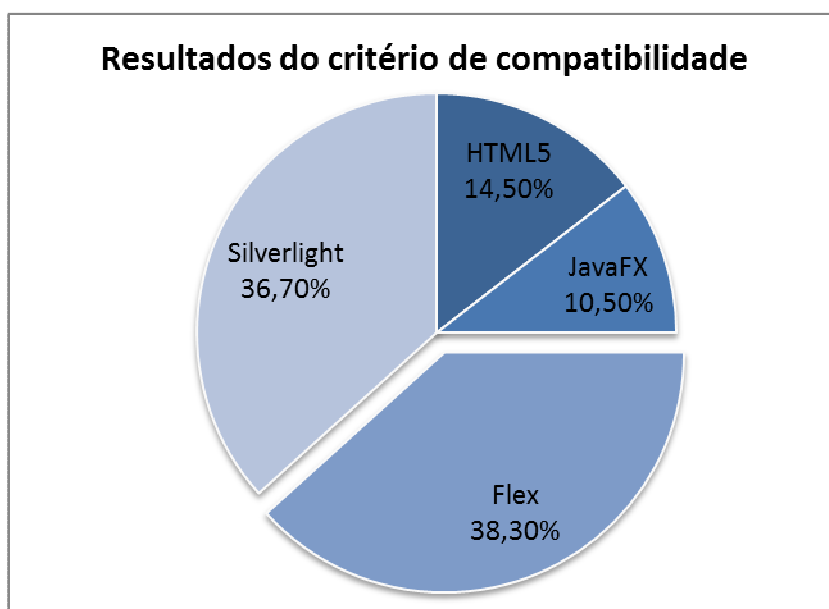


Figura 41 - Resultados obtidos em relação ao critério de compatibilidade.

O critério de facilidade demonstrou uma maior regularidade entre as tecnologias. Os resultados obtidos estão relacionados na Figura 42. Estes mostram pouca diferença entre o

Flex, o *HTML5* e o *Silverlight* apresentados com 28,8%, 27,8% e 25,5%. Entre tanto, o *JavaFX* novamente aparece em último com apenas 17,9%.

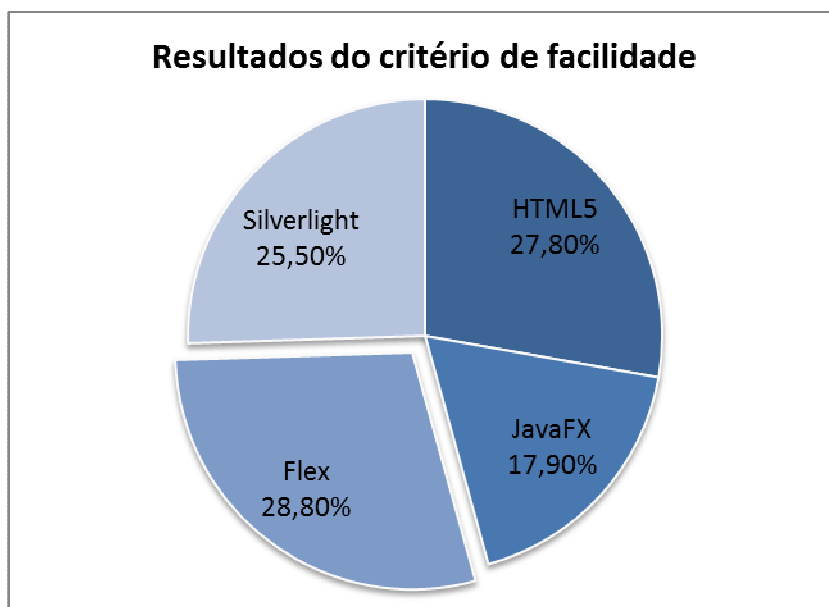


Figura 42 - Resultados obtidos em relação ao critério de facilidade de desenvolvimento.

As avaliações do critério de semelhança com aplicações *desktop* também obtiveram resultados mais padronizados. Porém neste se destaca o *Flex* novamente em primeiro com 31,6%. Apresentando resultados muito próximos entre si, aparecem o *Silverlight* com 24,8%, o *JavaFX* com 21,9% e, pela primeira vez em último, o *HTML5* com 21,7%.

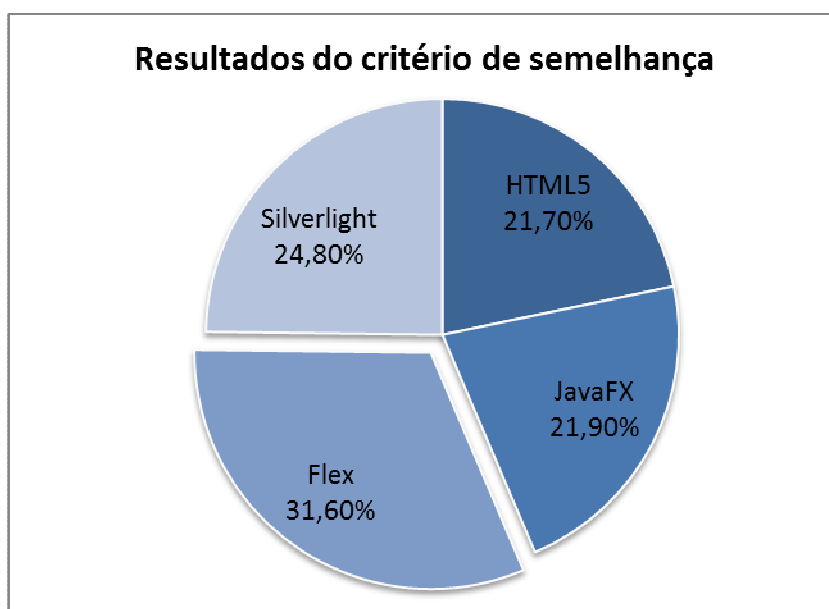


Figura 43 - Resultados obtidos em relação ao critério de semelhança com aplicações *desktop*.

Baseado nas relevâncias de cada critério definidas no início do Capítulo 6, o resultados

refletem a necessidade de uma aplicação rica mais genérica possível. Nesta, definiu-se que a performance possuiria maior relevância seguida da semelhança, da compatibilidade e por fim da facilidade. É preciso atentar que para um ambiente onde estas premissas sejam diferentes, os valores finais eventualmente podem mudar.

Utilizando as relevâncias definidas, os resultados apontaram como melhor tecnologia para desenvolvimento de aplicações ricas foi o *Adobe Flex* com 30%. Em seguida, não apresentando muita diferença, encontra-se o HTML5 com 28,3% juntamente com o *Silverlight* com 25,2%. Por último, o *JavaFX* recebeu resultado de 15,9%.

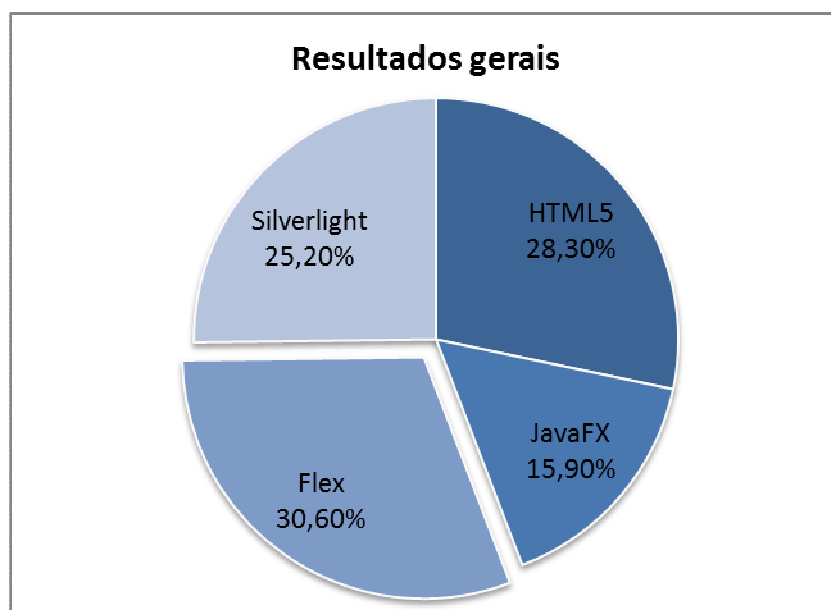


Figura 44 - Resultados gerais da avaliação.

Analisando os dados obtidos é possível observar que no critério de performance a predominância é total do HTML5, isto se deve ao fato de ser uma especificação a qual é suportada nativamente em cada navegador sem a necessidade de *plug-ins*. Contudo, o HTML5 possibilita alta customização, porém necessita realizar implementações manuais de controles avançados como uma *grid*, por exemplo.

A especificação do HTML5 ainda não está concluída, mas pode-se apontar que sua utilização será vantajosa após a conclusão se suas especificações e se houver a adoção dos fabricantes dos navegadores. Além disso, é interessante estudar a utilização de *frameworks* de terceiros baseadas em HTML5. Estas servem para facilitar sua utilização, resultando em maior produtividade de desenvolvimento. Exemplos destes frameworks são o *Kendo UI*⁵⁹ e o

⁵⁹ <http://www.kendoui.com/web.aspx> Disponível em 19/06/2012

*JQuery*⁶⁰.

Segundo as avaliações, a performance é o critério com maior relevância nesta avaliação. Mesmo havendo uma enorme vantagem para o HTML5 neste critério, a regularidade da tecnologia *Flex* lhe deixou com o melhor resultado geral. O *Flex* disponibilizou os recursos necessários para que a implementação ocorresse conforme planejado. Além disso, não utilizou de forma exagerada os recursos de *hardware* e apresentou resultados que suprem as necessidades definidas na aplicação de referência.

Em diversas características o *Silverlight* apresentou avaliações semelhantes aos do *Flex*. Isto o deixou muito próximo do HTML5 nos resultados finais. Porém, em várias avaliações esse apresentou baixo desempenho, principalmente nos critérios de facilidade e performance onde suas medições foram inferiores. O destaque ficou para compatibilidade onde juntamente com o *Flex* a variação para com o *JavaFX* e o HTML5 foi relativamente alta. Com isso concluiu-se que há maior maturidade dos *plug-ins* utilizados por ambos.

O *JavaFX* apresentou os piores resultados em praticamente todas as avaliações, isto o deixou na última posição. Pode-se atribuir esta colocação a sua imaturidade, onde houve diversas mudanças na versão avaliada (2.0). A falta de compatibilidade com o navegador Safari é um exemplo disso. Outra questão foi a utilização de recursos no critério de performance de forma exagerada, exatamente pelo fato da aplicação necessitar a utilização da máquina virtual do *Java*.

Sumarizando, os resultados obtidos estão apresentados na Tabela 60. Nesta estão relacionadas as tecnologias estudadas para com os critérios avaliados. Ainda na última coluna estão os resultados gerais apontando a vantagem do *Flex* que alcançou 30,60%.

	Performance	Compatibilidade	Facilidade	Semelhança	Geral
HTML5	47,90%	14,50%	27,80%	21,70%	28,30%
JavaFX	10,10%	10,50%	17,90%	21,90%	15,90%
Flex	24,50%	38,30%	28,80%	31,60%	30,60%
Silverlight	17,50%	36,70%	25,50%	24,80%	25,20%

Tabela 60 - Resultados finais das tecnologias para com os critérios avaliados.

Graficamente pode-se observar na Figura 45 as tendências dos resultados. Neste é clara a vantagem do HTML5 no critério de performance. O *Flex* demonstra maior

⁶⁰ <http://jqueryui.com/> Disponível em 19/06/2012

proximidade na semelhança, facilidade e compatibilidade mantendo certa constância. Próximo ao *Flex* aparece *Silverlight* e mais ao centro do gráfico está representado o *JavaFX* por ter obtido avaliações inferiores na maioria dos critérios.

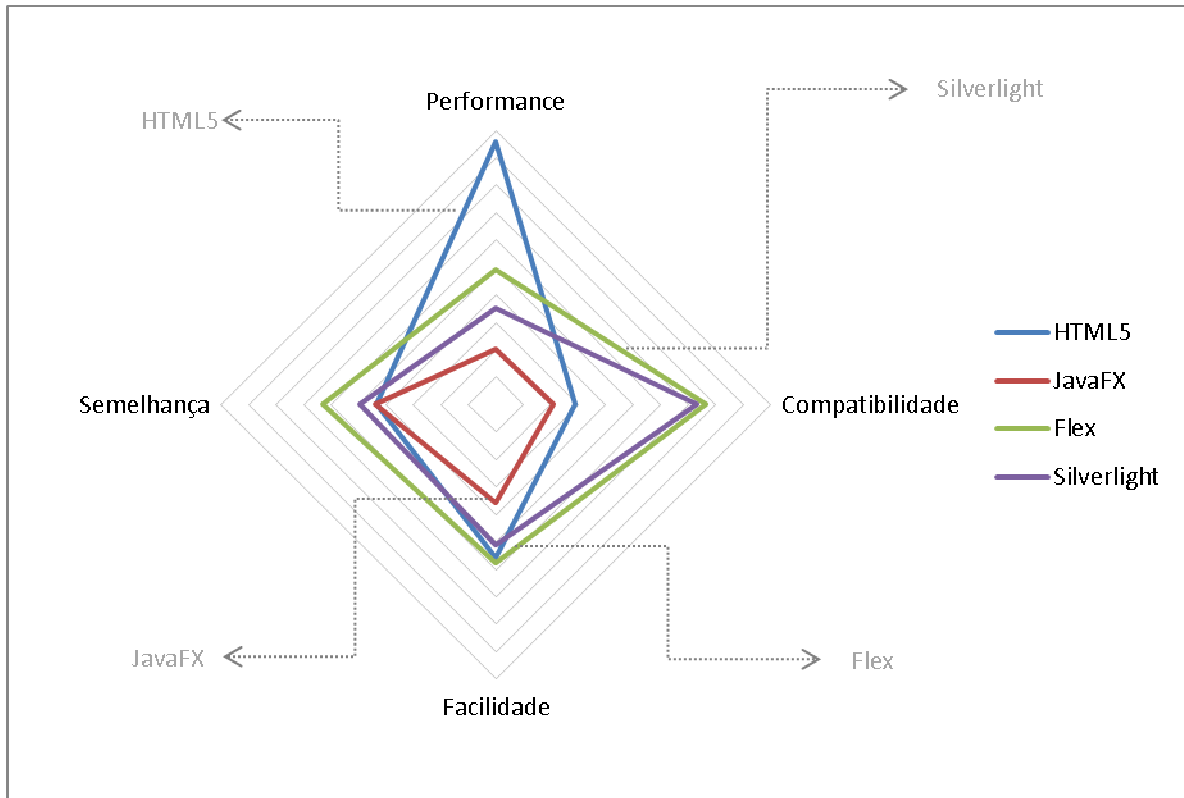


Figura 45 - Desempenho de cada tecnologia para os critérios avaliados.

7 CONSIDERAÇÕES FINAIS

Os estudos realizados neste trabalho envolveram diversas tecnologias. Tendo o ambiente *web* como base do trabalho foram detalhadas tecnologias pertinentes como o HTML, *JavaScript*, CSS, DOM. Estas também tiveram maior relevância por estarem relacionadas ao AJAX, que é tecnologia chave para a *Web 2.0* e aplicações ricas. Foram levantadas as características principais das tecnologias a serem comparadas. Houve uma maior preocupação em relação ao HTML5 por não se tratar de uma tecnologia e sim de uma especificação tecnológica.

Para realizar os comparativos foi escolhido e detalhado um método de avaliação. O método utilizado foi o Processo Analítico Hierárquico que auxilia diminuindo a subjetividade do avaliador. Baseado neste método foi definido um modelo de avaliação. Neste foram propostos quatro critérios de avaliação: performance, compatibilidade facilidade de desenvolvimento e semelhança com aplicações *desktop*.

Uma aplicação de referência foi especificada a fim de aplicar o modelo de avaliação proposto. Algumas características foram aprofundadas na finalidade de obter um maior detalhamento das funcionalidades a serem implementadas. A aplicação foi dividida em características, estas foram especificadas para serem utilizadas nos critérios propostos no modelo de avaliação.

Esta aplicação de referência foi implementada nas tecnologias *JavaFX*, *Silverlight*, *Flex* e HTML5. Todas as experiências foram relatadas e avaliadas conforme o método de avaliação. As avaliações foram confrontadas alcançando o resultado que aponta o *Flex* como tecnologia mais apropriada para o desenvolvimento de aplicações ricas seguido pelo HTML5, *Silverlight* e *JavaFX*.

Este trabalho objetiva contribuir para um melhor conhecimento das tecnologias abordadas. Ao realizar os comparativos, os resultados auxiliam na tomada de decisão ao adotar determinada tecnologia para o desenvolvimento de aplicações ricas.

A realização deste trabalho apresentou alguns não cumprimentos em relação a proposta inicial. Nem todos os objetivos, descritos na Seção 1.1, foram contemplados. O critério de escalabilidade proposto não foi avaliado devido ao foco ter sido direcionado para apenas a aplicação cliente, desconsiderando instalações e comunicação com servidores. Já o

critério de custo, teve sua não avaliação devido a dificuldade de comparar valores de tecnologias com especificações técnicas, além da necessidade de considerar custos de implantação e manutenção. As demais propostas foram avaliadas excluindo o que diz respeito a dispositivos móveis onde, conforme descrito anteriormente, o escopo foi reduzido ao ambiente *desktop*.

7.1 TRABALHOS FUTUROS

As tecnologias estudadas estão em constante evolução. Algumas se demonstraram imaturas em determinadas avaliações. Como complemento a este trabalho é interessante efetuar novamente esta avaliação em um prazo de 6 meses à 1 ano. Com isso poderá ser observada a evolução das tecnologias envolvidas e se houveram mudanças nos resultados obtidos.

O referencial teórico apontou o aumento da utilização de dispositivos móveis para acesso as aplicações *web*. A avaliação das aplicações ricas nesta utilização foi removida do escopo conforme descrito anteriormente. Com isso, mostra-se relevante a aplicação do modelo de comparação proposto em ambientes de dispositivos móveis. Isto viria complementar a avaliação realizada neste trabalho.

8 REFERÊNCIAS

ADOBE **Estrutura do Flex** Disponível em:
http://www.adobe.com/br/products/flex/flex_framework/. acessado em 22/11/2011.

ALLAIRE, Jeremy **Macromedia Flash MX – A Next-Generation Rich Client**. Macromedia White Paper, 2002.

ALVES, Lysandra G. K.; NYKIEL, Thiago P.; BELDERRAIN, Mischel C. N. **Comparação Analítica Entre Métodos de Apoio Multicritério à Decisão** Instituto Tecnológico de Aeronáutica, ITA, São José dos Campos, São Paulo, 2007.

BERNERS-LEE, Tim **Tim Berners-Lee** - <http://www.w3.org/People/Berners-Lee>, acessado em 12/09/2011.

BERNERS-LEE, Tim **Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor**. San Francisco: HarperCollins, 1999.

BEVAN, Nigel **Usability is Quality of Use** Anzai & Ogawa (eds) Proc. 6th International Conference on Human Computer Interaction, Yokohama, 1995.

BUSCH, Marianne; KOCH Nora **Rich Internet Applications – State-of-the-Art** Ludwig-Maximilians-Universität München (LMU), Germany, Dezembro 2009.

BUSH Vannervar **As We May Think** Atlantic Magazine, Julho 1945.

CAMERON, Rob **Pro Windows Phone 7 Development** Apress, EUA, 2011.

CASARIO, Marco et al. **HTML5 Solutions: Essential Techniques for HTML5 Developers** Springer, 2011.

CERN, **Where the web was born** Disponível em:
<http://public.web.cern.ch/public/en/about/web-en.html>. Acesso em 14/09/2011

CICCONI, Sergio **Hypertextuality**. Mediapolis. Ed.Sam Inkinen. Merlino & New York: De Gruyter, 1999. Disponível em: <http://www.cisenet.com/?p=246>. Acesso em 24/10/2011.

COMER, Douglas E. **Redes de computadores e Internet**. 4. ed. Porto Alegre: Bookman, 2007.

DAVID, Matthew **HTML5: Designing Rich Internet Applications**. Focal Press, 2010.

DEITEL, Paul J.; DEITEL, Harvey M. **AJAX, Rich Internet Applications e Desenvolvimento Web para Programadores**. Pearson, 2009.

FAIN, Yakov, **Java Programming 24-Hour Trainer** Wrox, 2011.

FERRARI, Pollyana; MARTINEZ, Adriana G. **Hipertexto, Hiperímídia: as novas ferramentas da comunicação digital**. Contexto, 2007.

FULTON, Steve; FULTON, Jeff **HTML5 Canvas** O'Reilly Media, 2011.

GARRETT, Jesse J. **Ajax: A New Approach to Web Applications**. Adaptive Path, 2005. Disponível em: <http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications>. Acesso em: 03/10/2011.

GASSNER, David **Adobe Flex 3 Bible** Wiley Publishing Inc., Indianapolis, Indiana, 2008.

GHODA, Ashish **Introducing Silverlight 4** Apress, EUA, 2010.

GALLO, Alessandro et al. **ASP.NET AJAX in Action** Manning Publications Co., 2008.

GUGLIELMETTI, Fernando R.; MARINS, Fernando A.; SALOMON, Valério A. P. **Comparação teórica entre métodos de auxílio à tomada de decisão por múltiplos critérios**. ENEGEP, 2003.

HICKSON, Ian **HTML5: A vocabulary and associated APIs for HTML and XHTML**. Google Inc. 2011. Disponível em: <http://www.w3.org/TR/html5/>. Acesso em: 12/08/2011.

HOLDENER, Anthony T. III **HTML5 Geolocation** O'Reilly Media, Março 2011.

JOBES, Steve **Thoughts on Flash.** 2010. Disponível em: <http://www.apple.com/hotnews/thoughts-on-flash/>. Acesso em: 12/08/2011.

JORDÃO, Bruno M. C.; PEREIRA, Susete R. **A Análise Multicritério na Tomada de Decisão - O Método Analítico Hierárquico de T. L. Saaty** Instituto Politécnico de Coimbra 2006.

KEITH, Jeremy **HTML for Web Designers**, A Book Apart, 2010.

LAIR, Robert **Beginning Silverlight 4 in C#** Apress, EUA, 2010.

LAWSON Bruce; SHARP, Remy **Introducing HTML5** New Riders, 2011.

MACDONALD, Matthew **HTML5: The Missing Manual** O'Reilly Media, 2011.

MARTINS, Leandro **Curso Profissional de Hardware** Digerati Books. São Paulo, 2007.

NELSON, Theodor H. **Literary Machines** Mindful Press, Sausalito, California, 1987.

NETSCAPE **Netscape and Sun Announce Javascript, the Open, Cross-Platform Object Scripting Language for Enterprise Networks and the Internet** California, 4 de Dezembro de 1995. Disponível em: <http://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newsrelease67.html>. Acesso em 03/08/2011.

NETSCAPE **Industry Leaders to Advance Standardization of Netscape's Javascript at Standards Body Meeting** California, 15 de novembro de 1996. Disponível em: <http://web.archive.org/web/19981203070212/http://cgi.netscape.com/newsref/pr/newsrelease289.html>. Acesso em 03/08/2011.

NIEDERAUER, Juliano **Web Interativa com Ajax e PHP** Novatec, 2007.

O'REILLY, Tim **What is Web 2.0** O'Reilly Media, Inc. 30/09/2007 Disponível em: <http://oreilly.com/web2/archive/what-is-web-20.html>. Acesso em 13/09/2011.

ORACLE **About JavaFX** Disponível em: <http://javafx.com/about-javafx/>. Acesso em: 23/11/2011.

PILGRIM, Mark **HTML5: Up and Running** O'Reilly Media, 2010.

PRESSMAN, Roger S. **Engenharia de Software** 6ª ed. Makron Books. São Paulo, 1995.

RAGGETT, David et al. **Raggett on HTML 4** 2 ed. Addison-Wesley Professional, 1997.

SANDERS, Willian B. **Smashing HTML5** Wiley, 2011.

SAATY, Thomas L. **The Analytic Hierarchy Process**, McGraw-Hill, New York, 1980.

SAATY, Thomas L. **Theory and Applications of the Analytic Network Process: Decision Making with Benefits, Opportunities, Costs, and Risks** RWS Publications, Pittsburgh, USA, 2005.

SAATY, Thomas L. **Decision making with the Analytic Hierarchy process**, Pittsburgh, USA, 2008.

SILVA, Roterdan M.; BELDERAIN, Mischel C. N. **Considerações Sobre Métodos de Decisão Multicritério** Instituto Tecnológico de Aeronáutica, ITA, São José dos Campos, São Paulo, 2005.

SINGH, Inderjeet; STEARNS, Beth; JOHNSON, Mark **Designing Enterprise Applications with the J2EE** 2 ed. Addison-Wesley, Março 2002.

SINOFSKY, Steven **Metro style browsing and plug-in free HTML5** MSDN, 14 de setembro de 2011. Disponível em: <http://blogs.msdn.com/b/b8/archive/2011/09/14/metro-style-browsing-and-plug-in-free-html5.aspx> Acesso em 03/08/2011.

VARGAS, Ricardo **Utilizando a Programação Multicritério (Analytic Hierarchy Process – AHP) para Selecionar e Priorizar Projetos na Gestão de Portfólio** PMI Global Congress, Washington – DC, EUA, 2010.

VOIDA, Stephen et al. **Getting Practical with Interactive Tabletop Displays: Designing**

for Dense Data, “Fat Fingers,” Diverse Interactions, and Face-to-Face Collaboration
Department of Computer Science, University of Calgary, Canada 2009.

W3C, **Biografia de Berners-Lee**. Disponível em: <http://www.w3.org/People/Berners-Lee/>.
Acesso em: 18/08/2011.

W3C, **FAQs: Is there support for digital rights management (DRM) in HTML5 video?**
Disponível em:
http://www.w3.org/html/wiki/FAQs#Is_there_support_for_digital_rights_management_.28DRM.29_in_HTML5_video.3F. Acesso em 17/10/2011.

W3C, **Geolocation API Specification** 10 de fevereiro de 2010. Disponível em:
<http://dev.w3.org/geo/api/spec-source.html> Acesso em 13/10/2011.

W3C, **What is CSS**. Disponível em:
<http://www.w3.org/standards/webdesign/htmlcss#whatcss>. Acesso em 03/08/2011.

WONG, Clinton **HTTP Pocket Reference: Hypertext Transfer Protocol** O'Reilly Media, 2006.

ZATTERA, Tiago S. **A Tecnologia da Informação como Ferramenta de Suporte à Gestão da Inovação** Bacharel em Sistemas de Informação da Universidade de Caxias do Sul, Caxias do Sul, 2011.

ANEXO A – Implementação da característica de informações gerais do pedido em Flex

A Figura 46 apresenta o código fonte da implementação da característica de informações gerais realizada em *Adobe Flex*. Neste foram utilizados controles do tipo *Label*, *ComboBox*, *TextArea* e *DateField*. Estes elementos estão dispostos e organizados através de suas posições *X* e *Y*.

```
<mx:Canvas id="viewInformacoesGerais" showEffect="{myWL}"
  show="viewInformacoesGerais_showHandler(event)" hideEffect="{myWL}" width="100%" height="100%"
  label="Informações Gerais" creationComplete="viewInformacoesGerais_creationCompleteHandler(event)">
  <s:Label x="10" y="17" text="Cliente"/>
  <s:Label x="10" y="67" text="Representante"/>
  <s:Label x="10" y="128" text="Forma de pagamento"/>
  <s:Label x="10" y="183" text="Observações"/>
  <s:ComboBox id="cboCliente" x="155" y="10" width="263"
    change="cboCliente_changeHandler(event)"
    dataProvider="{clientes}" errorString="" />
  <s:ComboBox id="cboRepresentante" x="156" y="62" width="263"
    change="cboRepresentante_changeHandler(event)" dataProvider="{representantes}" errorString="" />
  <s:ComboBox id="cboFormaPagamento" x="156" y="122" width="263"
    change="cboFormaPagamento_changeHandler(event)" dataProvider="{formasPagamento}" errorString="" />
  <s:TextArea id="txtObservacoes" x="156" y="183" width="265" height="127"/>
  <s:Label x="458" y="16" text="Emissão"/>
  <s:Label x="458" y="67" text="Situação"/>
  <mx.DateField id="dtfEmissao" x="540" y="10" width="142"/>
  <s:ComboBox id="cboSituacao" x="539" y="63" width="142"
    change="cboSituacao_changeHandler(event)" dataProvider="{situacoes}" errorString="" />

  <s:Label id="lblDistancia" x="458" y="437" width="395" height="33" text="Aqui mesmo!">
  </s:Label>
</mx:Canvas>
```

Figura 46 - Código fonte das informações gerais do pedido em Flex.

Na Figura 47 é possível observar o resultado obtido a partir do código representado na Figura 46. Todos os controles estão dispostos atendendo as necessidades da característica proposta.

Cliente	<input type="text"/>	Emissão	<input type="text"/>
Representante	<input type="text"/>	Situação	<input type="text"/>
Forma de pagamento	<input type="text"/>		
Observações	<input type="text"/>		

Figura 47 - Resultado da implementação de informações gerais do pedido em Flex.

ANEXO B – Implementação da característica de informações gerais do pedido em *Silverlight*

O código fonte do desenvolvimento da característica de informações gerais do pedido em *Silverlight* está exposto na Figura 48. É possível observar a utilização dos controles *Label*, *ComboBox*, *TextBox* e *DatePicker*. Estes estão organizados em uma *Grid* onde o posicionamento se dá pela margem de cada elemento.

```
<sdk:TabItem Header="tabItem1" Name="tabItem1" Visibility="Collapsed" >
  <Grid HorizontalAlignment="Right" Width="1048" >
    <Grid.DataContext>
      <local:Pedido/>
    </Grid.DataContext>
    <sdk:Label Content="Cliente" Height="20"
      Margin="6,8,8,0" VerticalAlignment="Top" HorizontalAlignment="Left" />
    <sdk:Label Content="Observações"
      Height="20" Margin="6,176,636,0" VerticalAlignment="Top" HorizontalAlignment="Left" />
    <sdk:Label Content="Forma de Pagamento"
      Height="20" Margin="6,58,589,0" VerticalAlignment="Top" HorizontalAlignment="Left" />
    <ComboBox Height="23" HorizontalAlignment="Left"
      Margin="142,6,0,0" Name="cboCliente"
      SelectedValue="{Binding Cliente, Mode=TwoWay, ValidatesOnExceptions=True}"
      VerticalAlignment="Top" Width="279" />
    <ComboBox Height="23" HorizontalAlignment="Left"
      Margin="142,58,0,0" Name="cboFormaPagamento" VerticalAlignment="Top" Width="279" />
    <ComboBox Height="23" HorizontalAlignment="Left"
      Margin="142,113,0,0" Name="cboRepresentante" VerticalAlignment="Top" Width="279" />
    <TextBox Height="139" HorizontalAlignment="Left" Margin="142,176,0,0"
      Name="textBox1" VerticalAlignment="Top" Width="279" />
    <sdk:Label Height="20" Margin="6,116,400,0"
      Content="Representante" VerticalAlignment="Top" HorizontalAlignment="Left" />
    <sdk:Label Content="Situação" Height="20"
      Margin="475,58,491,0" VerticalAlignment="Top" HorizontalAlignment="Left" />
    <ComboBox Height="23" HorizontalAlignment="Left" Margin="558,55,0,0"
      Name="vboSituacao" VerticalAlignment="Top" Width="159" />
    <sdk:Label Content="Emissão" Height="20" Margin="475,9,491,0"
      VerticalAlignment="Top" HorizontalAlignment="Left" />
    <sdk:DatePicker Height="23" HorizontalAlignment="Left" Margin="558,8,0,0"
      Name="datePicker1" VerticalAlignment="Top" Width="159" />
    <m:Map CredentialsProvider="{StaticResource CredentialsProvider1}" x:Name="MyMap"
      Center="-29.2237892150879,-51.3457412719727" ZoomLevel="12" HorizontalAlignment="left"
      Margin="475,116,0,0" Width="560" VerticalAlignment="Top" Height="360">
      <m:MapLayer x:Name="RouteLayer"/>
    </m:Map>
    <sdk:Label Height="40" HorizontalAlignment="Left" Margin="475,482,0,0"
      Name="lblDistancia" FontSize="12" VerticalAlignment="Top" Width="560" />
  </Grid>
</sdk:TabItem>
```

Figura 48 - Código fonte das informações gerais do pedido em *Silverlight*.

O resultado obtido com a utilização deste código-fonte pode ser visto na Figura 49. Nesta todos as necessidades especificadas na característica de informações gerais foram atendidas.

Cliente	<input type="text"/>	Emissão	<input type="text" value="<dd/MM/yyyy>"/> <input type="button" value="15"/>
Forma de Pagamento	<input type="text"/>	Situação	<input type="text"/>
Representante	<input type="text"/>		
Observações	<input type="text"/>		

Figura 49 - Resultado da implementação de informações gerais do pedido em *Silverlight*.

ANEXO C – Implementação da característica de informações gerais do pedido em HTML5

Neste anexo esta exposta a implementação da característica C1 realizada em HTML5. O código fonte está exibido na Figura 50 e na Figura 51. Neste foram utilizados os seguintes controles: *select*, *textarea* e *input*. Estes estão organizados dentro de *tables*.

```
<li id="divInfGerais">
  <form id="formDadosGerais" action="Cadastro.html" >
    <table>
      <tr>
        <td style="vertical-align: top;">
          <table class="tableContent">
            <tr>
              <td>
                Cliente
              </td>
              <td>
                <select id="cboCliente" class="field" ...>...</select>
              </td>
            </tr>
            <tr>
              <td>
                Representante
              </td>
              <td>
                <select id="cboRepres" class="field">...</select>
              </td>
            </tr>
            <tr>
              <td>
                Forma de pagamento
              </td>
              <td>
                <select id="cboFormaPgto">...</select>
              </td>
            </tr>
            <tr>
              <td style="vertical-align: top;">
                Observações
              </td>
              <td>
                <textarea id="txtObs" rows="0" cols="50"
                  style="height: 80px" class="field"></textarea>
              </td>
            </tr>
            <tr style="display:none;">
              <td>
                <input type="submit" value="submit" id="formSubmit" />
              </td>
            </tr>
          </table>
        </td>
      </tr>
    </table>
  </li>
```

Figura 50 - Código fonte das informações gerais do pedido em HTML5 Pate 1.

```

        </td>
    </tr>
</table>
</td>
<td style="vertical-align: top;">
    <table class="tableContent">
        <tr>
            <td>
                Emissão
            </td>
            <td>
                <input type="date" />
            </td>
        </tr>
        <tr>
            <td>
                Situação
            </td>
            <td>
                <select id="Select2" class="field">...</select>
            </td>
        </tr>
    </table>
    <div id="mapPlace" style="height: 300px; width: 500px;
        border: 1px solid black; vertical-align: middle;
        text-align: center;">
    </div>
    <div id="outputDiv">
        aqui oh
    </div>
</td>
</tr>
</table>
</form>
</li>

```

Figura 51 - Código fonte das informações gerais do pedido em HTML5 Pate 2.

O resultado obtido com a renderização do código apresentado está na Figura 52. Nesta é possível observar os controles conforme especificação. Porém, o campo de emissão que deveria representar um valor de data não possuiu representação adequada.

Cliente	<input type="text"/>	Emissão	<input type="text"/>
Representante	<input type="text"/>	Situação	<input type="text"/>
Forma de pagamento	<input type="text"/>		
Observações	<input type="text"/>		

Figura 52 - Resultado da implementação de informações gerais do pedido em HTML5.


ANEXO D – Implementação da característica de informações gerais do pedido em *JavaFX*

A Figura 53 representa o código gerado para implementar as informações gerais do pedido em *JavaFX*. Nesta são utilizados controles de *Label*, *ChoiceBox* e *TextArea* todos organizados pelas linhas e colunas do controle *GridPane*.

```
<GridPane fx:id="grpDadosPedido" prefWidth="900" hgap="8" vgap="8" style="-fx-border: 1px" >
  <children>
    <Label text="%cliente" GridPane.columnIndex="0" GridPane.rowIndex="0" labelFor="$clienteChoice"/>
    <ChoiceBox fx:id="clienteChoice" prefWidth="320" GridPane.columnIndex="1" GridPane.rowIndex="0">
      <items>
        <FXCollections>
      </items>
    </ChoiceBox>
    <Label text="%representante" GridPane.columnIndex="0" GridPane.rowIndex="1" labelFor="$representanteChoice"/>
    <ChoiceBox fx:id="representanteChoice" prefWidth="320" GridPane.columnIndex="1" GridPane.rowIndex="1">
      <items>
        <FXCollections>
      </items>
    </ChoiceBox>
    <Label text="%formaPagamento" GridPane.columnIndex="0" GridPane.rowIndex="2" labelFor="$formaPagamentoChoice"/>
    <ChoiceBox fx:id="formaPagamentoChoice" prefWidth="180" GridPane.columnIndex="1" GridPane.rowIndex="2">
      <items>
        <FXCollections>
      </items>
    </ChoiceBox>
    <Label text="%observacoes" GridPane.columnIndex="0" GridPane.rowIndex="3" labelFor="$observacoesField"/>
    <TextArea fx:id="observacoesField" prefWidth="320" prefHeight="80" GridPane.columnIndex="1" GridPane.rowIndex="3"></TextArea>
    <Label text="%situacao" GridPane.columnIndex="2" GridPane.rowIndex="1" labelFor="$situacaoChoice"/>
    <Label text="%situacao" GridPane.columnIndex="2" GridPane.rowIndex="1" labelFor="$situacaoChoice"/>
    <ChoiceBox fx:id="situacaoChoice" prefWidth="130" GridPane.columnIndex="3" GridPane.rowIndex="1">
      <items>
        <FXCollections>
      </items>
    </ChoiceBox>
  </children>
</GridPane>
```

Figura 53 - Código fonte das informações gerais do pedido em *JavaFX*.

A Figura 54 relata o resultado obtido da característica de informações gerais em *JavaFX*. Neste também não foi possível representar o controle de data para o campo de emissão.



The image shows a JavaFX form with the following fields:

- Cliente**: A dropdown menu.
- Representante**: A dropdown menu.
- Situação**: A dropdown menu.
- Forma de pagamento**: A dropdown menu.
- Obserevações**: A text area.

Figura 54 - Resultado da implementação de informações gerais do pedido em *JavaFX*.

ANEXO E – Implementação da característica de itens do pedido em *Flex*

Este anexo exhibe a implementação da característica de itens do pedido implementada na tecnologia *Flex*. Esta contempla a listagem de itens e a grade dos itens adicionados no pedido. Para isso foram utilizados os controles *TileGroup* e *DataGrid*, conforme é possível observar na Figura 55.

```
<mx:Canvas id="viewItensdoPedido" showEffect="{myWL}" hideEffect="{myWL}" show="viewItensdoPedido_showHandler(event)" width="100%"
height="100%" label="Itens do pedido">
  <s:TextInput x="34" y="10" width="484" change="txtFiltro_changeHandler(event)" id="txtFiltro"/>
  <s:Image x="10" y="10" height="22" smooth="false" source="resources/find.png"/>
  <s:Scroller x="34" y="49" verticalScrollPolicy="auto" width="481" height="398" >
    <s:TileGroup orientation="rows" id="tgpProdutos" creationComplete="tgpProdutos_creationCompleteHandler(event)">
    </s:TileGroup>
  </s:Scroller>
  <s:BorderContainer x="526" y="49" width="357" height="397" backgroundColor="#D2D2D2"
borderStyle="solid" borderVisible="true" borderWeight="2"
cornerRadius="10" dropShadowVisible="false"
dragEnter="bordercontainer2_dragEnterHandler(event)" dragDrop="bordercontainer2_dragDropHandler(event)">

  <s:Label x="67" y="14" color="#5A5A5A" text="Arraste aqui para adicionar ao pedido"/>
  <s:Image x="278" y="10"
source="../bin-debug/resources/cart_add.png"/>
  <mx:DataGrid x="10" y="44" width="333" height="339" rowHeight="30" dataProvider="{itensPedido}" editable="true" dragEnabled="true"
id="grdItensPedido" creationComplete="grdItensPedido_creationCompleteHandler(event)">
  <mx:columns>

    <mx:DataGridColumn headerText="Produto" editable="false">
      <mx:itemRenderer>
        <fx:Component>
          <mx:HBox verticalAlign="middle">
            <mx:Image height="20" width="30" source="{data._produto.urlImagem}" />
            <mx:Label text="{data._produto.nome}" />
          </mx:HBox>
        </fx:Component>
      </mx:itemRenderer>
    </mx:DataGridColumn>
    <mx:DataGridColumn dataField="qtdeSolic" editorDataField="value"
headerText="Quantidade" width="100" >
      <mx:itemEditor >
        <fx:Component>
          <mx:NumericStepper stepSize="1" maximum="999" change="numericstepper1_changeHandler(event)">
        </fx:Component>
      </mx:itemEditor>
    </mx:DataGridColumn >
    <mx:DataGridColumn editable="false" textAlign="right" dataField="_valorTotal" headerText="Total" width="100"
labelFunction="currencyLabelWithCents">
    </mx:DataGridColumn>
  </mx:columns>
</mx:DataGrid>
</s:BorderContainer>
  <mx:Canvas x="643" y="10" width="221" height="31" borderVisible="true"
dragDrop="grpExcluirItem_dragDropHandler(event)"
dragEnter="grpExcluirItem_dragEnterHandler(event)">

  <s:Label x="8" y="9" color="#5A5A5A" text="Arraste aqui para excluir um item"/>
  <s:Image x="194" y="6"
source="../bin-debug/resources/cart_delete.png"/>
</mx:Canvas>
  <s:Button x="526" y="10" label="Promoção" click="button1_clickHandler(event)"/>
</mx:Canvas>
```

Figura 55 - Código fonte dos itens do pedido em *Flex*.

Para exibir os itens na listagem de produtos foi necessária a implementação de código *Action Script*. Neste código os controles necessários são criados e recebem os atributos necessários para que a lista possa ser exibida. Isto pode ser visto no código apresentado na Figura 56.

```

public function MontaListaProdutos(filtro:String):void
{
    var content:BorderContainer;
    var descricao:Label;
    var preco:Label;
    var img:Image;
    tgpProdutos.removeAllElements();
    for each (var prod:Produto in produtos)
    {
        if (filtro == "" || prod.nome.toUpperCase().search(filtro.toUpperCase()) >= 0)
        {
            //Cria controles do produto
            content = new BorderContainer();
            descricao = new Label();
            preco = new Label();
            img = new Image();

            //Define propriedades do controle/borda
            content.id = "bctProduto_" + prod.GetCodigo();
            content.width = 144;
            content.height = 60;
            content.addEventListener(MouseEvent.CLICK, bordercontainer1_mouseDownHandler);
            content.setStyle("cornerRadius", 10);

            //Define propriedades da Descrição
            descricao.text = prod.GetNome();
            descricao.id = "lblDescricao_" + prod.GetCodigo();
            descricao.x = 55;
            descricao.y = 10;
            descricao.setStyle("fontSize",15);

            //Define propriedades do preço
            preco.text = cFormat.format(prod.GetValor());
            preco.id = "lblPreco_" + prod.GetCodigo();
            preco.x = 55;
            preco.y = 37;

            //Define propriedades da imagem
            img.source = prod.GetUrlImagem();
            img.x = 10;
            img.y = 10;
            img.width = 40;
            img.height = 40;














            //Monta hierarquia dos controles e adiciona ao TileGroup
            content.addElement(descricao);
            content.addElement(preco);
            content.addElement(img);
            tgpProdutos.addElement(content);
        }
    }
}

```

Figura 56 - Código Action Script para exibir os produtos disponíveis a venda.

Os resultados destes códigos estão representados na Figura 57. Nesta é possível observar as definições da característica de itens do pedido implementadas. À esquerda está a listagem dos produtos e a direita os itens adicionado ao pedido. Além destas também pode-se ver o filtro dos itens onde se está sendo filtrado pela letra 'r'.

Promoção
Arraste aqui para excluir um item

 Despertador R\$ 1.200,00	 Livre R\$ 50,00	 Celular R\$ 900,00
 Tesoura R\$ 23,15	 Retrato R\$ 65,00	 Bandeira R\$ 154,00
 Filmadora R\$ 1.066,00	 Câmera R\$ 3.000,00	 Microfone R\$ 80,00
 Relógio R\$ 664,00	 Cachorro R\$ 2.000,00	 Computador R\$ 3.450,00
 Cadeira R\$ 400,00		

Arraste aqui para adicionar ao pedido




Produto	Quantidade	Total
 Retrato	1	R\$ 65,00
 Celular	2	R\$ 1.800,00
 Filmadora	1	R\$ 1.066,00

Figura 57 - Resultado da implementação de itens do pedido em Flex.

ANEXO F – Implementação da característica de itens do pedido em *Silverlight*

A implementação realizada para atender os requisitos da característica de itens do pedido estão dispostos neste anexo. Na Figura 58 e na Figura 59 estão expostos os códigos em XAML para organizar os elementos visuais na tela. Neste observa-se o uso dos controles *ListBox* para representar a lista de produtos e o *DataGrid* exibir os itens adicionado no pedido.

```
<Grid HorizontalAlignment="Right" Width="1048">
  <Image Height="19" HorizontalAlignment="Left" Margin="9,10,0,0"
    Name="image1" Stretch="Fill" VerticalAlignment="Top" Width="19"
    Source="/AplicacaoRicaSL;component/Images/find.png" />
  <TextBox Height="23" HorizontalAlignment="Left" Margin="34,6,0,0"
    Name="txtPesquisaProd" VerticalAlignment="Top" Width="496"
    TextChanged="txtPesquisaProd_TextChanged" />
  <Button Content="Promoção" Height="23" HorizontalAlignment="Left"
    Margin="546,6,0,0" Name="btnPromocao" Click="btnPromocao_Click"
    VerticalAlignment="Top" Width="75" />
  <toolkit:PanelDragDropTarget AllowDrop="True" Drop="DeletePanel_Drop"
    Name="DeletePanel" Margin="829,6,6,468"
    AllowedSourceEffects="Copy" >
    <StackPanel Orientation="Horizontal" FlowDirection="RightToLeft"
      Name="stpDeletePlace">
      <Image Height="19" Width="19" HorizontalAlignment="Left"
        Name="image2" Stretch="Fill" VerticalAlignment="Center"
        Source="/AplicacaoRicaSL;component/Images/cart_delete.png" />
      <sdk:Label Height="28" Name="label12" Padding="5,0,0,0"
        Content="Arraste aqui para excluir um item" />
    </StackPanel>
  </toolkit:PanelDragDropTarget>
  <toolkit:ListBoxDragDropTarget Margin="7,48,0,0"
    HorizontalAlignment="Left"
    AllowDrop="True" Height="457"
    VerticalAlignment="Top" Width="523">
    <ListBox x:Name="listaProdutos" ItemTemplate="{StaticResource ProdutosItemTemplate}"
      ItemsPanel="{StaticResource ProdutosItemsPanel}" Margin="0,0,0,0"
      VerticalAlignment="Top" HorizontalAlignment="Left"
      ScrollViewer.HorizontalScrollBarVisibility="Disabled">
    </ListBox>
  </toolkit:ListBoxDragDropTarget>
  <toolkit:PanelDragDropTarget AllowDrop="True" Drop="PanelDragDropTarget_Drop"
    AllowedSourceEffects="Copy" >
    <Grid>
      <Border BorderBrush="Silver" CornerRadius="15" BorderThickness="5"
        Background="WhiteSmoke" Height="457" HorizontalAlignment="Left"
        Margin="546,48,0,0" Name="brdAddItem" VerticalAlignment="Top"
        Width="501">
        <Grid HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
          AllowDrop="True" Drop="brdAddItem_Drop" >
        </Grid>
      </Border>
    </toolkit:PanelDragDropTarget>
  </Grid>
```

Figura 58 - Código fonte dos itens do pedido em *Silverlight* parte 1.

A primeira parte, exposta na Figura 58, é apresentada a parte do controle *ListBox* para dispor os produtos. Além disso, é possível observar a implementação dos controles de pesquisa de produtos através dos elementos *Image*, *TextBox* e *Button*. Já na Figura 59, a continuação do código representa a utilização do *DataGrid*.

```

<Grid HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
  AllowDrop="True" Drop="brdAddItem_Drop" >
  <StackPanel>
    <StackPanel AllowDrop="True" FlowDirection="RightToLeft"
      HorizontalAlignment="Center" VerticalAlignment="Top"
      Name="stackPanel1" Orientation="Horizontal">
      <Image Height="19" HorizontalAlignment="Left" Name="image4"
        Source="/AplicacaoRicaSL;component/Images/cart_add.png"
        Stretch="Fill" VerticalAlignment="Center" Width="19" />
      <sdk:Label Content="Arraste aqui para adicionar um item"
        Height="28" Name="label2" Padding="5,0,0,0" />
    </StackPanel>
    <toolkit:DataGridDragDropTarget
      Margin="5,5,5,5" Height="409"
      VerticalAlignment="Top" Drop="DataGridDragDropTarget_Drop"
      HorizontalAlignment="Left" Width="480">
      <sdk:DataGrid AutoGenerateColumns="False" Name="grdItensPed"
        ItemsSource="{Binding Mode=TwoWay, Path=itensPedido}"
        HorizontalAlignment="Left" Height="409" Width="480">
        <sdk:DataGrid.Columns>
          <sdk:DataGridTemplateColumn Width="300" Header="Produto">
            <sdk:DataGridTemplateColumn.CellTemplate[...]>
          </sdk:DataGridTemplateColumn>
          <sdk:DataGridTemplateColumn Width="80" Header="Qtde">
            <sdk:DataGridTemplateColumn.CellTemplate>
              <DataTemplate>
                <toolkit:NumericUpDown
                  Value="{Binding Path=QtdeSolicitada, Mode=TwoWay}"
                  Maximum="{Binding Path=Produto.QtdeEstoque}"/>
              </DataTemplate>
            </sdk:DataGridTemplateColumn.CellTemplate>
          </sdk:DataGridTemplateColumn>
          <sdk:DataGridTextColumn
            Binding="{Binding Path=ValorTotal}" Header="Total"
            Width="95"></sdk:DataGridTextColumn>
        </sdk:DataGrid.Columns>
      </sdk:DataGrid>
    </toolkit:DataGridDragDropTarget>
  </StackPanel>
</Grid>

```

Figura 59 - Código fonte dos itens do pedido em Silverlight parte 2.

Outra funcionalidade exposta nestes códigos é o uso de componentes para a realização de *drag-and-drop*. Os controles *ListBoxDragDropTarget* e o *DataGridDragDropTarget* podem ser observados para suprir a característica de adição de itens. Na Figura 60 está representada a execução da característica de tela dos itens do pedido desenvolvida em Silverlight.

Promoção
Arraste aqui para excluir um item

Despertador 1200	Livro 50	Celular 900
Tesoura 23.149999618531	Retrato 65	Bandeira 154
Filmadora 1066	Câmera 3000	Microfone 80
Relógio 664	Cachorro 2000	Computador 3450
Cadeira 400		

Arraste aqui para adicionar um item

Produto	Qtde	Total
Lupa	1	21,53
Celular	2	1.800,00

Figura 60 - Resultado da implementação de itens do pedido em Silverlight.

ANEXO G– Implementação da característica de itens do pedido em HTML5

Este anexo exibe a implementação da característica de itens do pedido implementada em HTML5. A Figura 61 mostra o código desenvolvido em HTML para contemplar os requisitos desta característica. Nesta é possível observar que tanto para representar a listagem de produtos quanto os itens adicionados no pedido está sendo utilizado o controle *table*.

```
<tr style="height: 100%; overflow: scroll; vertical-align: top;">
  <td>
    <div style="height: 380px; width: 530px; overflow: auto;">
      <table id="tableProdutos" style="border: 0px solid black;
        height: 100%; width: 100%;">
        <tr>
          <td>
            <div class="produto">...</div>
          </td>
          <td>
            <div class="produto">...</div>
          </td>
          <td>...</td>
        </tr>
      </table>
    </div>
  </td>
  <td colspan="2">
    <div id="AdicionarItensPlace" class="itensPlace"
      ondrop="dropProd(event)" ondragover="allowDrop(event)">
      <div style="font-size: medium; padding: 4px; width: 100%;
        text-align: center">
        Arraste aqui para adicionar ao pedido
      </div>
      <div class="gridItensPlace">
        <table id="tableItens" cellpadding="0" cellspacing="0"
          class="gridItens">
          <tr class="rowItens rowHeader">
            <td>...</td>
            <td>...</td>
            <td>...</td>
          </tr>
        </table>
      </div>
    </div>
  </td>
</tr>
```

Figura 61 - Código fonte dos itens do pedido em HTML.

Para popular as tabelas exibidas na Figura 61 houve a necessidade da implementação manual em *JavaScript*. A representação disso está relatada na Figura 62. Nesta está exemplificada a montagem da lista de produtos avaliando o filtro informado.

```

function MontaListagemProdutos(filtro) {
    var countAux = 1;
    var row;
    var table = document.getElementById("tableProdutos");
    table.innerHTML = "";
    for (p in produtos) {
        var produtoValido = true;
        if (filtro != "") {
            var nome = produtos[p].nome.toUpperCase();
            produtoValido = nome.indexOf(filtro.toUpperCase()) >= 0;
        }
        if (produtoValido) {
            var divProd = document.createElement('div');
            divProd.setAttribute('class', 'produto');
            divProd.setAttribute('draggable', 'true');
            divProd.setAttribute('ondragstart',
                "dragProdStart(event, " + produtos[p].codigo + ")");
            var img = document.createElement('img');
            img.setAttribute('class', 'imgProduto');
            img.setAttribute('src', produtos[p].urlImagem);
            var nome = document.createElement('label');
            nome.setAttribute('class', 'nomeProduto');
            nome.appendChild(document.createTextNode(produtos[p].nome));
            var preco = document.createElement('label');
            preco.setAttribute('class', 'valorProduto');
            preco.appendChild(document.createTextNode(produtos[p].valor));
            divProd.appendChild(img);
            divProd.appendChild(nome);
            divProd.appendChild(preco);
            if (countAux == 1) {
                row = document.createElement("tr");
                table.appendChild(row);
            }
            var col = document.createElement('td');
            col.appendChild(divProd);
            row.appendChild(col);
            if (countAux == 3)//nro de colunas
                countAux = 1;
            else
                countAux++;
        }
    }
}


```

Figura 62 - Código para montar a listagem de produtos em JavaScript.


A renderização obtida com as implementações relacionadas está exibida na Figura 63. Nesta, assim como nas outras tecnologias é possível observar o cumprimento dos requisitos definidos na característica de itens do pedido.

Filtro


Promoção Arraste aqui para excluir um item




Despertador
1200




Livro
50




Celular
900




Tesoura
23.15




Retrato
65




Bandeira
154




Filmadora
1066




Câmera
3000




Microfone
80




Relógio
664



Cachorro
2000



Computador
3450



Cadeira
400

Arraste aqui para adicionar ao pedido



Produto	Quantidade	Total
 Bola	1 <input type="text"/>	23
 Livro	2 <input type="text"/>	100

Figura 63 - Resultado da implementação de itens do pedido em HTML5.

ANEXO H – Implementação da característica de itens do pedido em *JavaFX*

O Anexo H demonstra a implementação da característica de itens do pedido desenvolvida com a tecnologia *JavaFX*. A Figura 64 exibe o código desenvolvido para representar as informações visualmente. Nesta observa-se a utilização do controle *TilePane* para listar os produtos e o *TableView* para exibir os itens adicionados ao pedido.

```
<GridPane fx:id="grpItensPedido" hgap="1" vgap="2"
style="-fx-padding: 4 0 0 0; -fx-wid">
  <children>
    <ScrollPane prefHeight="400" prefWidth="440" GridPane.columnIndex="0" GridPane.rowIndex="1">
      <content>
        <TilePane fx:id="produtosPlace" orientation="HORIZONTAL" prefWidth="430">
          <children>
            <children>
              <ImageView fitHeight="55.0" fitWidth="35.0" layoutX="6.0" layoutY="5.0">
                <image>
                </image>
              </ImageView>
              <Label id="label1" layoutX="45.0" layoutY="6.0" text="Lanterna">
                <font>
                </font>
              </Label>
              <Label id="label2" layoutX="45.0" layoutY="35.0" text="R$ 2.000,00" />
            </children>
            <Rectangle id="rectangle1" arcHeight="30.0" arcWidth="30.0" fill="TRANSPARENT"
height="60.0" width="120.0" stroke="BLACK" strokeType="INSIDE" strokeWidth="2.0"
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0" />
          </children>
          <padding>
          </padding>
        </TilePane>
      </content>
    </ScrollPane>
    <AnchorPane id="GridItensPlace" prefHeight="400" prefWidth="390" GridPane.columnIndex="1" GridPane.rowIndex="1" onI
<children>
  <Rectangle id="rectangle1" arcHeight="32.0" arcWidth="21.0" fill="#bfbfbf" height="400.0" stroke="BLACK" st
  <VBox id="vBox1" alignment="TOP_CENTER" prefHeight="400" prefWidth="400" AnchorPane.bottomAnchor="0.0" Anch
  <children>
    <HBox>
      <ScrollPane id="scrollPane1" prefHeight="350" prefWidth="380">
        <content>
          <TableView fx:id="tableItensPedido" prefHeight="345.0" prefWidth="378" />
        </content>
      </ScrollPane>
    </children>
  </children>
  <padding>
  </padding>
  </VBox>
</children>
</AnchorPane>
</children>
</GridPane>
```

Figura 64 - Código fonte dos itens do pedido em *JavaFX*.

A implementação em *JavaFX* também necessitou popular os dados de forma dinâmica. Na Figura 65 está exposto como foram montados os dados a serem exibidos na grade de itens adicionados ao pedido. É possível observar que cada coluna foi criada dinamicamente atribuindo as propriedades necessárias.

```

@FXML
TableView tableItensPedido;

private void MontaGridItens() {

    tableItensPedido.setEditable(true);

    TableColumn nomeProdCol = new TableColumn();
    nomeProdCol.setText("Produto");
    nomeProdCol.setMinWidth(220);
    nomeProdCol.setCellValueFactory(new PropertyValueFactory("produto"));
    nomeProdCol.setCellFactory(new Callback<TableColumn, TableCell>() { ... });

    Callback<TableColumn, TableCell> cellFactory =
        new Callback<TableColumn, TableCell>() {
            @Override
            public TableCell call(TableColumn p) {
                return new EditingCell();
            }
        };

    TableColumn qtdCol = new TableColumn();
    qtdCol.setText("Quantidade");
    qtdCol.setCellValueFactory(new PropertyValueFactory("qtdSolicitada"));
    qtdCol.setCellFactory(cellFactory);
    qtdCol.setOnEditCommit(new EventHandler<CellEditEvent>() { ... });

    TableColumn valorCol = new TableColumn();
    valorCol.setText("Valor");
    valorCol.setMinWidth(60);
    valorCol.setCellValueFactory(new PropertyValueFactory("valorTotal"));

    tableItensPedido.setItems(itensPedido);
    tableItensPedido.getColumns().addAll(nomeProdCol, qtdCol, valorCol);

    tableItensPedido.setRowFactory(this.createRowCallback());
}


```


Figura 65 - Código para montar a listagem de produtos em Java.


O resultado obtido nesta implementação pode ser visto na Figura 66. De forma semelhante as outras tecnologias, o *JavaFX* também supriu as especificações propostas. Esta figura exemplifica o uso do filtro de produtos e mostra os itens adicionados ao pedido.


Filtro


Promoção Arraste aqui para excluir um item


 **Despertador**
R\$ 1200,00


 **Livro**
R\$ 50,00


 **Celular**
R\$ 900,00


 **Tesoura**
R\$ 23,15


 **Retrato**
R\$ 65,00


 **Bandeira**
R\$ 154,00


 **Filmadora**
R\$ 1066,00


 **Câmera**
R\$ 3000,00

 **Microfone**
R\$ 80,00

 **Relógio**
R\$ 664,00

 **Cachorro**
R\$ 2000,00

 **Computador**
R\$ 3450,00

 **Cadeira**
R\$ 400,00

Arraste aqui para adicionar ao pedido




Produto	Quantidade	Valor
 Livro	3	150,00
 Filmadora	1	1066,00
 Bola	1	23,00

Figura 66 - Resultado da implementação de itens do pedido em *JavaFX*.