UNIVERSIDADE DO SUL DE SANTA CATARINA

CÉSAR AUGUSTO DA SILVA MOSER

JÚLIO CÉSAR DA SILVA MOSER

MULTIMÍDIA NA WEB: ANÁLISE DA FERRAMENTA DE DESENVOLVIMENTO ADOBE FLASH CATALYST

Florianópolis

2010

CÉSAR AUGUSTO DA SILVA MOSER
JÚLIO CÉSAR DA SILVA MOSER

MULTIMÍDIA NA WEB: ANÁLISE DA FERRAMENTA DE DESENVOLVIMENTO ADOBE FLASH CATALYST

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof.ª Vera Rejane N. Schuhmacher.

Co-orientador: Prof. Ricardo Villarroel Davalos.

Florianópolis

2010

CÉSAR AUGUSTO DA SILVA MOSER
JÚLIO CÉSAR DA SILVA MOSER

MULTIMÍDIA NA WEB: ANÁLISE DA FERRAMENTA DE DESENVOLVIMENTO ADOBE FLASH CATALYST

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Graduação em Ciência da Computação da Universidade do Sul de Santa Catarina.

_____ , ___ de _____ de 20___.

Local, dia mês ano

_____.

Prof.ª Vera Rejane N. Schuhmacher, Msc. Eng.

Universidade do Sul de Santa Catarina

_____.

Prof. Ricardo Villarroel Davalos, Dr. Eng.

Universidade do Sul de Santa Catarina

_____.

Prof.ª Maria Inés Castineira, Dra.

Universidade do Sul de Santa Catarina

Dedicamos este trabalho ao nosso pai, Dálcio Moser (*in memoriam*).

AGRADECIMENTOS

À nossa professora, Prof.ª Vera Rejane N. Schuhmacher, pela compreensão, incentivo e ajuda constantes.

À nosso co-orientador, Prof. Ricardo Villarroel Davalos, pelo apoio e ajuda constante.

Aos nossos irmãos Marco Aurélio e Rafael Alberto da Silva Moser, pelo companheirismo e aprendizado de vida.

Aos amigos e professores do Curso de Ciência da Computação pela convivência inspiradora ao longo dos anos.

E, especialmente, ao nosso pai, Dálcio Moser (*in memoriam*), e à nossa mãe, Evanilda da Silva Moser, por nos terem apoiado e amparado nos momentos mais difíceis.

RESUMO

O presente trabalho analisou e avaliou a ferramenta de desenvolvimento *Adobe Flash Catalyst*. Para a análise, e posterior avaliação da ferramenta, os autores deste trabalho criaram um exemplo de *Website*, utilizando-se da ferramenta do estudo. A modelagem do *Website* se fez necessária para o *design* do *software* empregado, através do modelo ICONIX e a linguagem UML. Na bibliografia, a tecnologia resultante da ferramenta estudada foi posta em contexto histórico, perante algumas das tecnologias utilizadas na criação de páginas *Web*. A problemática da criação de páginas *Web* por *Webdesigners* e desenvolvedores também foi posta em discussão, visto que a ferramenta alvo do estudo visa facilitar o desenvolvimento de um aplicativo com linguagem *Flex* e tecnologia *Flash*. Como alegada pela empresa criadora da ferramenta, a facilidade e rapidez de criação, sem a necessidade de codificação do aplicativo criado, e sem a necessidade de intervenção de um desenvolvedor no trabalho do *Webdesigner*, foi observada, discutida e confirmada por este trabalho. Como requisito para o entendimento dos objetivos levantados em seu objetivo principal, demonstrou-se passo a passo o processo de criação do aplicativo e *Website*, que foram usados como peças chave para a avaliação da ferramenta alvo do estudo.

Palavras-chave: *Adobe Flash Catalyst*. *Flex*. *Flash*. *Webdesigner*. Análise. *Website*. UML.

ABSTRACT

This study examined and evaluated the development tool Adobe Flash Catalyst. For the analysis, and subsequent evaluation of the tool, the authors of this paper created a sample website using the targeted tool of the study. The modeling of the website was needed to design the employed software, through the ICONIX model and the UML language. In its bibliography, the resulting technology of the studied tool was placed into historical context in front of some of the technologies used in the creation of Web pages. The issue of creating Web pages for Web designers and developers has also been called into question, since the aim of the studied tool is to facilitate the design and development of an application with Flex language and Flash technology. As alleged by the company which created the tool, the ease and speed of creation, without the need for coding of the application created, and without intervention of a developer in the work of the web designer, was observed, discussed and confirmed by this work. As a prerequisite to the understanding of the objectives raised in its primary goal, it was shown a step by step process of creation to the application and Website, which were used as key parts to the evaluation of the targeted tool of the study.

Keywords: Adobe Flash Catalyst. Flex. Flash. Webdesigner. Examination. Website. UML.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

AJAX - Asynchronous Javascript And XML

ASP - Active Server Pages

CMS - Content Management System

CSS - Cascading Style Sheets

HTML - HyperText Markup Language

jpeg - Joint Photographic Experts Group

LCD - Liquid Crystal Display

PC - Personal Computer

PHP - Hypertext Preprocessor

W3C - World Wide Web Consortium

SGML - Standard Generalized Markup Language

SMIL - Synchronized Multimedia Integration Language

SGC - Sistema de Gestão de Conteúdo

SWF - Shockwave Flash

UML - Unified Modeling Language

URL - Uniform Resource Locator

XML - eXtensible Markup Language

xHTML - eXtensible Hypertext Markup Language

SUMÁRIO

1 INTRODUÇÃO

O crescimento da internet, ao longo das últimas duas décadas, é consequência do surgimento da necessidade das pessoas obterem mais informação e conteúdo, através do uso de plataformas computacionais. O aparecimento de novas tecnologias facilitou a visualização do conteúdo presente nos meios virtuais, tornando a *Web* um ambiente mais interativo, com elementos mais coerentes e funcionais.

A *Web* é mutável, variável de acordo com a tecnologia empregada e a ferramenta utilizada em sua construção. O desenvolvimento de interfaces para a *Web* envolve uma ampla gama de profissionais e ferramentas dependendo da complexidade da interface a ser desenvolvida, sendo que a comunicação, entre os profissionais dentro da cadeia de produção, é um dos fatores mais importantes para satisfação de metas de produção definidas nas etapas iniciais de um projeto.

Com o passar do tempo, viu-se que esta cadeia apresentava falhas. A ideologia de designers gráficos era muito complexa aos olhos dos programadores, e a programação de linguagens específicas era muito complexa aos olhos do designer gráfico. A ampla gama de ferramentas e técnicas de produção dificultou ainda mais a capacidade desses profissionais de desenvolverem uma interface em conjunto, sem que ambos entrassem em domínio alheio.

Na última década, com o surgimento do conceito da *Web 2.0*, apesar da tecnologia fundamental da *Web* nunca ter mudado, sendo adotada por linguagens como o HTML, PHP, etc, os elementos visuais e a interação com os navegadores se tornou mais organizada do ponto de vista de informação e mais interativa do ponto de vista do usuário. Assim, a interação entre os usuários em interfaces propícias para a troca instantânea de informação tornou-se uma necessidade e algo que os desenvolvedores de páginas *Web* almejaram desde sempre.

A existência atual de aplicativos, como Google, Flickr, Gmail e Orkut, são consequência direta do surgimento de ferramentas e tecnologias de programação, que é compreendida por uma série de tecnologias em sua base, como exemplo, o CSS, XHTML, XML e *Javascript*.

Com a imensa variedade de aplicativos, a Internet enriqueceu ao ponto que cada aplicativo virtual se compara aos aplicativos nativos da plataforma PC, ao passo que a evolução das ferramentas de programação atuais convergem para a criação de novos aplicativos para o meio virtual.

As Interfaces *Web* atuais cresceram e se tornaram mais atraentes, graças ao surgimento de ferramentas que permitiram o seu desenvolvimento, citando o *Adobe Photoshop*, *Adobe Illustrator*, *Adobe Dreamweaver* e *Adobe Flash*.

O que se vê adiante para o meio profissional é o contínuo surgimento e atualização das ferramentas de desenvolvimento, que tornarão as interfaces da *Web* cada vez mais ricas em mídia e, também, mais fáceis de serem implementadas. O usuário final usufrui desse benefício diretamente, com um novo conceito de interação e páginas sendo implementadas a todo o tempo, vindo do termo “beta para sempre”.

Se comparado aos tempos antigos, em que era necessário grande conhecimento sobre programação e horas seguidas de trabalho para atingir objetivos factíveis, hoje, graças às novas ferramentas, tornou-se mais rápido e fácil proporcionar ao usuário uma experiência agradável de navegação.

Com o surgimento e amadurecimento da tecnologia *Flash*, ao longo dos últimos 10 anos, observa-se uma plataforma de interface rica sobresair-se frente as suas concorrentes. Com a adoção da comunidade de desenvolvimento, a plataforma vem tomando forma, não somente nos aplicativos para a *Web*, mas também em interfaces protótipo para apresentações de portfólio e até interface de jogos. Isto é possível graças à constante evolução de suas ferramentas de trabalho, sendo o *Adobe Catalyst* o alvo deste estudo.

1.1 PROBLEMA

O risco de perder o Projeto Gráfico, no momento da implementação, é determinante para a finalização ou não do mesmo.

A falta de conhecimento do programador sobre regras de *Web design* e a falta de conhecimento do *Web designer* sobre programação afetam o resultado final. Há dificuldade de utilização das ferramentas para implementação dos trabalhos e testabilidade da interface por parte do *Web designer*. Novas ferramentas surgem para apoiar o desenvolvimento de sites, dentre elas, o Adobe Flash Catalyst.

Observa-se que a ferramenta *Adobe Flash Catalyst* apresenta pouquíssimo material de referência disponível, que permita corroborar ou mesmo desabonar o que é hoje apresentado em sua divulgação. Por isso, pretende-se analisar e avaliar esta ferramenta.

Sua principal característica, citada, é a facilidade para o profissional exercer sua atividade de edição gráfica e posterior conversão em código, sem a necessidade de conhecimento prévio sobre a linguagem de programação *Flex*, empregada no aplicativo.

Portanto, o profissional que utiliza a ferramenta, não necessita de conhecimentos de programação para executar a tarefa da criação do aplicativo, sendo esta tarefa opcional e terceirizada para outra ferramenta.

O código gerado automaticamente pela ferramenta visa a facilitar o processo de criação de interfaces, porém não permite edição do código na mesma ferramenta.

Não existem estudos atuais que determinem o impacto dessa nova ferramenta de desenvolvimento na cadeia de produção de Interfaces e páginas *Web*, nem no mercado de profissionais da área.

1.2 OBJETIVOS

Este projeto tem por objetivo analisar e avaliar a ferramenta de desenvolvimento Adobe Flash Catalyst a partir da criação de um Website funcional com sua tecnologia.

1.2.1 OBJETIVOS ESPECÍFICOS

São objetivos específicos deste projeto:

- estudar a ferramenta *Adobe Flash Catalyst* e sua capacidade de cumprir o que sugere;
- desenvolver um protótipo de interface multimídia *Web*, utilizando a ferramenta *Adobe Flash Catalyst*;
- realizar uma análise qualitativa de desempenho da ferramenta *Adobe Flash Catalyst*.
- Utilizar um Sistema de Gerenciamento de Conteúdo para abrigar o aplicativo criado com a ferramenta objeto do estudo.
- Desenvolver o aplicativo a partir de um protótipo gráfico.
- Gerar material de referência para futuros profissionais da área.

1.3 JUSTIFICATIVA

Este trabalho aborda a ferramenta de desenvolvimento *Adobe Flash Catalyst*, que, no presente momento, encontra-se na versão 1.0 de desenvolvimento. Segundo a Adobe Systems (2010), a ferramenta clama por executar o trabalho de desenvolvimento de uma interface *Web*

sem que o *Web Designer* precise efetivamente entrar em contato com código de programação. Esse fato ajuda a discriminar ainda mais a função de cada profissional na cadeia de produção de uma interface multimídia, visto que não há necessidade de comunicação direta entre as partes, *Web Designer* e *Web Developer*, durante as etapas iniciais do desenvolvimento de uma interface. Isso permite ao *Designer* experimentar e prototipar interfaces sem interferência inicial de um programador.

Portanto, este trabalho pretende estudar a ferramenta, promovendo uma discussão sobre sua facilidade, ou não, de criação. E, também, trazer ao mercado um material de referência que dê apoio aos desenvolvedores que pretendam fazer uso da ferramenta, visto que o material sobre a mesma ainda é escasso.

Para atingir os objetivos propostos pretende-se estudar a ferramenta em suas funções básicas, desenvolvendo um exemplo de interface *Web*, ou página *Web*, a fim de comprovar sua eficácia. Para isso pretende-se utilizar ferramentas gráficas da família Adobe (Photoshop e Illustrator), compatíveis com a ferramenta objeto do estudo.

1.4 ESTRUTURA

Capítulo 1: Introdução – Neste capítulo, é apresentada a Introdução, que aborda a problemática, o Objetivo principal e específicos, a Justificatíva e a Estrutura da monografia.

Capítulo 2: Revisão Bibliográfica – Este capítulo, aborda os diversos tópicos e material consultados de bibliográfia sob a temática de *Web 2.0*, Animação para *Web*, As tecnologias da *Web*, Profissionais da Área e a tecnologia *Flash*. Tem-se como objetivo, nesse capítulo, contextualizar o uso da ferramenta, foco deste estudo.

Capítulo 3: Metodologia do trabalho – Neste capítulo, é apresentada a metodologia empregada na pesquisa da ferramenta de estudo. Nele, também, destacam-se as etapas da pesquisa, com a escolha do tema, temática do protótipo, análise da ferramenta, etc.

Capítulo 4: Apresentação da Ferramenta – Neste capítulo, é apresentada a interface da ferramenta de desenvolvimento, o *Adobe Flash Catalyst*. Nele, demonstra-se a interface de trabalho e as principais características e funções da ferramenta, destacando seu potencial para os profissionais da área.

Capítulo 5: Modelagem – Neste capítulo, apresenta-se a modelagem, utilizando a metodologia ICONIX adaptada para apresentação do protótipo. Também, destaca-se o UML, os Requisitos Funcionais, Não Funcionais, Atores, Casos de uso e Protótipo da interface *Web* a ser desenvolvida.

Capítulo 6: Desenvolvimento do Aplicativo e Website – Este capítulo, aborda o desenvolvimento do Aplicativo a partir da ferramenta de estudo e a criação do exemplo de *Website*. Nesse, descreve-se o uso da tecnologia *Flash* gerada pela ferramenta e são apresentadas as conclusões acerca das hipóteses levantadas no capítulo 3 deste trabalho.

Capítulo 7: Conclusões – Conclusões do trabalho e considerações finais sobre a ferramenta.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, é apresentada a revisão bibliográfica sobre as áreas de domínio da monografia e compreende os tópicos sobre "*Web 2.0* – Aplicativo e Plataforma", as Tecnologias da *Web*, que aborda a tecnologia por trás da ferramenta de estudo, o *Adobe Flash Catalyst*. Também, é apresentada a evolução da *Web*, com o HTML5, que dá argumento ao tema de animação para *Web*, mesmo campo de atuação do *Adobe Flash*, a animação na *Web*, o principal propósito da ferramenta estudada. Também, compreende o tópico sobre os profissionais da área de *Web Design* e *Web Development*.

2.1 WEB 2.0 – APLICATIVO E PLATAFORMA

Com o crescimento da Internet, ao longo dos anos 90, veio, também, o desenvolvimento de tecnologias que facilitasse a sua navegação e assimilação de informação, que se faz basicamente por meio de Navegadores *Web*, ou *Web Browsers*. O primeiro *Web Browser* a ser criado, por Tim Berners-Lee, data de 1990 e se chamava *WorldWideWeb*, segundo Berners-Lee (2010):

> O primeiro navegador web – ou browser-editor - foi chamado *WorldWideWeb*, afinal, quando ele foi escrito em 1990, era a única maneira de ver a web. Mais tarde, ela foi renomeada Nexus, a fim de evitar confusão entre o programa e o espaço abstrato de informações (que agora é soletrado como World Wide Web ,com espaços).

Os *Web Browsers*, ou Navegadores *Web*, ditaram a tecnologia e o desenvolvimento de aplicativos em suas plataformas e seu mercado foi controlado posteriormente por Microsoft e Netspace. Os desenvolvedores viam os navegadores como oportunidade para crescimento da indústria e, também, como meio para o crescimento de novas mídias e ferramentas de desenvolvimento específicas. Segundo O'Reilly (2005):

> [...]Netscape enquadrou 'a web como plataforma' nos termos do velho paradigma de software: o seu principal produto foi o navegador web, uma aplicação desktop, e sua estratégia era usar seu domínio no mercado de navegadores para criar um mercado para produtos de alto custo para servidores.

A fase denominada "Web 1.0", que no começo ainda nem possuia tal denominação, surgiu após o crescimento da Internet em meados da última década, quando o conceito da "Web 2.0" começou a tomar forma, após ser proposto por Tim O'Reilly, em um *brainstorming* com a *MediaLive International*.

Outros autores também previram a mudança da *Web*, como Darcy DiNucci, primeiro mensionando a “Web 1.0”:

> A Web como a conhecemos hoje é uma coisa fugaz: Web 1.0. Sua relação com a Web que estaremos familiarizados nos próximos anos é aproximada do Pong em relação ao The Matrix. A Web como a conhecemos é essencialmente um protótipo, uma prova de conceito. (DINUCCI, 1999).

E, posteriormente, previndo o surgimento de novo conceito:

> Os primeiros vislumbres da Web 2.0 estão agora começando a aparecer, e podemos começar a ver como que o embrião possa se desenvolver. As primeiras fases da mitose começaram, e as células do organismo começam a se diferenciar. (DINUCCI, 1999).

O conceito por trás de ambas as definições da *Web* é abstrato, sendo difícil defini-las sem citar tecnologias e a evolução das páginas *Web* e sua interação com o usuário.

Para Pisani e Piotet (2008, p. 16),

> [...]é necessário sublinhar a diferença entre a internet e a web.[...]A internet é a rede de informática mundial que nos permite acessar correios eletrônicos ou websites[...]. A web, ou World Wide Web, é uma das maiores aplicações permitidas pela internet. É um sistema que possibilita consultar, por meio de um navegador, páginas contidas em sites.

Como dito por O'Reilly (2004), “A web é uma plataforma”, e ela se define da seguinte forma:

> De início por causa da migração de padrões proprietários para padrões abertos. A internet é hoje uma plataforma global, que repousa sobre padrões estabelecidos, abertos e compartilhados.[...] Passamos de uma primeira geração de sites estáticos (o conteúdo só muda pela intervenção de seu administrador, quer dizer, muito raramente), para uma segunda geração dinâmica (mais rápida e mais rica, o conteúdo muda também de forma automática, interagindo com os algoritmos e os usuários).(PISANI, PIOTET, 2008).

Portanto, “Web 1.0” caracteriza esses “sites estáticos”, de “caráter proprietário fechado”, sem interação com o usuário, ou seja, tudo que o conceito de “Web 2.0” não emprega. A “Web 2.0”, por sua vez, não possui uma cronologia específica, ou seja, não possui uma data inicial imposta para funcionamento, seu conceito apenas evoluiu há seis anos atrás, quando O’Reilly iniciou as discussões acerca da mudança de conceito da *Web*:

> O Conceito de “Web 2.0” começou com uma sessão de ‘brainstorm’ durante uma conferência entre O’Reilly e a MediaLive International. Dale Dougherty, pioneiro da web e O’Reilly VP, comentou que longe de ter ‘quebrado’, a Web era mais importante do que sempre foi, com excitantes novas aplicações e sites surgindo com uma regularidade assustadora. O que há a mais, as companhias que sobreviveram o colapço pareciam ter algo em comum. Poderia ser que o colapço da ‘ponto-com’ marcou um ponto de divergência para a Web, sendo que uma chamada de ‘Web 2.0’ possa fazer

> sentido? Nós concordamos que isto aconteceu, e então a Conferência da Web 2.0 nasceu. (O'Reilly ,2005)

Segundo O'Reilly (2005), "Você pode visualizar a Web 2.0 como um conjunto de princípios e práticas que unem um autêntico sistema solar de sites que demonstram alguns ou todos aqueles princípios, variando a distância daquele núcleo."

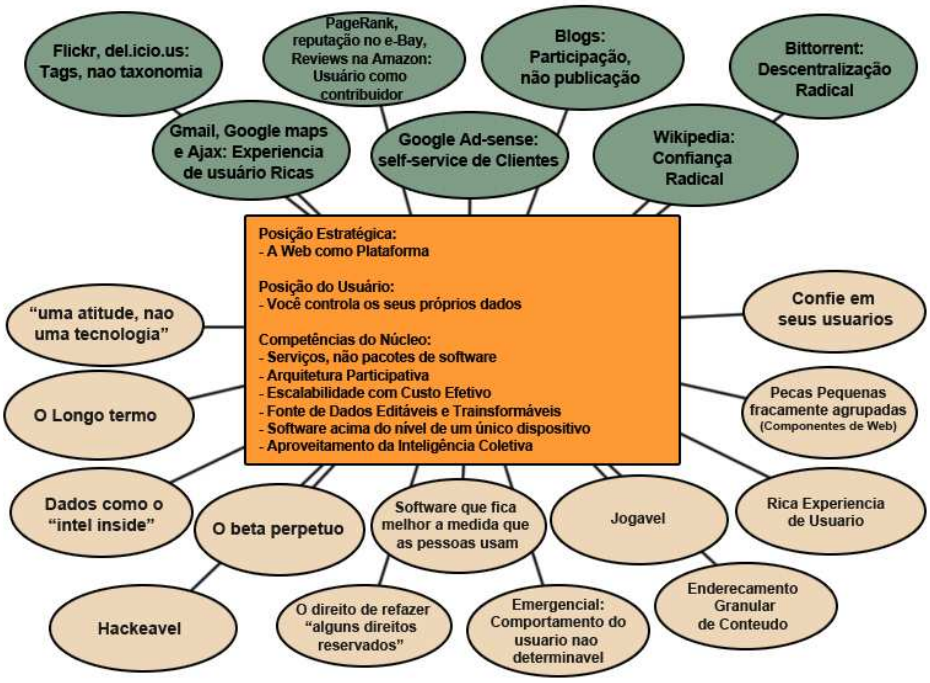


Figura 1 - Web 2.0 Meme Map.
Fonte: Tim O'Reilly (2005).

Segundo O'Reilly (2005), "A Figura 1 apresenta um 'mapa meme' da *Web 2.0* que foi desenvolvida durante uma sessão de '*brainstorm*' durante o *FOO Camp*, uma conferência na *O'Reilly Media*. A figura mostra as muitas idéias que irradiam de dentro do núcleo da *Web* 2.0".

Para Pisani e Piotet (2008, p. 17), " Os usuários passaram do status de viajantes na internet (internautas) para o status de atores da *web*,[...] propondo serviços e conteúdos próprios, comentando ou discutindo as informações disponíveis."

Portanto, é certo dizer que a *Web 2.0* é uma consequência da mudança e evolução dos conceitos de interação humana no meio computacional. Também, é certo dizer que essa mudança aconteceu graças a empresas engajadas no meio, destacando Netscape e Google, como os principais atores em seu desenvolvimento. O grande papel da Netscape, na época

regida pela *Web 1.0*, era o de criar uma fatia de mercado capaz de rivalizar com a alternativa "Desktop PC", segundo O'Reilly (2005):

> Se o Netscape foi o porta-estandarte para a Web 1.0, Google é certamente o porta-estandarte para a Web 2.0, se apenas porque seus respectivos IPOs foram eventos definitivos para cada Era. Então vamos começar uma comparação entre essas duas empresas e seu posicionamento.
> Netscape enquadrou "a web como plataforma" nos termos do antigo paradigma de software. Seu principal produto foi o Navegador Web, uma aplicação desktop, e sua estratégia era usar seu domínio no mercado de navegadores para criar um mercado para produtos de alto custo para servidores. O Controle sobre os padrões para a exibição de conteúdo e aplicações no Navegador seria, em teoria, dar ao Netscape o tipo de poder de mercado desfrutado pela Microsoft no mercado de PCs. Bem como a "carruagem sem cavalo" enquadrou o automóvel como uma extensão das famílias, a Netscape promoveu um "webtop" "para substituir o desktop, e planejou povoar o 'webtop' com atualizações de informações e 'applets' empurrado para o 'webtop' por provedores de informação que iriam comprar servidores Netscape.

No quadro a seguir, destacam-se os principais exemplos de aplicativos de cada conceito de *Web*. À esquerda, estão os principais aplicativos que marcaram a era da *Web 1.0* e à direita os principais aplicativos que ditaram a *Web 2.0*, proposto por Tim O'Reilly.

Web 1.0		Web 2.0
DoubleClick	-->	Google AdSense
Ofoto	-->	Flickr
Akamai	-->	BitTorrent
mp3.com	-->	Napster
Britannica Online	-->	Wikipedia
personal websites	-->	blogging
evite	-->	upcoming.org and EVDB
domain name speculation	-->	search engine optimization
page views	-->	cost per click
screen scraping	-->	web services
publishing	-->	participation
content management systems	-->	wikis
directories (taxonomy)	-->	tagging ("folksonomy")
stickiness	-->	syndication

Quadro 1 - Transição entre Aplicativos Web.
Fonte: Tim O'Reilly (2005).

O que, no entanto, não pode ser levado ao pé da letra, visto que aplicativos ou serviços *Web* como o Akamai e DoubleClick, como dito por O'Reilly (2005), "foram precursores do conceito de *Web 2.0*, ou seja, foram pioneiros que marcaram a transição entre um conceito antigo para um conceito novo, tratando a *Web* como plataforma".

Segundo O'Reilly (2005):

> Dois de nossos exemplos de Web 1.0, DoubleClick e Akamai, foram ambos pioneiros em tartar a Web como uma plataforma. As pessoas não costumam pensar nisso como "serviços da web", mas na verdade, a veiculação de

> anúncios foi o primeiro “serviço web” a ser amplamente utilizado.[...] Ambos DoubleCkick e Akamai foram pioneiros da Web 2.0, mas também podemos ver como é possível perceber mais as possibilidades adicionais adotando padrões da Web 2.0.

Em meados dos anos 90, os profissionais da área buscavam novas formas de exibir conteúdo através de imagens, muitas delas no formato padrão ‘jpeg’, principal formato para exibição de imagens atualmente.

Como instruído por Clark (2007), no site oficial do padrão “Jpeg”:

> Normalmente, os usuários da Internet são impedidos de descarregar grandes imagens de alta qualidade, devido ao tamanho físico do arquivo. Muitas vezes, os fornecedores de imagens deve criar três ou mais versões de uma imagem, variando de uma miniatura pequena até uma imagem de tamanho da página.

Do ponto de vista de visualização, essa técnica é utilizada desde os primórdios da *Web* para amostragem de conteúdo em formato de imagem. Mídias, como o Vídeo, no final da década de 90, podiam ser visualizados somente após longos *downloads* e, posteriormente, visualizados na própria máquina do usuário, graças à inexistência da “Banda Larga” e do conceito da “Web 2.0”. Nos dias atuais, esse processo foi amplamente substituído pela plataforma anexa de vídeo, no próprio Navegador *Web*, através de aplicativos.

Como dito por Pisani e Piotet (2008, p. 80):

> Contrariamente ao modelo tradicional, que impõe dispor no seu computador do programa necessário para o funcionamento de um aplicativo,[...] uma plêiade cada vez maior de programas está disponível na internet, não sendo necessário o download.[...] Eles são frequentemente gratuitos e ricos em aplicações que permitem aos usuários colaborar on-line, compartilhando os seus arquivos e enriquecendo-os.

Pode-se dizer que o surgimento da Banda Larga regeu e incentivou a mudança da *web*.

Como dito por Pisani e Piotet (2008, p. 27), sobre a “Banda Larga”:

> Os “grandes troncos” pelos quais transitam textos, imagens, música e vídeo atraem cada vez mais os usuários. O essencial é que permitam que os usuários estejam sempre conectados (always on). As redes móveis estão em via de acrescentar uma dimensão ao fenômeno.

E sobre as “Contribuições” à *web*:

> A banda larga incentiva as contribuições e facilita as modificações da plataforma.[...] as contribuições se adicionam, a ponto de criar um conjunto que é maior do que a soma de suas partes. Sociedades e tecnologias exploram o conteúdo gerado pelos usuários para desenvolver novos tipos de negócios. (PISANI e PIOTET, 2008, p. 27).

O surgimento da “longa rede”, proposto inicialmente por Tim O’Reilly e comentado por Pisani e Piotet (2008) “A web dá lugar a novas oportunidades de criação de valores, notadamente sobre os mercados de nichos, abrindo caminho a uma economia da diversidade e da abundância”.

Outro conceito que começou a surgir a partir do momento em que a “Web 2.0” tomou forma foi o fenômeno “mashup”, conceito inicialmente levantado por Tim O’Reilly.

Segundo Pisani e Piotet (2008, p. 93):

> Devemos o primeiro mashup ao programador de software Paul Rademacher pelo serviço do HousingMaps.com, [...] Frustrado por não conseguir associar um mapa aos anúncios que o interessavam, ele devenvolveu um aplicativo que mescla (mash up em inglês) as listas da Craigslist e o aplicativo de localização GoogleMaps automaticamente.
> Esse aplicativo, nascido da necessidade de um usuário, demonstra bem o potencial das tecnologias quando os programadores podem apropriar-se dos serviços e dos dados on-line e mesclá-los para atender a novas aplicações. Isso ilustra o fenômeno do loosely coupled (“associado segundo links folgados”). Dois ou mais aplicativos associados de forma muito aberta encontram novo valor.

Desde os primórdios da internet, no entanto, os usuários são dependentes dos Navegadores *Web*, repositórios de aplicativos *web*. Eles por si só são interfaces específicas para visualização de conteúdo dotadas de uma linguagem de programação específica, o *HyperText Markup Language* (HTML), visto adiante neste trabalho. Por trás do HTML, há outras linguagens e *Plug-ins* que permitem a visualização de vídeos, animações, até interações em tempo real via texto, imagem e vídeo, graças aos aplicativos em *Asynchronous Javascript And XML* (Ajax) e JAVA. O que leva ao encontro das tecnologias da *Web*.

A partir da falta de regulamentação do meio de comunicação, em meados da última década, começaram a surgir tecnologias e ferramentas capazes de suprir as deficiências da *Web*, do HTML e, também, do meio profissional, que viu surgir uma gama enorme de ferramentas capazes de produzir conteúdo para o meio.

Ao passo que a padronização da *Web* ocorreu, devida ao *World Wide Web Consortium* (W3C), órgão responsável pelos padrões do HTML e páginas *Web*. Foi reconhecida diretamente por empresas como Microsoft, Apple, entre outras. O HTML começou assim a ter seu código aperfeiçoado rumo aos padrões atuais e, também, ao anexo de mídias mais ricas, que facilitaram sua interação e entendimento.

2.2 TECNOLOGIAS EMPREGADAS NA WEB

Muitas são as tecnologias que surgiram com o desenvolvimento da Internet. Muitas delas tiveram seu uso ditado por grandes corporações, como a extinta Netscape e, atualmente, a Microsoft. Tim Berners-Lee foi o precursor da linguagem de programação HTML, alicerce fundamental para a *Web*, que pode ser acessado pela rede da Internet, via protocolo *Hypertext Transfer Protocol* (HTTP).

A Internet, que teve sua origem em campo militar, foi utilizada para a implementação do protocolo e, posteriormente, sua utilização por desenvolvedores, que, em sua grande maioria, utilizam o HTML como linguagem de programação principal para a amostragem de páginas *Web*.

Segundo Berners-Lee (2007):

> Do ponto de vista técnico, a Web é uma grande coleção de páginas Web (escritas em formato HTML padrão), ligada a outras páginas (com os documentos *linkados* com o chamado padrão URL), e acessado através da Internet (utilizando o protocolo de rede HTTP). Em termos simples, a Web tem crescido porque é fácil escrever uma página Web e de fácil ligação com outras páginas.

Desde sua concepção, o conceito fundamental permanece o mesmo, ou seja, utilizamos uma ferramenta de navegação que se conecta ao protocolo HTTP e, consequentemente, à plataforma *Web*. O que vemos adiante é o resultado de toda a evolução que tivemos ao longo dos últimos anos; páginas interativas, acesso à mídia, fotos, vídeos, textos, e blogs pessoais.

Tudo que acessamos, atualmente, de fato, é comandado por linguagens como o HTML, o *eXtensible Markup Language* (XML), *Standard Generalized Markup Language* (SGML), o *script Hypertext Preprocessor* (PHP), linguagens JAVA, Ajax, *Cascading Style Sheets* (CSS), a tecnologia *Flash*, entre outras linguagens específicas. Todas permitem uma variedade de opções para desenvolvedores contruírem seus aplicativos para a *Web* e, o mais importante, que os usuários interajam com as tecnologias, através de links de conteúdos, textos, imagens e vídeos.

Segundo Berners-Lee (2007):

> O que torna fácil criar links de uma página para outra é que não há limite para o número de páginas ou o número de ligações possíveis na web. Adicionando uma página Web não requer coordenação com qualquer autoridade central, e tem um baixo, muitas vezes zero, custo adicional.
>
> Além do mais, o protocolo que nos permite seguir estas ligações (HTTP) é um protocolo não-discriminatório. Permite-nos seguir qualquer ligação,

> independentemente do conteúdo ou propriedade. Então, porque é tão fácil de escrever uma página Web, link para outra página, e seguir estes links, as pessoas têm feito muito disto. Adicionando uma página fornece conteúdo, mas acrescentando um link fornece a organização, estrutura e apoio à informação na Web que transformam o conteúdo como um todo em algo de grande valor.

As principais tecnologias, em foco atualmente, são firmadas pelo mercado de profissionais e, mais importante, pelo público em geral, que tornou os aplicativos como Flickr, Facebook, Google, Twitter, *blogs* e tantos outros, como o padrão de Aplicativos *Web* a se seguir, através do uso massivo de *Links* e informação textual, como dito por Berners-Lee (2007).

Como dito por Pisani e Piotet (2008, p. 24):

> As ferramentas de criação de blogs, de compartilhamento de fotos, de mensagens instantâneas, de telefonia levam um número espantosamente elevado de usuários a se tornar web atores(sic), porque são mais simples, mais acessíveis, mais claras. Conectados em rede, permitem criar ligações, estabelecer relações quer entre dados, quer entre pessoas, ou entre pessoas e dados.

Nesses aplicativos, quem fornece os dados e contribui com informação é o ator alvo das companhias, o usuário da *web*. É possível que o futuro da tecnologia da *Web* se derive desses aplicativos, mas, o mais importante, qual o seu futuro como ferramenta e em quais plataformas elas estarão disponíveis.

Segundo Berners-Lee (2007):

> No futuro, a Web vai parecer que está em todo lugar, não apenas no nosso desktop ou dispositivo móvel. Enquanto a tecnologia LCD se torna mais barata, as paredes dos quartos, e até mesmo paredes dos edifícios, serão superfícies mostruárias de informações da web. Grande parte das informações que recebemos hoje através de uma aplicação especializada como um banco de dados ou uma planilha virá diretamente da web.

Portanto, a *Web*, como plataforma, tornou-se realidade nos últimos anos e deve continuar sua evolução a partir de novas ferramentas, novos aplicativos e, também, novos meios de comunicação.

O horizonte do HTML e o *eXtensible Hypertext Markup Language* (xHTML), no entanto, não nos guardam mais surpresas. Com a vinda do HTML5, uma evolução da linguagem HTML deve trazer mais dinamismo aos aplicativos do ponto de vista de estrutura e multimídia para aplicativos, tornando o HTML uma linguagem mais completa e pronta para receber o código de desenvolvedores, que tornarão as páginas *Web* mais interativas através de

um padrão universal, comum a todos os navegadores e homologado pelas grandes empresas do meio, membros do W3C.

Atualmente, grande parte do trabalho de interação no ambiente *Web* é executado por *softwares* terceirizados, ou seja, *Plug-ins* e aplicativos são anexados ao código HTML, que os executa e permite a interação de usuários com páginas *Web* interativas. Tal processo pode ser modificado, em breve, com a evolução do HTML.

2.2.1 A evolução do HTML

O HTML 5 vem para mudar a estrutura do código HTML e para dar nova semântica para os atuais “divs” utilizados na estrutura do HTML4, atual versão do HTML na maioria das páginas atuais. Suas mudanças permitirão maior flexibilidade na construção estrutural das páginas, e tornarão possíveis a implementação de elementos multimídia, antes executados exclusivamente por programas externos.

Segundo Hunt (2007):

> HTML 5 introduz um conjunto de novos elementos que tornam muito mais fácil estruturar páginas. A maioria das páginas HTML 4 inclui uma variedade de estruturas comuns, tais como cabeçalhos, rodapés e colunas e, hoje, é bastante comum marcá-los usando elementos div, dando a cada uma id ou classe descritiva.

Figura 2 - Estrutura do HTML4.
Fonte: Lachlan Hunt (2007).

A figura 2, acima, demonstra a atual estrutura utilizada no HTML4. Todas as sessões de uma página *Web* são divididas em “divs” específicas. Cada um dos elementos especifica seu conteúdo, “header” para o cabeçalho da página Web, “nav” para o menu de navegação, “sidebar” para os elementos secundários da página e, assim, por diante. Dentro delas é

inserido o código necessário para executar determinado aplicativo ou simplesmente adicionar informação textual e *links*.

A semântica atual do HTML4 é insuficiente para descrever o conteúdo de sua classe, ou seja, para o desenvolvedor a mudança para nomes específicos torna o código mais claro para a programação de cada conteúdo e, consequentemente, melhor visualização e navegabilidade por parte do usuário final.

Segundo Hunt (2007):

> A utilização de elementos div é grande, parte porque as versões atuais do HTML 4 carecem da semântica necessária para descrever essas peças mais especificamente. HTML 5 aborda esta questão através da introdução de novos elementos para representar cada uma dessas diferentes sessões.

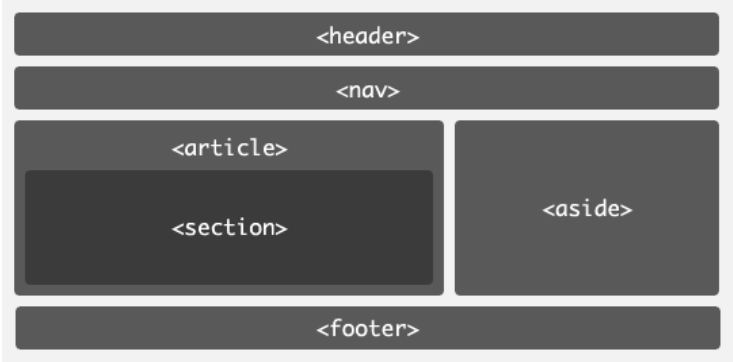


Figura 3 - Estrutura do HTML5.
Fonte: Lachlan Hunt (2007).

A figura 3 nos mostra a nova estrutura a ser adotada no HTML5. As divisões não são mais nomeadas como “divs”, e cada uma delas recebe seu nome específico e significado, tornando o código melhor estruturado visualmente para os programadores.

```
<body>
  <header>...</header>
  <nav>...</nav>
  <article>
    <section>
      ...
    </section>
  </article>
  <aside>...</aside>
  <footer>...</footer>
</body>
```

Quadro 2 - Código estrutural HTML5.
Fonte: Lachlan Hunt (2007).

Segundo Hunt (2007):

> Ao identificar o propósito das seções na página utilizando elementos de sessão específicos, a tecnologia assistida pode ajudar o usuário a navegar

> mais facilmente na página. Por exemplo, eles podem saltar facilmente a sessão de navegação ou pular rapidamente de um artigo para o outro sem a necessidade dos autores fornecerem links de pulo. Os autores também se beneficiam, porque substituindo muitas das divs no documento com um dos vários elementos distintos podem ajudar a tornar o código fonte mais claro e fácil para o autor.

Portanto, ao eliminar os elementos “div” do código estrutural do HTML, teremos páginas mais claras e de fácil manutenção por parte dos autores do código. Também, teremos páginas *Web* mais intuitivas e fáceis de serem navegadas por parte do usuário, como consequência da melhor implementação de elementos visuais.

O código HTML5 deve, com a mudança estrutural, facilitar o acesso de mídias diretamente no código. A meta dos desenvolvedores seria a de colocar elementos como controle de Som, Vídeo e Progressão diretamente no Navegador *Web*, de forma que o usuário não necessite de programas externos para acessar a mesma funcionalidade. Veremos o surgimento de gráficos animados, vídeos e música sendo controlados diretamente pelo código HTML e suportados por *codecs* proprietários como o padrão de compressão de vídeo H.264, nativos nos Navegadores *Web*.

Outro aspecto positivo seria a de tornar os Navegadores *Web* mais amigáveis às mídias digitais, de forma que controles aderidos ao *browser* tornariam a experiência multimídia mais rápida e sem necessidade de programas externos.

2.2.2 A Tecnologia Adobe Flash

Atualmente, a tecnologia *Flash* monopoliza a estrutura multimídia da maioria das páginas *Web*, com suas próprias funcionalidades, customizadas pelo autor e compatível com a maioria das plataformas utilizadas na atualidade. Seu uso mais frequente é a interação animada com as páginas *Web*. Originada da fusão da antiga empresa Macromedia, com a Adobe, hoje a ferramenta monopoliza o meio profissional que visa às animações de páginas *web*.

Segundo Jonathan Gay, profissional da extinta Macromedia:

> Em 2001, o Flash passou por cinco versões na Macromedia, e ainda tem muito do código que foi escrito para os computadores à caneta. Há agora 50 pessoas construindo o Flash em vez de 3, quando começamos a FutureWave. Ela evoluiu de um simples pacote de desenho e animação para Web em um ambiente completo de desenvolvimento de multimídia, com 500 mil colaboradores e mais de 325 milhões de usuários do Flash Player. Flash se tornou sinônimo de animação na Internet.

Em *websites*, como o YouTube e Vimeo, atualmente, o *Flash* é a principal tecnologia adotada para amostragem de vídeos interativos, isso graças ao comportamento dos desenvolvedores.

Segundo Hunt (2007), “Como evidenciado pelos vários mídia players baseados em Flash, os autores estão interessados em fornecer as suas interfaces de usuário em design personalizado, que geralmente permitem aos usuários reproduzir, pausar, parar, procurar e ajustar o volume.”

Em termos de popularidade, o *Adobe Flash Player* domina o mercado da *web*, segundo pesquisa feita pela Millward Brown, em março de 2010.

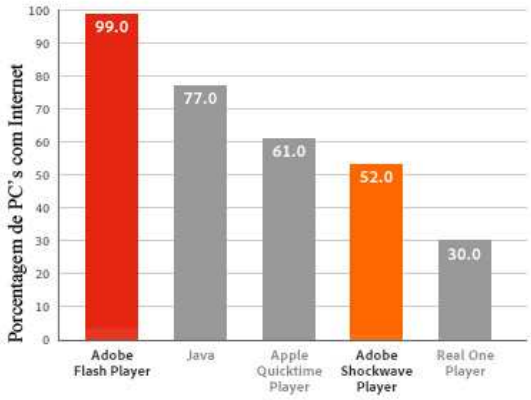


Figura 4 - Alcance da tecnologia Flash.
Fonte: Adobe.com (2010).

A exigência dessa tecnologia, no entanto, é a instalação do *Plug-in* necessário para visualização da mídia no formato “.swf ”, o *Adobe Flash Player*, um *software* externo ao Navegador *Web*, que permite que o código seja executado, e a página *web* apreciada pelo usuário, como explicado pela Wikipedia (2010), “Os arquivos Flash são em formato SWF, tradicionalmente chamada "Shockwave Flash" e pode ser usado sob a forma de uma página Web plug-in , estritamente "jogado" em um Flash Player”.

A tecnologia *Flash* teve início em 1996. Nessa época, os Navegadores *Web* careciam de elementos animados, vídeos e música, que fossem comum e compatíveis entre diversos Navegadores e plataformas. Na época, o meio também carecia de uma padronização, o HTML seguia ainda sem regulamentação sobre o seu conteúdo e o *Flash* tirava proveito desta situação.

Como dito pela empresa Adobe (2010) “O Adobe Flash Player é um runtime de aplicativo baseado em navegador entre plataformas que oferece uma exibição isenta de aplicativos expressivos, conteúdo e vídeos entre telas e navegadores”

Atualmente, a tecnologia *Flash* pode ser executada em diversos tipos de aparatos tecnológicos, desde *Personal Computers* (PC's) *desktop*, Apple Macintosh, *Notebooks*, dispositivos celulares, Liquid Crystal Displays (LCD's), entre outros. A função básica do *Flash* é executar uma interface interativa e proporcionar a audição de músicas e visualização de vídeos através de um reprodutor customizado e proprietário.

Os profissionais da área de *Web Design* utilizam as ferramentas da Adobe há anos, e sua própria evolução pode ser testemunhada em uma crescente biblioteca de jogos interativos para *Web*, páginas animadas, e a utilização da tecnologia por parte de sites que fornecem *streaming* por vídeo, como o Youtube e Vimeo, entre outros.

2.2.2.1 Alternativas e Opções em Padrão Aberto

Como alternativa ao *Flash*, alguns sites usam diversas outras tecnologias para exibir multimídia, como o *Microsoft Silverlight*, que, assim como o *Flash*, foi desenvolvido para a criação de apresentações na *Web*.

O *Java FX*, que pode ser usado para desenvolver aplicações para a internet em diversos dispositivos, conta também com plugin para as ferramentas da Adobe, que por sua vez exporta o arquivo no formato do java fx. (ORACLE, 2010).

Antes de abordar algumas opções em padrão aberto, é preciso definir o significado de um aplicativo livre, que é definido por Pisani e Piotet (2008) como sendo um aplicativo que pode circular livremente, não significando que o mesmo seja gratuito ou aberto.

Como uma das alternativas em padrão aberto, destaca-se o *Scalable Vector Graphics* (SVG) que é definido pela W3C(2003):

> SVG é uma linguagem em XML que descreve gráficos em duas dimensões. O SVG permite três tipos de objetos gráficos: formas gráficas em vetor (e.g., caminhos que consistem de linhas retas e curvas), imagens e texto. Objetos gráficos podem ser agrupados, estilizados, transformados e compostos em objetos previamente renderizados.

A maioria dos Navegadores *Web* oferecem suporte ao *SVG*. Atualmente, o grupo por trás do desenvolvimento da linguagem trabalha em uma nova especificação, o *SVG2.0* que deve combinar funcionalidades em um maior número de plataformas. (W3C, 2009).

Outra alternativa de desenvolvimento, o *Synchronized Multimedia Integration Language* (SMIL) é definido pela W3C(2006) da seguinte maneira:

> [...]uma linguagem baseada em XML que permite ao autor escrever apresentações multimídia. Usando o SMIL, um autor pode descrever um comportamento temporal de uma apresentação multimídia, associando

> hyperlinks com objetos de mídia e descrevendo a interface da apresentação em uma tela.

Conhecidos programas como *RealPlayer*, *QuickTime Player* e *Windows Media Player* estão habilitados para a visualização de apresentações em SMIL.

Não menos importante, o já citado HTML5 que segundo a W3C(2009), “representa um grande passo no desenvolvimento do HTML, introduzindo uma ampla gama de novas utilidades dentro da linguagem”. Em 2006, a W3C uniu-se no desenvolvimento da nova linguagem com a *Web Hypertext Application Technology Working Group* (WHATWG), organização formada pelas companhias Apple, Mozina e Opera, com objetivo de desenvolver o HTML5.

Portanto, a tecnologia que rege a internet continua evoluindo ao passo que a regulamentação das tecnologias e a adoção de um padrão comum são inevitáveis.

2.3 MULTIMÍDIA

Multimídia é um termo que ganhou popularidade com os computadores, mas não é necessariamente um termo novo, multimídia era visto, nos anos 60, na forma de apresentações com *slideshows*. Na época, shows com música, luzes e experiências táteis já eram realizados pela comunidade artística. (WISE, STEEMERS, 1999). O termo foi resgatado mais tarde na década de 80 para definir produtos especializados na indústria de computadores.

Feldman (apud WISE, STEEMERS 1999) define a multimídia como a “integração de dados, texto, imagens e som todos dentro de um mesmo dispositivo de informação digital”.

Segundo Mayer (2009), “Por centenas de anos, mensagens verbais – como palestras e aulas impressas – tem sido a principal maneira de explicar idéias aos estudantes.” Mayer (2009), também, afirma que “as pessoas aprendem melhor quando os gráficos e figuras são adicionados ao texto”.

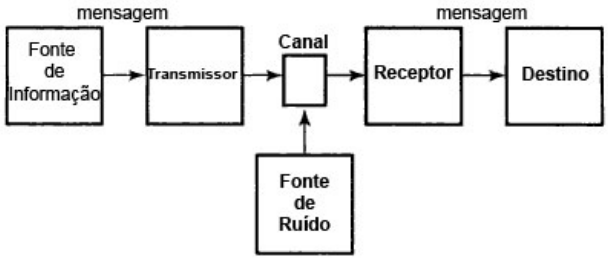
Hoje a multimídia se tornou uma força poderosa em nossa sociedade, quando usada da maneira correta, possui grande potencial para informar e educar as pessoas. Estamos em constante contato com ela todos os dias. (BANERJIE e GHOSH, 2010).

A palavra “multimídia” é um termo bastante saturado, quase todo mundo tem alguma idéia sobre o que ela é, embora seu significado seja diferente para diferentes pessoas. Para alguns profissionais, ela pode ser vista como um meio artístico, para outros como uma apresentação de negócios e ferramenta de comunicação. Para a maioria de nós, a multimídia é

vista como ferramenta de entretenimento e aprendizado. A multimídia de fato pode ser vista como ferramenta de comunicação e transferência de informação, e a comunicação é uma das principais responsaveis pelo desenvolvimento de nossa sociedade. (BANERJIE e GHOSH, 2010).

Segundo Banerjie e Ghosh (2010, p. xiv), “O campo da Multimídia culmina em um número de disciplinas e tecnologias. Cobre quase que todas as áreas da ciência da computação (tanto hardware quanto software) e comunicações”. Disciplinas fora do meio da informática como artes, psicologia, também, estão envolvidas, o que, segundo Banerjie e Ghosh(2010, p. xiv)”, tornam esse tema multi-disciplinar um tópico difícil de ser ensinado”.

No ramo de comunicações, tudo começou com o Modelo de Shannon, escrito em 1948, Shannon propôs um modelo para identificar modos de comunicação disponíveis naquela época.



Fluxograma 1 - Modelo de Comunicação de Shannon.
Fonte: Banerjie e Ghosh (2010).

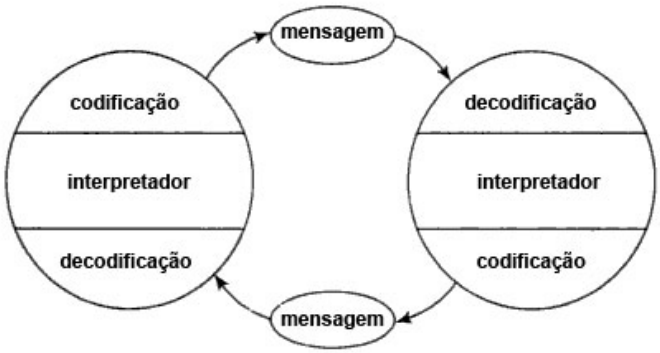
Banerjie e Ghosh(2010, p. 13) destacam o seguinte sobre o modelo de Shannon:

> Shannon providenciou um framework unificado para visualizar os distintos modos de comunicação até aquele momento (o telegrafo, telefone, radio e televisão). Ele estava interessado nos aspectos técnicos da comunicação. Embora, em seu modelo ele defina em termos matemáticos o que a informação é e como a informação pode ser transmitida em face do ruído. Sua teoria matemática da comunicação logo tornou-se conhecida como Teoria da Informação.

O Modelo de Shannon chamou atenção para o estudo da comunicação, que cresceu com o aparecimento de novos estudos, como o “Modelo de Schramm” de 1965, a contribuição de Schramm é destacada da seguinte maneira:

> Schramm providenciou diversos modelos, que essencialmente eram uma elaboração do Modelo de Shannon. Schramm viu a comunicação como um esforço com propósito de estabelecer um meio comum entre fonte e receptor. Banerjie e Ghosh(2010, p. 14).

É no modelo de Schramm que vemos pela primeira vez o emprego da palavra “feedback”. É através do *feedback* que Shramm tenta se livrar do problema do ruído presente no modelo de Shannon, assim, o fluxo da mensagem passa a ser em mão dupla.



Fluxograma 2 - Modelo de comunicação de Shramm.
Fonte: Banerjie e Ghosh (2010).

Como exemplo desse modelo, podemos ver que um palestrante observa a platéia e modifica suas informações de acordo com a resposta ou *feedback* que obtém dela.

As tecnologias de multimídia representam hoje um novo paradigma, sendo que suas mudanças trouxeram uma nova forma de comunicação, transformando nossas interfaces baseadas em texto em ricas e intuitivas interfaces gráficas. A sociedade está testemunhando a convergencia de sistemas e tecnologia da informação em sistemas de informação multimídia. (BANERJIE e GHOSH, 2010).

2.4 OS PROFISSIONAIS DO DESENVOLVIMENTO NA WEB

Web Designers e *Web Developers* são profissionais com metas distintas, porém relacionadas, ambos trabalham na construção de interfaces ou páginas para a *Web*. Seu perfil é o de criação de soluções para o usuário final, seja através de sites interativos ou funcionais.

Segundo a definição do *Web Design* de Implied by Design (2010):

> Web design tipicamente se refere ao processo de concepção de um web site ou layout de páginas web e muitas vezes inclui os elementos gráficos em uma página. O projeto pode ser desenvolvido usando um programa gráfico como o Adobe Photoshop, e fornece o ambiente que resulta da aparência de uma página web. O produto final do projeto de design normalmente não contém código. Pelo contrário, a representação gráfica da página web é usada por outra (ferramenta) ou do mesmo partido, como base para o código. A representação é dividida em áreas que podem ser representados pelo código web, e outras áreas que são puramente gráficos.

O aspecto criativo que rege o processo de desenvolvimento é quase que monopolizado pelo *Web Designer*, visto que este cria conceitos que serão aplicados posteriormente pelo *Web Developer*.

Como definido pela Inplied by Design (2010), acerca do *Web Development*, ou Desenvolvimento Web:

> Desenvolvimento Web é normalmente usado para descrever a programação necessária para construir o "back end" de um site. O “back-end” é a área do site que não é visto pelos visitantes, mas que faz o trabalho necessário para apresentar a informação certa no formato correto para os visitantes. Desenvolvimento Web é usado para descrever qualquer web design dirigido por base de dados utilizando linguagens dinâmicas de scripting como PHP, ASP, ASP.NET e Coldfusion. Também abrange a concepção e desenvolvimento de banco de dados. O termo também pode ser usado para script do lado do cliente, como JavaScript e Java.

Portanto, a função do *Developer*, ou desenvolvedor *Web*, é prestar funcionalidade ao *website* gráfico, ambos os profissionais da área de desenvolvimento *Web* devem buscar entendimento para saber se determinado projeto é possível, ou demasiado utópico, do ponto de vista de implementação tecnológica. Muitas vezes, uma idéia inicial do *Designer* pode parecer utópica, afinal é ele quem determina que a ação ou função seja executada a partir de uma implementação gráfica, ao passo que o Desenvolvedor encontra barreiras, muitas vezes tecnológicas, para a construção da funcionalidade.

Atualmente, um bom *Web Designer*, que tem por meta a criação de uma página *Web*, constrói-a inicialmente através de um projeto gráfico em uma ferramenta que torne seu trabalho possível. Existem diversas ferramentas gráficas que podem ser utilizadas, citando as mais comuns: O *Adobe Photoshop*, *Adobe Illustrator* e *Adobe Flash Builder*. Também é requisitado do profissional alguma noção de programação, principalmente aqueles que fazem o seu trabalho em caráter autônomo ou que não fazem parte de um processo de produção, com mais profissionais capacitados na área.

Por sua vez, o *Web Developer* tem, em sua meta, principalmente em empresas ou pequenos negócios, dar funcionalidade aos projetos gráficos efetuados pelo *Web Designer*. Em seu domínio estão, geralmente, mais de uma linguagem de programação, capacidade de gerenciar e administrar uma base de dados e, o mais importante, a comunicação constante com o *Designer* e sua meta inicial ao construir um gráfico. Cabe ao Desenvolvedor atribuir ao gráfico ou elemento multimídia, a sua devida funcionalidade, compatibilidade e manutenção.

Segundo Jenkins (2008):

> Nos primeiros estágios do design, não é sabido o custo final do projeto, até que é visto o resultado final. A solução vista seria a de levar ao

> conhecimento do desenvolvedor os custos esperados do projeto durante a fase do Design.

Portanto um bom projeto de *Web Design* e desenvolvimento deve levar em consideração os seguintes aspectos: a Ferramenta, os Profissionais engajados, o Tempo de produção e o Orçamento inicial, antes do início do projeto. (JENKINS, 2008).

Portanto, um dos fatores mais importantes para atingir essas metas é a boa comunicação entre os profissionais envolvidos.

2.5 CONSIDERAÇÕES FINAIS

A ferramenta *Adobe Flash Catalyst* é o alvo de estudo deste trabalho, visto a escassez de trabalhos sobre a própria ferramenta e seu impacto no meio profissional. Os tópicos vistos, neste capítulo, e as tecnologias citadas são importantes para dar conotação e posicionamento histórico para a tecnologia *Flash* e a ferramenta alvo do estudo. O capítulo seguinte aborda a metodologia do trabalho e as hipóteses a serem comprovadas com a criação de uma página funcional, utilizando-se a ferramenta.

3 METODOLOGIA

Neste capítulo, é apresentada a metodologia do trabalho.

3.1 METODOLOGIA CIENTÍFICA

Segundo GIL et al.(1999 apud Silva e Menezes 2005):

> Método científico é o conjunto de processos ou operações mentais que se devem empregar na investigação. É a linha de raciocínio adotada no processo de pesquisa. Os métodos que fornecem as bases lógicas à investigação são: dedutivo, indutivo, hipotético-dedutivo, dialético e fenomenológico.

Este projeto é classificado da seguinte forma:

Do ponto de vista de sua natureza, como uma pesquisa Aplicada, (SILVA, MENEZES 2005), "objetiva gerar conhecimentos para aplicação prática e dirigidos à solução de problemas específicos. Envolve verdades e interesses locais."

Do ponto de vista da forma de abordagem do problema, classifica-se como uma pesquisa Qualitativa, segundo Silva e Menezes (2005):

> Considera que há uma relação dinâmica entre o mundo real e o sujeito, isto é, um vínculo indissociável entre o mundo objetivo e a subjetividade do sujeito que não pode ser traduzido em números. A interpretação dos fenômenos e a atribuição de significados são básicas no processo de pesquisa qualitativa. Não requer o uso de métodos e técnicas estatísticas. O ambiente natural é a fonte direta para coleta de dados, e o pesquisador é o instrumento chave. É descritiva. Os pesquisadores tendem a analisar seus dados indutivamente. O processo e seu significado são os focos principais de abordagem.

Do ponto de vista de seus objetivos, é classificado como uma pesquisa Exploratória, conforme Gil (1991):

> Visa proporcionar maior familiaridade com o problema com vistas a torná-lo explícito ou a construir hipóteses. Envolve levantamento bibliográfico; entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado; análise de exemplos que estimulem a compreensão. Assume, em geral, as formas de Pesquisas Bibliográficas e Estudos de Caso.

Do ponto de vista dos procedimentos técnicos, é uma pesquisa Bibliográfica (GIL, 1991), "quando elaborada a partir de material já publicado, constituído principalmente de livros, artigos de periódicos e, atualmente, com material disponibilizado na Internet."

3.2 ETAPAS DA PESQUISA

Este trabalho é dividido em etapas que são: Escolha do tema, revisão literária, identificação do tema do protótipo, análise da ferramenta, implementação do protótipo e análise dos resultados à luz das hipóteses.

3.2.1 Escolha do tema

O presente trabalho aborda uma exemplificação de uma interface *Web*, construída à partir da ferramenta *Adobe Flash Catalyst*, objeto do estudo. A ferramenta foi escolhida com base em sua possível facilidade para implementação de interfaces animadas, que utilizam a tecnologia *Adobe Flash*.

São hipóteses deste projeto:

- a ferramenta é de fácil aprendizado;
- o tempo de execução, ou criação, da interface *Web* pode ser abreviada, com base na facilidade de criação da interface;
- a ferramenta permite uma fácil implementação;
- os custos do projeto podem ser abreviados em termos de “hora-homem”;
- a interface criada pela ferramenta é de fácil manutenção.

3.2.2 Revisão literária

No meio acadêmico, a literatura sobre a ferramenta de estudo ainda é bastante escassa, tornando este trabalho de pesquisa inédito. Embasamos a pesquisa em literaturas já publicadas sobre a tecnologia *Flash* e seu lugar na história da tecnologia para a *Web*. Abordamos, na Revisão Bibliográfica, os temas necessários para posicionar a tecnologia *Adobe Flash* em seu lugar histórico e paralelo com outras tecnologias.

3.2.3 Identificação do tema do protótipo

O protótipo a ser implementado é uma interface para *Web*, ou página *Web*, em formato de página Inicial e Sub-páginas. Por página inicial entende-se que a interface terá os seguintes elementos: Banner principal, Menu de navegação, Corpo de alteração ou Leitura de texto e

“Footer”, ou rodapé.

A página a ser construída será do curso de pós-graduação da Unisul, Especialização em Engenharia de Projetos de *Software* e deve conter todos os dados do curso e turmas de maneira organizada através de áreas específicas, discriminadas por botões em um Menu de navegação.

A parte principal do *Website* será construída com a ferramenta *Adobe Flash Catalyst*, que utiliza a linguagem de programação *Flex*. A partir da implementação desse aplicativo em *Flex*, poderemos analisar a ferramenta a contento.

3.2.4 Análise da ferramenta

A análise da ferramenta será feita de forma empírica, observando suas funcionalidades e suas principais características. Sua capacidade de cumprir seu objetivo deve ser avaliada e constatada com a implementação de um exemplo prático de página *Web*, advinda do protótipo.

Durante o processo de avaliação, serão coletados dados qualitativos em quatro tópicos principais: tempo de aprendizagem, tempo de uso gasto na construção da página *Web*, qualidade da implantação e eficiência.

3.2.5 Implementação do protótipo

O protótipo será implementado pela ferramenta gráfica de desenvolvimento *Adobe Photoshop CS5*. O resultado será uma imagem estática, não interativa e não animada.

Posteriormente, o protótipo será aberto para edição na ferramenta *Adobe Flash Catalyst* e, a partir desta, a construção e animação da interface será executada e seu código gerado em *Flex*.

Destaca-se, nesse processo, o fato da ferramenta de estudo não ser capaz de criar um protótipo de uma interface para a *Web* de forma autônoma, sendo totalmente dependente de uma ferramenta externa para a execução desse processo.

3.2.6 Análise dos resultados

A principal meta deste trabalho é a análise da ferramenta *Adobe Flash Catalyst* à partir da construção de um exemplo prático de *Website*. Após a coleta dos dados quantitativos e

qualitativos da ferramenta e a criação da interface para o *Website*, pretende-se analisar os resultados obtidos no cumprimento da tarefa de construção do aplicativo, considerando as características da ferramenta, sua funcionalidade e as hipóteses levantadas neste capítulo. Levado em consideração o tempo de aprendizado e tempo de execução do trabalho na ferramenta, a partir desse processo, será possível estabelecer argumentos para a defesa deste trabalho.

3.2.7 Delimitação

Pretendemos com esta monografia estudar os seguintes elementos da ferramenta de desenvolvimento *Adobe Flash Catalyst*:

- tempo de estudo da ferramenta, ou tempo médio gasto para aprender a utilizar a ferramenta em suas funções básicas;
- tempo gasto ao criar um exemplo de página *Web*, da disciplina de pós-graduação da Unisul, em suas funções básicas como Página principal e Sub-páginas;
- avaliar a ferramenta sob o aspecto de sua funcionalidade e de sua capacidade de cumprir o que levantamos como Hipótese.

Este trabalho não aborda elementos de código de programação, como também não pretende-se criar uma página *Web* com a ajuda de outro programa externo que não sejam os utilizados para criação gráfica do protótipo.

No entanto, para a construção da página, faz-se necessário o uso de um sistema de *Content Management System* (CMS) para postagem de notícias e abrigo da tecnologia *Flash*, como pode ser visto no próximo capítulo.

4 A FERRAMENTA ADOBE FLASH CATALYST

Neste capítulo, será apresentada a ferramenta que serve como estudo de caso desta monografia, o *Adobe Flash Catalyst*. Apresentamos a ferramenta sob um aspecto básico de suas funções, destacando sua interface, suas funções, seus componentes básicos e sua estrutura dedicada ao código de programação. Além disso, destacamos a potencialidade da ferramenta frente ao mercado em que os profissionais se situam, a fim de comprovar nossas hipóteses.

4.1 INTRODUÇÃO À FERRAMENTA DE DESENVOLVIMENTO ADOBE FLASH CATALYST

A Ferramenta de desenvolvimento *Adobe Flash Catalyst* é recente e passou a ser comercializada para o público no ano de 2010.

A *Adobe Systems* sugere que a sua ferramenta, voltada para o mercado de *Web Design*, compreenda um nicho de *Web Designers* que visam à facilidade de implementação, em termos, seu público alvo é leigo na implementação do *Adobe Flash*, no entanto, ela também se destina a profissionais mais avançados que, de alguma forma, pretendem poupar tempo na construção de interfaces *Web* animadas.

Como dito pela *Adobe Systems* (2010), em seu *Website* oficial:

> Crie rapidamente protótipos funcionais ou iteração em projetos de interface que podem ser aproveitados no produto final, e participe na concepção e desenvolvimento de fluxo de trabalho com os desenvolvedores que usam o Adobe ® Flash ® Builder™

Esses desenvolvedores que usam o *Adobe Flash Builder*, como alega a empresa, são profissionais de Desenvolvimento para a *Web*.

Uma das afirmações sobre a ferramenta é sua capacidade de abreviar o processo de produção de uma interface, poupando algum trabalho do profissional que o cria. Todo o conteúdo gráfico criado previamente é exportado da ferramenta de edição gráfica para o *Flash Catalyst*, e nesta, é gerada automaticamente o código de programação do aplicativo.

Segundo a *Adobe Systems* (2010):

> O software Adobe® Flash® Catalyst™ CS5 é uma nova ferramenta de criação de interação mais acessível. Transforme a arte-final do Adobe Photoshop®, Illustrator® e Fireworks® em projetos expressivos e totalmente interativos sem a necessidade de escrever códigos e aumente o alcance e a consistência com o Adobe Flash Platform.

E, como dito por Tapley (2010):

> O fluxo de trabalho do Flash Catalyst começa pela criação de artwork e estruturação do aplicativo usando ferramentas de design já familiares, o Adobe Illustrator, Adobe Photoshop e o Adobe Fireworks. A composição de design é então importado para o Flash Catalyst. Camadas, grupos, nomes, a estrutura e as posições da obra são preservadas no Flash Catalyst.
> Com seu trabalho artístico original e design estrutural preservados no Flash Catalyst, agora você pode começar a tornar arte estática em componentes de aplicações funcionais.

Portanto, dois fatores chave aparecem, os quais são motivo de estudo deste trabalho, o primeiro, a rapidez de construção de uma interface e sua animação na ferramenta, o segundo, a não necessidade de trabalho com código de programação, o que elimina o *Web Designer* de tal tarefa, visto que o código *Flex*, gerado pela ferramenta, não pode ser editado na própria ferramenta e, sim, na ferramenta de desenvolvimento do *Web Developer*.

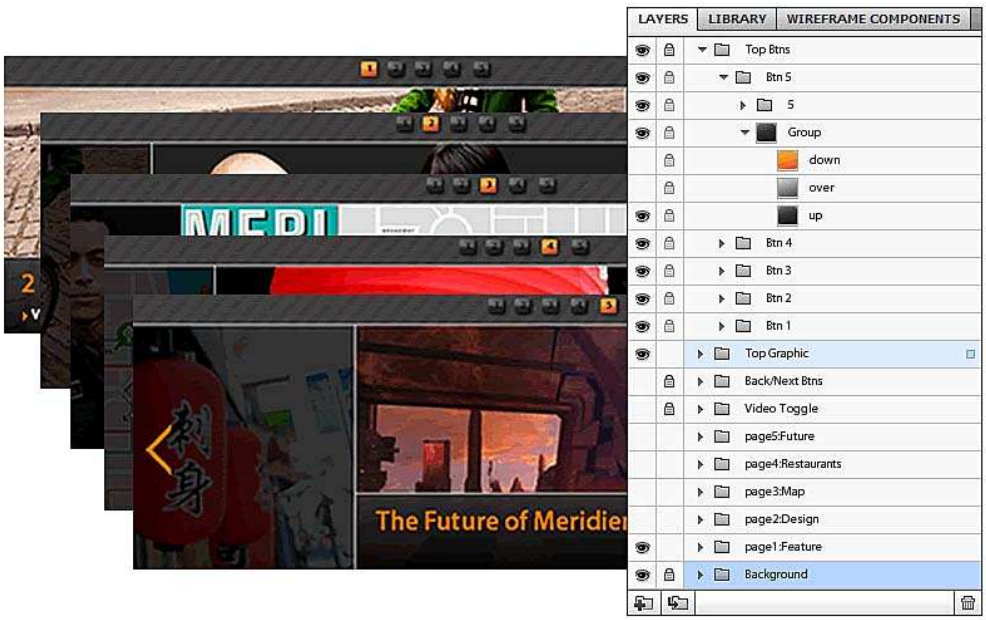


Figura 5 – Exemplo da estrutura de camadas, importada da ferramenta gráfica.
Fonte: Scott Taplay (2010).

A escolha da ferramenta, para a construção de um exemplo de página *Web*, que visa a confirmar as hipóteses, é obvia. A ferramenta, ainda, é nova e suas características e funcionalidades precisam ser pesquisadas e trabalhadas, como dito por Taplay (2010):

> Muitos criadores já começaram a trabalhar com o Flash Catalyst, e a resposta é muito positiva. Usando o Flash Catalyst, os designers podem publicar aplicações Flex sem escrever uma única linha de código. Alguns exemplos são anúncios interativos, guias de produto, portfólios de design e interfaces de usuário para aplicações de mídia rica.
> [...]Um desafio comum para os designers de aplicações é entregar obras estáticas e descrever a experiência de usuário intensionada aos desenvolvedores. A idéia original é deixada para a interpretação e, por vezes

> perdido na tradução. Ao usar o Flash Catalyst, os designers podem agora publicar aplicações para internet, ricas e completas, ou fornecer aplicações de elemento funcional para desenvolvedores, mantendo a fidelidade de design por todo o fluxo de trabalho.

Desse modo, o trabalho do *Web Designer* se torna simples e fiel ao que ele propôs inicialmente para que a experiência do usuário seja completa e da forma que ele intensionou, deixando para o Desenvolvedor *Web* apenas a aplicação e correção de código de programação *Flex* gerado pelo programa. Isso posto em prática, mudaria o perfil dos profissionais da área que pretendem trabalhar com estas ferramentas.

4.1.2 Apresentação da Interface Gráfica

A ferramenta de desenvolvimento *Adobe Flash Catalyst* foi criada pela *Adobe Systems*, inicialmente, com o propósito de criar interfaces interativas e animadas para a *Web*, a exemplo de sua ferramenta principal de desenvolvimento para a plataforma *Flash*, o *Adobe Flash Builder*, desta vez a companhia optou por um enfoque diferente, levando a crer que os profissionais da área necessitam de uma ferramenta mais simples de atuar e mais fácil de transformar gráficos animados para o ambiente *Web*.

Desta forma, a *Adobe Systems* criou a ferramenta sob o foco da facilidade, em que o profissional da área de *Web Design* não necessita de qualquer conhecimento sobre código de programação, trabalhando na ferramenta apenas sob o ponto de vista gráfico. Este mesmo profissional deverá ter conhecimento prévio de sua ferramenta de criação gráfica, o *Adobe Photoshop*, ou o *Adobe Illustrator*, visto que o *Flash Catalyst* possui recurso limitado para a criação de gráficos complexos, limitando-se somente para formas geométricas básicas e focando sua ação na animação e transformação desses objetos.

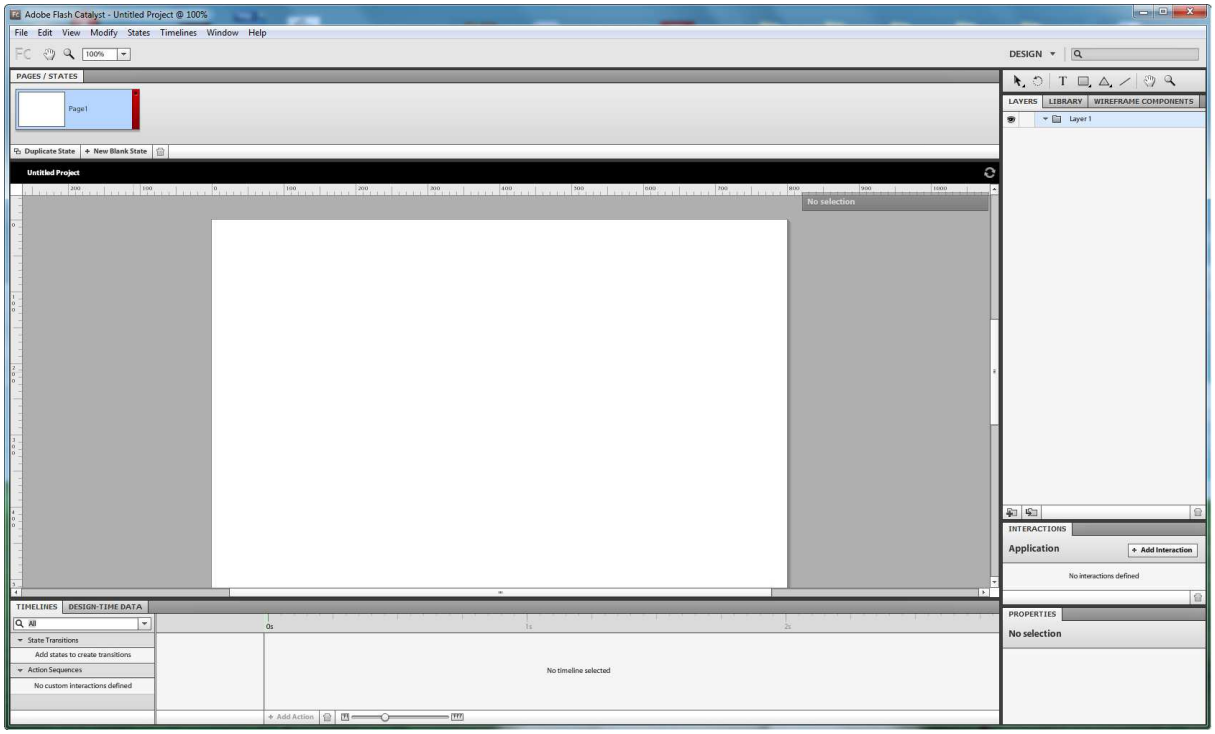


Figura 6 – Interface de trabalho do Adobe Flash Catalyst.
Fonte: Autores do Projeto.

A ferramenta se apresenta de forma semelhante às ferramentas gráficas *Adobe Photoshop* e *Adobe Illustrator* de forma que sua organização se assemelha. A posição das *Layers*, ou camadas, situam-se no quadrante direito da interface, quase identica às ferramentas gráficas, o que faz sentido para o profissional trabalhar nelas de forma semelhante. É importante que o profissional mantenha as Camadas organizadas, visto que o trabalho de animação e funcionalidade é feito com base no posicionamento individual dessas Camadas. Exemplificando, é possível tranformar gráfico em botões animados, desde que os elementos estejam individualizados nas Camadas, como exemplificado por Scott Taplay (2010):

> O fluxo de trabalho do Flash Catalyst começa pela criação de artwork e estruturação do aplicativo usando ferramentas de design familiares - Adobe Illustrator, Adobe Photoshop e Adobe Fireworks. A composição do design é então importada para o Flash Catalyst. Camadas, grupos, nomes e a estrutura e posição da *artwork* são preservadas no Flash Catalyst.

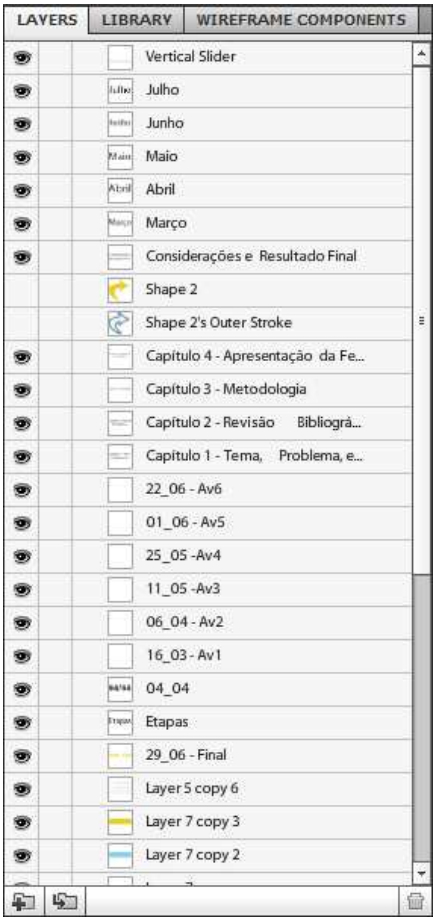


Figura 7 – Formato das *Layers* – Camadas, importadas da ferramenta gráfica.
Fonte: Autores do Projeto.

As camadas são parte importante do programa, elas são individualmente selecionadas, sendo que cada um desses elementos pode receber uma animação ou função específica da ferramenta *Flash Catalyst*, ao gosto do profissional, de forma que a interface é criada a partir da manipulação gráfica pela ferramenta e, ao mesmo, tempo gerada em código no *background*. Dessa forma, é possível animar e dar funcionalidade a qualquer elemento gráfico advindo de um protótipo externo ao programa. Todas as camadas criadas no programa advêm das ferramentas gráficas utilizadas na criação do protótipo.

4.1.3 Funcionalidades da ferramenta

As funções que podem ser atribuídas às camadas individuais são delimitadas pela ferramenta e compreendem uma série básica de características, que fazem sentido à navegação de uma página *Web*. As funções do programa são básicas, porém a gama de animações possível aos elementos é variável ao gosto do profissional, que pode animar os elementos da forma que achar necessário através do uso de "timelines". Cada elemento passa

por estados, ou páginas, definidas previamente, e sua ativação depende às vezes do clique, ou apenas o apontamento do cursor em um dos elementos efetuado pelo usuário. Esse processo é melhor explicado no Capítulo 6 deste trabalho. Vale destacar que estas funções, na forma que se apresentam, visam a poupar tempo ao profissional que as implementa e permitem-lhe construir uma interface completamente customizada e diferente dos padrões atuais dos Navegadores *Web*.

Logo acima das camadas, o profissional conta com uma limitada coleção de Ferramentas para manipulação gráfica. Por se tratar de uma ferramenta de animação de estados e páginas, o *Flash Catalyst* não necessita criar gráficos complexos, visto que os arquivos importados de ferramentas gráficas já desempenham esse papel. Porém, é possível, caso seja da intenção do profissional, começar um projeto em branco e construído apenas a partir do *Flash Catalyst*, apesar desse modo apresentar diversas limitações se comparando-as com ferramentas dedicadas a este propósito.

Figura 8 – Funções componentes básicas da ferramenta.
Fonte: Autores do Projeto.

Os componentes básicos da ferramenta visam a proporcionar uma variedade de elementos já conhecidos que são aderidos a uma página *Web*, como dito por Scott Tapley (2010):

> [...]Flash Catalyst fornece um conjunto de componentes embutidos com os estados e comportamentos pré-definidos, como o up, over, down, e os estados desabilitados de um simples componente de botão. Você seleciona a *Artwork* à incluir no componente, diz ao Flash Catalyst que tipo de componente deve criar, e depois indica qual *Artwork* deve mostrar ou ocultar, em cada estado do componente.

As funções básicas da ferramenta permitem anexar ao gráfico elementos de interação, dessa forma o trabalho de edição é facilitado. Um bom exemplo seria atribuir uma barra de rolagem vertical ou horizontal em uma Camada gráfica estática, prototipada anteriormente. É essa facilidade que permite poupar tempo no decorrer do processo de produção de uma página *Web*. E, também, nesse nível de entendimento, é possível customizar uma página *Web* de forma completa, ao gosto pessoal, de forma única e original. A ferramenta, também, torna possível customizar seu próprio componente para exercer determinada função, como exemplificado por Scott Taplay (2010):

> [...]Se a opção do componente disponível não atender às suas necessidades, você pode utilizar a opção componente *Custom / Generic* (sic) para projetar componentes originais com até 20 estados. Usando componentes personalizados, você pode adicionar profundidade e alcance ilimitado ao seu aplicativo assentando componentes dentro de outros componentes.

4.1.4 Interface de código

A interface de código da ferramenta visa apenas à informação, ou seja, não se trata de uma área editável pelo autor do projeto. Como especificado pela Adobe, o usuário da ferramenta não necessita do conhecimento sobre o código de programação.

No caso do *Adobe Flash Catalyst*, a linguagem *Flex* é adotada como padrão e pode somente ser visualizada. Dessa forma, um profissional mais ciente do código pode corrigir pequenos *bugs* ou adicionar pequenas alterações em outra ferramenta específica. Também pode se tornar um fato rotineiro a checagem final do código, para verificar elementos desnecessários ao Projeto gráfico ou apenas a simples correção de animações ou otimização de elementos que ocupam muitas linhas de código.

O simples esquecimento de um elemento oculto dentro das camadas pode ser visto no código de programação, adicionando peso ao projeto final. Dessa forma, torna-se trivial ao profissional buscar elementos desnecessários na Biblioteca de imagens do programa e deletá-las, se necessário.

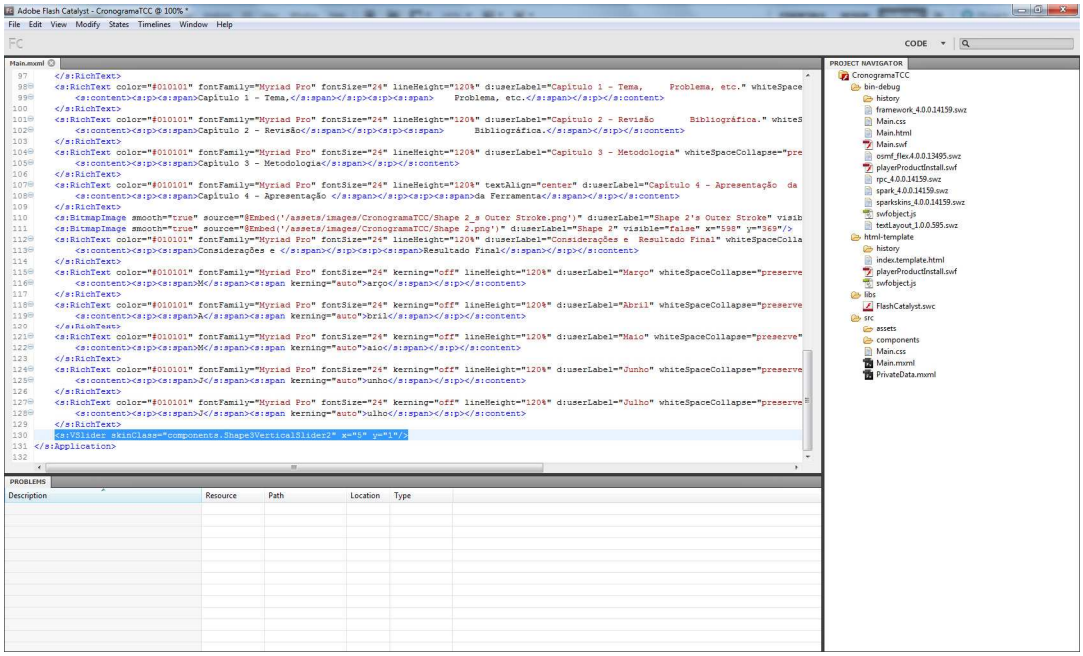


Figura 9 – Interface de código.
Fonte: Autores do Projeto.

Portanto, o profissional alvo da ferramenta pode levar ao conhecimento do Desenvolvedor o código de programação, uma alternativa viável, caso o profissional deseja determinada funcionalidade que o programa não pode suprir, ou correção de *bugs*.

Dessa forma, a ferramenta se situa apenas aos profissionais de *Web Design*, que têm em mãos o controle essencial na hora de dar funcionalidade e animação aos componentes desejados em uma Página *Web*. A seguir, será descrito com quais sistemas o *Website* e o aplicativo, criado pela ferramenta, são gerenciados e postos em prática.

4.2 SISTEMA DE GERENCIAMENTO DE CONTEÚDO - CMS

O sistema de Gerenciamento de Conteúdo, ou comumente chamado de CMS (*Content Management System*), permite ao cliente gerenciar um portal, *Website* ou *Blog* na *Web*. A partir do uso desse sistema, é possível inserir e editar conteúdo em tempo real na internet, sem a necessidade de programas externos, como exemplificado por Millarch (2005):

> [...]um sistema de gerenciamento de conteúdo, ou em inglês, CMS – Content Management System, cujo objetivo é exatamente o de estruturar e facilitar a criação, administração, distribuição, publicação e disponibilidade da informação.
>
> [...]Parece complexo, mas, sob o ponto de vista do usuário final, não é. Um CMS oferece ferramentas simples, todas acessadas através de qualquer

> navegador (Internet Explorer, Netscape, Firefox), que permite realizar todo o processo de gerência, desde a criação até o arquivamento do conteúdo.
>
> [...]Em termos simples, um CMS permite que a empresa tenha total autonomia sobre o conteúdo e evolução da sua presença na internet e dispense a assistência de terceiros ou empresas especializadas para manutenções de rotina. Nem mesmo é preciso um funcionário dedicado (o famoso webmaster), pois cada membro da equipe poderá gerenciar o seu próprio conteúdo, diluindo os custos com recursos humanos.

Dessa forma, aliada a outras linguagens de programação e a recursos multimídia, um portal para a *Web* pode ser criado de forma que sua autonomia dependerá somente do Cliente que o utilizará. Também é notável que dessa forma não é necessário dispendiar esforços na criação de um sistema semelhante, poupando recursos financeiros e tempo de trabalho.

Este projeto baseia seu exemplo de *Website* na ferramenta de CMS *Wordpress*. A ferramenta foi escolhida por sua versatilidade e popularidade no mercado de *Websites* e Portais profissionais e, também, por sua facilidade na aceitação de conteúdo *Adobe Flash*. Outro fator levado em consideração é o fato da ferramenta ser Código livre ou (*Open Source*), podendo ser usada para fins comerciais ou não, livrando o cliente de taxas ou *royalties.*

4.2.1 Sistema de Gerenciamento de Conteúdo - Wordpress

O sistema *Wordpress* tornou-se conhecido nos últimos anos e logo popularizou-se perante a comunidade de *Webdesign*. Ele tornou possível a gerência de *Websites* e Portais de conteúdo, com custo zero de implementação. Seu "*template"* ou modelo básico de desenvolvimento é gratuito e fornece o alicerce para a criação de uma grande variedade de Websites.

Segundo o Wordpress (2010):

> O núcleo do software é construído por centenas de voluntários da comunidade,[...] existem milhares de plugins e temas disponíveis para transformar seu site em quase qualquer coisa que você pode imaginar. Mais de 25 milhões de pessoas escolheram o WordPress para alimentar o local na web que eles chamam de "casa".
>
> [...]WordPress começou em 2003 com um único bit de código para melhorar a tipografia da escrita[...]. Desde então tem crescido a ser a maior auto-hospedada ferramenta de blogging do mundo, usado em milhões de sites e vista por dezenas de milhões de pessoas todos os dias.
>
> Tudo que você vê aqui, desde a documentação para o código propriamente dito, foi criado pela e para a comunidade. WordPress é um projeto Open Source, o que significa que existem centenas de pessoas em todo o mundo trabalhando nisso.

> [...]WordPress começou apenas como um sistema de blogging, mas evoluiu para ser usado como um completo sistema de gerenciamento de conteúdo.

Tornando possível o anexo do aplicativo *Adobe Flash*, o *Wordpress* é a escolha ideal para a implementação do exemplo de *Website* deste trabalho, facilitando sua posterior manutenção e usabilidade. Serve como uma plataforma para aplicativos de interface animada e proporciona a avaliação da ferramenta de estudo deste trabalho, o *Adobe Flash Catalyst*.

Escolhe-se o *Wordpress* como base para o desenvolvimento, pois o mesmo corrobora a necessidade de integração e a difusão de idéias e informações entre os usuário. Nos *Websites* atuais, tornou-se necessário o uso de uma ferramenta já amplamente testada e segura, abreviando, assim, o tempo de desenvolvimento de projetos.

Outro fato sobre o sistema *Wordpress* é sua extensa biblioteca de informações online e o comprometimento de sua comunidade com atualizações do sistema. Seu *Website* principal, acessado por *Wordpress.org*, fornece documentação e ajuda para iniciantes, tornando seu sistema cada vez mais difundido entre as comunidades virtuais.

4.3 CONSIDERAÇÕES FINAIS

Neste capítulo, vimos uma introdução teórica para a implementação das etapas de construção do *Website* e do Aplicativo *Flash* no Capítulo 6 deste trabalho. No próximo capítulo, abordamos a modelagem do trabalho. Nele podemos visualizar melhor a interface do *Website* através do Protótipo criado na ferramenta gráfica e exportada para a ferramenta do estudo.

5 MODELAGEM DO PROTÓTIPO

Neste capítulo, é apresentada a modelagem do protótipo por meio do modelo ICONIX adaptado à situação do caso da monografia.

5.1 ICONIX

Rosenberg, Stephens e Collins-Cope (2005, p. 3) falam que “O Processo ICONIX é uma análise direcionada a casos de uso e de metodologia de design”.

O ICONIX apareceu anos antes do UML como uma síntese das melhores técnicas das metodologias originais que formaram a UML: (OMT) *Object Modeling Technique* de Jim Rumbaugh, o método Objectory de Ivar Jacobson e o método Booch de Grady Booch. (ROSENBERG; STEPHENS; COLLINS-COPE, 2005, p. 40).

Rosenberg, Stephens e Collins-Cope (2005, p. 41) ressaltam que “a medida que esse material era ensinado, a maior parte das pessoas preferia aprender (e usar) o menor conjunto de diagramas”.

Então, chegou-se aos diagramas de UML abaixo (ROSENBERG; STEPHENS; COLLINS-COPE, 2005, p. 41):

- Diagramas de Classe;
- Casos de Uso (diagramas e texto);
- Diagramas de Sequência;
- Diagramas de Robustez.

5.1.1 Teoria do ICONIX

O ICONIX visa a guiar o *design* de *software* através de requisitos de comportamento, um passo de cada vez. (ROSENBERG; STEPHENS; COLLINS-COPE, 2005, p. 41).

Segundo Rosenberg, Stephens e Collins-Cope (2005, p. 42), no ICONIX, tudo tem um propósito primário:

- Texto dos Casos de Uso: Definem os requisitos de comportamento.
- Modelo de Domínio: Descreve os objetos do mundo-real e suas relações.
- Diagrama de Robustez: Desambigua os requisitos de comportamento(e amarra eles ao modelo de objetos).
- Diagrama de sequência: Aloca comportamento.

5.1.2 Casos de Uso e Atores

Para Rosenberg, Stephens e Collins-Cope (2005, p. 42), os "Casos de Uso são sequências de ações que um ator realiza dentro de um sistema para alcançar algum resultado".

> [...] O resultado da modelagem de casos de uso deve ser a de que todas as funcionalidades requeridas do sistema estão descritas nos casos de uso. Se você não aderir a este princípio básico você corre o risco de criar um sistema que não é o que o cliente quer. (ROSENBERG; STEPHENS; COLLINS-COPE, 2005, p. 42).

Dentro do diagrama de casos de uso, os atores representam papéis que os usuários podem realizar dentro de um sistema, podendo ser, também, outro sistema ou base de dados, que irá residir fora do sistema que está sendo modelado. (ROSENBERG; STEPHENS; COLLINS-COPE, 2005, p. 43).

5.2 LINGUAGEM UNIFICADA DE MODELAGEM (UML)

A UML tem como objetivo a visualização, especificação, construção e documentação de sistemas de *software*. Proporciona a preparação de projetos de sistemas, incluindo aspectos conceituais como processos de negócios e funções do sistema e, também, itens concretos como classes escritas em determinada linguagem de programação. (BOOCH; RUMBAUGHT; JACOBSON, 2006).

Segundo Booch, Rumbaught e Jacobson (2006), "A modelagem é uma parte central de todas as atividades que levam à implantação de um bom software."

Com a modelagem, Booch, Rumbaught e Jacobson (2006, p. 6) dizem que alcançamos os seguintes objetivos:

1. Os modelos ajudam a visualizar o sistema como ele é ou como desejamos que seja.
2. Os modelos permitem especificar a estrutura ou o comportamento de um sistema.
3. Os modelos proporcionam um guia para a construção do sistema.
4. Os modelos documentam as decisões tomadas.

Guedes (2009, p.19) ressalta que:

> A UML não é uma linguagem de programação, e sim uma linguagem de modelagem, uma notação, cujo objetivo é auxiliar os engenheiros de software a definirem as características do software, tais como seus requisitos, seu comportamento, sua estrutura lógica, a dinâmica de seus processos e até mesmo suas necessidades físicas em relação ao equipamento sobre o qual o sistema deverá ser implantado. Tais características podem ser definidas por meio da UML antes do software começar a ser realmente desenvolvido.

5.3 LEVANTAMENTO E ANÁLISE DE REQUISITOS

Segundo Guedes (2009, p.21):

> Uma das primeiras fases de um processo de desenvolvimento de software consiste no Levantamento de Requisitos. As outras etapas, sugeridas por muitos autores são: Análise de Requisitos, Projeto, que se constitui na principal fase da modelagem, Codificação, Testes e Implantação. Dependendo do método/processo adotado, essas etapas ganham, por vezes, nomenclaturas diferentes, podendo algumas delas ser condensadas em uma etapa única, ou uma etapa pode ser dividida em duas ou mais etapas. [...] As etapas de levantamento e análise de requisitos trabalham com o domínio do problema e tentam determinar "o que" o software deve fazer e se é realmente possível desenvolver o software solicitado.

Guedes (2009, p.22) ressalta que "a fase de levantamento de requisitos deve identificar dois tipos de requisitos: os funcionais e os não-funcionais."

5.3.1 Requisitos Funcionais

Os requisitos funcionais correspondem ao que o cliente quer que o sistema realize, as funcionalidades do software. (GUEDES, 2009, p. 22).

São requisitos funcionais deste projeto, em termos de protótipo:

Diagrama 1 - Requisitos Funcionais.
Fonte: Autores do Projeto.

Os requisitos listados são as funcionalidades do *Website*, portanto, cada um dos elementos pode ser entendido como uma página específica para anexar conteúdo. No caso do exemplo de *Website* a ser implementado, cada um dos elementos é tratado como uma Página ou Estado. Dentro da tecnologia *Flash*, vários desses elementos listados são estados, alguns com página própria, e utilizam a tecnologia *Flash* para dar introdução ao conteúdo e tornar a interação com o *Website* mais rica e funcional.

5.3.2 Requisitos Não-Funcionais

Os requisitos não-funcionais correspondem às restrições que devem ser levadas a efeito sobre os requisitos funcionais. (GUEDES, 2009, p. 22).

São requisitos não funcionais do protótipo:

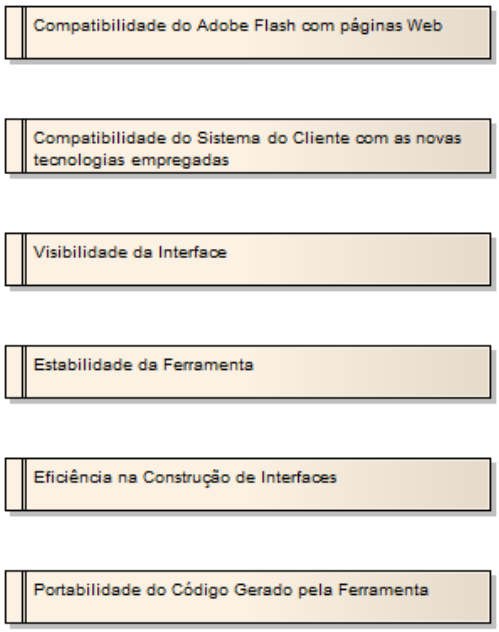


Diagrama 2 - Requisitos Não-Funcionais
Fonte: Autores do Projeto

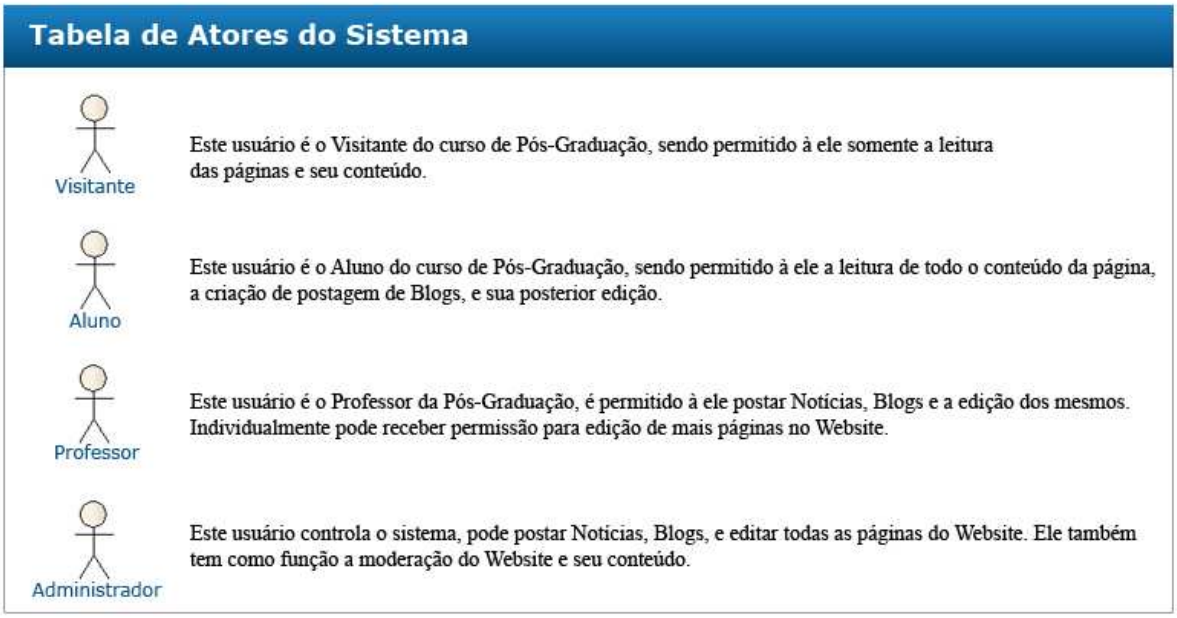
Os requisitos não-funcionais são elementos necessários para o bom funcionamento do *Website*, portanto, eles servem de pré-requisito para que a interação do usuário ocorra exatamente da maneira intensionada pela tecnologia.

5.3.3 Atores do Sistema

Segundo Guedes (2009, p.56):

> Os atores representam os papéis desempenhados pelos diversos usuários que poderão utilizar, de alguma maneira, os serviços e funções do sistema. Eventualmente um ator pode representar algum hardware especial ou mesmo outro software que interaja com o sistema, como no caso de um agente de software ou um sistema integrado, por exemplo. Assim, um ator pode ser qualquer elemento externo que interaja com o software.

São atores do sistema:



Infográfico 1 - Atores do Sistema.
Fonte: Autores do Projeto.

Da capacidade do usuário, que são distintos e podem ter diferentes regras de acesso, podemos ver o quadro, a seguir, para níveis de permissão das páginas:

Páginas/Permissão	Visitante	Aluno	Professor	Administrador
Notícias	L	L	L R E	L R E
Blogs	L	L R E	L R E	L R E
Curso	L	L	L	L R E
Professores	L	L	L	L R E
Turmas	L	L	L	L R E
Localização	L	L	L	L R E
Contato	L	L	L	L R E
	L - Leitura	R - Redação/Escrita		E - Edição

Quadro 3 - Tabela de Permissão dos Atores.
Fonte: Autores do Projeto.

Portanto, cada Ator dentro do sistema tem seu nível de permissão, isso não significa que o resultado final do sistema terá essa nomenclatura. Um exemplo seria dar previlégios de administrador para o Professor, para que este possa editar mais páginas.

Vale destacar que o sistema de CMS *Wordpress* já possui uma nomenclatura de acesso para seus usuários, podendo ser alterada ao gosto do administrador ou cliente do *Website*, como pode ser visto no Capítulo 6, sub-tópico 6.2.4 Permissões de Acesso.

5.3.4 Diagrama de Casos de Uso

Segundo Guedes (2009, p56):

> Os casos de uso referem-se aos serviços, tarefas ou funcionalidades que podem ser utilizados de alguma maneira pelos atores que interagem com o sistema, sendo utilizados para expressar e documentar os comportamentos pretendidos para as funções deste.

São Casos de Uso do Protótipo:

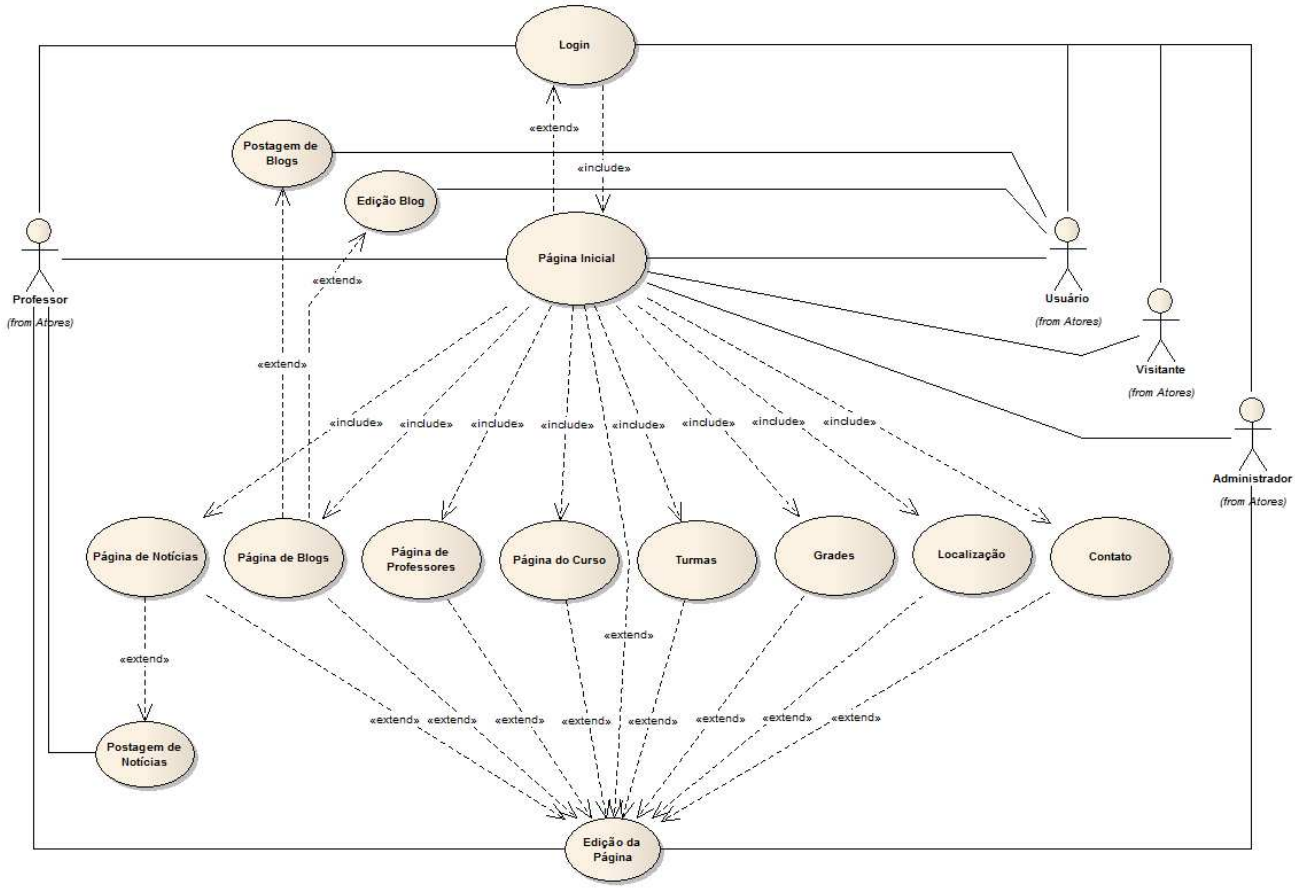


Diagrama 3 - Casos de Uso.
Fonte: Autores do Projeto.

No diagrama apresentado, destaca-se a interação imediata dos Atores do sistema com os elementos do *Website*. No gráfico, são listados os requisitos funcionais e como eles se relacionam com os usuários do sistema. Cada elemento pode ser interpretado como uma

página ou estado específico. Como intencionado pelos autores, cada seção tem a seguinte função no sistema:

notícias; Nesta página, o usuário visitante e alunos apenas têm permissão para leitura, Professores com a devida permissão podem postar notícias relacionadas ao curso, seja de caráter informativo a respeito do curso ou de atualidades que se relacionam com o mesmo.

blogs; Nesta seção, alunos e professores tem permissão de leitura e escrita. Aqui é possível que ambos discutam assuntos do cotidiano, notícias sobre o curso de cunho pessoal e até histórias de cada aluno. O tema é livre e se relaciona com o usuário de maneira informal.

curso; Esta página é dedicada à história do curso e seu objetivo perante os alunos e a instituição. Nesta página são apresentadas informações relevantes ao curso.

professores; Nesta seção, é possível ver a listagem de professores do curso, suas informações pessoais mais importantes e seu currículo *Lattes*.

turmas; Esta página é dedicada às turmas do curso, administradas pelo professor. Aqui constam os dados mais relevantes ao horário das aulas e seu conteúdo.

localização; Nesta página, é informado a localização do curso e dados relevantes à instituição.

contato; Esta página é dedicada para a forma de contato com o curso, através de *e-mail* e telefone.

login; Nesta página, é possível efetuar *Login* no sistema CMS, também, é possível registrar-se no sistema, caso seja aluno novo.

Sob a diretriz de que o usuário visualizará a página *Web*, através de uma página inicial, vendo todos os elementos listados, ele pode interagir com o aplicativo e ter acesso às páginas individualmente, sem entrar, inicialmente, na página específica de cada seção.

5.3.5 Documentação dos Casos de Uso

O website é composto por muitas páginas. Tais páginas são idênticas em termos de estruturação de conteúdo, ou seja, a página de notícias será igual à página do curso, ocorrendo assim, apenas variação de conteúdo entre elas. Então, fez-se necessário a documentação, apenas, de Casos de Uso chave. Páginas de administração do *software* de CMS *Wordpress*, também, não foram documentadas, tendo em vista que tal *software* é externo ao domínio desta modelagem.

5.3.5.1 Documentação do Caso de Uso - Página Inicial

Nome: Página Inicial

Descrição: os usuários acessam o *website*.

Ator primário: Usuário, Professor, Visitante ou Administrador.

Fluxo Principal:

1. o Usuário acesso o *website* através de seu endereço;
2. página inicial do *website* é mostrada na tela.

5.3.5.2 Documentação do Caso de Uso - Página de Notícias

Nome: Página de Notícias

Descrição: os usuários acessam a sub-página de notícias.

Ator primário: Usuário, Professor, Visitante ou Administrador.

Fluxo Principal:

1. o Usuário acesso a sub-página Notícias através do Botão no *Menu*;
2. sub-página é lida pelo navegador, mostrando a relação de notícias

Fluxo Alternativo;

1. o Usuário clica em voltar, exibindo a página inicial.

5.3.6 Telas do Protótipo

Segundo Guedes (2009, p.24):

> A Prototipação é uma técnica bastante popular e de fácil aplicação. Essa técnica consiste em desenvolver rapidamente um "rascunho" do que seria o sistema de informação quando ele estivesse finalizado.[..] A utilização de um protótipo pode, assim, evitar que, após meses ou, até, anos de desenvolvimento, descubra-se, ao implantar o sistema, que o software não atende completamente às necessidades do cliente devido, sobretudo, as falhas de comunicação durante as entrevistas iniciais.

O *Website* que serve de exemplo à construção refere-se a Especialização em Engenharia de Projetos de Software da Universidade do Sul de Santa Catarina, como apresentado na Figura a seguir.

Figura 10 - Protótipo da Interface Web.
Fonte: Autores do Projeto.

A figura 11, a seguir, é o exemplo de interface *Flash*, criada com o uso da ferramenta de edição gráfica *Adobe Photoshop CS5* e editada graficamente na ferramenta de desenvolvimento *Adobe Flash Catalyst*, alvo de nosso estudo, para aplicar animações e funcionalidades aos elementos gráficos.

Figura 11 - Interface em Flash.
Fonte: Autores do Projeto.

Como ressaltado por Guedes (2009, p25):

> Um protótipo pode induzir o cliente a acreditar que o software encontra-se em um estágio bastante avançado de desenvolvimento. Com frequência ocorre de o cliente não compreender o conceito de um protótipo. Para ele, o esboço apresentado já é o próprio sistema praticamente acabado.

Porém, vale destacar uma particularidade do sistema que este trabalho emprega. A tela de protótipo, visto acima na figura 11, foi construída a partir da ferramenta de desenvolvimento gráfico *Adobe Photoshop CS5*. Pela característica da Ferramenta de estudo, o *Adobe Flash Catalyst* aproveita a tela gerada, nessa ferramenta de edição, para criar a codificação da página e dar funcionalidade e animação à mesma. Todos os elementos criados fazem parte do processo de desenvolvimento do próprio *Website*, o que acaba por poupar tempo de desenvolvimento ao criar a interface *Web*.

Outra característica da ferramenta é a possibilidade, em estágios avançados da construção da interface, que possibilita fazer ajustes gráficos na ferramenta gráfica de quaisquer elementos construídos, possibilitando uma maior flexibilidade quanto a mudanças, seja por correção de erros ou mudanças cosméticas, sem a necessidade de recomeçar o projeto do início.

5.4 CONSIDERAÇÕES FINAIS

Por tratar-se de um trabalho de avaliação de uma Ferramenta de desenvolvimento, a modelagem do sistema foi limitada aos elementos de Requisitos Funcionais, Não-Funcionais e Casos de Uso necessários à construção do *Website*.

O *design* para a construção do exemplo de interface não foi considerado como primordial, sendo vista apenas como útil para a avaliação da ferramenta, baseado apenas na experiência pessoal dos autores na construção de páginas *Web*. Portanto, a ferramenta é avaliada de forma empírica, levando em consideração elementos característicos da tecnologia empregada e questões levantadas na metodologia deste trabalho.

Para a construção do *Website*, os autores aprenderam a utilizar a ferramenta. A duração do aprendizado, o tempo gasto na implementação do protótipo e o término da construção do *Website* podem ser vistos adiante nas seções desta Monografia. Porém, vale ressaltar que o *Website*, em questão, pode sofrer alterações até a apresentação deste trabalho, tornando as figuras dos tópicos, obsoletas.

No próximo capítulo, abordamos a construção do Aplicativo com o uso da ferramenta, a implementação do *Website* e sua tecnologia. Também, concluímos o trabalho, analisando a ferramenta e avaliando seu desempenho de acordo com os tópicos levantados, no Capítulo 3 deste trabalho, levando em consideração as hipóteses.

6 DESENVOLVIMENTO DO WEBSITE

Neste capítulo, abordamos passo-a-passo o desenvolvimento do aplicativo para a construção do *Website* que serve de exemplo para a aplicação da ferramenta *Adobe Flash Catalyst*. O aplicativo criado, a partir do uso da ferramenta, é anexado ao *Website* e, desta forma, são levadas em consideração nossas hipóteses levantadas para a avaliação da ferramenta.

6.1 DESENVOLVIMENTO A PARTIR DO PROTÓTIPO

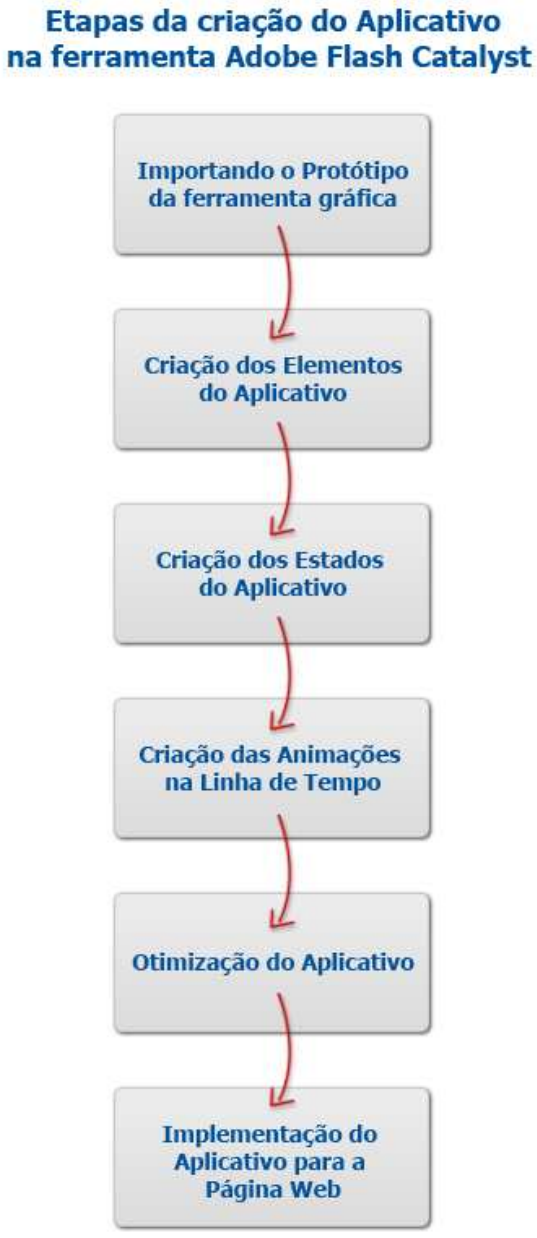
Como visto no capítulo anterior, a partir do protótipo do *Website*, o profissional tem plenas condições para desenvolver o aplicativo final dentro da ferramenta de desenvolvimento. O objetivo da ferramenta de edição gráfica foi a criação do protótipo, porém deve-se levar em consideração os seguintes aspectos listados abaixo:

- para tornar viável o trabalho na ferramenta, foi necessária a criação de camadas, ou *layers*, para cada elemento necessário na criação do aplicativo. *Ex:* Para a criação de um botão interativo, foi necessário dedicar uma camada específica para tal execução, de forma que o trabalho na ferramenta *Adobe Flash Catalyst* pudesse interpretar tal camada como elemento funcional;
- fez-se necessário o uso de regras básicas na criação de uma interface. *Ex:* Para a criação de páginas e estados específicos, esses, também, foram separados em camadas específicas, para posterior aplicação na ferramenta de desenvolvimento;
- outro aspecto levado em consideração foi a temática necessária para o desenvolvimento do *Website*. Durante a criação do protótipo, foram levadas em consideração as Cores, Temática e Austeridade do *Website* da Universidade do Sul de Santa Catarina.

Pequenas alterações, seja por erro de projeto ou mudança repentina de última hora puderam ser feitas sem precisar começar o projeto do início. A ferramenta tornou possível a alteração de elementos de uma forma unidirecional, podendo alterar elementos na ferramenta de edição gráfica com a ajuda de um *plug-in* fornecido pela *Adobe Systems* e posterior alteração na ferramenta de desenvolvimento, poupando tempo na criação do aplicativo. O uso do *Plug-in* se fez necessário somente na ferramenta gráfica *Adobe Photoshop*, sendo a

ferramenta *Adobe Illustrator* apenas necessária para a edição de conteúdo textual do aplicativo.

As etapas do processo de criação podem ser vistas no diagrama:



Fluxograma 3 - Processo de criação do aplicativo.
Fonte: Autores do Projeto.

6.1.1 Importando o protótipo para a ferramenta

Após a criação e finalização do protótipo e suas camadas, aberto a ferramenta de desenvolvimento *Adobe Flash Catalyst* e importado o arquivo no formato .PSD (formato adotado pela ferramenta de edição gráfica), a interface está pronta para edição de seus elementos, como se vê na figura abaixo. O mesmo pode ser feito para a ferramenta de edição e criação *Adobe Illustrator* e, também, para projetos criados, previamente, na ferramenta *Flash Catalyst*, no formato .FXG.

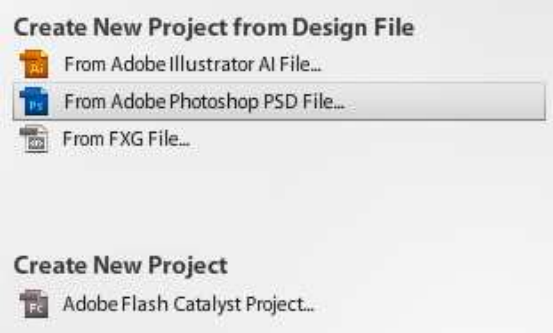


Figura 12 - Importando o protótipo.
Fonte: Autores do Projeto.

A ferramenta irá, automaticamente, importar camadas ocultas no protótipo. Esse passo é importante, pois nem todas as camadas estarão visíveis nesse processo, porém serão utilizadas, posteriormente, na criação dos estados ou páginas do aplicativo.

A interface da ferramenta é básica, contendo quase todos seus elementos listados na tela e posicionados de forma que possam facilitar o processo de edição e criação. Alguns elementos são característicos das ferramentas da Adobe, como o posicionamento da lista de Camadas. Outros elementos são paliativos e foram reposicionados, como a barra de ferramentas, bem mais limitada, se comparada a outras ferramentas.

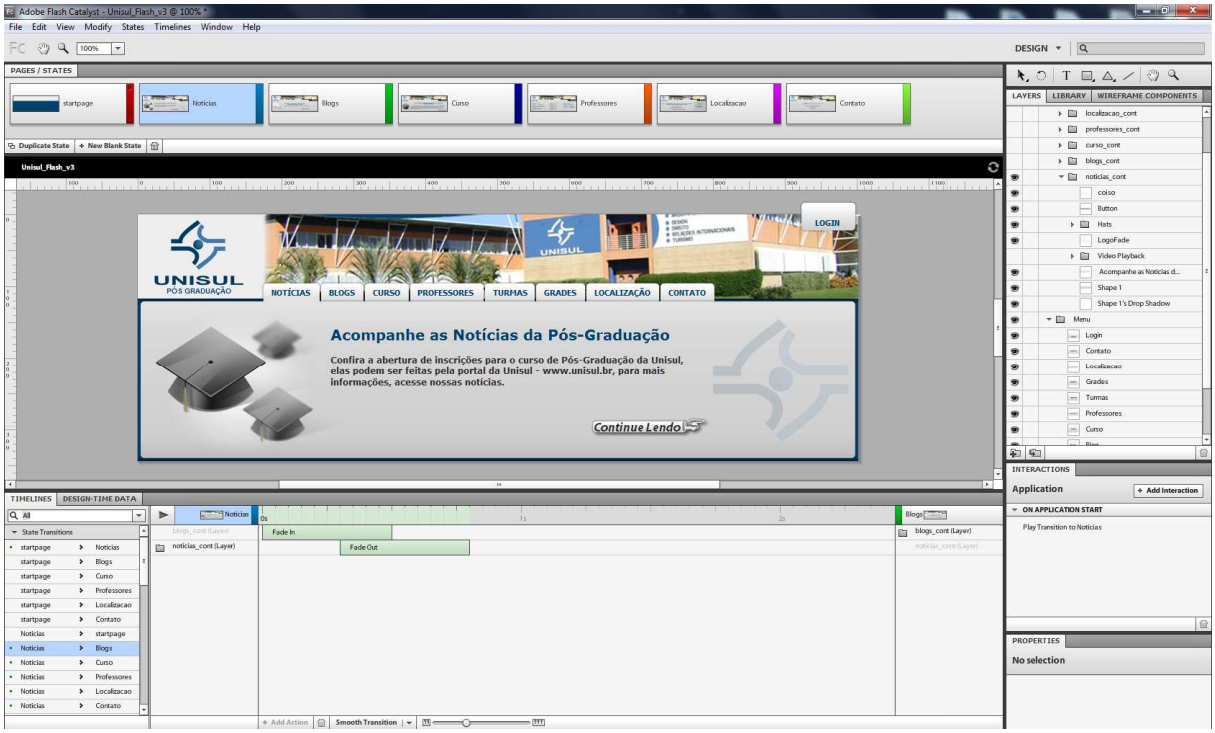


Figura 13 - Interface da Ferramenta.
Fonte: Autores do Projeto.

Ao importar o protótipo na ferramenta, este torna-se disponível para o começo do trabalho. No caso do aplicativo criado para o *Website*, nesta etapa do processo, teve início a criação dos Elementos e posteriormente os Estados (States), ou simplesmente as páginas do aplicativo, localizado no canto superior da ferramenta. Dessa forma, a ferramenta toma conhecimento das páginas que o *Webdesigner* tem interesse em criar. Neste projeto, foram seis páginas/estados criados e, no início da aplicação, uma página em branco, para a criação da animação inicial do aplicativo, visto nos tópicos subsequentes.

6.1.2 Criação dos elementos do aplicativo

Botões, elementos interativos e elementos animados, puderam ser definidos de uma forma simples, através de Componentes pré-existentes e apenas com o clique do mouse sobre determinada camada.

Como exemplo, a camada "Contato", a atribuição a esta camada como sendo um elemento interativo Botão, ou *Button*, como referenciado pela ferramenta na língua inglesa, deu-se de forma que o elemento, ao clique do mouse, acionasse determinado Estado/Página criada. Dessa forma, a ferramenta reconhece o cursor do mouse, aciona determinada animação

e, ao clique do dispositivo, aciona o estado a ser mostrado e sua consequente animação. Como se vê na imagem, a seguir, o processo é simples:

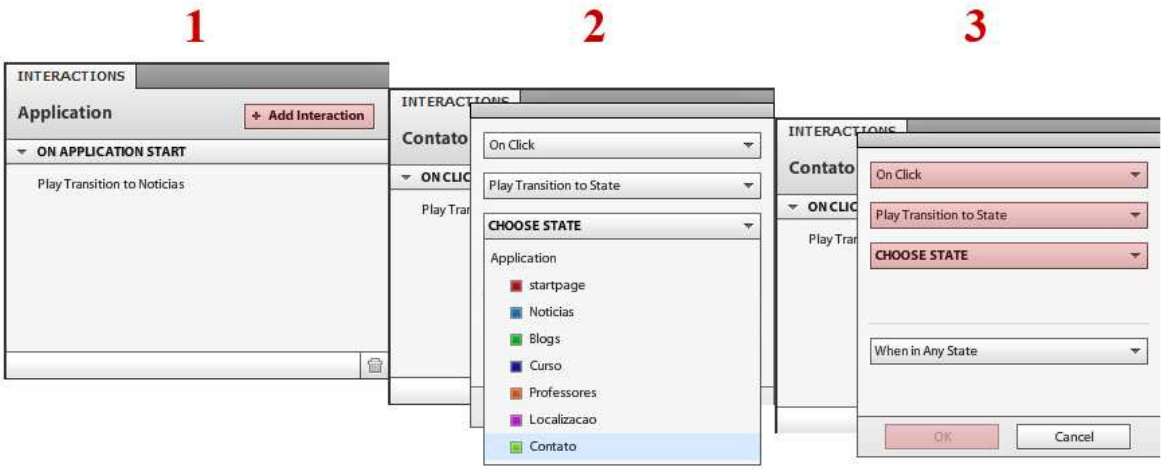


Figura 14 - Criação da interação de um botão para a transição de estado.
Fonte: Autores do Projeto.

Portanto, qualquer elemento gráfico, desde que em sua devida camada, pode conter uma gama de elementos referenciados, desde simples botões interativos até barras de rolagem vertical, rolagem horizontal, *checkboxes*, *sliders*, entradas de texto, entradas em *Uniform Resource Locator* (URL), etc. Cada elemento dessa lista de componentes pode ter interação e animação anexados, sendo do gosto do desenvolvedor aplicá-los, como se vê abaixo:

Figura 15 - Componentes.
Fonte: Autores do Projeto.

Esses elementos podem ser utilizados diretamente no gráfico, aproveitando a arte gráfica criada e a intuição do desenvolvedor. Dessa forma, o aplicativo torna-se completamente customizado do ponto de vista gráfico, artístico e funcional.

6.1.3 Criação dos estados do aplicativo

A criação dos estados foi feita de forma que o *Menu* principal de navegação do exemplo de *Website* contivesse páginas virtuais de pré-visualização de conteúdo, ou seja, ao clicar do mouse, é exibido uma introdução à área específica, como espécie de informativo ao usuário. O processo se fez necessário para a criação de uma página mais rica em conteúdo e *feedback* das seções empregadas.

Outro aspecto levado em consideração foi a complexidade do aplicativo, ele não poderia ser muito simples sob o risco de não utilizarmos a ferramenta em toda a sua capacidade e, também, não poderia ser demasiadamente complexo, sob o risco de comprometer a visualização rápida da informação e peso em *kilobites* do *Website*. Vale destacar novamente que o objetivo do aplicativo é a avaliação da ferramenta, portanto, a tentativa de anexar o máximo possível de funcionalidades.

Exemplo de lista de estados/páginas criadas abaixo:

Figura 16 - Estados ou páginas do aplicativo.
Fonte: Autores do Projeto

Dessa forma, cada estado possui sua determinada camada ativa na lista de camadas, previamente montada na ferramenta gráfica. Todo o seu conteúdo foi previamente criado e imaginado na ferramenta gráfica, sendo o uso da ferramenta de desenvolvimento do aplicativo apenas uma ponte entre o gráfico estático e a animação, para posterior aplicação na *Web*.

Neste trabalho foram criados seis estados e um estado inicial para montagem da animação inicial do aplicativo. Dessa forma foi possível animar a entrada do aplicativo,

escondendo camadas iniciais e mostrando-as na primeira página padronizada. O elemento inicial, aqui, é marcado com um ponto na aba gráfica, como se vê na figura 16. A criação dos estados foram possíveis apenas com o clique no botão "Duplicate State", visto na Figura 16, e posterior escolha das camadas que seriam visíveis em cada estado, ou página, do aplicativo. O mesmo poderia ser feito com o botão "New Blank State" para a criação de estados completamente novos. A deleção de estados foi possível através do ícone de lixeira, localizado no rodapé de sua interface, como se vê também na figura 16.

6.1.4 Criação das animações na linha de tempo

Nesse processo, entra o ponto crucial na utilização da ferramenta, a animação dos elementos criados. Aqui é feita a animação dos componentes que interagem com o ponteiro do mouse e, também, as transições animadas entre os Estados/Páginas do aplicativo. Dada a seriedade do exemplo de página *Web* criada, foi pensado em animações de bom gosto, que apenas relatassem *feedback* da interação do usuário e mudança suave de páginas do aplicativo, no entanto, é possível, dado a criatividade do profissional, elevar o nível de efeitos gráficos e animações para atingir determinado objetivo. Visto o uso atual da tecnologia *Flash*, esse tipo de efeito criado no aplicativo só pode ser criado nesse tipo de ferramenta, em que a interface rica é o foco principal.

O manuseio da linha de tempo é pensada de forma que o efeito é aplicado entre estados específicos. Ex: Transição da página Notícias para a Página Blog. E a Transição da posição de um botão ao acionamento do cursor do mouse.

Como visto a seguir, a animação é criada no momento em que o aplicativo é lido pela primeira vez na página *Web*, *Startpage* para o estado Notícias.

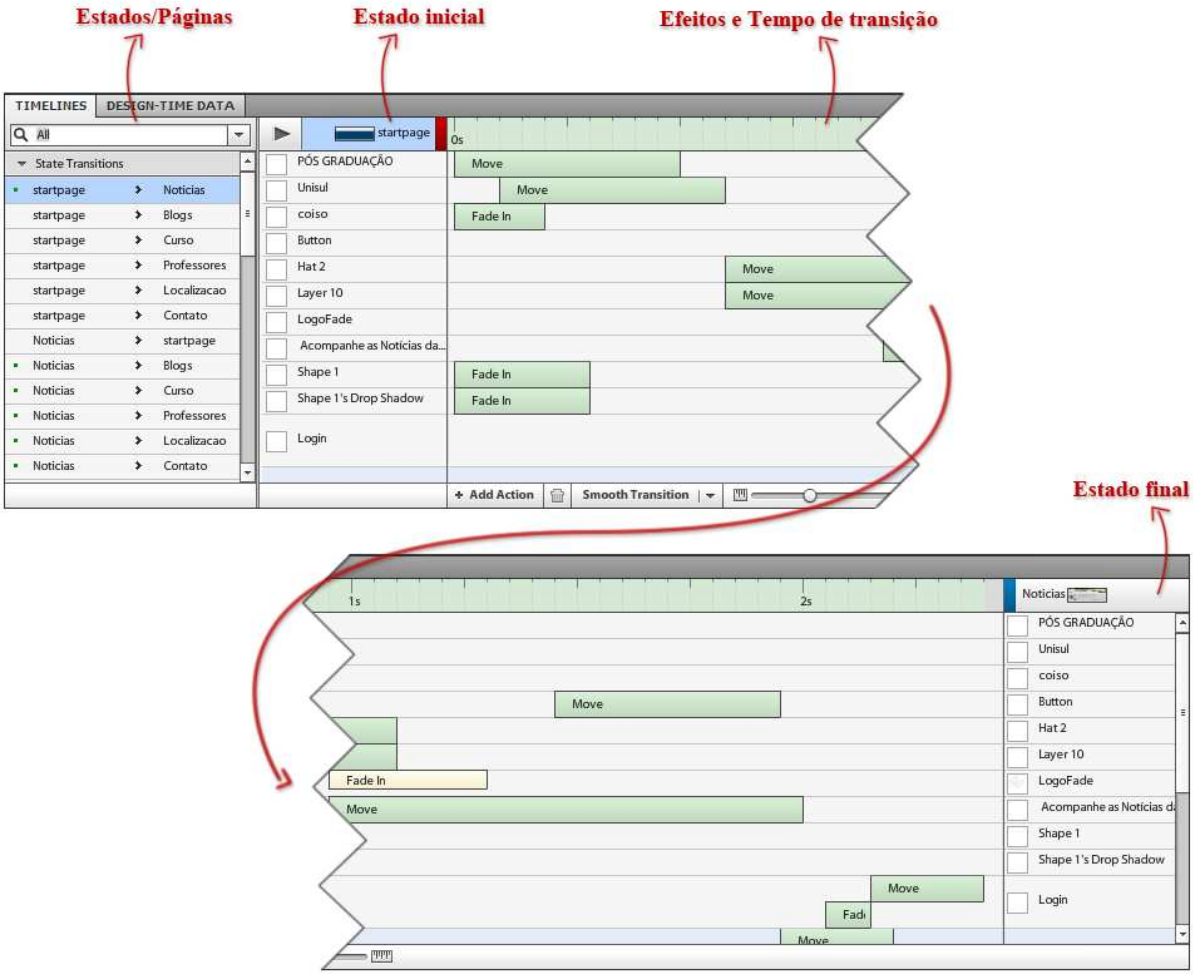


Figura 17 - Linha do Tempo para aplicação da animação
Fonte: Autores do Projeto

Como se vê na figura acima, o Estado inicial e Estado final são importantes para o efeito final, portanto, deve ser levado em consideração a criatividade na aplicação desses efeitos e animações. Cabe ao profissional escolher o melhor método ou lógica para aplicação de efeitos e seu objetivo principal ao mostrá-los. Nosso intuito, nesta aplicação, foi fazer uso discreto de animações, de forma sutil e minimalista, para que o *layout* final não viciasse a interação com o usuário e se tornasse o centro das atenções.

Portanto, esse é o último processo da criação do aplicativo, em todas as etapas do processo de criação é permitido e ideal que se faça a pré-visualização do aplicativo, para se ter idéia de como o aplicativo adquire forma e de como o resultado fica de acordo com a intenção do desenvolvedor.

6.1.5 Otimização e Codificação

Além desse processo, é feita a otimização das Camadas, aquelas que não foram utilizadas podem ser apagadas do projeto, algumas demasiadamente pesadas, podem ser editadas e otimizadas na ferramenta gráfica. Também, vale destacar a necessidade de deleção das artes não utilizadas na biblioteca da ferramenta, pois nela ainda constam os gráficos referenciados no código, como se vê na figura 18:

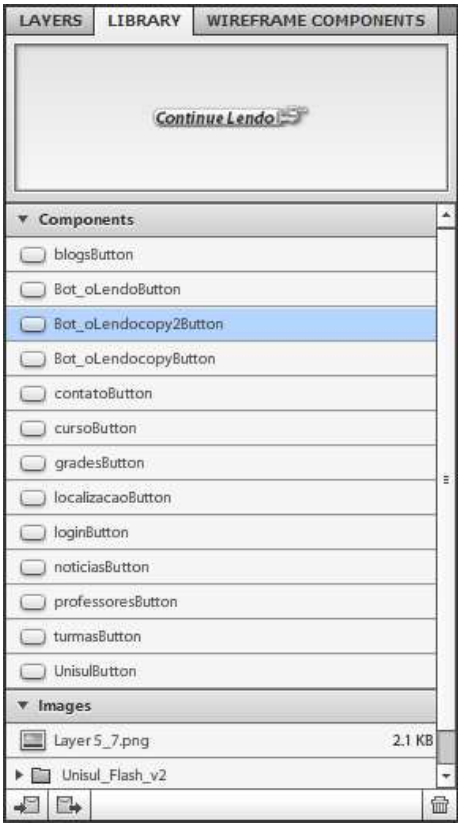


Figura 18 - Biblioteca de componentes e imagens.
Fonte: Autores do Projeto.

O código de programação, por sua vez, foi sendo construído pela ferramenta ao longo das etapas. Ele segue a lógica de construção por etapas, portanto, não sofre alterações significativas caso o desenvolvedor prefira iniciar o processo pela edição de Estados/Páginas do projeto. Para essa aplicação e posterior avaliação da ferramenta, o código gerado não foi modificado, sendo o resultado final exatamente aquilo que a ferramenta programou automaticamente. Como se vê abaixo, o aplicativo finalizado gerou 695 linhas de código e o executável do aplicativo com o tamanho em disco de 453kb, sendo ainda possível sua otimização.

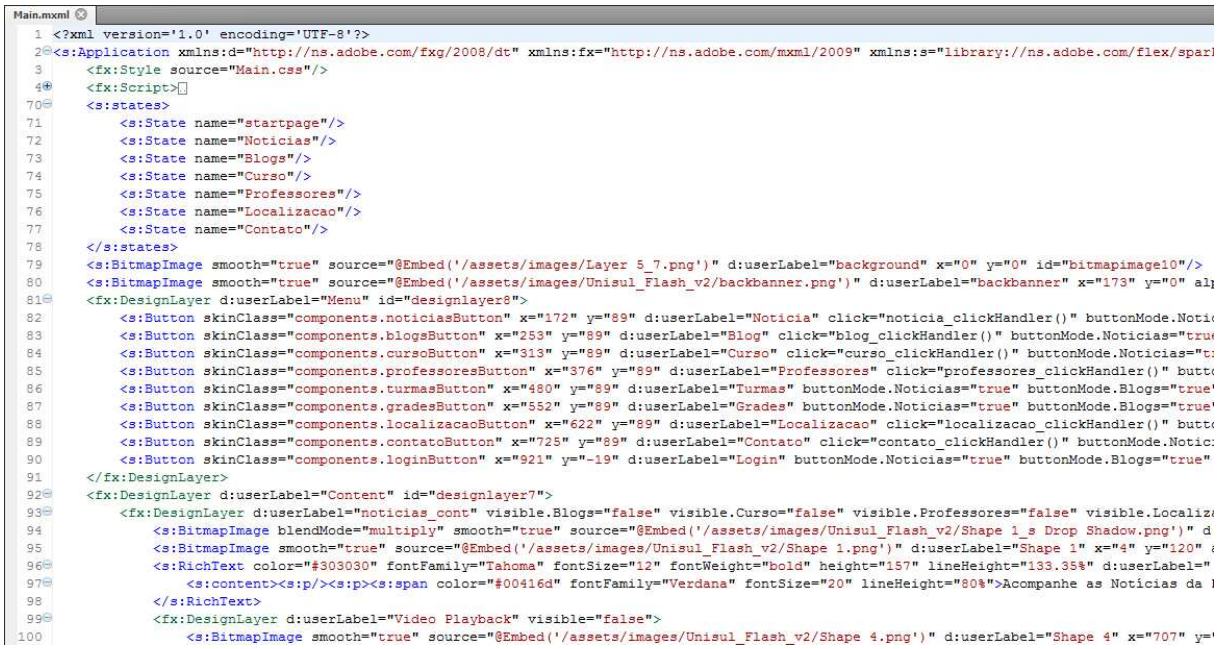


Figura 19 - Código gerado pela ferramenta.
Fonte: Autores do Projeto.

O tamanho final do aplicativo pode variar de acordo com o tamanho e ambição do projeto, dependendo muito do profissional que for utilizar a ferramenta.

Para evitar qualquer erro do projeto, é recomendável que o profissional salve seu trabalho regularmente ao longo do processo, com nomes diferentes para cada arquivo, e dessa forma é fácil voltar atrás caso algum processo seja falho ou se perca por algum motivo.

6.1.6 Implementação do aplicativo

Após esses passos para a construção do aplicativo em *Flash*, deu-se a implementação do mesmo no sistema de CMS *Wordpress*. Após colocar os arquivos gerados pela ferramenta no servidor, o aplicativo é, então, anexado ao cabeçalho da página, tornando o website misto em sua tecnologia. Para a implementação, porém, a ferramenta proporcionou os arquivos para o formato .SWF, ou seja, o projeto em *Flash* pode ser anexado ao *Host*, servidor ao qual são guardadas as pastas e arquivos necessários para a execução do *Website*, em uma pasta própria para a visualização do conteúdo. Qualquer navegador *Web*, atualizado e compatível com a tecnologia *Flash*, pode visualizar seu conteúdo. Aqui incluem-se navegadores para todos os sistemas operacionais mais populares usados na atualidade.

Como se vê a seguir na Figura 20, é possível publicar o aplicativo em 3 formatos distintos para simples visualização e aplicação.

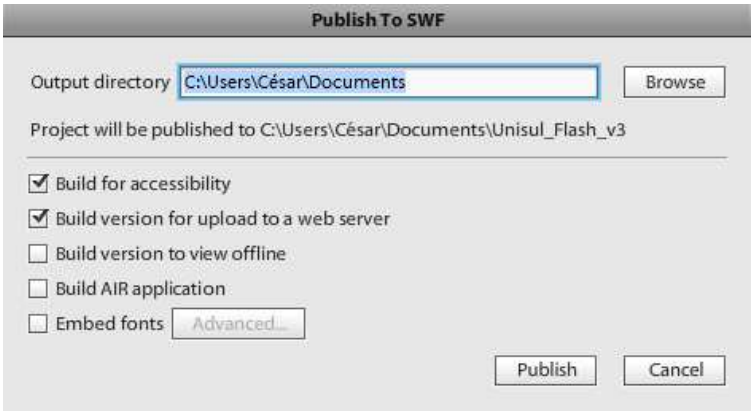


Figura 20 - Versões do aplicativo para publicação
Fonte: Autores do Projeto

Portanto, é possível, com o projeto pronto, publicá-lo em três versões distintas. A primeira opção, "*upload to a web server*", permite anexar o aplicativo a uma página *Web*. A segunda opção permite visualizar em modo "*offline*", e, por último, pode ser publicado para o formato AIR, em que não é necessário um Navegador *Web* para visualizar o aplicativo, esse sendo mais útil para apenas a amostragem do aplicativo para possíveis clientes ou apresentações do aplicativo quando não há conexão ou navegador disponível.

6.2 IMPLEMENTAÇÃO DO EXEMPLO DE PÁGINA WEB - WORDPRESS

Com a instalação do *Wordpress* em um servidor *Web*, tem-se uma base para começar o desenvolvimento do *Website*. O *wordpress* já possui um tema (*layout*) padrão que pode ser usado e modificado de acordo com as necessidades do projeto. Neste projeto, apenas os aspectos visuais do tema foram modificados, de modo que não houve a necessidade de programação da estrutura do website desde seu princípio, pois a mesma já está presente no tema inicial.

A estrutura, aparência e filosofia do *Website* foram concebidos de acordo com o objetivo de mesclar as duas tecnologias, no site foi anexado o aplicativo criado na ferramenta alvo do estudo, como mostrado na figura 21 a seguir:

Figura 21 - Tecnologias do Website.
Fonte: Autores do Projeto.

6.2.1 Página Inicial - Index

Na página inicial do *Website*, ou *Index*, é onde está situado o objeto de estudo da monografia, a aplicação desenvolvida na ferramenta *Adobe Flash Catalyst*, que pode ser vista como um grande retângulo onde estão presentes o menu de navegação e as abas de informação. O aplicativo pode ser identificado facilmente, pois é o único elemento no *website* que possui algum tipo de animação. A aplicação está inserida dentro de uma página de tema (estilo) que é gerenciada pelo *Wordpress*, onde é possível ter várias temáticas diferentes dentro de uma mesma instalação, sendo que novos temas podem até mesmo ser derivados, possibilitando a implementação de modificações ou correções de *bugs* sem que o usuário veja o trabalho em andamento.

Nessa página inicial, optou-se por colocar a aplicação *Flash* em sua forma completa, sendo que cada sub-página mostrada dentro da aplicação possui uma explicação de seu conteúdo e a relação que o usuário tem com a mesma, para que a navegação seja mais tranquila, e o usuário tenha certeza de seu papel dentro do *Website*. Por exemplo, na aba

Blogs, escrevemos que Professores e Alunos podem trocar experiências e histórias a respeito de seus cursos, enfatizando, assim, que os usuários terão condições de expor suas opiniões e, até mesmo, publicar novas informações. Optamos por seguir essa linha, para evitar que usuários descobrissem seu papel dentro do *website* por tentativa e erro.

6.2.1.1 Páginas Secundárias Dinâmicas

Sempre que um *website* é criado, existe o problema das páginas estáticas, em que o cliente fica impedido de modificar o conteúdo da página por não ter conhecimento de como fazer. Com o uso da ferramenta de CMS, o cliente passa a gerenciar as páginas secundárias através do *Wordpress*, tornando o conteúdo dessas páginas editável. Então, as páginas Curso, Professores, Grades e Turmas passam a ser constantemente atualizadas sem a necessidade de suporte por parte do desenvolvedor.

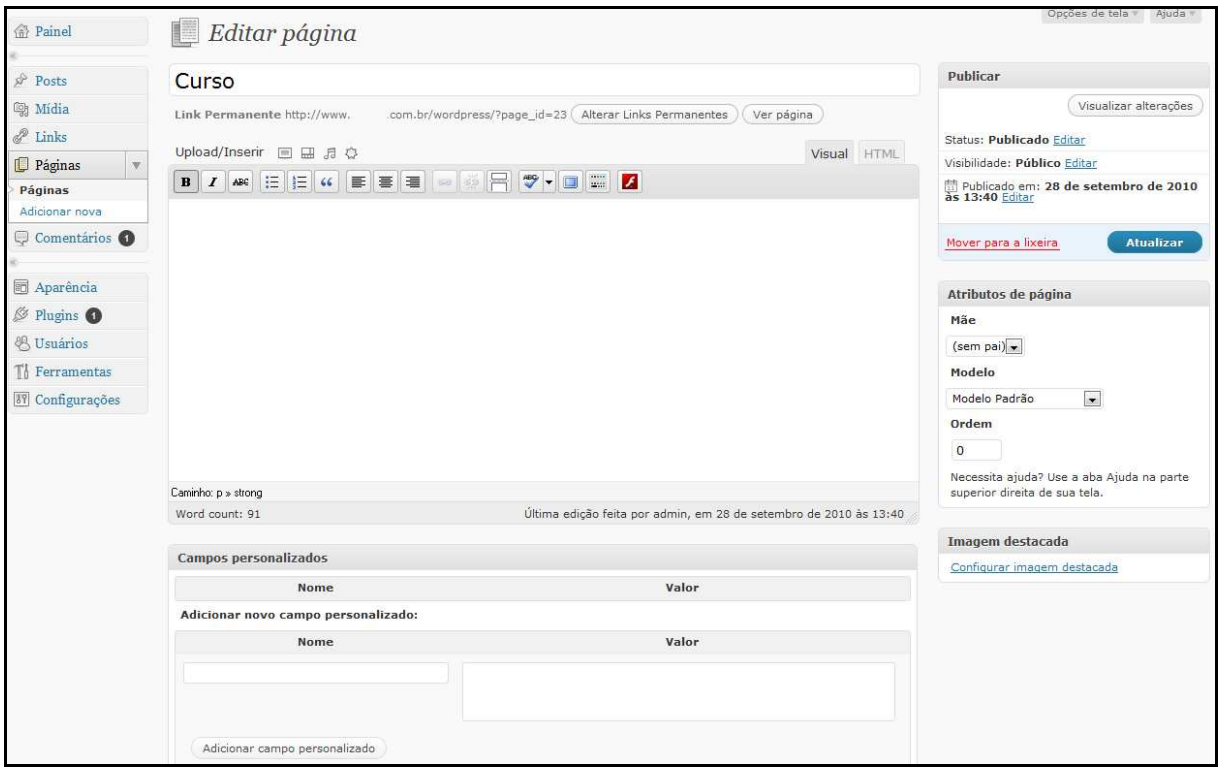


Figura 22 - Tela de edição de Páginas.
Fonte: Autores do Projeto.

Nas Páginas Secundárias, não é mais necessário explicar para o usuário qual a relação dele com a seção. Por motivos de organização, optamos por simplificar o cabeçalho das

páginas, de modo que o usuário precisa então voltar para a página inicial (*index*) para visualizar novamente informações pertinentes às sub-seções ou outras informações.

6.2.1.2 Organização da Interface no Wordpress

O *Wordpress* tem como foco a criação de *Blogs*, e *blogs*, em sua maioria, têm como característica a exposição casual de informações, de modo que na maioria das vezes o visual da página pode ser um pouco caótico, pois o conteúdo é exibido diretamente na página inicial.

Optou-se, então, por organizar melhor a maneira de como os tópicos de notícias são exibidos para o usuário, replicando o estilo das páginas de arquivos, em que o título dos tópicos é seguido por uma breve imagem e um breve texto sobre o assunto abordado, seguido então de um *link* (Leia-mais) para só, então, ser exibido o conteúdo completo do tópico em uma página secundária.

Figura 23 - Exemplo de Tópico.
Fonte: Autores do Projeto.

6.2.1.3 Permissões de Acesso

A filosofia da página de Pós-Graduação da Unisul é a integração de Professores e Alunos, para tal, o registro, no *website*, está aberto para qualquer pesssoa. Cabe ao Administrador, então, organizar a hierarquia dos usuários dentro do *website*. A hierarquia em páginas do *Wordpress* é a seguinte: Administrador, Editor, Autor e Assinante.

Como Administrador, o usuário está apto a modificar o *Website* completamente, inclusive editar e modificar a aparência.

Como Editor, seu papel passa a ser limitado, com opções como gerência de usuário e da estrutura do website sendo exclusivas ao Administrador.

Como Autor, o usuário passa a ser um Colaborador avançado, com permissão para a adição de conteúdo sem supervisão de um Editor ou Administrador, já, como Colaborador, o usuário não tem permissão suficiente para adicionar conteúdo. As informações precisam

passar pela supervisão de um usuário com nível hierárquico superior, de modo que todo o conteúdo do *website* passa a ser revisado e aprovado antes de ser exibido ao público.

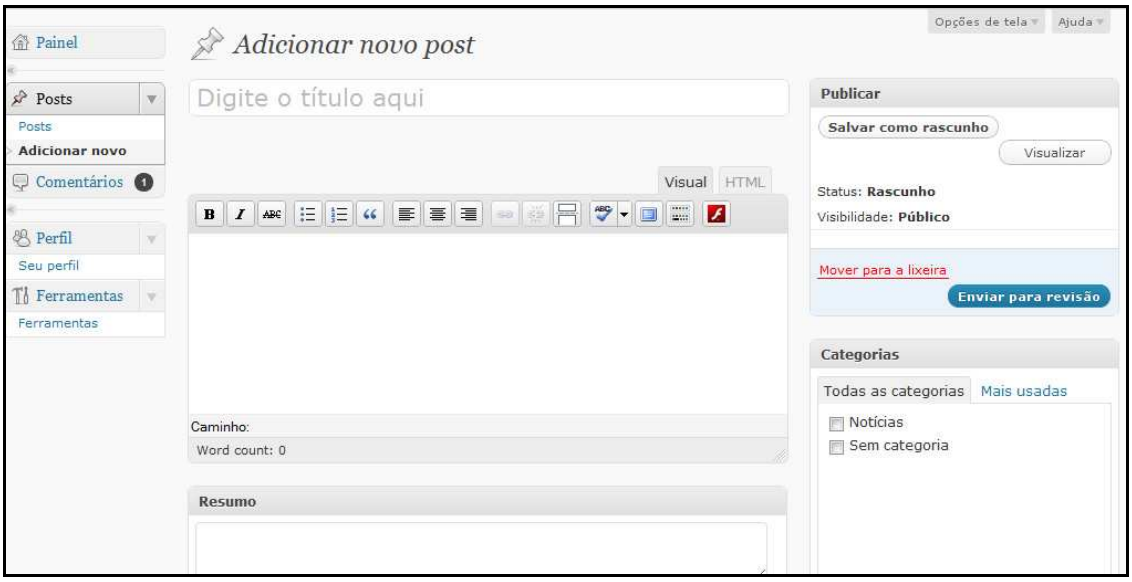


Figura 24 - Adição de Conteúdo dos Colaboradores.
Fonte: Autores do Projeto.

Por último, os usuários Assinantes têm permissão, apenas, para comentar os tópicos adicionados ao *Website*, de modo que cabe ao Administrador definir se usuários não registrados poderão ou não participar das discussões.

6.3 APRENDIZADO E AVALIAÇÃO DA FERRAMENTA

O aprendizado da ferramenta aconteceu através da experiência dos autores com as ferramentas de desenvolvimento da Adobe, entre elas o *Adobe Photoshop CS5* e *Adobe Illustrator CS5*. Também, foi levado em condideração suas experiências prévias com aplicativos *Flash*, suas experiências com sistemas CMS e práticas de *Webdesign* e *Webdevelopment*, ou Desenvolvimento para a *Web*. Os autores fizeram uso de vídeo aulas sobre a ferramenta, os quais podem ser acessados através do *website lynda.com*, como consta na bibliografia deste trabalho.

Ambos os autores possuem experiência contínua com a área de atuação da ferramenta, e suas opiniões e considerações encaixam-se nos quesitos necessários para avaliar a ferramenta deste capítulo.

6.3.1 Tempo de Aprendizado

Levando em consideração a experiência necessária para operar a ferramenta alvo do estudo, é possível estipular um tempo médio necessário para o aprendizado completo da mesma. Tendo em vista que a ferramenta é mais completa se usada em conjunto com ferramentas gráficas, então, é justo afirmar que o profissional que a utilizar tem por pré-requisito saber aplicá-las corretamente, mesmo que de forma parcial ou apenas suficiente para gerar um protótipo.

Com base no tempo de aprendizado dessa ferramenta para a criação do exemplo de interface, verificamos os seguintes dados para a realização do trabalho:

- tempo de experiência em ferramentas de edição gráfica: 3 anos;
(Opcional, visto que seria suficiente conhecimento apenas para a criação de um protótipo estático.)

- tempo de experiência na ferramenta de desenvolvimento *Adobe Flash Catalyst*: 3 meses; (aqui inclue-se tempo de estudo da ferramenta e trabalho na ferramenta para a criação do Aplicativo finalizado.)

- tempo médio necessário para aprender a operar a ferramenta de Estudo *Adobe Flash Catalyst* em suas funções básicas, com base em vídeo aula: 10 Horas;

- tempo médio na construção do Aplicativo final: 3 Horas.

Os dados acima são aproximados, sendo que, para a criação do aplicativo final, o autor também passou pelo processo de aprendizado da ferramenta, fazendo uso de suas funções ao longo de 10 horas em média de estudo. Vale destacar sua experiência prévia na construção de protótipos na ferramenta gráfica, o que poupou tempo na implementação da ferramenta de estudo.

6.3.2 Análise da Ferramenta

Para a análise e posterior avaliação da ferramenta, levamos em consideração quatro variáveis principais, como estipulados no capítulo 3. São eles, Tempo de aprendizagem, Tempo de uso gasto na construção da página *Web*, Qualidade da implantação e, por fim, a Eficiência da ferramenta, como apresentado nas seções a seguir.

6.3.2.1 Tempo de aprendizagem

Como visto na seção anterior, o tempo médio para aprender a utilizar a ferramenta *Adobe Flash Catalyst* pode ser considerada baixo, sendo necessárias apenas alguma horas de estudo e posterior prática com exemplos na ferramenta para o seu completo domínio. Experiência prévia em ferramentas de edição gráfica seria recomendável, visto que o alvo principal da ferramenta são justamente os profissionais da área de *Webdesign*.

6.3.2.2 Tempo de uso gasto na construção da página Web

Desde a criação do protótipo na ferramenta gráfica até a construção do aplicativo na ferramenta alvo do estudo e posterior aplicação da mesma na página *Web*, foi gasto aproximadamente 1 mês de trabalho. Vale ressaltar, porém, que este tempo é estimado, podendo ter sido abreviado, caso a ferramenta já fosse dominada previamente. De acordo com a experiência do autor, o mesmo projeto poderia ter sido executado em apenas alguns dias de trabalho.

6.3.2.3 Qualidade da implantação

De acordo com a experiência e vivência dos autores no meio de desenvolvimento, a qualidade final do aplicativo pode ser considerada Ótima, se levado em consideração o tempo gasto no trabalho e a facilidade de integração das ferramentas utilizadas. Certamente, o uso da ferramenta poupou tempo e o resultado final se aproximou bastante de aplicativos criados em outras ferramentas mais complexas, como o *Adobe Flash Builder*. Porém, com ressalvas para a incapacidade da ferramenta de anexar uma base de dados à sua implementação, o que torna o aplicativo de interface rica estático e dependendo de atualizações constantes do profissional para sua manutenção.

Outro ponto levado em consideração, é que o método convencional para a criação de um *Website* de interface rica tem sempre seu ponto inicial na criação da imagem estática do Aplicativo, ou Página *Web*, em um programa de edição gráfica, com excessão dos métodos que partem da programação dos aplicativos utilizando-se código de programação. Portanto, o processo de criação do *Website* na ferramenta *Flash Catalyst* é o mesmo do método convencional, exceto pelo trabalho coletivo do *Webdesigner* e *Developer* nesta tarefa.

Em trabalhos prévios, ambos tinham que trabalhar em conjunto para obter resultado

prático semelhante ao alcançado pela ferramenta *Adobe Flash Catalyst*, porém, vale ressaltar, que tal fato só se comprova caso o *Webdesigner* não tenha conhecimento ou *know-how* necessário para programar e trabalhar com código de programação. Como visto na figura abaixo, o mesmo resultado pode ser alcançado, exceto pela animação do *menu*, que não se faz presente, visto que a construção do menu de navegação não se utilizou da ferramenta *Adobe Flash Catalyst*.

Figura 25 - Menu convencional.
Fonte: Autores do Projeto.

O *menu* visto acima não faz uso da ferramenta, sendo necessário, assim, a intervenção do profissional de desenvolvimento. Este, por sua vez, torna o gráfico estático em código de programação através da ferramenta de desenvolvimento *Adobe Dreamweaver*, por exemplo, ou apenas por código de programação manual, tornando a tarefa do *Webdesigner*, nesse processo, dispensável. É neste tipo de processo, que parte da criatividade e intuição do *Designer* se perde, dando espaço para uma implementação desprovida de animações e/ou recursos visuais de *feedback* mais apurados. Também adiciona-se tempo à construção, visto que tal fato acarreta no trabalho de dois profissionais.

A figura, a seguir, exemplifica a animação adotada em um botão no projeto da interface criada em *Flash*, tal efeito é possível pelo método convencional de criação, exceto que o movimento do botão é seco, instantâneo e sem fluidez. O fato do aplicativo conter seis páginas de pré-visualização de conteúdo são possíveis, somente, na plataforma *Flash*, apresentando dificuldades de implementação por métodos convencionais em HTML.

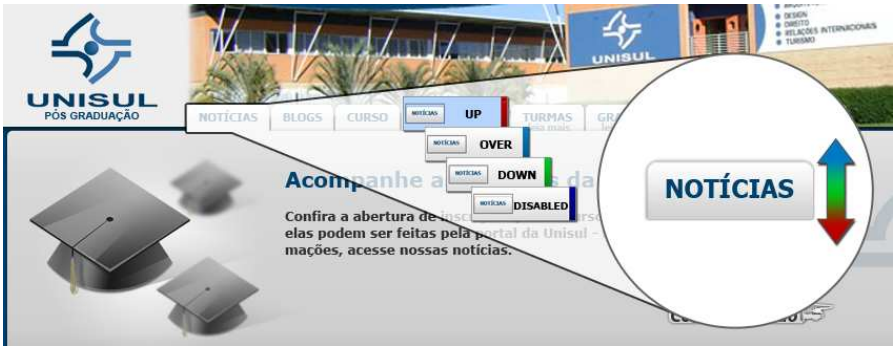


Figura 26 - Animação de um botão.
Fonte: Autores do Projeto.

O processo de criação convencional, devido à necessidade de comunicação constante entre *Designer* e *Developer*, torna o trabalho demorado e burocrático, nem sempre atingindo o objetivo final desejado.

6.3.2.4 Eficiência da ferramenta

A ferramenta demonstrou-se bastante eficiente na criação do aplicativo, uma vez dominado o processo de execução, na criação de elementos, estados/páginas e animação, a ferramenta provou que pode poupar tempo de execução, tornando o trabalho de um profissional mais simples e focado no protótipo.

6.3.3 Avaliação da Ferramenta

Os autores deste trabalho avaliam a ferramenta de desenvolvimento *Adobe Flash Catalyst* como Ótima para a criação de interfaces animadas para a *Web*. A ferramenta proporcionou a integração com as ferramentas gráficas *Adobe Photoshop* e *Adobe Illustrator*, tornando o processo de prototipagem e posterior animação facilitadas. Também, tornou o processo de edição e correção de erros gráficos em ambas as ferramentas gráficas sem risco de comprometimento do arquivo trabalhado, tornando o processo mais ágil e facilitado.

6.3.3.1 Análise dos Resultados com base nas hipóteses

As hipóteses levantadas, apresentadas no capítulo 3, agora são respondidas com base na visualização dos resultados práticos na implementação da página Web:

a ferramenta é de fácil aprendizado; a ferramenta é de fácil aprendizado, porém com ressalvas à experiência adquirida pelo profissional em ferramentas de edição gráfica, o que torna esse processo facilitado. Profissionais do meio não devem ter grandes dificuldades de aprendizado, e aqueles que ainda não dominam uma ferramenta gráfica devem priorizar seu aprendizado nessa antes de fazer uso da ferramenta de animação.

o tempo de execução, ou criação, da interface *Web* pode ser abreviada, com base na facilidade de criação da interface; a criação da interface *Web*, ou aplicativo *Web*, pode ser abreviada, uma vez que o protótipo do mesmo, advindo da ferramenta gráfica, pode ser alterado, e suas funções anexadas com a ferramenta alvo do estudo. Dessa forma, poupou-se tempo na execução da tarefa e posterior implementação do aplicativo.

a ferramenta permite uma fácil implementação; a ferramenta permite uma fácil implementação, porém vale ressaltar que com ela advém, quase que por obrigação, o uso das ferramentas gráficas da mesma família, o que pode tornar sua aplicação mais cara e demorada.

os custos do projeto podem ser abreviados em termos de "hora-homem"; os custos de projeto podem ser abreviados, uma vez que o profissional de *Webdesign* parte de seu protótipo gráfico estático para um protótipo animado e funcional apenas ao utilizar a ferramenta, sem necessidade de codificação do mesmo e podendo aplicá-lo a qualquer página ou projeto.

a interface criada pela ferramenta é de fácil manutenção; a interface criada é de fácil manutenção, podendo ser modificada posteriormente com ajuda das ferramentas gráficas. Uma ressalva, porém, é o fato da ferramenta não possibilitar a implementação de base de dados em seu projeto, esse sendo possível apenas em ferramentas mais complexas.

6.4 CONSIDERAÇÕES FINAIS

Com este capítulo, encerramos a monografia através da implementação da Página *Web* e do Aplicativo em *Flash* criado na ferramenta alvo do estudo, o *Adobe Flash Catalyst*. Os processos usados para a criação do exemplo de interface *Web* foram importantes para a análise e posterior avaliação da mesma. No capítulo seguinte, concluímos o trabalho levando em consideração os dados obtidos.

7 CONCLUSÕES E RECOMENDAÇÕES

A seguir são apresentados as conclusões e recomendações de trabalhos futuros desta monografia.

7.1 CONCLUSÕES

Conclui-se neste trabalho uma perspectiva otimista da ferramenta. Ela foi capaz de proporcionar aos autores a construção de um aplicativo com a tecnologia *Flash* e código de programação *Flex* sem qualquer necessidade de trabalho com código de programação, poupando tempo de trabalho e possibilitando melhor foco no aplicativo gráfico criado. O resultado final é um aplicativo capaz de ser demonstrado com a tecnologia *Adobe AIR* em qualquer dispositivo com sistema operacional compatível e, também, sua implementação em uma página *Web*.

A página *Web*, criada como exemplificação neste trabalho, tornou-se mais rica na amostragem de conteúdo, adicionada à praticidade do aplicativo. O usuário final pode fazer uso da página de forma mais fluída, com animações ricas e de bom gosto, exatamente de acordo com a intenção do *Webdesigner*. Somado a isso, o tempo poupado na criação, a página *Web* pode ser montada muito rapidamente e com resultados bastante satisfatórios.

No entanto, algumas ressalvas devem ser feitas: a ferramenta não possibilita a adição de banco de dados pré-existentes e, também, não possibilita o trabalho de edição do código de programação criado pela própria ferramenta, sendo essas tarefas terceirizadas por um programa mais complexo. Por sorte, o formato criado pela ferramenta *Flash Catalyst* possibilita a exportação do projeto para a ferramenta *Flash Builder*, em que essas tarefas podem ser executadas, adicionando tempo de trabalho ao projeto.

Outro fator a ser levado em consideração acerca da tecnologia *Flash* é sua evolução ao longo do tempo. A ferramenta do estudo não possui tempo de vida suficiente para ser avaliada de acordo com seu resultado dentro das comunidades e empresas de *Webdesign*, sendo, ainda, considerada nova no meio profissional e alvo de atenção por suas características e seu foco de trabalho. Evoluções da ferramenta devem ser esperadas ao longo do tempo, caso a *Adobe Systems* pretenda seguir esse nicho de mercado.

Tecnologias, como o HTML5, por exemplo, devem a partir do ano de 2010 se tornarem cada vez mais populares, e virão para rivalizar com a tecnologia *Flash* e seus aplicativos, devendo lutar pela supremacia da *Web*. Portanto, a ferramenta de

desenvolvimento, também, sofrerá com as mudanças de mercado e foco da comunidade que as aplica, sendo necessário cautela ao se dizer que determinada tecnologia ditará o mercado ou dominará por muito tempo. O futuro da tecnologia *Flash* dependerá do bom uso das ferramentas pela comunidade de desenvolvimento e, também, pela aceitação dos usuários da *Web*.

7.2 RECOMENDAÇÕES

Além do escopo deste trabalho, para o futuro, podem ser feitos estudos de impacto em empresas que aderirem à ferramenta em seu meio de trabalho, levando em consideração a produtividade de seus empregados, também, é possível, com a evolução da ferramenta ao longo dos anos, reavaliar sua eficácia no meio profissional.

Também é possível abranger mais o foco do estudo, realizando uma análise quantitativa da ferramenta, levantando novas hipóteses para estudo da mesma.

REFERÊNCIAS BIBLIOGRÁFICAS

ADOBE. **Adobe Flash Player**. Disponível em: http://www.adobe.com/br/products/flashplayer Acesso em: 31 mai 2010.

ADOBE. **Flash content reaches 99% of Internet viewers**. 2010. Disponível em: http://www.adobe.com/products/player_census/flashplayer/ Acesso em 31 mai. 2010.

BANERJIE, A.; GHOSH, A.M.. **Multimedia technologies**. Nova Deli: McGraw Hill, 2010.

BERNERS-LEE, T.. **The Future of the World Wide Web**. 2007. Disponível em: http://dig.csail.mit.edu/2007/03/01-ushouse-future-of-the-web.html .Acesso em 31 mai. 2010.

BERNERS-LEE, T.. **The WorldWideWeb browser**. Disponível em: http://www.w3.org/People/Berners-Lee/WorldWideWeb.html .Acesso em 31 mai. 2010.

BOOCH, Grady; RUMBAUGHT, James; JACOBSON, Ivar. **UML: Guia do Usuário**. Rio de Janeiro: Campus, 2006.

BULTERMAN, D. et al. **Synchronized Multimedia Integration Language (SMIL 3.0)**. W3C: World Wide Web Consortium. Dezembro 2006. Disponível em: http://www.w3.org/TR/2006/WD-SMIL3-20061220/ Acesso em 31 mai. 2010.

CLARK, R.. Internet. **JPEG: Joint Photographic Experts Group**. 2007. Disponível em: http://www.jpeg.org/apps/internet.html Acesso em 31 mai. 2010.

DINUCCI, D.. **Print: Design & New Media**. 2009. Disponível em: http://www.cdinucci.com/Darcy2/articles/Print/Printarticle7.html Acesso em 31 mai. 2010.

FERRAIOLO, J.. **Scalable Vector Graphics (SVG) 1.1 Specification**. W3C: World Wide Web Consortium. Janeiro 2003. Disponível em: http://www.w3.org/TR/SVG/ Acesso em 31 mai. 2010.

GUEDES, Gilleanes T. A.. **UML 2: Uma Abordagem Prática**. São Paulo: Novatec, 2009.

HUNT, L.. **A Preview of HTML 5**. A List Apart: A List Apart Magazine and the authors. Dezembro, 2007. Disponível em: http://www.alistapart.com/articles/previewofhtml5 Acesso em 31 mai. 2010.

HUNT, L.. **A Web Developer's Guide to HTML 5**. W3C: World Wide Web Consortium. Março 2009. Disponível em: http://dev.w3.org/html5/html-author/ Acesso em 31 mai. 2010.

Implied By Design. Disponível em: http://www.impliedbydesign.com/ Acesso em 31 mai. 2010.

JENKINS, Caleb. **Developers are from Mars and Designers are from Venus**. Agosto 2008. Disponível em: http://www.creativeedge.com/web-development/usability/00000spm3ddz02001 Acesso em: 31 mai 2010.

LILLEY, C.; SCHEPERS, D.. **Scalable Vector Graphics (SVG)**. W3C: World Wide Web Consortium. Disponível em: http://www.w3.org/Graphics/SVG/ Acesso em 31 mai. 2010.

MAYER, R. E.. **Multimedia Learning**. Nova York: Cambridge University Press, 2008.

MORONEY, L.. **Introdução ao Silverlight**. MSDN: Microsoft Development. Maio 2007. Disponível em: http://msdn.microsoft.com/pt-br/library/cc580591.aspx Acesso em 31 mai. 2010.

MILLARCH, F. **O que é CMS e por que você precisa de um**. Webinsider. 2005. Disponível em: http://webinsider.uol.com.br/2005/06/08/o-que-e-cms-e-porque-voce-precisa-de-um/. Acesso em 2010.

O'REILLY, T.. **What Is Web 2.0**. O'REILLY. Setembro 2005. Disponível em http://oreilly.com/web2/archive/what-is-web-20.html Acesso em 31 mai. 2010.

ORACLE. **Develop Expressive Content with the JavaFX Platform**. Disponível em: http://javafx.com/about/overview/index.jsp Acesso em: 29 out. 2010.

ORACLE, **Java FX**. Disponível em: http://www.oracle.com/us/products/tools/050854.html Acesso em 31 mai. 2010.

PISANI, F.; PIOTET, D.. **Como a web transforma o mundo: A Alquimia das Multidões**. São Paulo: SENAC, 2010.

ROSENBERG, Doug; STEPHENS, Matt; COLLINS-COPE, Mark. **Agile Development with ICONIX Process: People, Process, and Pragmatism**. Nova York: Apress, 2005.

SILVA, Edna Lúcia da; MENEZES, Estera Muszkat. **Metodologia da Pesquisa e Elaboração de Dissertação**. 4. ed. Florianópolis: 2005.

TAPLEY, S.. **Adobe® Flash® Catalyst™ CS5 - A Smart Choice for Web Designers**. Peachpit: Pearson Education. Abril, 2010. Disponível em: http://www.peachpit.com/articles/article.aspx?p=1583176 Acesso em 31 mai. 2010.

W3C. **A vocabulary and associated APIs for HTML and XHTML**. Disponível em: http://www.w3.org/TR/html5 .Acesso em 31 mai. 2010.

WEITZNET, D.J.. **Hearing on the "Digital Future of the United States: Part I -- The Future of the World Wide Web"**. DIG: Decentralized Information Group. 2006. Disponível em: http://dig.csail.mit.edu/2007/03/01-ushouse-future-of-the-web.html Acesso em 31 mai. 2010.

WIKIPEDIA. **Adobe Flash**. Disponível em: http://en.wikipedia.org/wiki/Adobe_Flash. Acesso em: 31 out. 2010.

WISE, Richard; STEEMERS, Jeanette. **Multimedia: A Critical Introduction**. Nova York: Routledge, 1999.

GLOSSÁRIO

Adobe Flash Catalyst: Ferramenta de design gráfico para criação de interfaces de usuários em aplicações ricas para a internet.

Adobe Flash Builder: Previamente conhecido como Flex Builder, é uma ferramenta de desenvolvimento de aplicações ricas para a internet.

Adobe Illustrator: Ferramenta de edição de gráficos e imagens vetorizadas.

Adobe Photoshop: Editor caracterizado pela sua capacidade de criar e modificar qualquer elemento gráfico, desde fotos até ilustrações. Seu funcionamento atua sobre *pixels*, não sendo capaz de reproduzir formatos vetorizados.

Adobe Fireworks: Editor de imagens bitmap(.bmp) e imagens vetorizadas.

Artwork: Tipo de imagem que serve como representação gráfica, ou artística.

ASP.NET: Framework(Ambiente de trabalho) desenvolvido pela Microsoft para aplicações na Web.

Background: Em websites, pode ser definido como o plano de fundo, ou papel de parede que fica atrás de todos os elementos.

Beta: Etapa do desenvolvimento de um sistema ou aplicação em que não está finalizada. É comumente utilizada em sistemas ou Websites incompletos ou em constante mudança.

Browser: Navegador Web, ou sistema responsável pela navegação pelas páginas da Internet através do código HMTL.

Blog: Uma mistura do termo "web log", pode ser considerado um tipo distindo de website, onde informações são adicionadas de uma maneira mais informal, visando a integração de pessoas e comentários.

Brainstorm: É um exercício individual, ou em grupo, que visa desenvolver idéias ou exercitar a criatividade.

Bug: Termo usado para expressar a falha de software de um programa específico, geralmente causada por erros lógicos no código de programação.

Checkbox: É uma caixa de seleção, que permite que os usuários selecionem múltiplas opções.

Codec: Dispositivo de hardware ou software capaz de codificar ou decodificar sinais.

Coldfusion: Aplicação comercial originalmente desenvolvida para tornar mais simples a conexão de páginas HTML a um banco de dados. Atualmente a ferramenta é distribuida pela Adobe e conta com opções mais avançadas para a criação de aplicações ricas para a internet.

CS5: CS é uma coleção de aplicativos de design gráfico, edição de vídeo e aplicações de desenvolvimento para a web, criado pela Adobe Systems.

Custom: Customização, é empregada no sentido de personalização.

Desktop: Significa superfície da mesa, geralmente é usado para definir o computador pessoal que foi desenvolvido para ficar em uma mesa. A palavra desktop também pode ser usada como analogia para uma área de trabalho.

Div: Elemento da linguagem HTML, é na verdade uma tag HTML usada para implementar elementos em bloco.

Down: Usado para expressar um elemento que está por baixo, como no caso de um botão que está sendo apertado, seu estado atual(apertado) é definido com o nome Down.

Download: Termo utilizado para espressar a descarga de dados de um computador remoto para o computador do usuário. O termo inverso do fluxo de dados, emprega-se o termo *Upload*.

Facebook: Serviço de rede social e website.

Feedback: Pode ser descrito como um evento de resposta, em decorrência de uma ação.

Flash: Software multimídia para criação de animações.

Flash Player: Programa destinado à visualização de animações e vídeos através do navegador web.

Flex: Linguagem de programação para desenvolvimento de interfaces ricas para a internet. Pode ser desenvolvido com o uso da ferramenta Adobe Flash Builder, ou compilado separadamente através do compilador Flex, disponível gratuitamente.

Flickr: Website de comunidade na Internet responsável pela hospedagem e troca de fotos.

Generic: Usado para determinar algo personalizado ou genérico a opção original.

Gmail: Aplicativo de correio eletrônico pessoal do Google.

Google: Sistema de Busca por Websites na Internet.

H.264: Padrão de compressão de vídeo.

Header: Cabeçalho da página.

Index: Página inicial.

JAVA: Linguagem de programação orientada a objetos.

Java FX: Plataforma destinada a criação de aplicações ricas na internet.

Kilobyte: Unidade de armazenamento, corresponde a 1.024 Bytes.

Layer: Camada, usado para separar elementos como se fossem folhas.

Layout: Esboço artístico dos elementos gráficos, ou visual de uma página na internet.

Link: Geralmente usado como diminutivo de Hyperlink, nome dado aos elementos textuais que contém, em sí, endereços para outras páginas. Por padrão, o elemento é sublinhado em uma página web para determinar a existência de um Hyperlink.

Login: Elemento de identificação de usuários dentro do website, geralmente usado na autenticação juntamente com a senha pessoal.

Nav: Usado para identificar elementos de navegação de um website.

Notebook: Computador pessoal portátil.

Open Source: Aplicativo de código aberto, também conhecido como software livre.

Orkut: Rede social de troca de informações pessoais do Google.

Over: Usado para expressar um elemento que está em foco, como quando o ponteiro está sobre um elemento, seu estado muda para indicar que o mesmo está prestes a ser selecionado, ou ativado.

Plug-in: Termo que pode também ser encontrado como add-in e add-on. É um programa ou aplicativo usado para adicionar novas funcionalidades que não existiam no programa original.

QuickTime Player: Aplicação desenvolvida pela Apple para a visualização de vídeos.

RealPlayer: Programa destinado a visualização de vídeos, músicas e rádios na internet.

Sidebar: Elementos secundários em uma página da web, geralmente colocados em uma coluna lateral ao conteúdo principal.

Slider: Elemento gráfico usado para rolar a página ou aplicação Web em todas as direções vetoriais.

Software: Programa armazenado e executado pelo computador. O software é comumente expressado pelo conjunto de linhas de código de programação que formam seu conjunto.

Startpage: Definição da página ou elemento inicial em uma animação.

Streaming: Termo definido para o fluxo de pacote de dados multimídia, por uma rede privada ou pública. A mídia é constantemente reproduzida durante o processo através do uso da memória *Cache*.

Symbian OS: Sistema operacional desenvolvido pela Nokia para celulares.

Template: Pode ser definido como o modelo inicial, ou padrão, geralmente usado na criação de um novo visual de página web.

Twitter: Rede social de microblogging, usado por usuários para o compatilhamento de pequenas notícias, informações ou hyperlinks.

Up: Usado para expressar um elemento que está sem foco, geralmente o estado inicial de todos os botões de navegação de uma página web.

Vimeo: Rede social baseada no compartilhamento de vídeos.

Web: Rede. Comumente descrito como a Rede da Internet, normalmente usada como abreviatura do termo WWW, ou World Wide Web.

WebDesign: Pode ser visto como uma extenção da prática de design, onde o foco do projeto é a criação de websites ou documentos presentes na web.

WebDeveloper: Termo usado para definir o profissional desenvolvedor de programas e aplicações para o ambiente Web.

WebSite: Local da internet que permite acesso por dispositivo remoto, normalmente um PC ou computador pessoal.

WHATWG: Web Hypertext Application Technology Working Group, caracterizado por um grupo de trabalho interessado e comprometido na evolução da linguagem HTML.

Windows Media Player: É um programa que reproduz mídias digitais, tais como vídeos e músicas.

Windows Mobile: Sistema operacional desenvolvido pela Microsoft para celulares smartphone.

YouTube: Rede social baseada no compartilhamento de vídeos.

APÊNDICE

APÊNDICE A - Modelagem do Website

Model Detail
This document provides a complete overview of all element details. For simpler and more focused reports, simply copy this initial template and turn off the sections not required.

Modelos de Casos de Uso
Type: **Package**
Status: Proposed. Version . Phase 1.0.
Package: Modelo
Detail: *Created on 15/08/2010. Last modified on 15/08/2010*
GUID: {ACAAA83C-7608-4fef-BAFE-9EA9F7E041D6}

Modelos de Casos de Uso - *(Use Case diagram)*
Created By: César *on* 15/08/2010
Last Modified: 25/10/2010
Version: 1.0. *Locked:* False
GUID: {5C49073F-8697-46d2-9048-01BB8CC043ED}

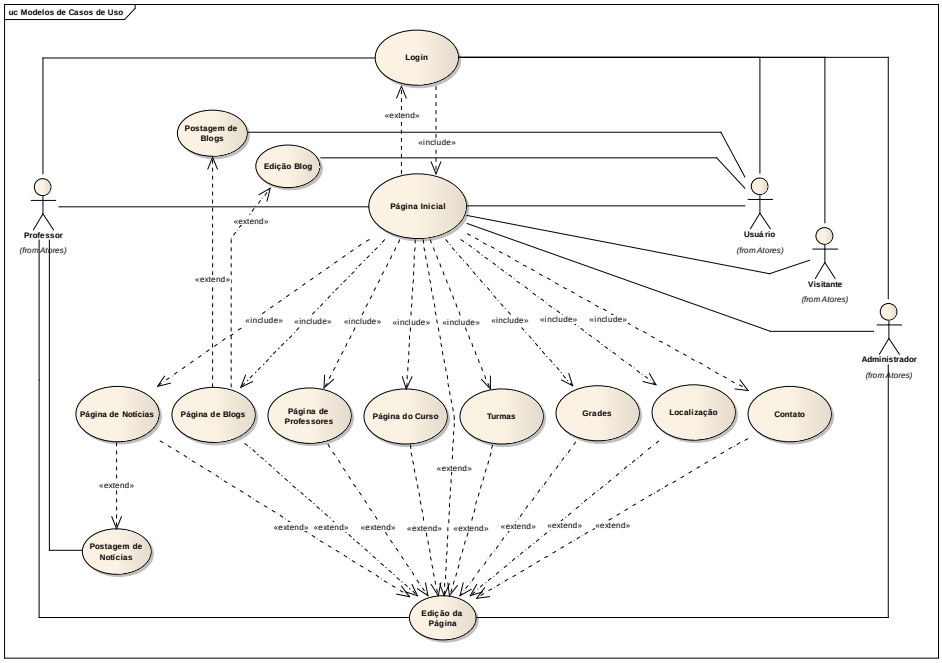


Figure: 1

Contato
Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {13BE7F91-F45B-48bf-B44E-92A221CF1C40}

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Contato	Public Edição da Página	
Include Source -> Destination	Public Página Inicial	Public Contato	

Edição Blog

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 17/09/2010. Last modified on 17/09/2010.*
GUID: {1897FBC7-2870-4e44-BD22-F6A276D83D4F}

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Página de Blogs	Public Edição Blog	
Association Unspecified	Public Usuário	Public Edição Blog	

Edição da Página

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {DF1AEA16-9E09-4976-B0A6-096167D110F7}

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Turmas	Public Edição da Página	
Extend Source -> Destination	Public Página do Curso	Public Edição da Página	
Extend Source -> Destination	Public Página de Professores	Public Edição da Página	
Extend Source -> Destination	Public Localização	Public Edição da Página	

Connector	Source	Target	Notes
Association Unspecified	Public Professor	Public Edição da Página	
Extend Source -> Destination	Public Contato	Public Edição da Página	
Extend Source -> Destination	Public Página de Blogs	Public Edição da Página	
Association Unspecified	Public Administrador	Public Edição da Página	
Extend Source -> Destination	Public Grades	Public Edição da Página	
Extend Source -> Destination	Public Página de Notícias	Public Edição da Página	
Extend Source -> Destination	Public Página Inicial	Public Edição da Página	

Grades

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {010A8ACD-8BFB-4d52-9F1D-E110D6B0E841}

Connections

Connector	Source	Target	Notes
Include Source -> Destination	Public Página Inicial	Public Grades	
Extend Source -> Destination	Public Grades	Public Edição da Página	

Localização

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {AD92CA4E-B86B-4a26-950C-F503311FC83B}

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Localização	Public Edição da Página	
Include Source -> Destination	Public Página Inicial	Public Localização	

Login

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 17/09/2010. Last modified on 17/09/2010.*
GUID: {956AF17C-E539-4493-8285-1D6E93FCBF5C}

Connections

Connector	Source	Target	Notes
Association Unspecified	Public Usuário	Public Login	
Association Unspecified	Public Professor	Public Login	
Association Unspecified	Public Administrador	Public Login	
Association Unspecified	Public Visitante	Public Login	
Extend Source -> Destination	Public Página Inicial	Public Login	
Include Source -> Destination	Public Login	Public Página Inicial	

Postagem de Blogs

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {847AB24B-7A6F-445f-A293-195538A016C4}

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Página de Blogs	Public Postagem de Blogs	
Association Unspecified	Public Usuário	Public Postagem de Blogs	

Postagem de Notícias

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {0F666AAE-4B53-48ef-920E-847EB2F5021E}

Connections

Connector	Source	Target	Notes
Association Unspecified	Public Professor	Public Postagem de Notícias	
Extend Source -> Destination	Public Página de Notícias	Public Postagem de Notícias	

Página Inicial

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 26/10/2010.*
GUID: {8F98ED9E-D401-467f-8F80-8204E2B468F6}

Nome: Página Inicial

Descrição: Os usuários acessam o website.

Ator primário: Usuário, Professor, Visitante ou Administrador.

Fluxo Principal:

1. O Usuário acesso o website através de seu endereço.

2. Página inicial do website é mostrada na tela.

Connections

Connector	Source	Target	Notes
Association Unspecified	Public Professor	Public Página Inicial	
Association Unspecified	Public Usuário	Public Página Inicial	
Association Unspecified	Public Administrador	Public Página Inicial	
Include Source -> Destination	Public Página Inicial	Public Página de Notícias	
Include Source -> Destination	Public Página Inicial	Public Contato	
Include Source -> Destination	Public Página Inicial	Public Localização	
Include Source -> Destination	Public Página Inicial	Public Grades	
Include Source -> Destination	Public Página Inicial	Public Turmas	
Include Source -> Destination	Public Página Inicial	Public Página do Curso	
Include Source -> Destination	Public Página Inicial	Public Página de Blogs	
Extend Source -> Destination	Public Página Inicial	Public Edição da Página	
Association Unspecified	Public Visitante	Public Página Inicial	
Extend Source -> Destination	Public Página Inicial	Public Login	
Include Source -> Destination	Public Login	Public Página Inicial	
Include Source -> Destination	Public Página Inicial	Public Página de Professores	

Página de Blogs
Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {2F002C3C-4604-4b6c-9DD7-8E4D13046C17}

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Página de Blogs	Public Edição Blog	
Extend Source -> Destination	Public Página de Blogs	Public Postagem de Blogs	
Extend Source -> Destination	Public Página de Blogs	Public Edição da Página	
Include Source -> Destination	Public Página Inicial	Public Página de Blogs	

Página de Notícias
Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 26/10/2010.*
GUID: {6B08F54D-EA22-4807-8112-D1FE87A3C5CE}

Nome: Página de Notícias

Descrição: Os usuários acessam a sub-página de notícias.

Ator primário: Usuário, Professor, Visitante ou Administrador.

Fluxo Principal:

1. O Usuário acesso a sub-página Notícias através do Botão no Menu.

2. Sub-página é lida pelo navegador, mostrando a relação de notícias.

Fluxo Alternativo:

1. O Usuário clica em voltar, exibindo a página inicial.

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Página de Notícias	Public Postagem de Notícias	
Include Source -> Destination	Public Página Inicial	Public Página de Notícias	
Extend Source -> Destination	Public Página de Notícias	Public Edição da Página	

Página de Professores

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {390A44AC-306F-4239-A0D1-138B70FF29C3}

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Página de Professores	Public Edição da Página	
Include Source -> Destination	Public Página Inicial	Public Página de Professores	

Página do Curso

Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {CE1C1723-1B73-4042-91BB-0CE981291B46}

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Página do Curso	Public Edição da Página	
Include Source -> Destination	Public Página Inicial	Public Página do Curso	

Turmas
Type: **UseCase**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {D72DC847-711D-446b-8C7B-30538268F7EB}

Connections

Connector	Source	Target	Notes
Extend Source -> Destination	Public Turmas	Public Edição da Página	
Include Source -> Destination	Public Página Inicial	Public Turmas	

Atores
Type: **Package**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Modelos de Casos de Uso
Detail: *Created on 19/11/2005. Last modified on 19/11/2005*
GUID: {B4ED934D-72E0-49fb-BE2B-7193EF39925B}

Administrador
Type: **Actor**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Atores *Keywords:*
Detail: *Created on 17/09/2010. Last modified on 17/09/2010.*
GUID: {4C43CF6D-DAF6-477a-B183-1CD532F7D318}

Connections

Connector	Source	Target	Notes
Association Unspecified	Public Administrador	Public Página Inicial	
Association Unspecified	Public Administrador	Public Edição da Página	
Association Unspecified	Public Administrador	Public Login	

Professor

Type: **Actor**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Atores *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 17/09/2010.*
GUID: {0A79FA92-0F76-44df-8417-C4095F1C8DF5}

Connections

Connector	Source	Target	Notes
Association Unspecified	Public Professor	Public Página Inicial	
Association Unspecified	Public Professor	Public Edição da Página	
Association Unspecified	Public Professor	Public Postagem de Notícias	
Association Unspecified	Public Professor	Public Login	

Usuário

Type: **Actor**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Atores *Keywords:*
Detail: *Created on 15/08/2010. Last modified on 15/08/2010.*
GUID: {97C01B2A-A5FD-41e8-A567-229271F0BA64}

Connections

Connector	Source	Target	Notes
Association Unspecified	Public Usuário	Public Postagem de Blogs	
Association Unspecified	Public Usuário	Public Login	
Association Unspecified	Public Usuário	Public Página Inicial	
Association Unspecified	Public Usuário	Public Edição Blog	

Visitante

Type: **Actor**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Atores *Keywords:*
Detail: *Created on 25/10/2010. Last modified on 25/10/2010.*
GUID: {E278A3E2-A4B8-4d36-AA33-47A5E1541344}

Connections

Connector	Source	Target	Notes
Association Unspecified	Public Visitante	Public Login	
Association Unspecified	Public Visitante	Public Página Inicial	

ANEXO

ANEXO A - Fluxo de trabalho do Adobe Flash Catalyst

Chapter 2: The Flash Catalyst workflow

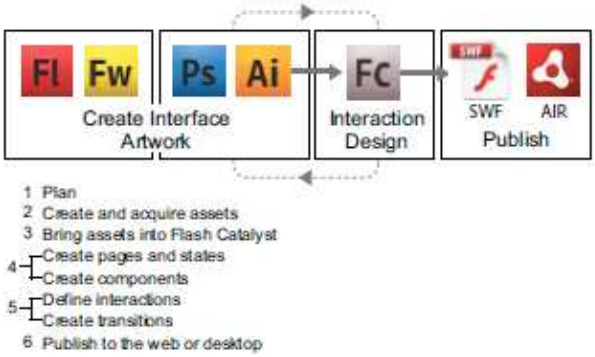
Adobe Flash Catalyst CS5 is a design tool for rapidly creating application interfaces and interactive content without coding. Examples include interactive ads, product guides, design portfolios, and user interfaces. While there is not a single, prescribed workflow for creating all projects in Flash Catalyst, there are tasks common to many basic projects. Here is an overview to get you up and running quickly:

General workflows

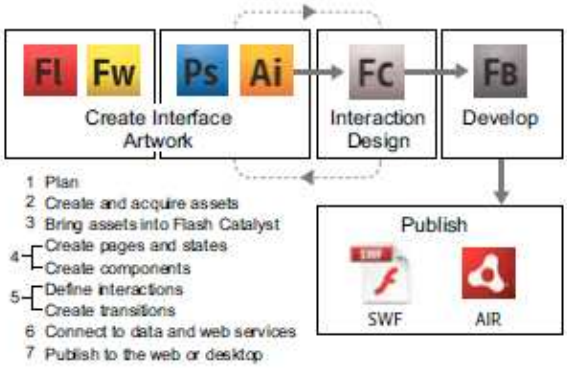
There are two main types of Flash Catalyst applications. These applications include micro sites and data-centric applications.

In this document, micro site is an application that is complete when published in Flash Catalyst. No additional development is required. A data-centric application requires additional development, such as integrating components with external data or web services. A Flex developer completes the development using Adobe Flash Builder.

The workflows for designing micro sites and data-centric applications are similar.



Micro site workflow



Data-centric designer/developer workflow

Workflow steps

Follow these general workflow steps when creating micro sites and data-centric applications with Flash Catalyst.

Plan the application Start with a detailed project specification. This specification describes each page or screen, the artwork and interactive components on each page, user navigation, and the different states of each component. The specification also describes any data list components used to retrieve and display external data.

Create or acquire artwork, video, and sound Create the artwork, video, and sound for the application. Create a layered design document or composition in Adobe Illustrator, Photoshop, or Fireworks.

Bring in artwork, video, and sound Start Flash Catalyst. Bring the layered artwork into Flash Catalyst. You can also import individual graphic files or create simple graphics using the built-in vector drawing tools. Import additional assets, such as video, sound, and SWF content. For data-centric components, such as a data list, import a representative sample of the data (text or images). For more information, see "Importing artwork" on page 6.

After importing or creating artwork in Flash Catalyst, you can launch and edit artwork in Illustrator or Photoshop, and then return the edited artwork to Flash Catalyst. Round-trip editing extends the graphic drawing and editing capabilities of Flash Catalyst and improves the iterative design process. For more information, see "Round-trip editing" on page 52.

Create and modify page states Create pages according to the project specification. For more information, see "Defining structure with pages and states" on page 13.

Create interactive components and define component states Convert artwork to ready-made components (buttons, scroll bars, data lists, and so on). Use the Wireframe Components panel to quickly add common components with a generic appearance. Create custom components for behaviors that you can't capture with the built-in components. For more information, see "Interactive components" on page 22.

For data-centric applications, use design-time data to design data list components. Design-time data allows the use of dummy content, such as sample database records or bitmap images, without having to actually connect to a back-end system. A Flex developer can replace the design-time data with real data from a database or web service. For more information on using Design-time data, see "Creating scrolling images, panels, and lists" on page 35.

Components can have multiple states, such as the Up, Over, Down, and Disabled states of a button. Create or modify the different states of each interactive component, according to your project specification

Note: *The steps of creating page states and creating interactive components are interchangeable. Some designers prefer to create all interactive components first, and then add those components to pages and states.*

Define interactions and transitions Add interactions that define what happens as users interact with the application. For example, you can add interactions that transition from one page or component state to another when a user clicks a button. You can also add interactions that play animation, control video playback, or open an URL. Use the Timelines panel to add and modify smooth animated transitions between pages and component states. For more information on interactions, see "Interactions and action sequences" on page 27. For more information on transitions, see "Transitions and the timeline" on page 31.

Test and publish the project When you finish creating a micro site, you can publish the final project as a web or desktop application. For more information, see "Preview and publish" on page 59.

Share the project with a Flex developer Save a data-centric Flash Catalyst project file (FXP) for further development in Adobe Flash Builder. Export the Flash Catalyst project library. Exporting a library package creates a single file containing every library item in the project. The package is saved as an FXPL file. For more information on exporting a Flash Catalyst library, see "Extending Flash Catalyst projects using Flash Builder" on page 62.

Start a new Flash Catalyst project

You can start a new project in two ways:

- Start with a blank artboard and build your application. This approach is useful for rapid wire framing of user interfaces. Catalyst provides wireframe components, drawing tools, and the ability to import various media to rapidly prototype an interface.
- Import a completed design document as layered artwork created in Adobe Photoshop, Illustrator, or an exported design from Fireworks. Using this approach, you can design in your favorite Adobe Creative Suite application and quickly convert the artwork into a functioning interactive application.

Start a project with a blank artboard:

1 Start Flash Catalyst. In the Create New Project section of the Welcome screen, choose Adobe Flash Catalyst Project.

Note: If you already have a project open, choose File > New Project to begin a new blank project.

2 In the New Project dialog box, name the project, enter values for the size and color of the artboard, and click OK.

You now have a new project with a blank artboard. By default, the Design workspace is open. You can build your application by importing artwork, adding pages, creating components, and adding interactions and transitions.

Note: You can change the artboard values later by choosing Modify > Artboard Settings.

Start a project by importing artwork in a layered design document:

1 Start Flash Catalyst.

2 In the Create New Project from Design File section of the Welcome screen, choose the type of file you want to import. Options include: Adobe Illustrator AI File, Adobe Photoshop PSD File, FXG File (FXG files can be exported from Adobe Fireworks, as well as other applications).

Note: If you already have a project open, choose File > Import > <File Type>.

All artwork in the design document is added to the new Flash Catalyst project. The Layers panel reflects the layer structure from the imported document, preserving the integrity of your original design.

You can now build your application by adding pages, creating components, and adding interactions and transitions.

For more information, see "Importing artwork" on page 6.

ANEXO B - Importando a Arte gráfica

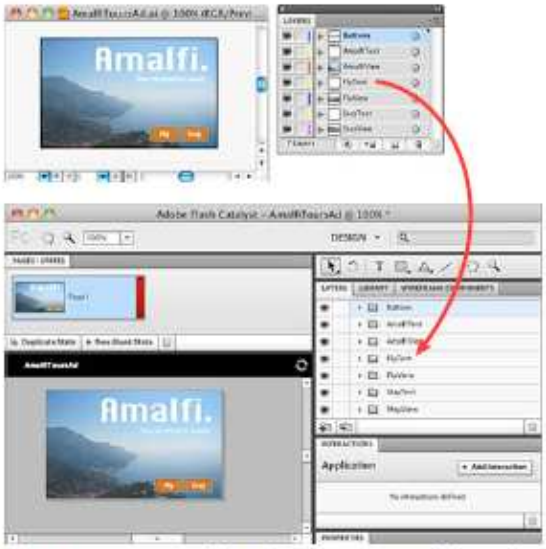
Chapter 3: Importing artwork

There are several ways to get your artwork into Flash Catalyst.

- Import a layered design document created in Adobe Photoshop or Adobe Illustrator.

  Note: *Flash Catalyst only imports design documents that are 40 MB or less.*

- Import a layered FXG file. You can export an FXG file from Adobe Fireworks and other Adobe Creative Suite applications.
- Import one or more bitmap images.
- Copy and paste graphics into the Flash Catalyst artboard.
- Import a SWF file.
- Import a Flash Catalyst library package.



When you import artwork from Illustrator CS5 into Flash Catalyst CS5, the layer structure remains intact and art can continue to be edited in Illustrator.

Import Adobe Illustrator files

You can start a new Flash Catalyst project by importing an Illustrator file.

1 Start Flash Catalyst.

2 In the Create New Project From Design File section of the Welcome screen, choose From Adobe Illustrator AI File.

If Flash Catalyst is already running, choose File > New Project From Design Comp. You can have only one project open at a time.

3 Browse to the file you want to import, select it, and click Open.

The Illustrator Import Options dialog box includes artboard settings and fidelity options. You can choose to import non-visible layers and include unused symbols.

Note: *Choosing Import Non-Visible Layers imports all layers, including layers that are hidden in the Illustrator file. Choosing Include Unused Symbols imports the graphic symbols that ship with Illustrator and the symbols you create.*

4 Specify a size and color for the artboard. Select import fidelity options and click OK.

Illustrator Import Options dialog box

The Illustrator file converts to the FXG format automatically, and then imports into a new Flash Catalyst project. If the Illustrator file includes a single artboard, all artwork is placed in the same Flash Catalyst page state. If the Illustrator file includes multiple artboards, the artwork in each artboard is placed in a separate Flash Catalyst page state.

You can copy and paste individual pieces of artwork from Illustrator into the Flash Catalyst artboard. The Illustrator Import Fidelity Options also appear when you copy and paste artwork.

Note: *Objects outside the Illustrator artboard are discarded when you import or copy and paste artwork into Flash Catalyst.*

Illustrator symbols import as Optimized Graphics. If your Illustrator file includes multiple instances of the same symbol, then your Flash Catalyst document will include multiple instances of the same optimized graphic. In Flash Catalyst, it is a best practice to use one instance of an object and then share that object to other states. You can remove all but one instance of the optimized graphic, share the same instance to other states, and then apply different properties in each state. To convert the optimized graphic into a Flash Catalyst component, you must first break it apart by choosing Modify > Break Apart Graphic.

Import Adobe Photoshop files

You can start a new Flash Catalyst project by importing a Photoshop file.

1 Start Flash Catalyst.

2 In the Create New Project From Design File section of the Welcome screen, choose From Adobe Photoshop PSD File.

If Flash Catalyst is already running, choose File > New Project From Design Comp. You can have only one project open at a time.

3 Browse to the file you want to import, select it, and click Open.

The Photoshop Import Options dialog box includes artboard settings and fidelity options. You can also choose to import non-visible layers.

Note: *Choosing Import Non-Visible Layers imports all layers, including layers that are hidden in the Photoshop file.*

4 Specify a size and color for the artboard. Select import fidelity options and click OK.

Photoshop Import Options dialog box

Click Advanced in the Photoshop Import Options dialog box to specify exactly which layers to import. You can select and deselect layers to import, regardless of their visibility in Photoshop. You can choose to import specific Photoshop Layer Comps. See Importing Photoshop layer comps.

Import FXG files

Flash Catalyst imports artwork in the FXG file format.

1 Start Flash Catalyst.

2 In the Create New Project From Design File section of the Welcome screen, choose FXG File.

If Flash Catalyst is already running, choose File > New Project From Design Comp. You can have only one project open at a time.

3 Browse to the file you want to import, select it, and click Open.

For information on exporting an FXG file from Fireworks, see Export FXG files.

Import bitmap images

Flash Catalyst accepts bitmap images with the PNG, GIF, JPG, JPEG, and JPE filename extensions.

1 Choose File > Import > Image.

2 Browse to locate the file, select it, and choose Open.

- When you import a single image file, it is placed in the project library and an instance is placed in the artboard in the current state. A new layer for the object is added in the Layers panel.
- When you import multiple image files, they are placed in the project library. No image is added the artboard. To add an instance of the image to the artboard, drag it from the Library panel to the artboard.

Import SWF files

1 Choose File > Import > SWF File.

2 Browse to locate the file, select it, and choose Open.

- When you import a single SWF file, it is placed in the project library and an instance is placed in the artboard in the current state. A new layer for the object is added in the Layers panel.
- When you import multiple SWF files, they are placed in the project library. No SWF file is added the artboard. To add an instance of the SWF file to the artboard, drag it from the Library panel to the artboard.
- You cannot preview a SWF file in Flash Catalyst Library panel. To preview the SWF file, run the project by choosing File > Run Project.
- Use interactions and effects in Flash Catalyst to control the playback of SWF files. You can also play or stop a SWF file at a specific frame. For more information, see "Interactions and action sequences" on page 27.
- Only SWF content written in ActionScript 3.0 and published using Adobe Flash Professional is controllable in Flash Catalyst.
- There is no direct integration between Flash Catalyst and Flash Professional. Edit the SWF file in Flash Professional, republish, and import the new file into Flash Catalyst. Use the Source link in the Properties panel to swap the old SWF file for the new one.

Import a Flash Catalyst library package

For information on importing artwork in a library package, see "Export and import a library package" on page 21.

Import fidelity considerations

When you import from Illustrator and Photoshop, you specify fidelity options that control how Flash Catalyst imports your artwork. It is helpful to know which attributes are supported in Flash Catalyst, and when to flatten or rasterize artwork before importing.

ANEXO C - Interface do Usuário

Chapter 4: User interface

The Flash Catalyst CS5 user interface has two workspaces. These workspaces include Design and Code. Use the Workspaces pop-up menu to change between workspaces.

Design workspace

The Design workspace shows a graphical representation of your pages and states. This workspace includes the panels and tools used for creating and editing projects. Use the Hand tool to grab and pan the artboard as an alternative to scrolling. Use the Zoom tool or Magnification menu to change the view from between 25% and 800% of actual size. Use the magnifying glass to zoom into a specific part of the artboard (Alt-click (Windows) or Option-click (Mac OS) to zoom out). When you enter a term in the Search box, the Adobe Community Help client appears. It gives you access to online Help and community resources.

Design workspace
***A.** Artboard **B.** Breadcrumbs bar **C.** Hand tool **D.** Zoom tool **E.** Zoom Magnification menu **F.** Heads Up Display **G.** Tools panel **H.** Workspaces pop-up menu **I.** Search box*

Artboard The artboard represents what users see when they view the published application. The artboard is where you place artwork, interactive components, and other objects that make up the application interface. It has rulers, grids, and guides for positioning and snapping elements. These features are available in the View menu. Use the Modify menu to align, group, and arrange (front to back) the objects on the artboard.

Breadcrumbs bar The Breadcrumbs bar, located directly above the artboard, tracks where you are as you work in Flash Catalyst. For example, when you open a component, you can use the Breadcrumbs bar to quickly close the component and return to the main artboard.

Pages/States panel The Pages/States panel displays a thumbnail for each page in the application. It shows the different states for any component you select. You can duplicate, remove, add, and rename pages and component states according to your project plan. For more information, see "Defining structure with pages and states" on page 13.

Tools panel The Tools panel includes tools for creating, selecting, and transforming objects, including simple lines, shapes, and text.

Layers panel The Layers panel is an organized collection of the objects in the application (artwork, components, video, and so on). When you import a design document created in Illustrator, Photoshop, or Fireworks, Adobe Flash Catalyst preserves the original layer structure. As you add pages and component states to the application, you use the Layers panel to show or hide objects in each state. For more information, see "Layers" on page 17.

Library panel The Library displays the entire list of graphics and other media available in the project, including your project skins and components. For more information, see "Library" on page 20.

Wireframe Components panel Wireframe components are ready-to-use interactive components with a simple default appearance. You can drag these components to the artboard and use them "as is" or modify them to fit the appearance of your application. For more information, see "Interactive components" on page 22.

Interactions panel Add interactions that define what happens as users interact with the application. For example, you can add interactions that transition from one page or component state to another when a user clicks a button. You can

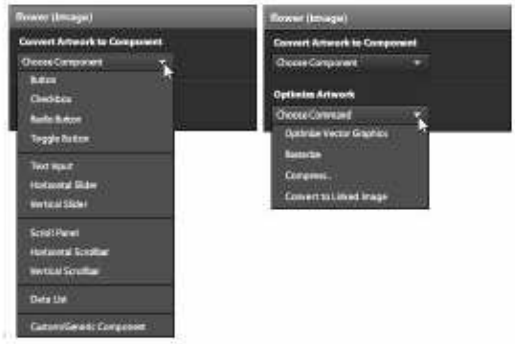
also add interactions that play animation, control video playback, or open an URL. For more information, see "Interactions and action sequences" on page 27"Transitions and the timeline" on page 31.

Timelines panel The Timelines panel provides controls for creating and editing transitions and action sequences. You can also use the Timelines panel to control the playback of video and SWF content, and to add sound effects. For more information, see "Transitions and the timeline" on page 31.

Design-Time Data panel After creating a data list component, use the Design-Time Data panel to control which data (images and text) appear in the data list. For more information, see "Creating scrolling images, panels, and lists" on page 35.

Properties panel Use the Properties panel to edit the properties for selected objects, such as graphics, text, and components. The available properties change as you select different objects in the artboard, Layers panel, or Timelines panel.

Heads Up Display (HUD) The HUD gives quick access to common commands related to the current action or currently selected object. It shows some of the key actions you can perform on the selected objects. For example, the HUD appears when you select artwork on the artboard, giving you the choice of converting the artwork to a component. Use the HUD to quickly create components.



The HUD displays context-appropriate commands.

- If you don't see the HUD when you select an object, select Window > HUD.
- When converting objects to components, the HUD displays a message if additional steps are required to complete the component.
- All of the functionality in the HUD is also available in the main menu. For example, you can choose Modify > Convert Artwork To Component.

Use the HUD to quickly:

- Convert artwork to components or component parts.
- Edit the parts and states of a component.
- Optimize graphics elements.
- Make the parts of a component the same in all states or copy changes from one state to another.

***A.** HUD with new data list component selected **B.** Component message*

For more information on using the HUD, see "Interactive components" on page 22, "Creating scrolling images, panels, and lists" on page 35, "Creating shapes, lines, and text" on page 40.

Code workspace

The Code workspace shows the underlying application code. This code is generated automatically as you work in Flash Catalyst.

The applications you build in Flash Catalyst are build on the Flex framework. Flex is an open source framework for building and deploying applications that run in all major browsers and operating systems. MXML is the language developers use to define the layout, appearance, and behaviors in Flex. ActionScript 3.0 is the language used to define the client-side application logic. When you publish a Flash Catalyst project, your MXML and ActionScript are compiled together as a SWF file.

Viewing the MXML code gives designers the opportunity to understand how the application is programmed. The Code workspace is read-only. To edit the code, open the project in Adobe Flash Builder. For more information, see "Extending Flash Catalyst projects using Flash Builder" on page 62.

Code workspace

Code panel Shows the MXML code in the Code panel.

Problems panel Shows any errors in the current MXML code.

You can double-click an error in the Problems panel to locate the error in the code.

Project Navigator panel Shows the Flex project directory structure and files being created as you design your project in Flash Catalyst.

ANEXO D - Definindo estruturas com páginas e estados

Chapter 5: Defining structure with pages and states

Most interactive projects and applications are designed to present information in more than one view, or page. The different views a user sees when navigating through an interactive project—or when clicking on an interactive component (such as a button) inside of your project—are called "states" in Flash Catalyst. There are two types of states:

Page states. A simple interactive ad might have one view that contains a sale offer, and another view that contains pricing details. Each view can be defined as a page state Think of page states as forming the top level structure of your project or application.

This interactive ad has 3 views defined as Page states. You can control what displays on any Page state by turning layers on and off in the layers panel.

Click a state in the Pages/states panel to select it; then use the layers panel to determine what displays on the selected state.

Component states. Continuing with the same example above, a sale offer page may include interactive button components for navigating, or the pricing details page may have a scrolling list component. These interactive components can also have states. "Component states" define the look and feel of an object (for example, a button) at a certain point in time; usually based on a mouse event or other user interaction.

This interactive "Stay" button has 4 Component states: Up, Over, Down, and Disabled. You can control the look and feel of each state of a button by selecting it in the Pages/states panel, then using the Properties panel to adjust the look and feel.

Add, duplicate, and delete pages and states

All Flash Catalyst pages and component states are created, modified, and managed in the Pages/States panel. A new Flash Catalyst project begins with one page state.

Note: *If you import an Illustrator file with multiple artboards, each artboard is added to a separate page state.*

- To add a new page based on an existing page, select the existing page in the Pages/States panel and click Duplicate State.
- To create a new blank page (one in which all layers are hidden and no objects are present), click New Blank State.
- To delete a page, select it in the Pages/States panel and click the Delete button (trash can).

When you duplicate a state, you are not duplicating objects. The objects persist across states. You can show or hide the objects in each state using the Layers panel.

An application or custom component can have no more than 20 states. Adding too many page states can slow performance. If the application requires more than 20 states, you can encapsulate them in custom components. For example, you can encapsulate menu bars and other components that appear on multiple pages.

In addition to improving application performance, there are other advantages to creating your different application views using custom components.

- A custom component is more versatile than a page state. It can have a unique set of properties (size, position, opacity) in each page or parent component where it's used.
- A custom component can appear to the viewer as if they are viewing another page or screen in the application.
- Components can be nested inside other components. Nesting components makes it possible to create a more efficient application with many states or views.
- Editing a component in Edit-In-Place updates the component throughout the application.

Note: *Some components have a fixed set of states, such as the Up, Over, Down, and Disabled states of a button. You cannot duplicate or delete these component states, but you can hide them by hiding all layers for the selected state.*

See the following for more information on creating and editing components:

"Interactive components" on page 22

Name pages and states

Consider the following when naming pages and component states:

- To rename a page or component state, double-click its name in the Pages/States panel, type a new name, and press Enter (Windows) or Return (Mac OS).
- State names must begin with a letter.
- State names cannot contain spaces.
- State names cannot contain special characters: @!#$%^&*().

Navigate between pages and states

- To view the contents of a page, select it in the Pages/State panel. The artboard shows all visible objects in the selected page.
- To view the different states of a component, double-click the component in the artboard to enter Edit-In-Place mode. When you edit a component in place, the Pages/States panel updates to show the states of the component.

Note: *When a component is in Edit-In-Place mode, the Layers panel splits into sections. It shows layers for both the project and any components you have open. You can drag objects from the main application layers into the component.*

For more information on Edit-In-Place and, see "Edit-In-Place" on page 25.

Show and Hide artwork in pages and states

When you import a design document, the artwork is added to a page state in Flash Catalyst.

To add additional objects to a state, do one of the following:

- Import new artwork.
- Drag assets from the Library panel to the artboard.

- Create new objects with the Flash Catalyst drawing tools.

  For more information on the drawing tools, see "Creating shapes, lines, and text" on page 40.

When you add objects to a page or component state, they are present. They exist in that state. When an object is present, it can be made visible or hidden. The following information helps identify the presence and visibility of objects in the current state using the Layers panel:

Present and visible The name of the object is listed using dark text (present), and the eyeball icon is dark (visible).

Present and hidden The name of the object is listed using dark text (present), but there is no eyeball icon showing (hidden). If the eyeball appears dimmed, the object's visibility is on but its parent layer is hidden. When a parent layer or group is hidden, its children are hidden automatically.

Not present The name of the object is listed using dimmed text (not present). The object is not present in the current state, but it does exist in one or more other states of the application.

Use the following techniques to display or hide objects in a state:

- Turn the eyeball icon on or off to show or hide an object.

  The eyeball icon is a toggle. Click the Show/Hide column (far left column) beside an object in the Layers panel to toggle its visibility.

- Turn off the eyeball icon for a parent layer or group to hide all of its children.
- Select an object and press Delete to remove the object from the current state. If the object exists in other states, its name turns dim in the Layers panel. If the object does not exist in any other states, it disappears from the Layers panel.
- Turn on the eyeball icon to share an object to the current state when its name is dimmed.
- Select an object and click the Delete button (trash can) in the Layers panel to remove it from all states and the Layers panel.

Note: *You can make an object appear invisible or partially transparent by changing its Opacity value in the Properties panel.*

For more information on the Layers panel, see "Layers" on page 17.

Share objects between pages and states

A single object can appear in multiple states. That object can have a different set of properties, such as size, position, color, and transparency in each state where it exists. In most cases, when you modify an object, those changes apply only to the object in the currents state. Once you position and modify an object to your liking, you can quickly share that object, along with its properties, to other states.

This technique makes it possible to create smooth transitions from one state to the next. For example, you can create the effect of an object fading in or out or morphing from one shape or position to another.

- To share an object to other states, select the object and choose States > Share To State. Select the states to which you want to share the object.
- To remove an object from a specific state, select the object in one state and choose States > Remove From State. Choose the state from which you want the object removed.
- To make an object the same in all states, modify its properties, and then choose States > Make Same In All Other States.

Note: *Some edits or changes apply to all states. Any change that affects the application hierarchy is shared across all states automatically. For example, if you group objects or convert objects to components, the change applies to all states. If you edit a component using Edit-In-Place mode, you edit the component definition in the project library. The changes apply to all instances of the component in all states.*

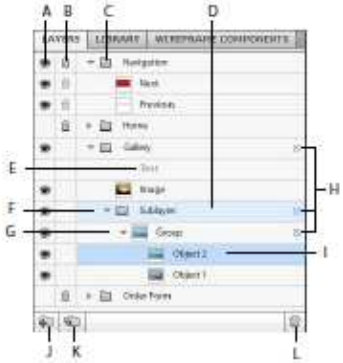
For more information on components, see "Set component properties" on page 23.

For more information on creating transitions, see Create transitions.

ANEXO E - Camadas

Chapter 6: Layers

The Flash Catalyst CS5 Layers panel serves a few primary functions. It provides an organized structure for viewing and managing every object in the application (or a component that you have open for editing). It also indicates which of those objects are present and visible in the current state. The current page or state is indicated in the Pages/States panel. See "Show and Hide artwork in pages and states" on page 14.



Layers panel
***A.** Show/Hide **B.** Lock/Unlock **C.** Layer **D.** Target layer **E.** Object is not present in the selected state **F.** Sublayer **G.** Group **H.** Contains selected objects **I.** Selected **J.** Create New Layer **K.** Create New Sublayer **L.** Delete*

Organize a project using layers

The Layers panel shows every object in the application using a collection of stacked rows. Rows can represent layers, sublayers, objects (images, shapes, text, components), and groups (grouped objects).

An eyeball icon to the left of a row indicates that objects are visible in the current page or state. If an eyeball is dimmed, it means that the row is visible, but its parent layer or group is hidden in the current state. Objects that do not exist in the current state are dimmed with no eyeball. Objects that do exist but are hidden in the current state are regular text with no eyeball. A padlock icon in the second column of a row means that it is locked. A locked layer can be viewed but not edited. Locking layers prevents content from being selected or moved accidentally. The target layer for new content that you add to the artboard is shaded light blue. The currently selected row/object is a slightly darker shade of blue. A small blue square means that the row includes a selected object. You can follow the blue squares as you drill down to find the selected item.

- To expand or collapse a layer or group, click the small arrow.
- To rename the row representing a layer, object, or group, double-click the current name, type a new name, and press Enter (Windows) or Return (Mac OS).
- To add a new layer, click the Create New Layer or Create New Sublayer button.
- You can delete a layer, object, or group from all states in the application. Select its row and click the Delete button (trash can) in the Layers panel. The object is removed from every state.

Flash Catalyst Layer panel terminology

The following list summarizes how Flash Catalyst layer terminology compares to layer terminology in Illustrator, Photoshop, and Fireworks.

- A Flash Catalyst Layer is the same as an Illustrator Layer, a Photoshop Layer Group, and a Fireworks Layer Folder.
- A Flash Catalyst Object is the same as an Illustrator Object, a Photoshop Layer, and a Fireworks Layer.

Selecting objects in the Layers panel

You can select layers, objects, or groups (grouped objects) in the artboard or in the Layers panel.

- To select a single object, click its row in the Layers panel.

  Note: *Use the Select or Direct select tool to select an object or group in the artboard.*

- Shift+click to select a contiguous range of objects in the Layers panel.
- To select non-contiguous objects in the Layers panel, Control+click (Windows) or Command+click (Mac) two or more objects.

  Note: *Shift+click to select multiple objects in the artboard.*

- Selecting a layer in the Layers panel selects the layer and all of its children.
- Selecting a group in the Layers panel selects the group as a single object. The group's Properties are active in the Properties panel.
- To select the individual children in a group, select their individual rows in the Layers panel.

  Note: *Use the Direct Select tool to select the children of a group in the artboard.*

Manage artwork using layers

When you import a design document created in Adobe Illustrator, Photoshop, or Fireworks (FXG), Flash Catalyst maintains the integrity of the original design. With your artwork organized into layers, you can begin creating the different pages, components, and component states for the application.

Use the Layers panel to determine which artwork is visible, hidden, or present on each page.

- Click the eyeball icon to hide or show an object in the current state. When you hide an object, it is still present in the current state, but not visible.
- To remove an object from the current state, select it and press Delete. The object is no longer present in the current state. If the item still exists in another state, the item's name appears dimmed in the Layers panel. You can return it to the current state by clicking its Show/Hide button.
- You can use the artwork in multiple layers to create interactive components, such as a button or scroll bar. Components can have multiple states, such as up, over, down, and disabled. When you edit a component in Edit-In-Place, the Layers panel expands to show the objects in the selected component. Use the Layers panel to hide or show artwork in each component state.
- As you modify the artwork in a page or component state, each object's size, position, and visual attributes are remembered for each state. You can have the same object appear differently from one state to another. For more information, see "Defining structure with pages and states" on page 13.

- To change the stacking order of objects in the application, you can drag rows up or down in the Layers panel. You can also change the stacking order of objects within a layer or group. Drag the object row or select the object and choose Modify > Arrange > Bring To Front, Bring Forward, Send Backward, or Send To Back.

Note: *The stacking order of layers is constant across all states. Unlike object properties, you can't have a different stacking order in different states.*

See the following for more information on components and editing objects.

"Edit-In-Place" on page 25

"Creating shapes, lines, and text" on page 40

ANEXO F - Expandindo o Projeto do Flash Catalyst

Chapter 17: Extending Flash Catalyst projects using Flash Builder

Although you can use Flash Catalyst CS5 to publish fully functional rich Internet applications many projects require additional development, such as binding components to a data source. A developer completes this phase using Adobe Flash Builder, an integrated development environment (IDE).

Flash Catalyst and Flash Builder

Design and development teams work closely to communicate the RIA design vision clearly to each other. After the design is complete, you save it as an FXP file containing MXML code. The developer can then import the FXP file into Flash Builder and add any functionality required by the design. For example, the developer can then add web services or connect to a database. Typically the developer does simple cleanup tasks before adding code. Cleanup includes removing metadata added by Illustrator, Flash Catalyst, and other applications.

After the developer adds services, data hookups, and other back-end functionality, further changes can only be made in Flash Builder. However, you can make design refinements in Flash Catalyst and hand off an updated FXP for the developer to merge with their copy of the project. Best practices make collaboration easier by keeping the developer's code mostly separate from the code created by Flash Catalyst.

- Before you design in Flash Catalyst, determine if the project requires development in Flash Builder.
- If the application includes large records of data, then store the data externally and bind components to the data using Flash Builder.
- If your application presents a list of items from an external source, you do not need to add every item to the component in Flash Catalyst. Instead, add a few design-time data items as a prototype.
- If designing a Flash Catalyst project for Flash Builder, meet with your developer first to discuss which Flash Catalyst components and properties to use.
- If the application uses web services, then create the input controls in Flash Catalyst and let your developer bind controls to the service using Flash Builder wizards.
- Use descriptive names for all layers and objects in the Layers panel.
- Give all library assets unique names that the developer can recognize.
- Remove all library content that is not used in the project.
- Give unique and identifiable names to all page states, components, and component states.

See the following for more information on Flash Builder integration:

Support for Catalyst projects

Importing a Catalyst FXPL project

Resolving font references when importing Catalyst projects

Importing a Flex project or Flex library project

Using Flash Builder