

Wagner Roberto de Carvalho

Informática

**Rede
Internet
Elaboração de páginas de internet
C#**

IESDE Brasil S.A.
Curitiba
2011

C33li Carvalho, Wagner Roberto de. / Informática. / Wagner Roberto de Carvalho. — Curitiba: IESDE Brasil S.A., 2011. 148 p.

ISBN: 978-85-387-1567-2

1.Informática. 2. Rede. 3. Internet. 4. Elaboração de Páginas de Internet.
5.C#. I. Título.

CDD 000.0

Capa: IESDE Brasil S.A.

Imagem da capa: IESDE Brasil S.A.

Todos os direitos reservados.



**Inteligência
Educacional**

IESDE Brasil S.A.

Al. Dr. Carlos de Carvalho, 1.482. CEP: 80730-200
Batel – Curitiba – PR
0800 708 88 88 – www.iesde.com.br

Wagner Roberto de Carvalho ■

Bacharel em Direito e Administrador de Empresas pela Universidade Mackenzie. Técnico Eletrônico pela Escola Técnica Federal. Consultor e instrutor especialista na área de Banco de Dados Relacional e Analista de Sistemas *Mainframes* com passagem pelas empresas Oracle, IBM, Unisys e Labo.

9

Introdução à rede

- 9 | LAN
- 10 | WAN
- 11 | Tipos de conexão
- 12 | Topologia estrela
- 13 | Como é composta uma rede Client/Server?
- 14 | Parte lógica de uma rede

21

Introdução à internet

- 21 | Profissões da área de web
- 22 | Por que criar um site
- 24 | O que é a internet
- 24 | Redes e conexão
- 30 | Plataformas
- 31 | Conceito de *software* livre
- 32 | História do Linux
- 34 | Segurança

41

Endereços e domínios

- 41 | Domínios de região
- 41 | Tipos de domínios
- 43 | Registrando um domínio
- 44 | Gerenciando domínios
- 46 | Linguagens utilizadas na web
- 47 | Ferramentas
- 48 | Projetando um site

53

A linguagem HTML

- 53 | Sintaxe básica de uma tag
- 54 | Criando documentos HTML

63

Imagens no site

- 63 | PhotoShop

71

Dreamweaver

- 71 | Iniciando o Dreamweaver
- 78 | Exercício – Configurando textos
- 79 | Criando *links*
- 80 | Exercício – *Links*
- 81 | Trabalhando com imagens
- 82 | Exercício – *Rollover*
- 84 | Formulários
- 86 | Exercício – Formulários
- 86 | Configuração de estilos
- 89 | Frames

93

Flash

- 93 | O que é o Flash
- 95 | Telas do Flash
- 97 | Introdução à animação
- 98 | Interpolação de forma
- 100 | Interpolação de movimento

103

Conhecendo páginas dinâmicas

- 103 | Plataforma .NET
- 104 | O que é XML?
- 105 | Entendendo o ASP.NET
- 110 | Propriedades dos objetos

115

C# .NET

- 115 | Características da linguagem C#
- 116 | Sintaxe do C#
- 117 | Estrutura condicional
- 118 | Montando um exemplo com ASP.NET e C#

121

Projeto final

- 121 | *Web service*
- 121 | O que faremos
- 122 | Iniciando
- 125 | Configurando o *web service*
- 126 | Página de cadastro – *layout*
- 146 | Montando o Index.html no Dreamweaver
- 147 | Finalizando





■ Introdução à rede

Rede é um sistema de comunicação que visa o compartilhamento de recursos, tais como: documentos, sistemas, banco de dados, impressoras, meios de armazenamento, acesso à internet, entre outros. Basicamente temos dois tipos de redes: LAN (*Local Area Network*) e WAN (*Wire Area Network*).

LAN

Muito utilizada dentro de empresas, permite o envio de informações de setor para setor com mais velocidade e segurança. A principal característica de uma rede LAN é que apenas os micros reconhecidos pelo servidor fazem parte dela. É formada pelas estações cliente, servidor, concentrador, placas de rede, cabos e sistema operacional de rede.

Numa rede LAN temos o processamento dos dados distribuído entre estações de trabalho e o controle dessas informações centralizado em um único computador, o servidor.

Vantagens da rede LAN

- **Economia** – não é necessário adquirir várias licenças de utilização dos softwares, pois uma licença pode ser compartilhada por vários usuários na rede.
- **Consistência** – o aplicativo é instalado e atualizado uma única vez, tendo a certeza de que todos os usuários estarão utilizando a mesma versão.
- **Economia de recursos** – com a instalação dos aplicativos na rede e o compartilhamento com vários usuários, o espaço do disco rígido do micro local pode ser aproveitado para armazenamento de outros arquivos.
- **Integridade dos dados** – a existência de uma única base de dados centralizada evita duplicidade de informações e as graves inconsistências geradas por esse problema.
- **Gerenciamento de backup** – o backup centralizado em um único ponto proporciona confiabilidade na atualização dos dados, além de facilitar a própria tarefa de realização deste.

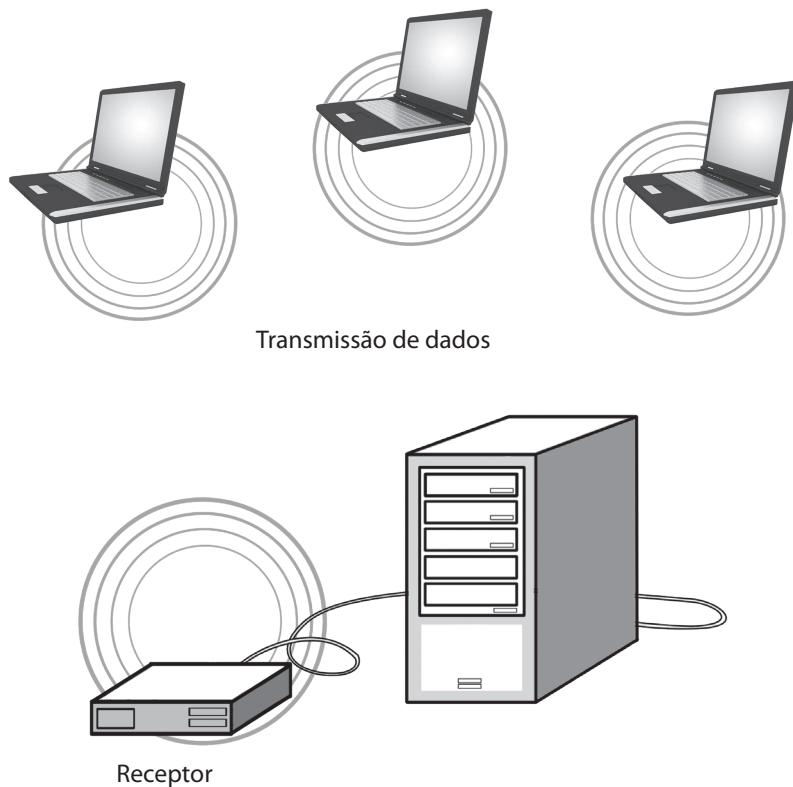
WAN

Caracteriza-se por uma conexão de longa distância, usando muitas vezes meios públicos de comunicação. Uma das maiores redes que utiliza esse método é a internet, onde a transferência das informações pode ser feita através da linha telefônica, permitindo que nos comuniquemos com qualquer parte do mundo.

As redes sem fio estão cada vez mais fazendo parte do nosso dia a dia. Um bom exemplo são as redes de celulares que hoje cobrem países inteiros através do sistema global de localização via satélite (*Global Positioning Satellite System* ou GPS). Conheça mais algumas tecnologias da rede *wireless*:

- **Bluetooth** – criada pela Ericsson na década de 1990, essa tecnologia utiliza transmissão via rádio de curto alcance, possibilitando o envio de dados entre computadores, telefones e outros. A tecnologia de rádio utilizada pelo Bluetooth usa um sistema de frequência de sinal que provê um *link* seguro e robusto, mesmo em ambientes com alto ruído e de grande interferência.
- **CDMA** (*Code Division Multiple Access*) – esta tecnologia oferece um sistema que, através de um código de identificação, diferencia várias conversas transmitidas ao mesmo tempo em um mesmo canal de rádio.
- **Cartão SIM** (*Subscriber Identity Module*) – podemos traduzir como *Módulo de Identificação do Assinante*. É um cartão inteligente que personaliza o telefone GSM, carregando todas as informações sobre o usuário (agenda, operadora, serviços aos quais tem direito etc.). Dessa forma o usuário pode levar consigo apenas o Cartão SIM e usá-lo em qualquer outro celular padrão GSM.
- **GSM** (*Global System for Mobile Communications*) – pela tradução: *Sistema Global para Comunicações Móveis*. Permite *roaming* mundial automático, ou seja, comunicação entre áreas de serviços diferentes.
- **SMS** (*Short Messaging Service*) – criado pela Nokia em 1993, oferece serviço de transmissão de mensagens curtas de texto entre telefones celulares.
- **WAP** (*Wireless Application Protocol*) – conhecido também como internet dos celulares, o *Protocolo para Aplicações Móveis* oferece transmissão de conteúdo da internet para telefones celulares, *palms* e outros aparelhos móveis.

A figura a seguir mostra como uma rede sem fio funciona.



Wagner Roberto de Carvalho.

Figura 1 – Funcionamento de uma rede wireless.

Perceba que temos o servidor (receptor) o qual se comunica com as estações através de sinais via rádio. Por sua vez, este pode estar conectado à internet oferecendo aos usuários da rede acesso ao e-mail, consultas bancárias e muito mais. Assim como em qualquer outra rede, os dados são sigilosos, portanto existem várias maneiras de se criptografar as informações que trafegam na rede.

Outro exemplo claro do uso dessa tecnologia é a utilização de aparelhos celulares para conexão com a internet e os *notebooks* que usam os serviços de Banda Larga em 3G para conexão.

Tipos de conexão

Existem alguns tipos de conexão disponíveis no mercado. Temos conexões com baixa taxa de transferência até conexões de altíssima velocida-

de. A seguir veremos alguns tipos de conexões e suas características.

- **Dial-up** – é a conexão por linha discada. É um tipo de conexão de baixo custo e necessita de um *modem* instalado na máquina além de uma linha telefônica para fazer a conexão. Essa conexão tem baixa taxa de transferência e muita instabilidade.
- **ISDN** – este tipo de conexão tem princípios similares à *dial-up*, mas oferece maior taxa de transferência e maior estabilidade já que a conexão é feita em duas vias, uma para envio e outra para recebimento de informação.
- **ADSL** – é uma tecnologia que utiliza uma linha telefônica digital para tráfego de dados em alta velocidade. O funcionamento desse serviço depende de um *modem* ADSL que é instalado na casa do usuário ou empresa, fazendo a conexão com a central telefônica, transformando a linha telefônica em apenas uma via de acesso entre os dois pontos. O *modem* ADSL encarrega-se de transmitir os dados em uma frequência diferente da frequência de transmissão de voz, o que permite ao usuá-rio navegar na internet e falar ao telefone simultaneamente.
- **Proxy** – é um sistema onde podemos dividir a conexão à internet com outras máquinas da rede. Essa é uma saída econômica utilizada em pequenas empresas. Porém, nesse tipo de conexão existe uma queda da taxa de transferência significativa.
- **Rádio** – a conexão via rádio está em plena expansão na atualidade, isso porque ela tem como princípio a distribuição de conexão em banda larga, sem a necessidade de cabos e fios. Somente uma pequena antena receptora e um decodificador de sinal são necessários para esse tipo de conexão.

Topologia estrela

A topologia estrela, entre as diversas topologias existentes (anel, barra-mento, estrela, árvore etc.) é a mais recomendada, já que os micros estão ligados a um dispositivo central (concentrador). Se algum cabo falhar não ocorrem danos graves, pois apenas aquela estação será desconectada da rede e não todas.

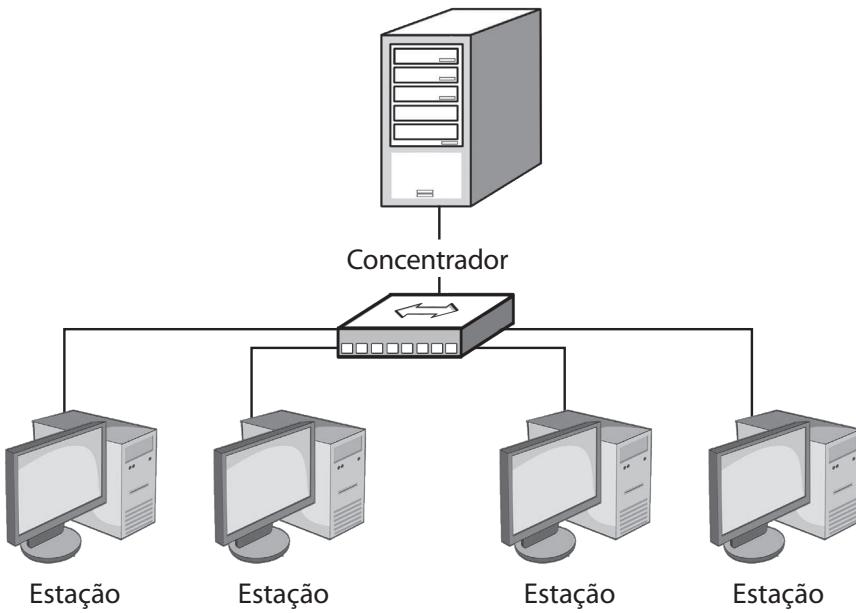


Figura 2 – Topologia estrela.

Essa topologia apresenta a vantagem de, se alguma das estações (computadores) apresentar problemas, somente esta ficará fora da rede. As demais estações permanecem conectadas. Nessa topologia, as estações são ligadas a um concentrador como ponto central da rede, o qual se encarrega de retransmitir os dados para todas as estações. Esse concentrador pode ser:

- **Hub** – permite a ligação de vários micros a um servidor.
- **Switch** – assim como o *hub*, permite a ligação de vários micros à rede, com a diferença que este também organiza o envio dos pacotes entre as estações.

Como é composta uma rede Client/Server?

- **Estações cliente/trabalho** – são os computadores utilizados pelos usuários, que utilizam os serviços do servidor.
- **Servidor** – é o principal computador na rede (geralmente possui alta velocidade). O servidor é quem irá processar o sistema operacional que gerencia a rede, além de gerar e administrar o fluxo de dados (pacotes).
- **Hub/switch** – para que seja possível a ligação de várias estações cliente a um servidor, é necessária a utilização do *hub* ou *switch*. Normalmente esses dispositivos se encontram próximo ao servidor.

- **Placas de rede** – é o componente de *hardware* que irá permitir a ligação entre as estações cliente e o servidor, através de um cabeamento.
- **Cabos** – é o meio físico por onde os dados (pacotes) irão trafegar entre as estações da rede.
- **Sistema operacional de rede** – é o sistema localizado no servidor que permite gerenciar todos os recursos das estações cliente e periféricos, como o acesso aos arquivos ou compartilhamento de periféricos (impressora, scanner etc.). Exemplos: Windows e Linux.

Parte lógica de uma rede

Após ter escolhido o meio físico da rede é necessário criar a rede lógica, ou seja, configurar os usuários, grupos de trabalho, servidor, acessos remotos e permissões. Com isso podemos garantir a segurança e integridade dos dados.

Uma das perguntas mais frequentes em relação a redes que escutamos é: como o servidor vai reconhecer outros micros conectados a ele? Como posso usar a impressora que está em outro micro? Por que não tenho direito aos mesmos arquivos e diretórios que os outros?

Configurando a rede

Sabemos que os arquivos no servidor não podem ser acessados por qualquer usuário da rede, pois isso poderia trazer graves problemas de segurança.

Depois de montada a parte física da rede, devemos fazer com que o servidor reconheça e identifique cada um dos micros e periféricos que estão conectados a ele. Cada micro terá um nome, pelo qual o servidor irá identificá-lo. Depois de identificados, devemos criar as contas de usuários. Veja alguns tipos de usuários de uma rede:

- **Administrador** – responsável pela manutenção do sistema, cadastramento e remoção de usuários, bem como as permissões que cada um terá.

- **Usuário** – todos os demais que trabalham na rede. Estes podem fazer parte de grupos de trabalho com permissões específicas para o grupo.

Cada usuário da rede terá um *log in* e senha para acessá-la, sendo que só poderá se logar na rede utilizando os micros identificados pelo servidor. Além disso, o usuário só poderá acessar os arquivos para os quais possui permissão. Veja alguns tipos de permissões abaixo:

- *Read Only* (Somente Leitura)
- *ReadWrite* (Leitura-Gravação)
- *Sharable* (Partilhável)
- *Copy Inhibit* (Inibir Cópia)

O administrador pode configurar as permissões para cada usuário da rede, sendo que cada um pode ter uma ou mais permissões em relação a cada arquivo compartilhado pelo servidor, além de permitir acesso à impressora, *scanner* e internet.

Exemplo de compartilhamento – impressora

Vamos imaginar que temos uma rede composta por 10 estações de trabalho mais o servidor. Vamos supor que na *estacao09* temos uma impressora instalada e queremos imprimir um documento que está na *estacao02*, como fazer?

Considerando que a rede está funcionando e que os usuários já foram configurados, basta indicarmos na *estacao09* que a impressora será compartilhada na rede, dando um nome a esse compartilhamento. Depois devemos incluir uma impressora na *estacao02* indicando que será uma impressora dentro da rede, conectada a outro micro, onde daremos o nome do compartilhamento.

Obs.: dependendo do sistema operacional os procedimentos para compartilhamento são diferentes.

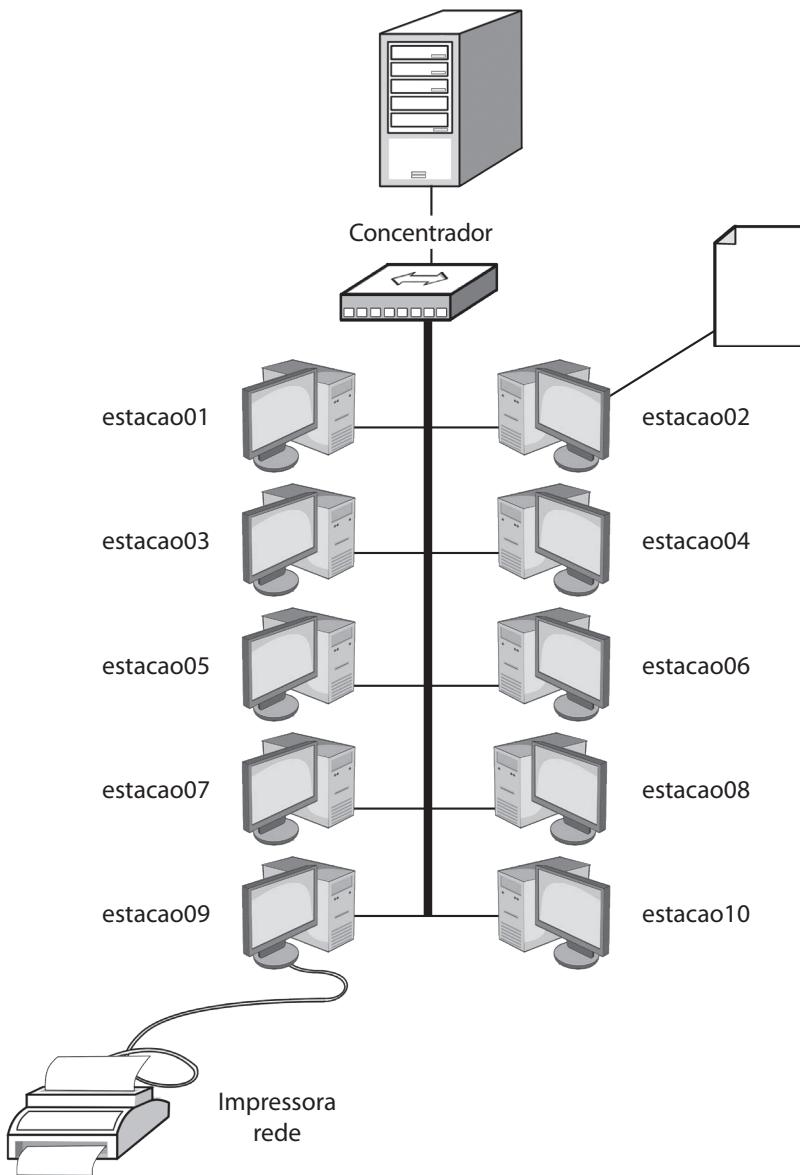


Figura 3 – Exemplo de compartilhamento em rede de uma impressora.

Atividades

1. Cite dois aparelhos usados para centralizar a transmissão dos dados na rede.
 - a) *Hub, servidor.*
 - b) *Servidor, switch.*

c) *Hub, switch.*

d) Placa de rede e cabos.

e) Servidor.

2. Como é composta uma LAN?

a) Estações cliente, servidor, concentrador, placas de rede, cabos e sistema operacional de rede.

b) Estações de trabalho, servidor, placas de rede, cabos e sistema operacional de rede.

c) Estações cliente, estações de trabalho, concentrador, placas de rede e sistema operacional de rede.

d) Usuários e administrador de rede.

e) Computadores.

3. Quais os usuários de uma rede e suas responsabilidades?

a) Usuários – utilizar os recursos que a rede oferece; Data base – responsável pela manutenção da rede.

b) Usuários – responsável pela manutenção da rede; Administrador – utilizar os recursos que a rede oferece.

c) Usuários – responsável por bagunçar a rede; Administrador – responsável pela manutenção da rede.

d) Usuários – utilizar os recursos que a rede oferece; Administrador – responsável pela manutenção da rede (física e lógica).

e) Pessoas.

4. Assinale os tipos de permissões que os usuários de uma rede podem ter.

a) *Read Only* e *ReadWrite*.

b) *Read Only*, somente leitura, *Sharable* e *Copy Inhibit*.

c) Somente leitura, somente escrita, partilhável, proibir cópia.

d) Permissão para deletar.

e) *Sharable, Read Only, Copy Inhibit* e *ReadWrite*.

5. Quais as vantagens que uma rede LAN oferece?
 - a) Não há vantagens.
 - b) Economia (custos e recursos), *backup*, consistência e integridade dos dados.
 - c) Só economia de custos.
 - d) Só economia de recursos.
 - e) Só facilita o *backup*.
6. Cite três tecnologias utilizadas na rede sem fio.
 - a) GPS, Cartão SIM e NÃO.
 - b) GPS, CDMA e WAN.
 - c) Internet, GPS e SMS.
 - d) Bluetooth, WAP e LMS (*Large Messaging Service*).
 - e) Bluetooth, GSM e WAP.
7. Quais aparelhos móveis utilizam as tecnologias de rede *wireless*?
 - a) Celular, *pendrive* e *palm*.
 - b) Celular, *notebook*, rádio e *palm*.
 - c) PC, *notebook*, *palm* e celular.
 - d) *Walkman*, BIP e celular.
 - e) Só o celular.
8. Quais das áreas abaixo utilizam uma ou mais das tecnologias citadas nesse capítulo?
 - a) Telefonia.
 - b) Transporte aéreo.
 - c) Telemarketing.
 - d) As alternativas a, b e c estão corretas.
 - e) Nenhuma empresa de respeito utiliza tecnologia.

9. Em uma rede sem fio, como impedir que qualquer aparelho receptor dentro da área de alcance possa ler os dados transmitidos?
- a) Não tem como outro aparelho, fora da rede, receber os dados.
 - b) Enviando as dados em diferentes frequências.
 - c) Através do uso de criptografia de dados.
 - d) Quebrando os dados em pacotes, e enviando esses pacotes fora de ordem.
 - e) Não transmitindo os dados.
-

Gabarito

- 1. C
- 2. A
- 3. D
- 4. E
- 5. B
- 6. E
- 7. C
- 8. D
- 9. C



■ Introdução à internet

Profissões da área de web

Hoje existem diversos cargos para o profissional de tecnologia que atua na área de internet. Esses cargos na maioria das vezes estão disponíveis em estúdios de criação, consultorias ou em empresas de grande porte, mas nada impede que você siga “carreira solo”, pois o mercado ainda consegue absorver um grande número de profissionais tendo em vista a demanda existente.

Web designer

É o profissional responsável pela criação do *layout* das páginas do site, tratamento das fotos, ajustes de tamanho de fonte, disposição do conteúdo das páginas, projeto do ambiente gráfico, identidade visual do site, entre outras atribuições. Esse profissional deve ter sólidos conhecimentos em ferramentas de criação de páginas (Dreamweaver), sólidos conhecimentos em Flash, além das ferramentas de editoração e preparação de fotos para web.

Em diversas empresas o *web designer* deve também ter conhecimento em algumas linguagens de programação como: JavaScript, ASP, JSP e noções de banco de dados como Oracle e SQL Server.

Web developer

É o profissional responsável pela criação da parte de *scripts*, que são os textos de criação de objetos no banco de dados como tabelas ou procedimentos por exemplo. Também trabalha nos processos e otimização de navegação do site, criação de rotinas para otimização, *scripts* de validação e envio, interação das regras do banco de dados com o site, desenvolvimento de aplicativos voltados à internet, intranets¹ e extranets, que são redes internas privadas mas que permitem o acesso de usuários externos através de softwares de controle de acesso que são chamados de *front-end*, entre outras atribuições.

¹ Redes internas e privadas.

Esse profissional deve ter sólidos conhecimentos em ferramentas de criação de páginas como Dreamweaver MX, InterDev, JBuilder. Além dessas ferramentas, o *web developer* deve ter sólidos conhecimentos das linguagens de programação utilizadas na internet como HTML, JavaScript, CSS, ASP, JSP, ASP. NET e conhecimentos intermediários em banco de dados como Oracle e SQL Server.

Web master

É o profissional responsável por projetar toda a estrutura dos sites e gerenciar todo o processo de criação do mesmo. Esse profissional deve ter sólidos conhecimentos nas principais ferramentas de criação e de programação, além de bancos de dados, redes e sistemas de segurança.

Por ser o profissional que organiza todo o projeto antes da criação, é também recomendável que ele tenha conhecimentos relacionados a UML, que é uma linguagem para especificação, documentação e desenvolvimento de sistemas orientados a objeto. Em diversos casos o *web master* gerencia todo o projeto, distribuindo as responsabilidades entre os desenvolvedores e *designers*.

Por que criar um site

Hoje vivemos em um mundo onde a concorrência é acirrada e todos necessitam de grande volume de informação *on-line*, ou seja, para uma empresa de qualquer porte é imprescindível estar na internet. Isso porque o cliente atual não é mais aquele que procura um produto por meio de uma “lista telefônica”, mas sim procura produtos e serviços na internet. Essa atitude se deve a vários fatores podendo ser citadas a comodidade, a agilidade etc.

A empresa que ainda está fora da *web* está perdendo grandes chances de negócio, porque com certeza um ou mais concorrentes diretos ou indiretos dessa empresa já estão divulgando seus produtos e serviços na rede mundial.

A agilidade na troca de informações também é um fator de grande “peso”, isso porque, as grandes indústrias têm um sistema totalmente interligado com principais clientes e fornecedores, agilizando assim todo seu processo e diminuindo seus custos operacionais, o que resulta em produtos mais baratos e mais acessíveis.

Vamos imaginar uma empresa que fabrica sapatos. Essa empresa precisa de matéria-prima como solas, couro, tecidos, tintas etc. No entanto, esta empresa não tem um sistema *on-line* como, por exemplo, uma extranet. As compras sempre são efetuadas de um mesmo grupo de fornecedores, mas toda vez que o estoque está baixo o comprador consulta em uma agenda telefônica os números de telefone de fornecedores e os contata, solicitando o preço dos itens em baixo nível no estoque. Após várias ligações e variados orçamentos, com diferentes preços e condições, o comprador analisa as informações obtidas e opta por um fornecedor.

Para todo esse processo o comprador perdeu horas e mais horas do seu dia (digamos que, no mínimo, umas seis horas). Agora vamos imaginar que esta empresa efetue compras duas vezes por semana. Sendo assim, serão 48 horas por mês perdidas para cada funcionário do departamento de compras, o que equivale a dois dias de trabalho no telefone, ou seja, a cada 30 dias trabalhados, dois ele passa no telefone.

Supondo que essa empresa tivesse um sistema interligado com a extranet. Quando o estoque ficasse abaixo do ideal, o sistema geraria uma solicitação de orçamento automaticamente com todos os itens e enviaria por e-mail para todos os fornecedores cadastrados. Esses fornecedores, por sua vez, enviariam as informações diretamente para o sistema o qual geraria automaticamente um relatório para o comprador analisar. Após a aprovação do orçamento, o comprador confirma o orçamento via sistema, o que gerará um comunicado para a logística de quantos dias serão necessários para a reposição do estoque, para o setor financeiro sobre o valor total da compra e o nome do favorecido e, também, o fornecedor recebe uma confirmação de efetivação da compra.

Nesse caso, a empresa teria uma redução de custo com pessoal significativa, além de ganhar agilidade em toda sua cadeia de produção. A mesma situação pode ser utilizada para o departamento de vendas: ao invés do vendedor preencher um pedido em papel e depois passar esse pedido para o computador, o vendedor poderia ter acesso à extranet e, em qualquer lugar com conexão à internet, passar os seus pedidos para a empresa, ganhando assim agilidade no processo de produção e logística. É claro que esse caso pode ser aplicado a diversos tipos de negócios, mas todo desenvolvimento deve ser minuciosamente projetado para evitar falhas. Veja um diagrama do exemplo que acabamos de conhecer:

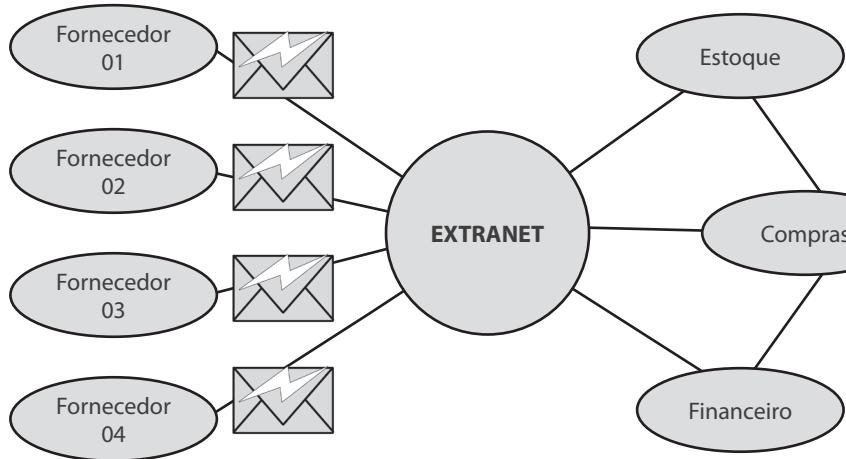


Figura 1 – Exemplo de extranet.

O que é a internet

Todos nós ouvimos falar sobre internet, acesso rápido, e-mail, entre outros termos que são usados diariamente para referir-se ao ambiente web. A internet nada mais é que uma rede mundial de computadores, que conta com diversas redes espalhadas pelo mundo todo. Essas redes, em conjunto ou não, trocam informações de todos os tipos 24 horas por dia, criando assim uma estrutura de informações que pode ser acessada de qualquer lugar do planeta. Em síntese, podemos dizer que a internet é uma rede mundial de computadores que colaboram entre si.

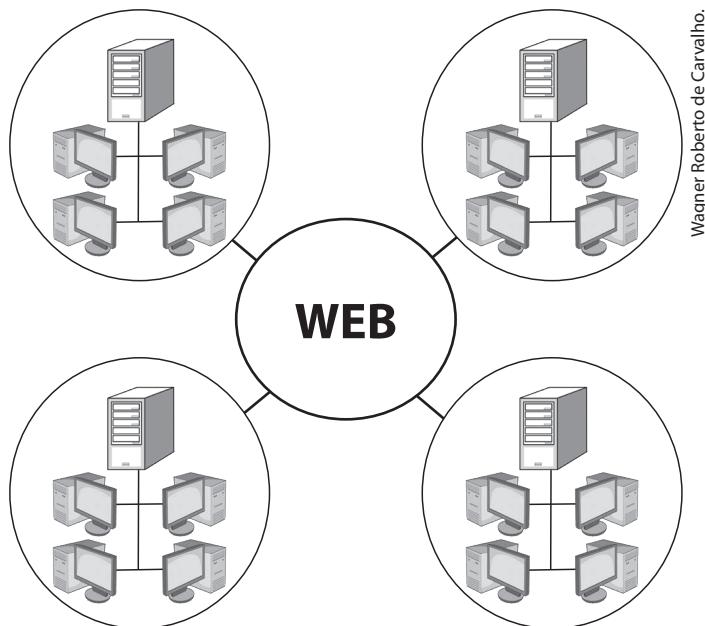
Com o passar do tempo foram surgindo necessidades de serviços adicionais dentro da internet, daí começaram a ser desenvolvidos sistemas para auxiliar o usuário da internet em tarefas do dia a dia. Um desses serviços que todos nós conhecemos é o e-mail, que hoje é usado diariamente por milhões de usuários da internet no mundo todo, agilizando a comunicação entre as pessoas e gerando negócios.

Redes e conexão

Redes de internet

As redes na internet são um pouco complexas porque existem redes diferentes colaborando entre si, trocando informações e organizando o envio

e recebimento no tráfego de informações. Na imagem a seguir temos uma ideia de como funciona isso:



Wagner Roberto de Carvalho.

Figura 2 – Exemplo de rede.

Tráfego

Toda informação que trafega na internet é dividida em várias partes que são chamadas de *pacotes*, com isso a transferência se torna rápida. Os pacotes são criados pelo TCP (*Transfer Control Protocol*) e todo arquivo que trafega na rede utiliza esse conceito.

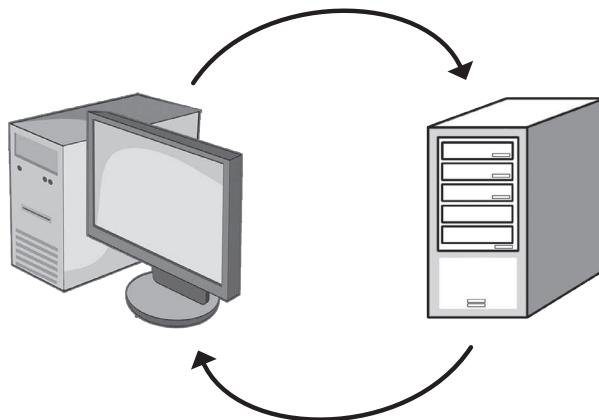
Quando você faz um *download*, o servidor que tem esse arquivo o divide em pequenas partes (pacotes). O cabeçalho é chamado de pacote “especial”, pois contém as informações como o nome, o tamanho e a quantidade de pacotes em que esse arquivo foi dividido.

Após serem criados os pacotes eles são enviados para seu computador, que recebe um a um até que todos sejam recebidos. Após este recebimento total, o arquivo é recomposto e está pronto para ser utilizado.

Cliente-servidor

A arquitetura cliente-servidor é a base da existência da internet. Quando você acessa um site você é o *cliente* que faz uma requisição de página para

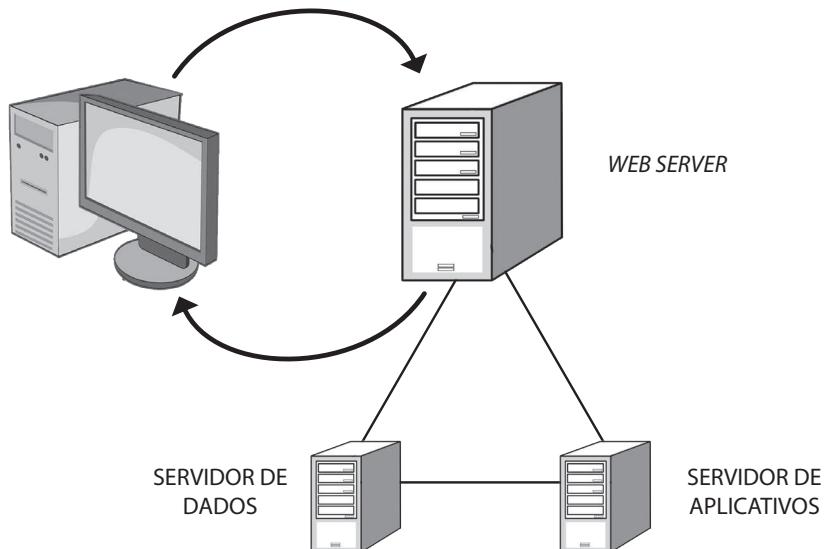
um *servidor*. Os servidores de *web* podem ser de diversos tipos: de páginas, de dados, de aplicativos etc. Veja como funciona:



Wagner Roberto de Carvalho.

Figura 3 – Exemplo de arquitetura cliente-servidor.

A arquitetura apresentada na figura acima poderia ser maior se ela contasse com outros servidores como o de aplicativos e o de dados, veja:



Wagner Roberto de Carvalho.

Figura 4 – Exemplo de arquitetura cliente-servidor considerando outros servidores.

Estrutura de web

Para a internet funcionar é preciso uma estrutura de redes para a qual são necessários vários equipamentos. Vejamos os principais:

Roteadores

Os roteadores têm a função de apontar um caminho, ou melhor, dizer uma rota entre um cliente e um servidor na web. Em qualquer conexão de web, vários roteadores trabalham em grupo para que seja feita uma conexão, isso porque os roteadores fazem requisições entre si para criar a conexão, como é demonstrado na figura abaixo:

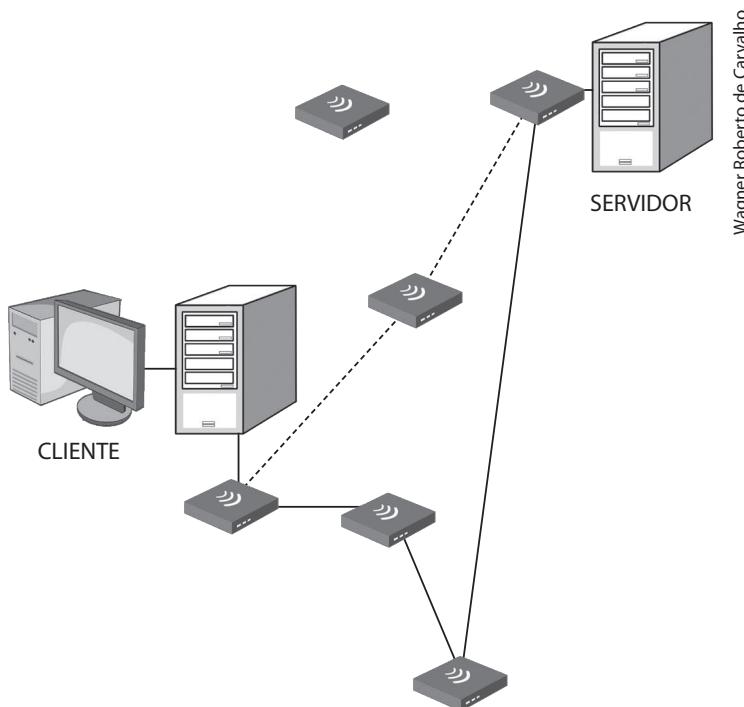


Figura 5 – Função dos roteadores.

Caso um roteador tenha uma queda durante a transmissão, a conexão é cancelada e uma nova “rota” é criada sem a utilização daquele roteador. Veja a seguir uma foto de um *rack* com vários roteadores:



Wikimedia Commons/Misternome

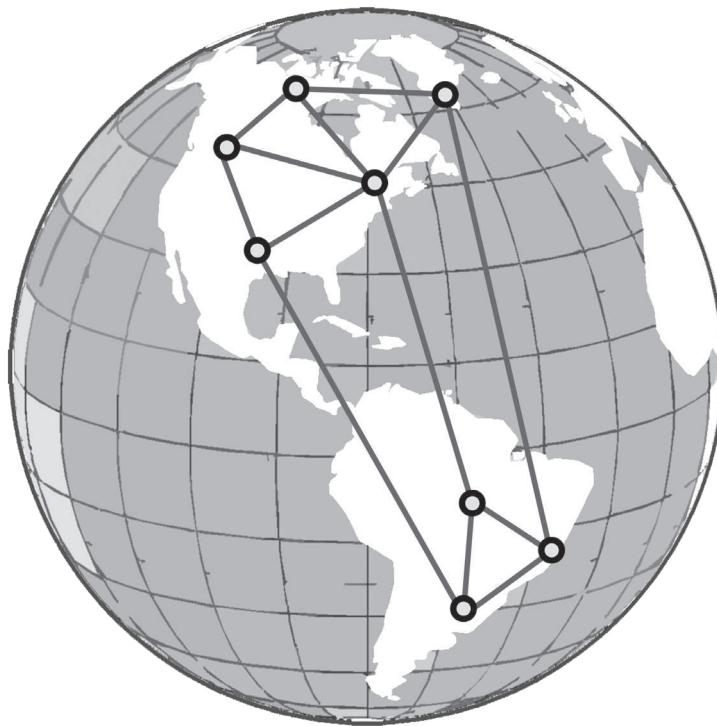
Figura 6 – Rack com vários roteadores.

Switch

O *switch* é peça fundamental de uma rede, pois nele são ligados todos os cabos de rede como uma “central”, a qual gerencia as informações que trafegam entre os computadores. Cada máquina possui seu endereço de IP (*Internet Protocol*) que é um endereço físico e único utilizado pelo protocolo TCP/IP para a comunicação e conexão entre as máquinas.

Backbone

O *backbone* é uma estrutura de distribuição de conexão de alta velocidade. É considerado como uma espinha dorsal com várias conexões que são subdivididas e distribuídas para os usuários. Os *backbones* também se comunicam entre si, via cabo, rádio ou satélite fechando assim a rede mundial de computadores. Veja a seguir uma pequena estrutura de *backbones*:



Wagner Roberto de Carvalho.

Figura 7 – Exemplo de estrutura *backbone*.

Tipos de acessos

Quando utilizamos recursos e serviços da internet, devemos usar endereços para acessar informações. Cada tipo de informação deve ser acessada de uma forma diferente através de um protocolo internet. Entre os principais podemos citar:

- **http** – é o protocolo que utilizamos para acessar páginas de internet em geral.
- **ftp** – é o protocolo utilizado para a transferência de arquivos de uma estação de trabalho para um servidor de web.
- **mailto** – é o protocolo que utilizamos para envio e recebimento de mensagens (e-mails), normalmente trabalha com mais dois serviços: o POP (*Post Office Protocol*) e o SMTP (*Simple Mail Transfer Protocol*).
- **news** – é um protocolo usado para o envio e recebimento de notícias.
- **telnet** – é um protocolo utilizado para a conexão de servidores remotos.

As páginas *web* e os *hosts* (endereços físicos dos computadores) que compõem a internet devem ter a mesma localização para que o seu computador possa localizar e recuperar as informações. A única coisa que permite que um *host* seja identificado é o endereço IP, e o único modo de identificar uma página ou um *host* é o URL (*Uniform Resource Locator*).

O URL funciona como um correio normal ou e-mail. Assim como os endereços de correio e de e-mail se referem a um lugar específico, um URL ou endereço *web* indica onde está localizado o computador, o nome da página *web* e o tipo de arquivo de cada documento. Normalmente, o URL tem o seguinte aspecto: <http://wwwaaaaaaaaa.com.br/imagens/logo.jpg>

Usando como exemplo a URL: <http://wwwaaaaaaaaa.com.br/imagens/logo.jpg>, sua primeira parte no URL indica que tipo de protocolo de transferência será usado para resgatar o documento ou arquivo especificado, neste caso o http.

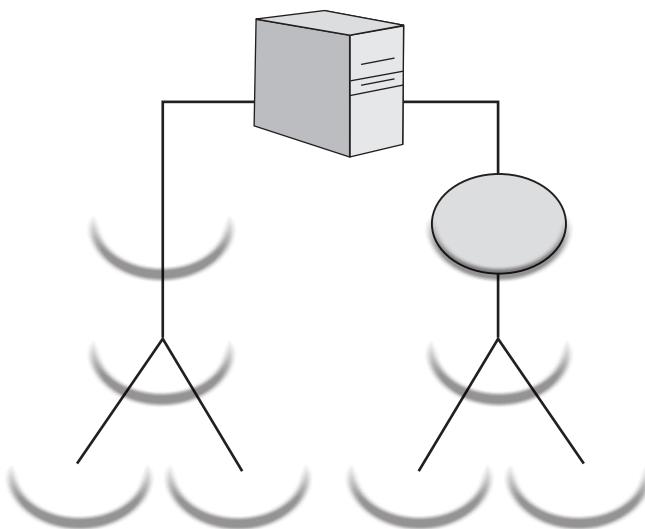
A segunda parte refere-se ao computador *host* específico onde reside o arquivo que deve ser conectado pelo navegador. Esta parte do endereço também é chamada de nome de referência e é constituída por nomes de domínios. No exemplo, wwwaaaaaaaaa.com.br

A terceira parte é o diretório do computador *host* que contém um *web site* específico ou múltiplos *web sites*. Ela fica sempre localizada depois do primeiro corte no URL e é essencialmente o subdiretório do disco rígido que abriga o *web site*. No exemplo dado é a parte “imagens”.

Os subdiretórios também podem ser indicados nesta parte do endereço. Obrigatoriamente, o subdiretório indicado no URL não precisa existir fisicamente. Ele pode ser um subdiretório virtual gerenciado pelo *webserver*. A última parte representa o arquivo que será retornado usando o protocolo indicado na primeira parte do URL.

Plataformas

Temos hoje na internet alguns tipos de servidores de *web* que “rodam” em duas plataformas distintas, os mais difundidos são o IIS (*Internet Information Server*) e o Apache. Os dois servidores são utilizados em toda a internet. Existem algumas diferenças entre eles, porém é difícil avaliar qual o melhor ou pior, pois isso vai depender de vários fatores. Cada um desses servidores tem compatibilidade com algumas linguagens de internet, veja:



Wagner Roberto de Carvalho.

Vamos imaginar o seguinte exemplo: Pedro desenvolve um *software* de envio de e-mail *opensource* e *freeware*. Marcelo pega uma cópia desse *software* e faz mudanças em seu código-fonte (que pode ser C, C++, VB, Delphi, Java etc.) de forma a melhorar o programa. Como o *software* é *opensource* ele não pode comercializá-lo, pois estaria obtendo lucro com o programa que Pedro desenvolveu. O que Marcelo deve fazer é enviar as alterações para o “dono” (Pedro) do *software*, que irá analisar as mudanças e, se necessário, lançar um novo produto.

História do Linux

Um dos maiores exemplos de *software opensource* (código aberto) no mundo é o sistema operacional Linux. Antes de falarmos sobre ele vamos ver um pouco de sua história. Em 1983, Richard Stallman fundou a *Free software Foundation* (Fundação de *software Livre*), cujo projeto GNU (GNU is not Unix – a sigla foi inspirada no gnu, um mamífero africano), teria a finalidade de criar um clone melhorado e livre do sistema operacional Unix, sem a utilização do seu código-fonte original. Esse era um grande desafio, já que precisaria desenvolver, por completo, o *kernel* (núcleo do sistema operacional que controla o *hardware*), utilitários de programação, administração do sistema, conexão em rede, comandos-padrão etc.

No final da década de 1980 o projeto havia fracassado, já que apenas uma parte do *kernel* havia sido desenvolvida. Na mesma época havia muitas outras tentativas para se desenvolver um clone do Unix, uma delas era do Dr. Andrew Tanenbaum, que desenvolveu o Minix. O Minix trazia pouquíssimos recursos, por isso era utilizado apenas como uma base de estudos.

Em agosto de 1991, Linus Torvalds, aluno da universidade de Helsinque, na Finlândia, estava decidido a construir um *kernel* clone do Unix que pudesse disponibilizar memória virtual, multitarefa e capacidade para ser multiusuário. Era um trabalho quase impossível para uma única pessoa, mesmo que estivesse familiarizado com as complexidades dos sistemas operacionais, então iniciou seu projeto particular, inspirado no seu interesse pelo Minix, definindo seu projeto como “um Minix melhor que o Minix”.

Após um bom tempo de trabalho em seu projeto solitário, Linus Torvalds foi capaz de criar um *kernel* que permitia executar utilitários de programação e os comandos-padrão do Unix clonados pelo projeto GNU. Reconhecendo

que não seria possível desenvolver sozinho, ele procurou por colaboradores na lista de discussão “comp.os.minix”, para participarem no desenvolvimento desse novo sistema. Em 5 de Outubro de 1991, lançou a versão “oficial” do Linux (versão 0.02), e a partir dessa data muitos programadores no mundo inteiro têm colaborado e ajudado a fazer o Linux que nos é disponibilizado hoje.

Linux de hoje

Podemos encontrar várias versões do Linux, todas elas *opensource* e algumas *freeware*, todas desenvolvidas em cima de um mesmo *kernel*, que pertence ao Linus Torvalds (chamado “pai do Linux”). Por todas essas versões possuírem parte do código que Linus Torvalds desenvolveu, elas não podem ser comercializadas. O que as empresas fazem é vender o suporte técnico para suas versões, ou seja, quando você paga para ter Linux em sua máquina, na verdade você está pagando pelo suporte técnico que a empresa oferece, caso haja algum problema.

Pelo Linux possuir uma falta de utilitários de configuração, esse suporte se torna necessário, já que muitas configurações de *hardware* devem ser feitas utilizando linha de comando. Diferentemente do Windows, que já possui uma variedade de *softwares* gráficos para essas finalidades. As versões mais conhecidas do Linux são:

- Red Hat
- Connectiva
- Kurumin
- SuSE
- Debian
- Mandrake

Com o passar do tempo muitas empresas procuraram desenvolver, cada vez mais, utilitários para ajudar na configuração e utilização do Linux. Muitos desses utilitários *opensource* e/ou *freeware*, outros não. Algumas ferramentas utilizadas no Linux são o *OpenOffice* (equivalente ao Office no Windows), *MySQL* (também usado no Windows), *Java*, *Mozilla* (navegador web) entre outras.

Vantagens e desvantagens de utilizar *software livre*

A maior vantagem de se utilizar um *software livre* seria o custo pela licença por utilizá-lo, o que em muitos casos é praticamente zero. Hoje o Linux possui um grande suporte para redes como servidor de sites, e-mail, serviços de *firewall*, serviços de compartilhamento e protocolo TCP/IP. A desvantagem, em se utilizar *software livre* está na responsabilidade pelo mesmo, pois caso você adquira uma versão desse *software* (gratuitamente) e ele venha lhe causar prejuízo no futuro, ninguém se responsabilizará pelos prejuízos. Esse é um dos maiores medos que as empresas possuem em relação a esse tipo de *software*. Outro motivo seria em relação ao suporte técnico, já que o *software* é *freeware* fica difícil encontrar especialistas capazes de oferecer um ótimo serviço.

Segurança

Toda vez que uma empresa conecta sua rede interna de computadores (LAN) à internet, enfrenta um perigo em potencial. Em virtude da abertura da internet, toda a rede da empresa conectada a ela fica vulnerável a ataques externos. Existem os *hackers* que são invasores de redes éticos e *crackers* que são invasores criminosos que roubam ou destroem dados. Os invasores então podem violar a rede corporativa e prejudicá-la de várias formas: roubar ou danificar dados importantes, danificar computadores pessoais ao longo de toda a rede, usar recursos dos computadores da empresa ou usar a rede e os recursos da empresa para se fazer passar por um funcionário da corporação.

A solução não é desligar a rede da internet. A empresa pode construir bloqueios para proteger sua rede destes ataques. Esses bloqueios são os *firewalls* (paredes de fogo) e permitem que qualquer um na rede corporativa accesse a internet, porém impedem que *hackers* ou outros usuários da internet obtenham acesso à rede da empresa e provoquem danos.

Os *firewalls* são combinações de *hardware* e *software* construídos a partir de roteadores, servidores e uma variedade de *softwares*. Eles são instalados no ponto mais vulnerável entre a rede corporativa e a internet, e podem ser simples ou complexos, conforme a vontade do administrador que os cria. Existem vários tipos diferentes de *firewalls*, mas a maioria deles tem alguns elementos em comum.

Um dos tipos mais simples de *firewalls* emprega um filtro de pacotes. No filtro, um roteador seletivo examina o cabeçalho de cada pacote de dados que viaja entre a internet e a rede corporativa. Os cabeçalhos dos pacotes contêm informações como o endereço IP do emissor e do receptor, o protocolo que está sendo usado para enviar o pacote e outras informações similares.

Com base nessas informações, o roteador sabe que tipo de serviço está sendo usado para enviar os dados, bem como a identidade do emissor e do receptor dos dados. Baseado nessas informações, o roteador pode impedir que certos tipos de pacotes sejam enviados entre a internet e a rede corporativa. O roteador pode, por exemplo, bloquear qualquer tráfego, exceto para correio eletrônico, ou bloquear o tráfego “de” e “para” destinos suspeitos.

Existem tipos ou gerações diferentes de proteção. Veja a descrição de algumas.

- **Filtros de pacotes:** essa geração de *firewall* somente controla a origem e o destino dos pacotes de mensagens da internet. Hoje, esse trabalho pode ser realizado facilmente por roteadores que são programados para filtrar os pacotes. Não é aconselhável o uso apenas de roteadores para garantir a segurança da rede.
- **Proxies:** são responsáveis pela conexão da rede corporativa com a internet através de estações seguras que realizam filtros do tráfego. Através deles, o usuário pode se comunicar com sistemas seguros que escondem as informações de qualquer outra pessoa que não possa ter acesso a elas.
- **Gateway:** além de funcionar como um proxy ele também é responsável por fazer a comunicação entre duas redes.
- **Firewall de terceira geração:** este tipo de proteção suporta uma tecnologia chamada SMLI (*Stateful Multi-Layer Inspection*). Isso significa que o *firewall* examina cada pacote em todas as suas camadas, desde o transporte até a aplicação, sem a necessidade de processar a mensagem. Para isso, ele usa algoritmos de verificação de tráfego otimizados e compara os pacotes a padrões conhecidos como amigáveis. Isso dá uma ótima *performance* como os filtros de pacotes, mas também uma ótima segurança assim como os *proxies*.

Atividades

1. Quais dos profissionais abaixo fazem parte da área de *web*?
 - a) *Web master, web developer* e administrador de rede.
 - b) *Web master, programador web e web developer.*
 - c) *Web master, web designer e web developer.*
 - d) *Web master, analista web e programador web.*
 - e) DBA *web, programador web e analista web.*
2. Qual o nome dado às estruturas de distribuição de alta velocidade da internet?
 - a) Servidor.
 - b) Roteador.
 - c) *Switch.*
 - d) *Backbones.*
 - e) *Hub.*
3. Cite cinco linguagens utilizadas no desenvolvimento de páginas para *web*.
 - a) JSP, PHP, ASP, HTML e VB.
 - b) Delphi, VB.NET, ASP.NET, JSP e HTML.
 - c) ASP.NET, JSP, PHP, ASP e VB.NET.
 - d) ASP.NET, JSP, PHP, ASP e C.
 - e) ASP.NET, Java, PHP, VB e Pascal.
4. Selecione a alternativa que contenha o conceito sobre o que é a internet.
 - a) Uma rede telefônica.
 - b) Uma rede sem fio de longa distância.
 - c) Só serve para mandar e-mail.

- d) Vários computadores ligados entre si, que acessam um único servidor central.
 - e) Uma rede mundial de computadores.
5. De que maneira as informações são transmitidas pela internet?
- a) Via e-mail.
 - b) Através de pacotes de dados.
 - c) O protocolo TCB quebra os dados em *packages* (pacotes).
 - d) Via protocolo IP.
 - e) Através dos protocolos TCP/IP de uma só vez.
6. Qual a diferença entre *roteador* e *switch*?
- a) Roteador escolhe a melhor rota para chegar no destino e o *switch* só indica qual é o destino.
 - b) *Switch* escolhe a melhor rota para chegar no destino e o roteador só indica qual é o destino.
 - c) Roteador é usado na internet e o *switch* não.
 - d) Roteador não é usado na internet e o *switch* sim.
 - e) Não há diferença.
7. Cite três tipos de acesso a páginas utilizados na internet.
- a) TCP, mailto e news.
 - b) HTTP, FTP e mailto.
 - c) ADSL, FTP e HTTP.
 - d) News, mailto e dial-up.
 - e) WWW, HTTP e FTP.
8. Cite três tipos de conexões usadas na internet.
- a) Dial-up, ADSL e WWW.
 - b) ISDN, Proxy e mailto.
 - c) ADSL, ISDN e TCP.

d) Via Rádio, Proxy e HTTP.

e) Dial-up, ADSL e via rádio.

9. Quais as plataformas *web* mais utilizadas hoje em dia?

a) IIS e Apache.

b) Linux e Windows.

c) Novell e Linux.

d) Apple e Solaris.

e) Internet Explorer e Mozilla.

10. O Linux é:

a) só *opensource*.

b) *opensource* e *freeware*.

c) só *freeware*.

d) nem *opensource* e nem *freeware*.

e) *opensource* (código-fonte fechado) e *freeware* (pago).

Gabarito

1. C

2. D

3. C

4. E

5. B

6. A

7. B

8. E

9. B

10. B



■ Endereços e domínios

Na internet temos milhões de sites espalhados no mundo todo e cada um desses sites tem um endereço o qual é conhecido por domínio (como, por exemplo, o endereço de uma residência). Os domínios de web são nomes que são seguidos por tipo e região conforme mostra a figura a seguir:



Figura 1 – Exemplo de estrutura de domínio.

Esses domínios foram criados para facilitar a navegação na internet, isso porque, na verdade, quando você acessa um site, existe um serviço que troca o nome que você digitou por um número (o TCP/IP). Esse serviço é conhecido como DNS (*Domain Name Service*), o qual foi desenvolvido pela Sun®, e sem ele seria muito complicado navegar na internet, pois seria necessário o auxílio de uma lista ou manual.

Domínios de região

Para cada região do mundo temos um domínio específico que faz referência ao nome do país de registro. Por exemplo, na Itália temos “*nome.com.it*”, na Inglaterra temos “*nome.com.uk*”, e assim por diante. No Brasil, o domínio de região é o “*br*”, ele define que os sites com essa terminação estão registrados no Brasil.

Tipos de domínios

Em cada região do mundo temos diversos tipos de sites que são criados todos os dias. Para facilitar o registro e organizar melhor toda a estrutura da internet, foram criados tipos de domínios que definem o tipo da empresa e do site publicado. No Brasil temos uma grande variedade desses tipos, veja:

Tipos para instituições

Aplicáveis somente para pessoas jurídicas.

AGR.BR	Empresas agrícolas, fazendas
AM.BR	Empresas de radiodifusão sonora
ART.BR	Artes: música, pintura, folclore
EDU.BR	Entidades de ensino superior
COM.BR	Comércio em geral
COOP.BR	Cooperativas
ESP.BR	Esporte em geral
FAR.BR	Farmácias e drogarias
FM.BR	Empresas de radiodifusão sonora
G12.BR	Entidades de ensino de primeiro e segundo grau
GOV.BR	Entidades do governo federal
IMB.BR	Imobiliárias
IND.BR	Indústrias
INF.BR	Meios de informação (rádios, jornais, bibliotecas etc.)
MIL.BR	Forças Armadas Brasileiras
NET.BR	Detentores de autorização para o serviço de Rede e Circuito Especializado da Anatel e/ou detentores de um Sistema Autônomo conectado à internet conforme o RFC1930 (RFC – Request for comments – documentos catalogados e seguidos como norma pela comunidade da internet.)
ORG.BR	Entidades não governamentais sem fins lucrativos
PSI.BR	Provedores de serviço internet
REC.BR	Atividades de entretenimento, diversão, jogos etc.
SRV.BR	Empresas prestadoras de serviços
TMP.BR	Eventos temporários, como feiras e exposições
TUR.BR	Entidades da área de turismo
TV.BR	Empresas de radiodifusão de sons e imagens
ETC.BR	Entidades que não se enquadram nas outras categorias

Tipos para profissionais liberais

Somente para pessoas físicas.

FND.BR	Fonoaudiólogos
FOT.BR	Fotógrafos

FST.BR	Fisioterapeutas
GGF.BR	Geógrafos
JOR.BR	Jornalistas
LEL.BR	Leiloeiros
MAT.BR	Matemáticos e estatísticos
MED.BR	Médicos
MUS.BR	Músicos
NOT.BR	Notários
NTR.BR	Nutricionistas
ODO.BR	Dentistas
PPG.BR	Publicitários e profissionais da área de propaganda e marketing
PRO.BR	Professores
PSC.BR	Psicólogos
QSL.BR	Radioamadores
SLG.BR	Sociólogos
TRD.BR	Tradutores
VET.BR	Veterinários
ZLG.BR	Zoólogos

Tipos para pessoas físicas

Somente para pessoas físicas.

NOM.BR Pessoas físicas

Registrando um domínio

Para registrar um domínio de *web* você deve antes de qualquer coisa fazer uma busca pelo nome que você deseja registrar, no Brasil o órgão gestor é o Registro.br.

Órgão gestor

O órgão gestor é responsável pelo registro e apontamento dos domínios no território nacional, é ele que faz o registro “de nascimento” do seu site.

O site da Fapesp (entidade mantenedora – pessoa jurídica – que controla este site) é <http://www регистрації.br>, lá você deve iniciar uma busca digitando o nome do site que você deseja registrar, caso o nome já esteja sendo utilizado o site retornará os dados do detentor desse domínio, como telefone, nome, endereço. Caso o domínio não tenha um detentor será exibida uma mensagem de domínio inexistente.

Criação de ID

Todo domínio deve ter alguém que responda por ele, e cada órgão gestor tem regras para o “administrador” do domínio. Esse administrador é encarregado de criar o apontamento do domínio para o servidor, também é ele que configura o DNS e que define quem será responsável pelo pagamento das taxas de inscrição e de manutenção do domínio. Para isso é necessária a criação dos ID's, que são usuários cadastrados no órgão gestor. Esse cadastro é gratuito e deve ser feito antes do cadastro do domínio. Existem atualmente três tipos de ID's que respondem pelo site: o responsável legal pelo site, que é o ADMIN; o que cuida da manutenção, que é o TÉCNICO, e é possível ter um específico para COBRANÇA.

Gerenciando domínios

Após ter criado seu ID você poderá prosseguir com o registro. Nessa etapa serão solicitadas informações da empresa detentora do domínio com CNPJ, endereço, responsável legal pela empresa, telefone etc., como mostrado na figura a seguir:

registro.br
Registro de Domínios
para a Internet no Brasil

Núcleo de Informação e Coordenação do Ponto BR
CGLbr - NIC.br - Registro.br - CERT.br - CETIC.br - CEPTRO.br - W3C.br

Ir para o conteúdo | [Impressa](#)

Você está em: Registro.br > Sistema > Novo Domínio

Escolha do domínio → Acesso ao sistema → Registro do domínio

Novo Domínio

Id: VEEBE2
15/10/2010 13:49:10
Tela Principal

Domínio

Domínio: autatest .com.br

Período:

Informações sobre a Entidade

Pessoa: Jurídica Física

CNPJ: **CHECAR**

Razão social:

Responsável:

Endereço

CEP: **CHECAR**

Logradouro:

Número: Complemento:

Cidade: UF:

Telefone

DDD: Telefone: Ramal:

Informações sobre os Contatos

Administrativo VEEBE2
PESQUISAR

Técnico: **PESQUISAR**

Cobrança: **PESQUISAR**

DNS
É obrigatória a delegação dos servidores Master e Slave 1.

Servidor	Nome
Master	<input type="text"/>
Slave 1	<input type="text"/>

+ DNS **EXIBIR IPS** **+ DNSSEC**

Contrato

\$Revision: 756 \$
\$Date: 2010-09-30 18:28:07 -0300 (Thu, 30 Sep 2010) \$
CONTRATO PARA REGISTRO DE NOME DE DOMÍNIO SOB O ".BR"

Li e aceito todos os itens do contrato

ENTRAR **LIMPAR**

Figura 2 – Página de registro do domínio do site <www.registro.br>.

Após o preenchimento de todos os campos você receberá um e-mail contendo instruções para a confirmação de sua solicitação. Quando você registra um domínio, a publicação dele é feita em até 20 horas, isso porque após a publicação pelo órgão gestor deve-se considerar o tempo para que esse apontamento seja distribuído por toda a internet.

Linguagens utilizadas na web

Atualmente temos várias linguagens que são utilizadas na internet. Essas linguagens têm como função criar páginas e conteúdos para os sites de internet. A seguir veremos algumas dessas linguagens.

HTML

HTML é a sigla de *Hyper Text Markup Language*. É a linguagem padrão para todos os navegadores de web. Considerada uma linguagem simples, trabalha normalmente com outras linguagens como JavaScript, ASP, JSP.

JavaScript

O JavaScript é uma linguagem de *script* voltada ao processamento de objetos na página. Sua utilização é mundial.

ASP

O ASP é uma linguagem voltada à criação de páginas dinâmicas na internet. Com ele é possível criar sites que são atualizados automaticamente, pois as informações exibidas nas páginas do site são fornecidas por um banco de dados.

JSP

O JSP também é uma linguagem voltada à criação de páginas dinâmicas e vem ganhando espaço por trabalhar com objetos baseados em Java, que é outra linguagem bastante usada na rede nas arquiteturas cliente-servidor do lado servidor.

PHP

Com o PHP é possível criar páginas dinâmicas com acesso a dados. Essa linguagem ganhou seu espaço na internet por ter compatibilidade com o servidor Apache.

ASP.Net

Atualmente o ASP.Net é a linguagem mais utilizada comercialmente. Ela é baseada em VB.Net (Visual Basic.Net). Existem inúmeras diferenças entre o ASP convencional e o .NET. Essa nova versão do ASP é baseada em objetos, como o JSP, porém ele é desenvolvido em ambiente VB.net, que também teve grandes mudanças.

Ferramentas

Para facilitar a criação dos sites, hoje contamos com várias ferramentas de desenvolvimento voltadas para internet. A seguir vamos conhecer algumas delas:

Dreamweaver

É uma ferramenta de criação de *layout* de páginas que aumenta a produtividade da criação do site. Com o Dreamweaver você pode além de criar páginas organizar e gerenciar diversos projetos de web ao mesmo tempo.

Flash

É a ferramenta padrão da internet quando o assunto é animação. Os sites feitos com Flash, priorizam experiências dinâmicas e intuitivas na internet e a ferramenta conta também com grande poder de programação, tornando-o indispensável.

JBuilder

Essa ferramenta da empresa Borland possui todas as facilidades de um desenvolvimento visual e gera o programa em código executável na linguagem Java e JSP, facilitando dessa forma e dando velocidade ao desenvolvimento de aplicações.

Visual Studio.NET

Esse produto da Microsoft possibilita a geração da aplicação em vários tipos de linguagem como VB.Net , C#, ASP.Net.

Projetando um site

Todo site deve ser baseado em um projeto para que não se perca tempo e dinheiro durante sua execução. Para isso devemos conhecer as etapas de um projeto de web, como mostramos na figura a seguir:

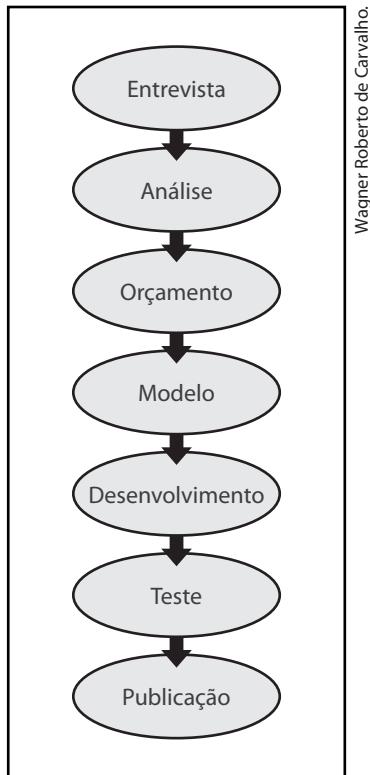


Figura 3 – Etapas de um projeto.

Além dos passos mostrados devemos fazer uma análise mais detalhada em alguns aspectos, veja:

Ambiente

O tipo de servidor que será utilizado para hospedar o site é um ponto importante para definir quais as linguagens e tecnologias poderão ser utilizadas.

zadas. Qual a base de dados que será utilizada também é um ponto que deve ser analisado, assim como a estrutura das tabelas.

Layout

O *layout* do site é também um fator de grande importância. O posicionamento dos objetos, imagens, textos e cores são itens que devem ser analisados antes da confecção do site.

Cores e padrões

As cores causam diversas sensações ao ser humano. Elas podem induzir você a sentir cansaço, sono ou deixar você impaciente, irritado, isso porque elas atuam de maneira direta no cérebro. Por essa razão deve-se utilizar as cores com cautela, um site com muitas cores diferentes na tela deixa o internauta perdido e confuso. Seja sóbrio, porém não abra mão de expor e destacar o que realmente deve ser destacado.

Identidade visual

Toda empresa tem um foco de mercado, um público que ela tenta seduzir diariamente e, para isso, utiliza propagandas, vendedores, contatos corporativos etc. Em todos os casos essa empresa se mostra ao mercado da mesma maneira através de seu material publicitário com cores e logomarcas que não mudam. Isso mostra a identidade visual da empresa.

Imagine você se, a cada mês, seu banco mudasse as cores e o *design* da logomarca, ou ainda se os fabricantes de automóveis mudassem seus brasões ano a ano. Isso não acontece, pois uma marca demora anos para se firmar no mercado e ninguém quer correr o risco de perder a solidez de uma marca já conceituada.

Na internet não é diferente. Você deve seguir os padrões e cores da empresa sempre. É possível criar um site todo utilizando apenas três cores e tendo um resultado muito bom, aí entra a criatividade de cada um.

Tipologia

Tipologia é o termo usado para falarmos sobre tipos de fontes. Na internet essas fontes são restritas, você não pode usar qualquer tipo de letra, pois pode acontecer que ela não seja exibida corretamente. Se isso ocorrer, é necessário que o internauta tenha a fonte que você usou instalada na máquina dele para que ele consiga ver os efeitos dessa fonte utilizada. Caso ele não tenha instalado, o sistema operacional troca por uma fonte padrão de sua biblioteca como a *Times New Roman* ou a *Arial*.

As fontes mais usadas na internet são *Times New Roman*, *Arial*, *Courier New* e a *Verdana*. Uma saída bastante usada por *designers* é a conversão da fonte em figura, mas isso só pode e deve ser aplicado em logomarcas e títulos.



A linguagem HTML

Linear media é um termo usado para descrever qualquer tipo de mídia que é definida como *começo*, *meio* e *fim*. Exemplos de mídia linear são filmes, música, fitas de vídeo e boa parte de livros. A *web*, por sua vez, é bem diferente.

Hypermedia é onde o usuário simplesmente seleciona o próximo item de seu interesse e é imediatamente transportado para aquele novo local. Um bom exemplo é um CD de áudio onde você pode escolher tocar a trilha 10 e ouvi-la imediatamente, sem ter que ficar procurando qual o ponto onde a música começa.

Quando esse conceito é aplicado sobre o texto, você tem então o *Hipertexto*. Um documento hipertexto é constituído de trechos de texto, muito parecido com aqueles que produzimos com um editor de textos. Na verdade, é o que vamos utilizar para produzir páginas HTML: NotePad, VI¹, WordPad ou qualquer editor de texto compatível. Um documento hipertexto diferencia-se de um documento comum pela possibilidade de “ligar” uma palavra ou um trecho de texto com outras partes desse ou de outros documentos. Essa ligação é chamada *link* de hipertexto, ou simplesmente *link*. Ao dar um clique em um *link*, o programa HTML irá se encarregar de desviar sua leitura para um novo trecho do documento ou até mesmo a outro documento, correlacionando com o assunto apontado pelo *link*.

¹ Editor de texto do Linux.

HTML possui características de uma linguagem de programação já que engloba comandos e diretivas. Mas ao contrário das linguagens de programação, HTML não controla recursos físicos do computador. Ela só envia informações de representação gráfica que são interpretadas pelo *browser*, que por sua vez, transforma seus comandos e diretivas na representação gráfica solicitada.

Sintaxe básica de uma tag

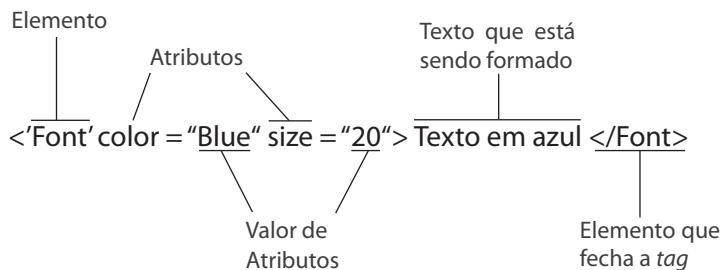
HTML é uma linguagem relativamente simples, possuindo aproximadamente uma centena de palavras reservadas entre um par de chaves angulares (“< >”). Essas marcações entre esses sinais são chamadas *tags*. Todo o

código HTML utiliza *tags* que não são *case sensitive*, ou seja, não faz diferença se você digitar em letras maiúsculas ou minúsculas; porém recomendamos um padrão de escrita para aumentar a legibilidade.

A maioria das *tags* trabalham em conjunto: uma abrindo e outra fechando. Essa última é distinguida pela utilização do caractere “/”.

A primeira palavra que aparece na *tag* é chamada *elemento*. Um elemento é um comando que pede ao *browser* para realizar alguma ação, como por exemplo . As palavras que seguem ainda dentro da *tag* são chamadas *atributos*. Atributos não podem ser repetidos em uma *tag* e descrevem as características de um elemento (como se fossem propriedades de um objeto).

Todos os atributos de um elemento devem ser separados por um espaço e seguidos de um sinal de igual (=) para atribuição de um valor. Veja um exemplo:



Há diferenças entre os diversos navegadores de *web*, de forma que nem todas as marcações e seus correspondentes recursos são suportados por todos eles. Quando um navegador não entende uma determinada marcação, ele simplesmente a ignora. Ao criar um documento, é importante testá-lo com os principais navegadores.

Criando documentos HTML

Todo documento HTML deve começar declarando a si mesmo como tal. Você faz isso com a *tag* <HTML>, a qual é colocada no começo do documento. A última diretiva do documento será, portanto, </HTML>.

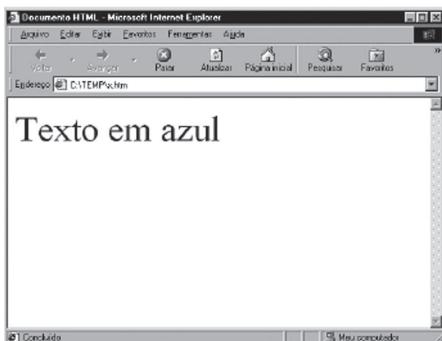
Perceba que toda *tag* que fecha uma estrutura utiliza a barra (/) para signalizar o fechamento do trecho que está sendo formatado. O topo do documento deverá conter também uma seção cabeçalho, delimitado pelo par <HEAD> </HEAD>. Você deverá inserir *sempre* um comando na declaração <HEAD>; trata-se do título do documento. Ele será contido pelo par <Title> </Title> e o texto que vai entre as duas *tags* será mostrado na barra de título do browser. O documento também deve ter uma área chamada “corpo” que é demarcado pelo par <BODY> </BODY>. Nessa parte do documento serão colocados todos os comandos para apresentação de uma página.

Vamos a um exemplo prático. Abra o aplicativo Wordpad. Nele, digite os seguintes comandos (exatamente como está demonstrado a seguir):

```
<HTML>
    <!—qualquer tipo de comentário sobre o documento-->
    <HEAD>
        <Title>Documento HTML</Title>
    </HEAD>
    <BODY>
        <Font color="Blue" size=20>Texto em azul</Font>
    </BODY>
</HTML>
```

Após digitar esses comandos, clique no menu arquivo e, em seguida, em salvar como. Ao selecionar o endereço para salvar o arquivo, no campo *nome do arquivo* digite o nome do arquivo seguido por “.html”. Na opção “salvar como tipo” altere para Documento de texto Unicode. Clique em salvar.

O resultado desses comandos no *browser* será o seguinte:



Fonte: Microsoft Internet Explorer 6.

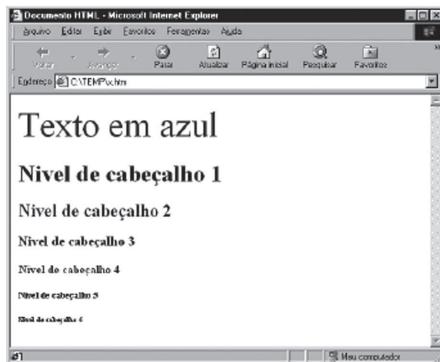
Figura 1 – Demonstração do resultado em HTML.

Cabeçalhos

Cabeçalhos normalmente são usados para títulos e subtítulos de uma página. Escreva seus documentos de forma que os níveis dos tópicos e subtópicos reflitam sua organização. HTML possui seis níveis de cabeçalhos, numerados de 1 a 6, sendo o número 1 o de maior destaque. A tag de cabeçalho é definida pelo par `<Hn> </Hn>`, onde *n* representa seu nível (1 a 6). Veja um exemplo:

```
<HTML>
  <!--qualquer tipo de comentário sobre o documento-->
  <HEAD>
    <Title>Documento HTML</Title>
  </HEAD>
  <BODY>
    <Font color="Blue" size="20">texto em azul</Font>
    <H1>Nível de cabeçalho 1</H1>
    <H2>Nível de cabeçalho 2</H2>
    <H3>Nível de cabeçalho 3</H3>
    <H4>Nível de cabeçalho 4</H4>
    <H5>Nível de cabeçalho 5</H5>
    <H6>Nível de cabeçalho 6</H6>
  </BODY>
</HTML>
```

Isso resultará em:



Fonte: Microsoft Internet Explorer 6.

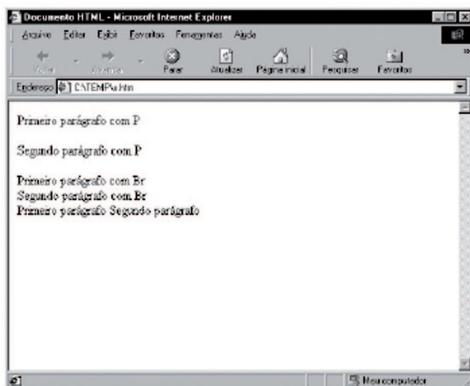
Figura 2 – Demonstração do resultado em HTML.

Ao definir o nível de um cabeçalho, você não está definindo o tamanho da letra. Você apenas define que ele aparecerá com maior tamanho e destaque que o resto do texto. O tamanho exato com que ele será visualizado é definido pelo *browser* de cada pessoa.

Parágrafos e quebras de linha

A marcação `<P>` é utilizada para definir o início de um novo parágrafo, deixando uma linha em branco entre cada parágrafo. HTML não reconhece o caractere de quebra de linha dos editores de texto. Mesmo que exista uma linha em branco, os *browsers* só reconhecem o início de um novo parágrafo mediante a marcação apropriada. Já a marcação `
` faz uma quebra de linha sem acrescentar espaço extra entre as linhas. Veja a diferença entre o uso de `<P>` e `
`:

```
<HTML>
    <HEAD>
        <Title>Documento HTML</Title>
    </HEAD>
    <BODY>
        Primeiro parágrafo com P<P>
        Segundo parágrafo com P<P>
        Primeiro parágrafo com Br<br>
        Segundo parágrafo com Br<br>
        Primeiro parágrafo
        Segundo parágrafo
    </BODY>
</HTML>
```



Fonte: Microsoft Internet Explorer 6.

Figura 3 – Demonstração do resultado em HTML.

Formatação de texto

Todo caractere que aparece na página utiliza tamanho, tipo e cor de fonte determinados pelo *browser*. Entretanto, existem notações que são usadas para mudar essas características. Veja uma lista com as principais:

Notação	Descrição
...	Texto em negrito
<I>...</I>	Texto em itálico
<TT>...</TT>	Fonte em tamanho fixo (normalmente Courier)

Notação	Descrição
<U>...</U>	Texto sublinhado
<BLINK>...</BLINK>	Texto com efeito piscante (somente para Netscape)
<MARQUEE>...</MARQUEE>	Rolar texto (somente no Internet Explorer)
<STRIKE>...</STRIKE>	Para texto riscado
<BIG>...</BIG>	Torna o texto maior
<SMALL>...</SMALL>	Torna o texto menor
_{...}	Texto subscrito
^{...}	Texto sobreescrito
<CENTER>...</CENTER>	Alinha texto ao centro da página

Veja um exemplo para diferenciar cada notação:

```
<HTML>
  <HEAD>
    <Title>Documento HTML</Title>
  </HEAD>
  <BODY>
    Texto normal –
    <B>Texto em negrito</B><P>
    Texto normal –
    <I>Texto em itálico</I><P>
    Texto normal –
    <TT>Fonte em tamanho fixo (normalmente Courier)</TT><P>
    Texto normal –
    <U>Texto sublinhado</U><P>
    Texto normal –
    <BLINK>Texto com efeito piscante (somente para Netscape)</BLINK><P>
    Texto normal –
    <marquee>para rolar o texto (somente para IE)</marquee><p>
```

Texto normal –

<STRIKE>Para texto riscado </STRIKE><P>

Texto normal –

<BIG>Torna o texto maior<BIG> e maior </BIG></BIG><P>

Texto normal –

<SMALL>Torna o texto menor<SMALL> e menor

</SMALL></SMALL><P>

Texto normal –

_{Texto subscrito}<P>

Texto normal –

^{Texto sobrescrito}<P> <CENTER>

Alinha texto ao centro da página.

</CENTER>

</BODY>

</HTML>



■ Imagens no site

Na internet podemos inserir fotos em nossas páginas de HTML. Esse recurso é utilizado na grande maioria dos sites existentes na internet em logomarcas, botões, fotos de produtos, além de outras inúmeras aplicações.

As imagens que são utilizadas na web devem receber uma atenção especial, isso porque você não pode utilizar qualquer tipo de imagem. As fotos e figuras que utilizamos devem ser tratadas e compactadas, visando assim uma boa qualidade de visualização e boa taxa de transferência para o usuário.

Atualmente existem vários formatos de imagens que são utilizadas na internet. Aqui abordaremos os dois formatos mais usados: o JPG (*Joint Photographic Experts Group*) e o GIF (*Graphics Interchange Format*). Antes de inserirmos as figuras na página vamos ter algumas noções sobre como tratar e compactar uma foto para uso na internet. Para essa finalidade vamos utilizar uma ferramenta específica para tratamento de imagens: o PhotoShop.

PhotoShop

O PhotoShop é um aplicativo gráfico com inúmeras ferramentas para edição, retoque e composição em fotos *bitmap*. Desde as primeiras versões lançadas ele sempre surpreendeu com novas ferramentas, filtros e aplicações. A partir da versão 3.0 ele se tornou preferência entre *designers*, fotógrafos e agências de publicidade. Por ser um aplicativo muito estável, que tem 100% de compatibilidade na plataforma PC (*Personal Computer*) ou MAC (*Macintosh da Apple*), ele é um dos mais bem elaborados softwares de edição no mercado.

Aplicações

A quantidade de aplicações do Photoshop é muito vasta. Com ele é possível calibrar uma foto mal revelada, corrigir uma imperfeição em uma modelo, editar ambientes, criar desde efeitos de sombra até montar mais de 10 fotos em apenas uma além de poder salvar seus trabalhos direto no formato JPG para web. Também é possível usá-lo como *software* de entrada para *scanner*, câmeras digitais entre outros.

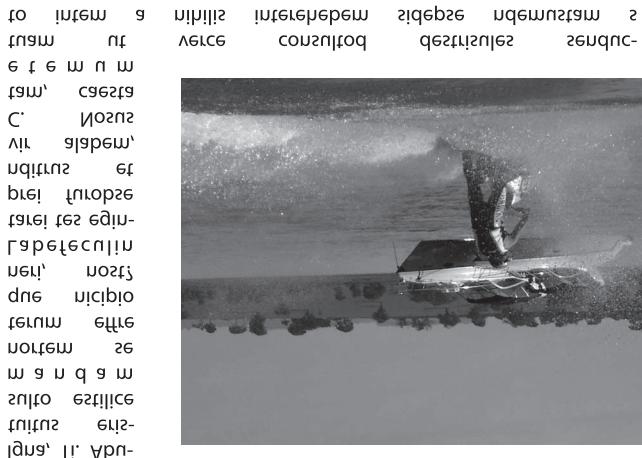
Abrir fotos

Para abrir uma foto, utilizamos o menu *file>open*, selecionamos a foto desejada e confirmamos. A foto será exibida em uma janela e já estará disponível para edição.

Recortar fotos

Na grande maioria das vezes, quando vamos editar uma foto precisamos recortá-la e, assim, descartamos partes dessa foto que são desnecessárias. Esse tipo de recurso também serve para tirar pequenas irregularidades das bordas da foto.

Para recortar uma foto utilizamos a ferramenta *crop* que está na caixa de ferramentas. Para demarcar a área que será cortada, clicamos e arrastamos o *mouse* sobre a foto. Após a primeira seleção podemos clicar sobre os marcadoreis da seleção e mudar sua posição. Veja:



Wikimedia Commons/José Reynaldo da Fonseca.

Figura 1 – Exemplo de imagem.

A foto acima foi escaneada e agora deverá ser recortada e rotacionada para ficar similar à foto a seguir:



Wikimedia Commons/José Reynaldo da Fonseca.

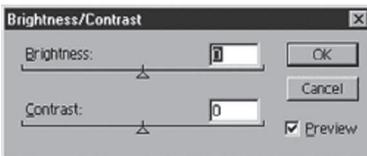
Figura 2 – Exemplo de imagem.

Brilho e contraste

Agora que já cortamos a foto é que começa o real trabalho de edição. Falha comum nas fotos é a falta de equilíbrio das cores; quase sempre isso se deve a má qualidade dos produtos químicos utilizados na revelação ou ainda por causa da baixa resolução óptica do scanner que foi utilizado.

Para minimizar essa falha podemos calibrar as cores das fotos de várias maneiras diferentes utilizando o PhotoShop. Uma das facilidades do PhotoShop é a praticidade que ele oferece em questão de editoração de fotos. Podemos, por exemplo, editar o brilho e contraste de uma foto a fim de poder melhorar sua aparência; podemos também calibrar as cores da foto em conjunto ou isoladamente ou ainda dessaturar (tirar a saturação) as cores da foto. São essas ferramentas que veremos agora no menu: *Image>Adjust>Brightness/Contrast*

Podemos editar o grau de brilho e de contraste de uma foto. Essa é uma importante ferramenta porque normalmente o cliente não tem uma boa foto em seu arquivo ou não dispõe de condições de contratar um fotógrafo, com modelos e estúdio ou, ainda, ele prefere usar fotos que ele mesmo produziu e, por não ter conhecimento técnico e equipamento adequado para fotografia, as fotos em sua maioria precisam ser editadas. Quando selecionamos a ferramenta de brilho e contraste é exibida essa caixa:



Fonte: Photoshop 6.

Figura 3 – Controle de brilho e contraste.

Para definirmos o grau de brilho e contraste da foto, basta movermos a barra em positivo e negativo e depois confirmar. Vamos ver agora alguns exemplos de brilho e contraste. Na imagem a seguir as definições de brilho e contraste estão bem equilibradas, proporcionando assim uma foto visível.



Domínio público.

Figura 4 – Exemplo de imagem com brilho e contraste equilibrado.

Nessa foto o brilho está irregular e existe muita luz sobre a imagem principal, proporcionando assim uma foto fraca.



Domínio público.

Figura 5 – Exemplo de imagem com brilho irregular.

Nessa foto o breu tomou quase toda a imagem. Em função de ser uma foto escura ela necessita de uma adição de brilho.

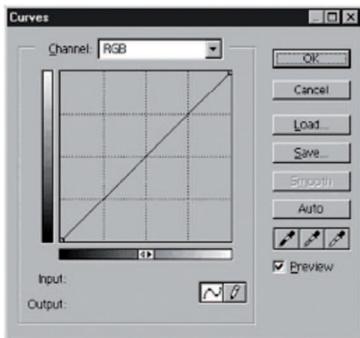
Dominio público.



Figura 6 – Exemplo de imagem com muito contraste.

Edição de cores em linha de curva

Essa é uma ferramenta que nos permite editar as cores de uma foto em conjunto, ou seja, todas as cores juntas ou isoladamente (cor a cor). Podemos, por exemplo, diminuir somente o azul de uma foto ou acrescentar o vermelho a outra. A ferramenta de edição de curvas é de grande interatividade, pois você vê na hora o resultado de como está ficando seu trabalho. Você pode acessar essa ferramenta no menu: *Image>Adjust>Curves*. Quando selecionamos a edição de curvas é exibida essa tela:



Fonte: Photoshop 6.

Figura 7 – Opção de edição de cores em linha de curva.

Channel determina qual cor queremos editar em conjunto ou individualmente. Na área de edição de curva temos uma linha diagonal que determina nossa linha inicial, para alterarmos o tom de curva basta selecionarmos essa linha pelas extremidades ou por um ponto qualquer em sua trajetória, no caso de selecionarmos a linha em um ponto de sua trajetória será inserido um "nó" de edição nesse ponto. Esse nó pode ser removido se for arrastado até a extremidade da linha. Após a edição podemos salvar nossas linhas para uma futura aplicação em outra foto, para isso é só clicarmos no botão salvar dentro da janela de edição de curvas.

Para carregarmos essa linha, basta clicar no botão *load* da janela de edição de curvas. Para terminarmos o processo e aplicarmos a edição é só clicar em ok. Caso você queira desprezar a edição, clique em cancelar.

Na janela de variações podemos adicionar uma gama a mais de cor em alguns aspectos da imagem e essa aplicação pode ser controlada por uma barra que determina se a aplicação será leve ou forte sobre a foto. É recomendado que utilizemos sempre uma aplicação leve e se for necessário repetimos a aplicação. Nessa janela também podemos aumentar ou diminuir a saturação de uma imagem, que segue o mesmo padrão, ou seja, pode ser leve ou forte. Vamos agora conhecer a ferramenta de edição de variação de cores no menu: *Image>Adjust>Variations*



Fonte: Photoshop 6.

Figura 8 – Ferramenta de edição de variação de cores.

Na janela de variação de cores podemos visualizar no lado superior direito um menu que nos permite determinar qual item da foto queremos alterar (sombras, luzes, meios-tonos, saturação). Abaixo temos a linha que determina se a aplicação será leve ou forte. Um pouco mais pra baixo são exibidos três quadros: no canto superior esquerdo é apresentada a foto antes de sua edição e ao seu lado um *preview* do resultado da edição. No quadro inferior esquerdo é mostrada ao centro a foto em estado normal e ao redor estão os exemplos do resultado da próxima inserção. No canto direito estão as opções de clarear e escurecer. Para adicionarmos o valor visualizado basta clicar sobre o item e ele será passado para o *preview*. Caso queira reverter uma aplicação errada basta clicar sobre a foto original que está no canto superior esquerdo da janela. Para analisarmos se uma foto está ou não calibrada, podemos optar por consultar a paleta de informação de cores no menu: *Window>Info*

Nessa paleta podemos inspecionar a quantidade de cada cor que está sobre a foto, em CMYK (abreviatura do sistema de cores formado por ciano (*Cyan*), magenta (*Magenta*), amarelo (*Yellow*) e preto (*Black*)) e em RGB (abreviatura do sistema de cores formado por vermelho (*Red*), verde (*Green*) e azul (*Blue*)). Essa ferramenta é muito útil em diversas etapas do nosso trabalho porque ela é o “olho clínico” das cores. As cores podem ser apresentadas em diferentes tons, em função do tipo de placa de vídeo, tipo de monitor, resolução e até mesmo em função da luz do ambiente em que está a estação de trabalho. Para eliminar esse problema o ideal é usar um programa calibrador de cores e uma escala de cores impressa.

Padrões de cores

O PhotoShop suporta diversos padrões de cores, porém na internet utilizamos somente o RGB, que é o padrão de cores utilizado por monitores de micro, televisores e *notebooks* em geral. O formato RGB é baseado na adição das cores básicas vermelho, verde e azul em um mesmo ponto de luz. Na internet os valores são convertidos para um código que chamamos de Hexadecimal.



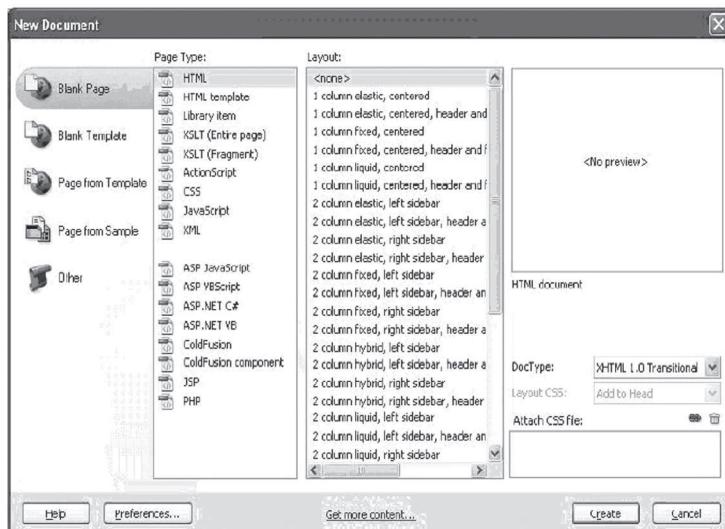
Dreamweaver

O Dreamweaver é uma ferramenta que proporciona grande facilidade de criação de páginas para internet. Além da criação, também gerencia vários projetos de *web* ao mesmo tempo, ou seja, você poderá utilizar o Dreamweaver para gerenciar todos os sites de seus futuros clientes de maneira rápida e fácil.

Por ser uma ferramenta da Macromedia, o Dreamweaver tem fácil interação com outras ferramentas voltadas ao desenvolvimento de internet como Flash, Fireworks e ColdFusion.

Iniciando o Dreamweaver

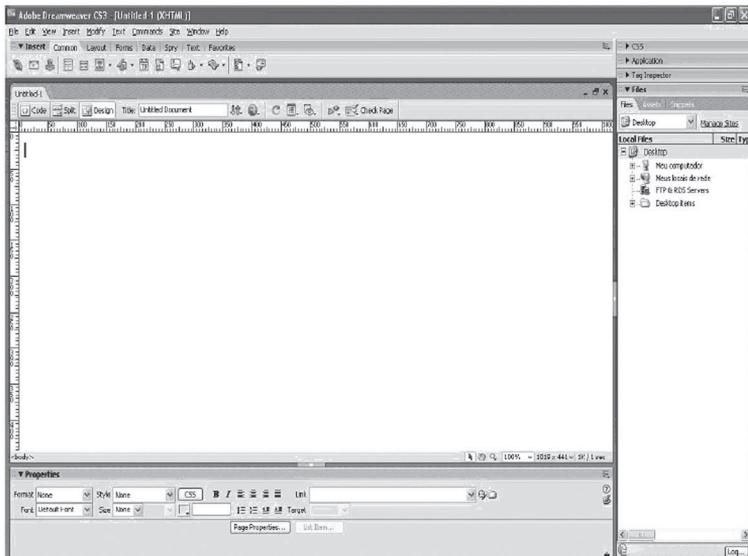
Quando iniciamos uma nova página no Dreamweaver, uma janela é exibida para definirmos que tipo de arquivo queremos criar. Essa janela está dividida em duas abas: na aba geral (General) temos a área de categorias que exibe o tipo de arquivo e a área básica (Basic Page), que exibe opções de criação de cada item selecionado; a segunda aba é a de modelos (*templates*) que exibe os modelos já definidos para as contas de site do Dreamweaver. É nessa janela que podemos definir se iniciaremos uma página HTML ou um arquivo de *script*, entre outras opções. Veja a seguir a figura dessa janela:



Fonte: Adobe Dreamweaver CS3.

Áreas do Dreamweaver

O novo Dreamweaver MX, tem uma interface de trabalho otimizada. Agora tudo fica organizado na própria tela e o acesso a todos os painéis e propriedades ficou muito mais fácil. Alguns itens se mantiveram inalterados, como a barra de documento, para fazer a alternância de visualização. Veja a seguir a área de trabalho do Dreamweaver:

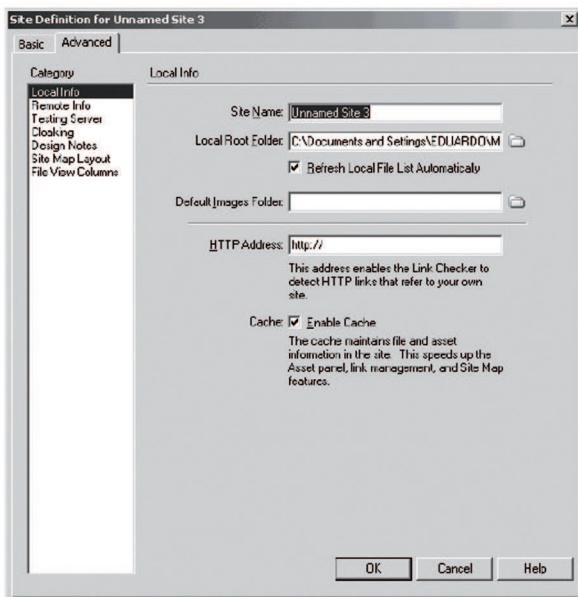


Fonte: Adobe Dreamweaver CS3.

Definindo o site

O Dreamweaver tem uma ferramenta própria para FTP¹, para a publicação do site na internet. Além disso, para evitar erros banais como *links* quebrados ou figuras que não carregam, devemos sempre começar nosso projeto definindo o site que será desenvolvido. O Dreamweaver tem como gerenciar vários projetos diferentes de *web*, sendo assim veremos como definir para cada site uma pasta raiz, um nome e as configurações necessárias para o início do projeto. Para definir um novo site devemos ir ao menu *site/new site* e será exibida a seguinte janela:

¹ File Transfer Protocol, em português, Protocolo de Transferência de Arquivos.



Fonte: Adobe Dreamweaver CS3.

Nessa janela utilizaremos somente a aba *Advanced*. Aqui definiremos todas as configurações do nosso site. A primeira configuração que devemos definir é o nome que utilizaremos para esse site. Esse não é o endereço do site, mas sim um nome de referência para o Dreamweaver gravar as configurações de pasta raiz, servidor, FTP e outros.

Para cada item selecionado na área de categoria, teremos as opções descritas a seguir. Veja os itens e suas configurações:

Local Info

Define as configurações locais do site e tem os seguintes itens:

- *Site name*: nome do projeto para o Dreamweaver.
- *Local Root Folder*: define a pasta raiz do site. É nessa pasta que definimos onde serão gravados todos os arquivos do site, páginas html, figuras, arquivos, CSS², JavaScript. Para alterar o nome da pasta raiz clique sobre o ícone de pasta ao lado da área do nome da pasta.
- *Default Images Folder*: define uma pasta padrão para as imagens do site que deve ser sempre uma subpasta da pasta raiz. Para alterar o nome da pasta raiz clique sobre o ícone de pasta ao lado da área do nome da pasta.

² Cascading Style Sheets – CSS – é uma linguagem de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação (HTML ou XML, por exemplo). Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento.

- *HTTP address*: define o endereço do site na internet (esse campo é opcional).
- *Cache enable*: liga e desliga o cache em máquina local, esse item tem como finalidade agilizar a leitura de toda a estrutura do site e das bibliotecas de Dreamweaver para o projeto.

Remote Info

Determina o tipo de acesso que será utilizado para o envio dos arquivos do site para o servidor de web. E tem as seguintes configurações:

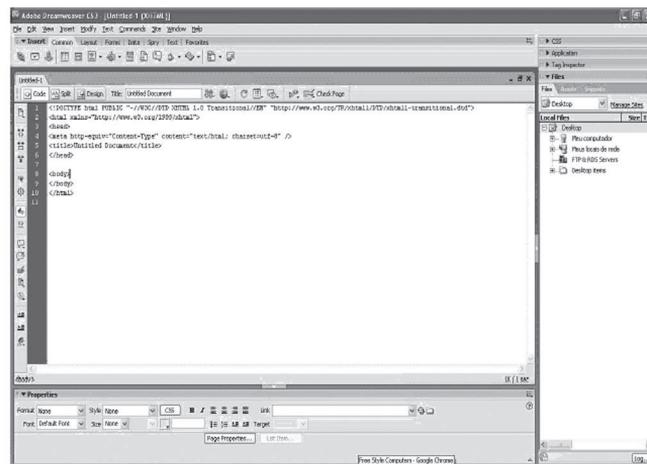
- *FTP*: protocolo mais utilizado para o envio de arquivos para o servidor. Esse protocolo tem as seguintes configurações:
 - *FTP Host*: aqui você deve determinar o nome do servidor de FTP em que será efetuada a conexão.
 - *Host Directory*: aqui você deve determinar qual a subpasta que será acessada na conexão.
 - *Login*: nome de usuário para a conexão FTP.
 - *Password*: senha para a conexão FTP.
 - *Test*: esse item é muito útil pois com ele podemos testar se a conexão com o servidor FTP está correta.
 - *Save*: esse item salva o *log in* e senha para essa conta de FTP, isso facilita na hora de conexão com o servidor.

Modos de exibição

Agora que já configuramos a pasta raiz do site, veremos os três modos de exibição do Dreamweaver.

Code

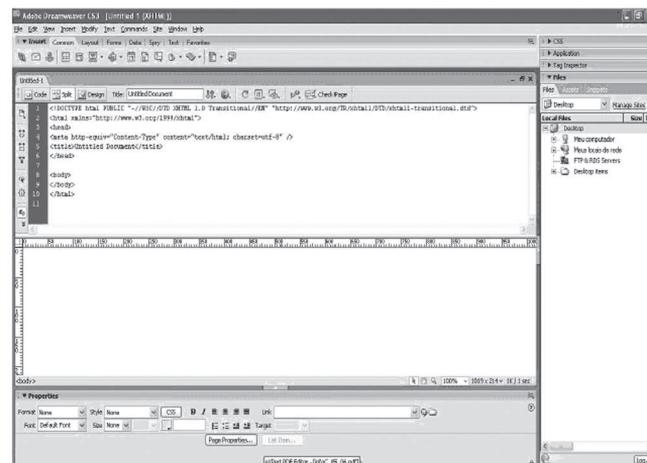
Exibe o código fonte da página que está sendo editada e no caso de alteração, ativa o auxiliar de digitação que ajuda na criação de código html e JavaScript.



Fonte: Adobe Dreamweaver CS3.

Code and design

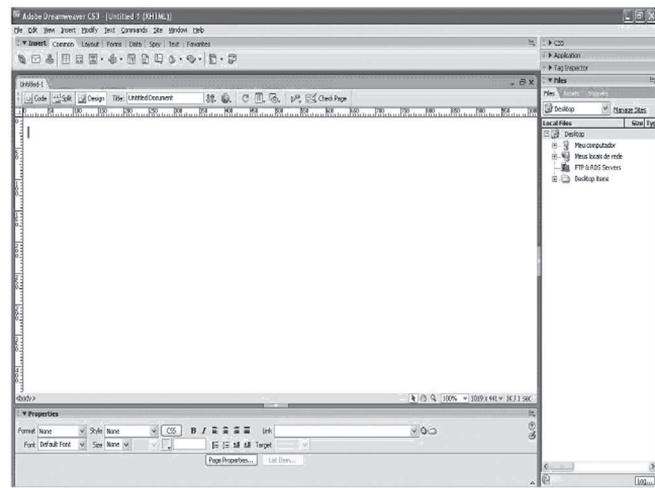
Exibe a tela dividida ao meio: na parte superior o código fonte da página e na inferior a área de *design*.



Fonte: Adobe Dreamweaver CS3.

Design

É a área de criação do Dreamweaver, onde criamos todo o *layout* das páginas.



Fonte: Adobe Dreamweaver CS3.

Alterando a aparência da página

Até agora conhecemos algumas características do Dreamweaver. Vamos começar a trabalhar o *layout* e o *design* da página. Para configurarmos as propriedades da página como título, cor ou imagem fundo, podemos utilizar a janela de configuração de página no menu *modify/page properties*. Nessa janela temos os seguintes itens:

- *Title*: título da página que aparece na barra superior do navegador.
- *Background image*: imagem de fundo para a página, essa imagem será duplicada até que ocupe todo o fundo da página.
- *Background*: cor de fundo para a página, que pode ser utilizado no lugar de uma imagem de fundo para a página, a cor de fundo será sempre em código hexadecimal, para selecionar a cor, basta clicar sobre o quadrado de exemplo de cor e escolher.
- *Text*: define uma cor padrão para os textos da página, esta cor por *default* é preta, para selecionar a cor, basta clicar sobre o quadrado de exemplo de cor e escolher.
- *Links*: define a cor padrão para os *links* da página, essa cor por *default* é azul, para selecionar a cor, basta clicar sobre o quadrado de exemplo de cor e escolher.

- *Visited Links*: define a cor padrão para os *links* já visitados na página, esta cor por *default* é lilás, para selecionar a cor, basta clicar sobre o quadrado de exemplo de cor e escolher.
- *Active Links*: define a cor padrão para os *links* ativos da página, essa cor por *default* varia de acordo com o navegador, para selecionar a cor, basta clicar sobre o quadrado de exemplo de cor e escolher.
- *Margins Left / Top / Width / Height*: define a margem de apresentação da página, por *default* seu valor é 2 pixels.
- *Document Encoding*: define qual a codificação de caracteres que será utilizado.
- *Tracing Image*: define uma imagem para ficar fixa no fundo da página, essa imagem poderá ter sua opacidade alterada porém ela não se repetirá, como o *background image*. Para utilizar o *tracing image* é aconselhável que seja utilizada uma imagem ou foto do tamanho da área útil do site.

Após definirmos essas configurações, basta clicarmos no botão OK da janela e as alterações serão aplicadas à página.

Configurando textos

Para a inserção de textos na página, devemos estar no modo de visualização de *design* e digitar diretamente o texto no Dreamweaver. Após a digitação podemos configurar vários itens desse texto, como alinhamento, cor, tipo de fonte, tamanho, identação³, criar *links* para outras páginas HTML, definir *targets* e criar listas. Veja a seguir a barra de propriedades de texto.

³ Recuo.



Fonte: Adobe Dreamweaver CS3.

Podemos definir se o texto será um header de página, pré-formatado ou um nome parágrafo, para isso utilizamos o item *Format*. Para definir o tipo de fonte do texto, devemos selecionar o tipo da fonte na área *Default font*. O tamanho do texto pode ser alterado na área *Size*. A cor do texto pode ser alterada no item *Text color*.

Podemos também definir se o texto ficará em negrito e/ou itálico, para isso basta clicar sobre as marcações *B* e *I*. O alinhamento do texto pode ser feito à direita ou esquerda, centralizado ou justificado, sendo que o justificado só pode ser aplicado quando o texto não for pré-formatado.

Exercício – Configurando textos

Para praticarmos o conteúdo visto até agora vamos criar uma página com textos configurados. Para isso siga as diretivas a seguir:

- 1) Defina um novo site que será utilizado durante todo o curso, o nome de referência para o Dreamweaver deverá ser dw_seunome (ex.: dw_maria).
- 2) Tipo de documento será um html básico.
- 3) O título da página será: Página de textos.
- 4) A cor de fundo dessa página será #003366.

Conteúdo da página

- 1) Cabeçalho com fonte tamanho 6, tipo de fonte Arial, alinhamento centralizado e cor #000000.
- 2) Textos na cor #FFFFFF, com fonte Times New Roman e tamanho 3, seguidos de textos formatados, na seguinte ordem.

Texto em negrito cor #FFFF00

Texto em itálico cor #FF0000

Texto negrito e itálico cor #33FFFF

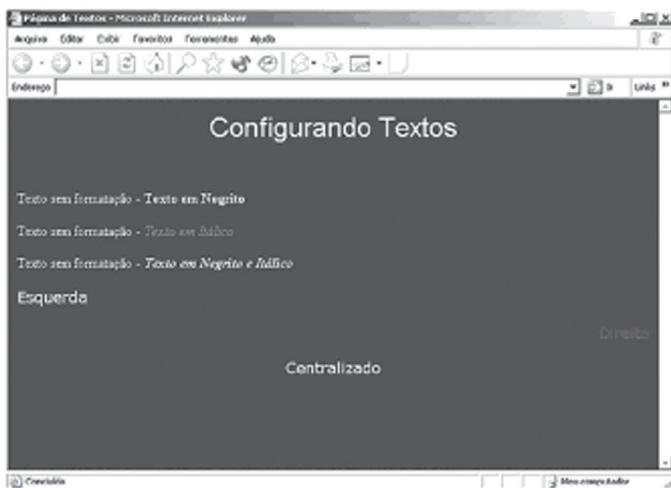
Texto alinhado à esquerda tamanho 4, tipo de fonte Verdana e cor #FFCC99

Texto alinhado à direita tamanho 4, tipo de fonte Verdana e cor #990000

Texto alinhado centralizado tamanho 4, tipo de fonte Verdana e cor #00FF00.

- 3) Salve o arquivo com o nome textos.htm

Veja qual a aparência desse exercício:



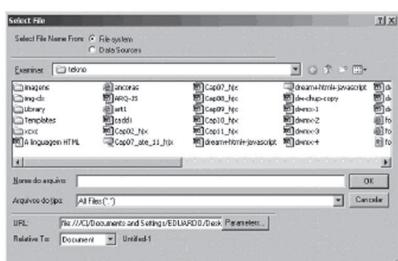
Fonte: Microsoft Internet Explorer 6.

Criando links

Com os *links* a navegação na internet fica muito mais fácil, permitindo ao internauta “saltar” de página em página, podendo ir direto ao que deseja com apenas um clique. Para criar um *link* utilizando um texto no Dreamweaver, devemos:

- 1) Digitar o texto que servirá de *link*.
- 2) Selecionar esse texto.
- 3) Clicar sobre a pasta que se encontra ao lado do item Link na barra Properties.

Veja a aparência da janela que será aberta:



Fonte: Adobe Dreamweaver CS3.

Nessa janela, selecione o arquivo que será aberto quando o internauta clicar no *link*. Caso você selecione um arquivo que não é interpretado pelo navegador, quando o usuário clicar no *link* será exibida a confirmação de início de *download* para o usuário, caso contrário, a página será exibida no próprio navegador.

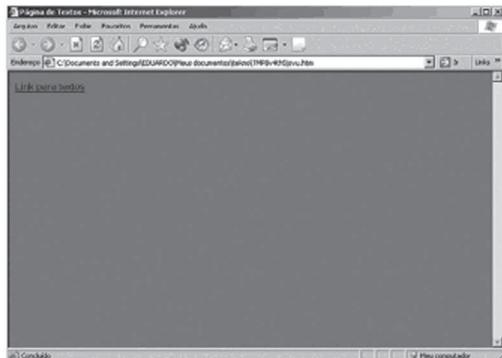
4) Clicar em OK.

Um detalhe de grande importância na criação de um *link* é definir para o Dreamweaver se a localização do arquivo apontado pelo *link* é relativa ao servidor ou relativa ao documento, normalmente o *link* é feito relativo ao documento. Fique atento a esse detalhe.

Exercício – *Links*

- 1) Tipo de documento será um html básico.
- 2) O título da página será Página Link.
- 3) A cor de fundo dessa página será #3399FF.
- 4) Digite o texto (Link para Textos).
- 5) Selecione todo o texto.
- 6) Clique na pasta ao lado da área de *link*, localize o arquivo textos.htm, selecione o arquivo e confirme.
- 7) Salve o arquivo com nome links.htm

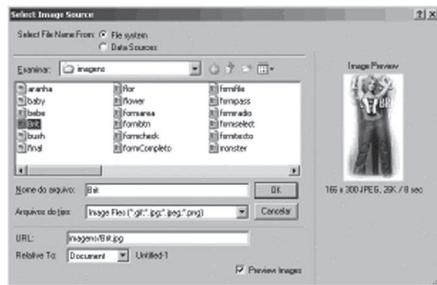
Veja a aparência dessa página:



Fonte: Microsoft Internet Explorer 6.

Trabalhando com imagens

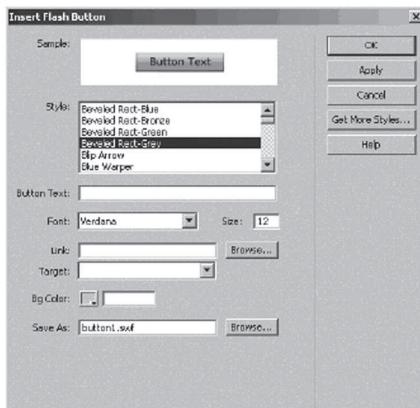
Na criação da estrutura do site podemos definir uma pasta padrão para armazenarmos as imagens e fotos. Isso evita erros na hora de carregar a página pois, toda vez que inserir uma imagem na página, o Dreamweaver perguntará se você deseja salvar uma cópia dessa imagem na pasta padrão de imagens definida anteriormente. Sempre devemos confirmar essa ação, pois isso organizará melhor o site. Para inserir uma imagem devemos clicar sobre o item *image*, da barra *insert* na aba *common*. Será exibida uma janela para selecionarmos a imagem desejada. Nessa janela teremos um *preview* da imagem. No caso das imagens também devemos ficar atentos ao caminho da imagem. Veja abaixo a janela que aparecerá quando clicarmos no botão mencionado acima:



Fonte: Adobe Dreamweaver CS3.

Flash Button

Mesmo sem conhecer o Flash, podemos inserir facilmente botões animados em Flash. Utilizamos no menu *insert*, a opção *media* e o item *flash button*. Para inserirmos esse botão é necessário que o arquivo esteja salvo, e a seguinte janela será exibida:



Fonte: Adobe Dreamweaver CS3.

⁴ Extensão de arquivos Flash.

Nessa janela definimos quais as configurações que esse botão terá, definimos qual a aparência, o texto que será exibido dentro do botão, tipo de fonte e tamanho, qual o *link* que ele deverá abrir, o alvo de destino do *link*, a cor de fundo do botão e o nome que o arquivo do botão terá, porque o Dreamweaver criará um arquivo swf⁴, automaticamente, quando a operação for confirmada.

Criando *links* com efeito *rollover*

Você já deve ter visto em sites alguns *links* feitos com imagens, que quando passamos o *mouse* sobre eles, as figuras mudam de cor, ou apresentam efeitos similares. Na verdade o que ocorre é que quando passamos o *mouse* sobre uma dessas imagens o atributo *src*⁵ da imagem é alterado, mudando assim a figura que é apresentada. Esse efeito é gerado com JavaScript, porém o Dreamweaver tem isso incorporado na barra *insert*, no item *rollover image*, o que nos permite criar com facilidade. Quando clicamos sobre o item, uma janela é exibida para definirmos qual a imagem será mostrada como *link*, a imagem que será exibida quando se passar o *mouse* sobre o *link*, qual o arquivo de destino do *link* e um texto alternativo para o *link*. Um item importante nessa janela é a definição do nome desse *link*, pois ele servirá de referência para o JavaScript. Veja a aparência dessa janela:



Fonte: Adobe Dreamweaver CS3.

Após confirmar, a imagem será apresentada na página. Porém, em tempo edição não é possível visualizar o efeito.

Exercício – *Rollover*

- 1) Título da página: imagem como *links*.
- 2) Cor de fundo #6600FF.
- 3) Crie um botão *rollover* com 80 pixels por 80 pixels e atribua um *link* para esse botão.

4) Crie um Flash Button e atribua um *link* para esse botão.

5) Salve o arquivo como *rollover.htm*.

Tabelas

A criação de tabelas ajuda no ajuste do *layout* do site, além de deixar as informações melhor organizadas. Porém, o processo de criação de tabelas faz com que esse recurso seja utilizado constantemente. Para inserir uma tabela utilizamos a aba *common*, o item *insert table*, onde será exibida uma janela com as configurações iniciais da tabela.



Fonte: Adobe Dreamweaver CS3.

Nessa janela definimos quantas colunas e linhas a tabela terá, o espaçamento entre as células (*cell spacing*), a distância entre a borda da célula e o conteúdo (*cell padding*), a largura da tabela que pode ser fixa (em pixels) ou relativa (em porcentagem) e a espessura da borda da tabela.

Mesmo depois de criarmos a tabela podemos alterar suas propriedades. Para isso devemos selecionar toda a tabela e na barra de propriedades, teremos várias opções como o nome da tabela (*id tabel*), a quantidade de linhas e colunas, as definições de *cell padding* e *cell spacing*, as dimensões da tabela, a cor de fundo ou figura de fundo para a tabela. Podemos também alterar a cor da borda e ajustar o tamanho da tabela ao conteúdo.



Fonte: Adobe Dreamweaver CS3.

Propriedades da célula

Para inserir conteúdo nas células da tabela, basta clicar sobre a célula e inserir as informações desejadas, sejam estas texto, foto ou figura. Cada célula tem algumas propriedades que são aplicadas para uma célula ou para um grupo de células da tabela. Veja a barra de propriedades:



Fonte: Adobe Dreamweaver CS3.

Nessa barra podemos mesclar mais de uma célula ou linha, com o *merge*, ou separar células com o *split*, além de definir o alinhamento horizontal e vertical do conteúdo em relação à célula, podemos definir um tamanho específico para a célula em função da tabela, bloquear a quebra de linha automática com *no wrap*, definir uma célula com *header*⁶, e definir cores de fundo e de borda para a célula.

⁶ Cabeçalho.

Formulários

Para inserir formulários na página utilizaremos a aba *forms*. Nessa aba teremos todos os campos relativos ao formulário. Antes de começar a inserir os campos de formulário na página, devemos sempre iniciar o formulário utilizando o primeiro elemento que aparece na aba. Esse elemento criará no código a tag *form* e exibirá as propriedades do formulário que estamos iniciando.



Fonte: Adobe Dreamweaver CS3.

Nessa barra podemos definir o nome do formulário, qual a ação do formulário, o método de envio, o alvo e o tipo de dados. Após inserirmos o *form*, podemos começar a inserir os campos do formulário.

Campo de texto

O campo do tipo *text field* é utilizado para inserir textos ou senhas, além de área de texto. Ele tem algumas propriedades que podemos definir, como

nome, máximo de caracteres que o campo aceita (*max chars*), o tamanho físico do campo em caracteres (*chars width*), o valor inicial e o tipo de campo (*single line, multi-line ou password*). Quando definimos que o tipo de campo é *multi-line*, podemos definir o tamanho do campo em linhas, e se ele possui quebra automática de linha ou não. A quebra automática (*wrap*) pode ser física ou virtual.

Checkbox

Este botão permite ao usuário selecionar um ou mais itens do formulário e tem como propriedades o nome, o valor que será retornado quando selecionado e seu estado inicial *checked*.

Radio

Este botão tem funcionalidade similar ao *checkbox*. Podemos associar mais de um botão, evitando assim que o usuário selecione dois itens do mesmo grupo.

Radio group

Ajuda na criação e associação de botões do tipo *radio*, isso porque ele permite de maneira rápida a criação de grupos de botões de seleção. Quando clicamos sobre ele uma janela é exibida para definirmos o nome do grupo e os botões e valores desse grupo. Veja a janela:



Fonte: Adobe Dreamweaver CS3.

Botões

Podemos inserir três tipos de botões: do tipo *submit*, *reset* e do tipo *button*.

- *Submit*: o botão de tipo *submit* tem por finalidade carregar a definição do *action* do formulário, e enviar o formulário.

- **Reset:** limpa todos os valores do formulário.
- **Button:** este botão não tem aplicação predefinida, isso nos permite associar a este botão funções em JavaScript entre outras.

Podemos criar formulários dentro de tabelas, isso facilita a criação do *layout* da página.

Exercício – Formulários

Criar uma página com formulário dentro de uma tabela, que deverá ser centralizada na página.

- 1) A cor de fundo para a página é #66ccff.
- 2) O título da página é cadastro.
- 3) Salve o arquivo com o nome form.htm.

Veja abaixo o *layout* do formulário:

The screenshot shows a Microsoft Internet Explorer window with the title "Untitled Document - Microsoft Internet Explorer". The page content is a form with the following fields:

- nome: [text input field]
- endereco: [text input field]
- telefone: [text input field]
- e-mail: [text input field]
- sexo:
 - masculino
 - feminino
- nasimento:
 - dia: [text input field]
 - mês: [text input field]
 - ano: [text input field]

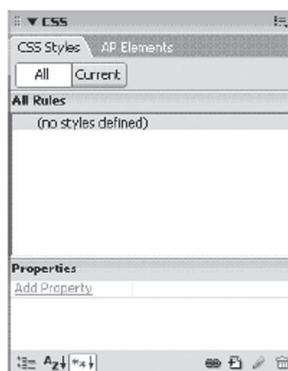
At the bottom of the form are two buttons: "VALIDAR" and "LIMPAR".

Fonte: Adobe Dreamweaver CS3.

Configuração de estilos

A utilização de estilos facilita a configuração e padronização das páginas. Essas configurações são facilmente inseridas com o Dreamweaver, seja para textos, tabelas ou formulários, além da possibilidade de criação das classes de formatação. O Dreamweaver tem uma grande galeria de efeitos e formações, algumas voltadas para textos outras para figuras, e outras ainda para

imagens. Aqui veremos as principais formatações utilizadas. Para definirmos uma formatação para os elementos de textos, como negrito, itálico, textos de cabeçalho, *links* e outros, devemos utilizar na barra de *Design* a aba *CSS Styles*.

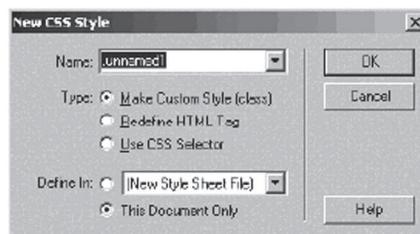


Fonte: Adobe Dreamweaver CS3.

Nessa barra temos alguns botões que veremos agora:

Attach style sheet: esse botão atribui um arquivo CSS, esse tipo de método é muito utilizado para padronizar todo o site utilizando somente um arquivo externo de estilos.

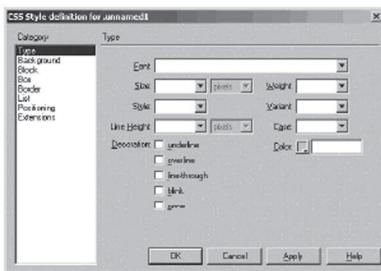
New CSS Style: esse botão abre uma janela para definirmos uma nova formatação, veja essa janela:



Fonte: Adobe Dreamweaver CS3.

Nessa janela temos a possibilidade de definir se a formatação será atribuída para uma *tag* do HTML ou para um comportamento de *link* para uma classe de formatação. Ainda nessa janela podemos definir se as formatações serão aplicadas somente para a própria página ou se será um documento externo de estilos. No caso de um novo arquivo externo de CSS, antes de confirmarmos, será aberta uma janela para nomearmos este arquivo.

Após definirmos o tipo de formatação, será exibida outra janela para agora criarmos as configurações de estilos. Veja essa janela:



Fonte: Adobe Dreamweaver CS3.

Perceba que essa janela contém uma lista do lado esquerdo com as categorias de formatação e do lado direito as formatações suportadas para cada categoria.

Criando classes

As classes de formatação têm aplicação diversa dentro de um site. É muito comum utilizarmos classes para configurar diversos elementos que vão desde textos simples até botões de formulário. Para criar uma classe devemos clicar sobre o botão *new css style*, que está na barra *design*, na aba *css style*. Quando a janela de tipo de configuração for exibida, selecione a opção *make custom style*, defina um nome para a classe no item *name*, defina se a classe será utilizada somente para esta página ou se será parte de um arquivo externo de estilos, e confirme. Após a confirmação defina as configurações que esta classe terá, e confirme novamente. Depois da criação da classe ela aparecerá na janela de *CSS Style*, para atribuir as configurações da classe para um objeto na página basta selecionar os objetos ou texto que você deseja configurar e clicar sobre o nome da classe.

Não existe limite para a criação de classes, porém devemos lembrar que quanto mais códigos o *browser* processar mais lenta fica a navegação.

Algumas configurações só têm efeitos quando visualizadas no navegador. Existem configurações que só funcionam *on-line* e outras que dependem de tipo e versão de *browser*.

Editando uma classe

Para editar uma classe criada basta selecionar essa classe na barra e clicar sobre o botão *edit Style Sheet*. Caso você clique sobre o botão de edição sem ter selecionado a classe que deseja editar será exibida uma janela de visualização de formatações e classes, nessa janela basta você selecionar a classe ou elemento que deseja editar e clicar no botão *edit*.

Atribuindo formatações para elementos

Para atribuir configurações para elementos da página, textos, tabelas, formulários, devemos definir qual o elemento queremos configurar, após clicar sobre o botão *new CSS style*, e selecionar o item *redefine HTML tag*. A aplicação da formatação escolhida é aplicada automaticamente ao elemento sempre que ele for utilizado.

Atribuindo comportamento para *links*

Os *links* por si só têm quatro comportamentos: *link* normal, *link* ativo, *link* visitado e sobre. Esses comportamentos podem ser alterados utilizando o item *use CSS Selector*. Nesse item podemos definir diretamente para um *link* de texto, comportamentos de *link* normal e sobre, diferentes, criando assim um efeito *rollover*. Esse tipo de técnica é utilizado em grande escala na web. Aparentemente o *link* parece um simples texto, mas quando passamos com o *mouse* sobre ele sua aparência é alterada. Para conseguir esse efeito devemos criar configurações diferentes para um *link* normal (*a:link*) e para o sobre (*a:hover*).

Frames

Até agora criamos páginas e mais páginas que são exibidas em uma janela do navegador, porém podemos exibir mais de uma página html na mesma janela do navegador e para isso utilizamos frames. Os frames são áreas que definimos dentro de uma janela do navegador, e essas áreas exibem outro arquivo html. Imagine uma moldura de quadro na parede, em uma mesma moldura você pode colocar vários tipos de pinturas, fotos, certificados. O conceito de frame é o mesmo: nós definimos a “moldura”, e dentro dessa moldura exibimos vários arquivos html diferentes.

A definição dessas molduras é feita em um arquivo chamado *frameset*. O *frameset* é também um arquivo html, com uma estrutura similar a que vimos até o momento, a diferença está no fato do arquivo não ter corpo. Na verdade o conteúdo que seria o corpo é substituído por uma chamada de outros documentos html. No Dreamweaver temos grande facilidade para a criação de frames. Na barra *insert*, temos a aba *frames*, nessa aba podemos definir um modelo predefinido da apresentação dos frames. Para inserir o *frameset* na página, é só clicar sobre o modelo desejado.

Quando inserimos o *frameset* na página, a barra de propriedades exibe as configurações de cada frame separadamente, para alternar o frame ativo basta selecioná-lo na barra de propriedades. Veja barra de propriedades do frame:

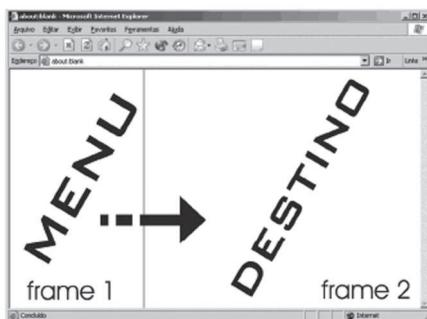


Fonte: Adobe Dreamweaver CS3.

Nessa barra podemos definir as propriedades dos frames de maneira independente para cada frame.

Target

Quando criamos uma página com frames, é comum uma das áreas ser destinada para um menu com *links*, porém se clicarmos sobre o *link*, ele abrirá na mesma área que foi chamado, mas é comum termos um menu que seja exibido durante toda a navegação e os *links* sejam apontados para outra área do *frameset*.



Fonte: Microsoft Internet Explorer 6.

Esse apontamento deve ser feito nas propriedades de cada *link* criado, e no item *target* devemos selecionar o nome do frame de destino, ou seja, qual é o alvo de *link*.



Flash

O que é o Flash

O Flash é a solução profissional para a produção de experiências de *web* de alto impacto. Com o Flash, você poderá criar sites e aplicativos interativos baseados em vetores. A interface com o usuário da Macromedia facilita a criação de ilustrações incríveis e permite criar, rapidamente, animações complexas, controles de navegação e até mesmo sites completos de multimídia. Você poderá usar o Flash para criar aplicações avançadas de *web*, utilizando *scripting*, formulários e conectividade no servidor. Filmes Flash são ilustrações vetoriais compactas que descarregam (por exemplo, fazem o *download*) rapidamente e são colocadas em escala para o tamanho de tela do visualizador.

Fluxo de animações

O Flash Player fará “fluir” os filmes Flash conforme eles descarregarem. Isso significa que a animação poderá começar antes do *download* ter terminado. Quando você criar seus filmes Flash de maneira adequada, os usuários poderão começar a visualizar o seu site em segundos e não terão que esperar para que termine todo o *download* do filme. Eles terão algo para ver de imediato, evitando que cliquem no botão voltar.

Ilustrações baseadas em vetores

O Flash suporta dois tipos de formatos de imagens: *vetorial* e em *raster*. Ilustrações vetoriais são criadas com linhas e curvas e com descrições de suas propriedades. Os comandos dentro da ilustração vetorial dizem ao seu computador como exibir as linhas e formas, que cores usar, qual deverá ser a largura das linhas e assim por diante.

As imagens em *raster*, também chamadas de *bitmaps*, são criadas com pixels. Ao criar uma imagem em raster, você mapará a colocação e a cor de cada pixel e o bitmap resultante será o que se vê na tela. O termo “bitmap” é normalmente usado para se referir às ilustrações em raster. Ele também é o

nome de um tipo de imagem, o formato Windows Bitmap (BMP). Usaremos o termo para nos referirmos às ilustrações em raster ao longo desse curso. Ao fazermos referência ao formato Windows Bitmap, iremos chamá-lo de BMP.

O tipo de imagem criada pelo Flash é vetorial, então quase tudo que você desenhar no Flash será uma ilustração vetorial (com algumas exceções, como os preenchimentos de bitmaps). As ilustrações vetoriais têm benefícios importantes – têm um pequeno tamanho de arquivo e são colocados em escala muito bem. Entretanto, elas também têm limitações – as ilustrações vetoriais muito complexas podem ter tamanhos muito grandes de arquivo e vetores não são tão úteis para arte-final com qualidade fotográfica. Você poderá desenhar todas as suas ilustrações vetoriais no Flash, usando as ferramentas de desenho e você poderá importar artes-finais vetoriais de outros aplicativos. Os formatos de ilustrações vetoriais que o Flash pode usar incluem Macromedia FreeHand (FH7, FH8, FH9), Adobe Illustrator (EPS, AI), AutoCAD DXF (DXF), Enhanced Metafile (EMF), Windows Metafile (WMF) e Flash Player (SWF, SPL).

Apesar do Flash ser uma ferramenta de autoria baseada em vetores, você também poderá usar imagens de bitmap no Flash. Imagens de bitmap não são criadas pelo Flash – você precisará usar qualquer aplicativo externo, como o Fireworks e o Photoshop, para criar os arquivos e então importá-los para o Flash. Ao contrário das ilustrações vetoriais, as imagens de bitmap não são muito redimensionáveis. As imagens simples de bitmap geralmente são maiores quanto ao seu tamanho de arquivo do que as simples ilustrações vetoriais, mas imagens muito complexas de bitmap – por exemplo, fotografias – muitas vezes são menores que as ilustrações vetoriais comparáveis. Os formatos de bitmap que o Flash pode usar incluem Bitmap (BMP), Imagem GIF, Imagem JPEG, Imagem PNG, Imagem Macintosh (PICT), Imagem MacPaint (PNT) e Imagem TIFF.

Um benefício do uso de ilustrações vetoriais ao invés de bitmaps é que você poderá facilmente redimensionar as ilustrações vetoriais sem distorcer a imagem. Quando você redimensiona um bitmap, o recorte dos pixels fica evidente enquanto que quando você redimensiona as ilustrações vetoriais, as bordas ainda permanecem bem suaves.

Além dessas aplicações o Flash também pode ser utilizado para a criação de CD multimídia, que pode conter catálogos de produtos em geral, propagandas, cases, portfólios entre outros itens. Veja alguns exemplos sites que utilizaram o Flash:



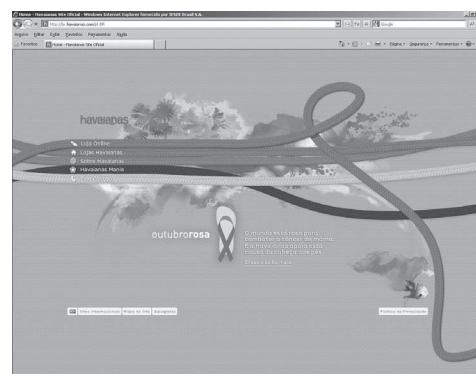
Fonte: Microsoft Internet Explorer 6.



Fonte: Microsoft Internet Explorer 6.



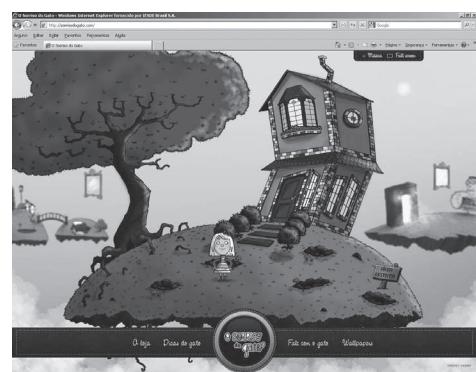
Fonte: Microsoft Internet Explorer 6.



Fonte: Microsoft Internet Explorer 6.



Fonte: Microsoft Internet Explorer 6.



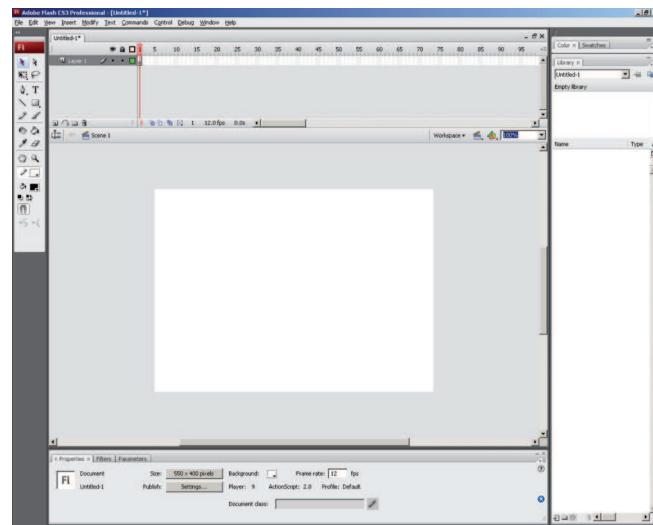
Fonte: Microsoft Internet Explorer 6.

Telas do Flash

No Flash temos duas áreas distintas para a criação do filme, cada uma tendo finalidades diferentes: uma para a criação e outra que permite o teste do filme em partes ou na íntegra.

Design

Nessa área damos forma e vida ao filme. É aqui onde é criado todo o filme, fotos, textos, animações e todo o conteúdo da animação.



Fonte: Adobe Flash CS3 Professional.

Esta área é dividida em três partes distintas: palco, barras e ferramentas.

Palco

O palco é a área onde efetivamente desenhamos nossos objetos e formas para a animação. Ele não se restringe à área branca da tela, que corresponde à área de visualização do usuário, ele tem áreas livres para todos os lados. Essas áreas livres têm grande utilização como área de descanso de objetos que ainda não foram animados em um pequeno filme ou de objetos que devem cruzar toda área de visualização. É no palco que inserimos tudo que deverá ser animado no filme.



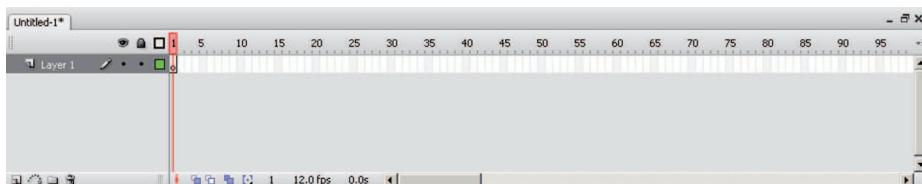
Fonte: Adobe Flash CS3 Professional.

Introdução à animação

A animação é o processo de criação e mudança das formas que ocorre durante um período de tempo. A animação atua sobre objetos alterando sua posição, tamanho, cor ou forma. Durante uma animação podem ocorrer vários eventos e baseado nesses eventos podem ocorrer ações de controle. Uma animação é como um filme, onde temos itens como tamanho, duração e cenas, porém no Flash temos controle total sobre esse filme: podemos avançar, voltar, pausar, mudar de filme, tudo isso usando *scripts* específicos que poderemos atribuir aos objetos.

Timeline

Toda a animação é baseada no conceito de tempo, e para cada intervalo de animação existe um número fixo de etapas que chamamos de quadros (*frames*), os quais são exibidos por segundo de animação. No Flash podemos ter esses valores personalizados. Mas para interagirmos com esses valores devemos sempre utilizar a linha de tempo que exibe com exatidão informações como número do quadro atual, tipo de quadro, se ele tem conteúdo ou não. Veja a *timeline* do Flash:



Fonte: Adobe Flash CS3 Professional.

Na figura acima visualizamos a linha de tempo do Flash exibindo o marcador de quadros, o primeiro quadro da animação. Todo o controle do filme é feito baseado na *timeline*.

Frames

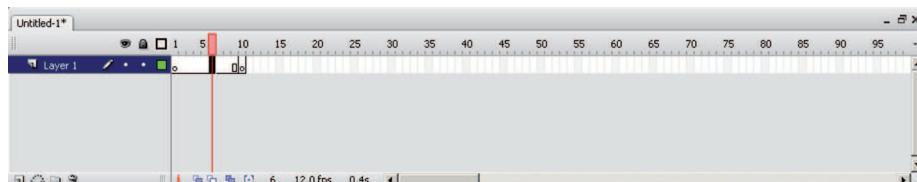
Como foi visto anteriormente, a *timeline* controla o fluxo do filme, baseada nos frames. No Flash temos dois tipos de frames: os frames normais e os frames chaves. Em todas as animações existem os dois tipos de frames. Na verdade a animação só começa a existir quando temos um grupo de frames com conteúdos diferentes.

Toda a animação começa com um frame chave (*key frame*). O *key frame* é uma marcação de início ou fim de um instante do filme. Toda vez que inserirmos um *key frame* o Flash sabe que ali ocorrerá alguma coisa, seja um início ou fim de animação.

Para inserirmos um *key frame* na *timeline* podemos utilizar a tecla de atalho F6. Automaticamente o Flash duplicará o conteúdo do frame atual para este novo frame. Podemos também inserir quadros chave em branco (*blank key frame*). Esse tipo de quadro tem a mesma função do *key frame* normal, porém nesse caso o Flash não copia o conteúdo do frame antecessor para o novo frame.

Para inserir um *blank key frame* utilizamos a tecla de atalho F7. Toda vez que você inicia um filme, o Flash cria automaticamente o primeiro *key frame*, no quadro 1 da primeira *layer* do filme, isso porque não é possível você começar um filme sem um quadro chave.

Além do *key frame* o Flash tem também o quadro normal, ou melhor dizendo, o frame simples. Esse tipo de frame sempre vem depois de um *key frame*. Ele não sofre alteração nenhuma durante toda a animação. Cada tipo de frame tem uma marcação diferente no Flash, como podemos ver abaixo:



Fonte: Adobe Flash CS3 Professional.

Na figura acima temos uma *layer* com um *key frame* com conteúdo seguido de um frame simples e um *blank key frame*.

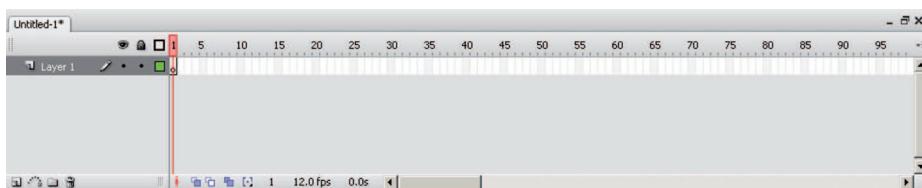
Interpolação de forma

As animações com formas do Flash podem ser criadas facilmente porque o Flash é capaz de calcular os vetores necessários para esse fim em tempo de execução. Basicamente falando, podemos definir qual a forma inicial, marcarmos o intervalo de quadros que desejamos e a forma final, e o Flash se encarrega de criar o restante dos quadros automaticamente. Além da interpolação de forma, o Flash também cria a interpolação de movimento.

Para criar uma interpolação de movimento devemos criar a figura que será interpolada no quadro 1 do filme, selecionar qual o quadro final da animação (o 20, por exemplo) e criar um quadro chave diretamente no quadro 20. Para isso basta selecionar o quadro 20 e pressionar F6.

A interpolação de forma pode ser feita com objetos diferentes, ou seja, você pode transformar elipse em quadrado ou linha em elipse, por exemplo. Para isso deve criar os quadros mostrados anteriormente, selecionar o último quadro, e desenhar a nova forma.

Depois de desenhar a forma, selecione a linha da interpolação como mostrado na figura a seguir:



Fonte: Adobe Flash CS3 Professional.

Agora na barra de propriedades selecione o *shape* no item *tween*. Quando você selecionar esse item a animação já estará criada, porém você pode definir mais dois parâmetros, o *ease*, que define a aceleração do objeto, e o *blend*, que define se a interpolação será distributiva ou angular. O item *blend* é aplicável em interpolações com linhas múltiplas e objetos complexos, nesse tipo de animação ele é dispensável.

Perceba que no intervalo de tempo que criamos a animação o Flash fez uma marcação verde. Isso porque temos uma interpolação de forma e cada tipo de animação tem uma cor diferente.

Além das transformações mostradas acima podemos criar diversas outras. Veja algumas animações de interpolação de forma bastante utilizadas:

Com elipse

- Desenhe uma pequena elipse no canto superior da tela.
- Defina 30 quadros para a interpolação.
- No quadro 30, desenhe duas elipses no canto inferior da tela.
- Apague a elipse superior do quadro 30.

- Selecione a linha da interpolação.
- Na barra de propriedades defina com *shape* o item *tween*.

Com quadrados

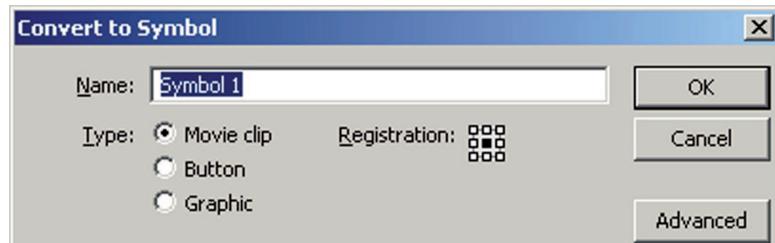
- Crie um quadrado que ocupe toda a tela.
- Defina 50 quadros para a interpolação.
- No quadro 50 delete o quadrado e desenhe duas linhas paralelas verticais.
- Selecione a linha da interpolação.
- Na barra de propriedades defina com *shape* o item *tween*.

Interpolação de movimento

Além da animação que já criamos podemos criar animações de movimento que podem também sofrer alterações de cores e opacidade, isso em um único intervalo.

Para criar esse tipo de animação temos que “proteger” o objeto que desejamos animar, para isso devemos converter este objeto em um símbolo. Os símbolos no Flash têm várias finalidades. A mais comum refere-se à possibilidade de reutilizar um mesmo símbolo várias vezes, ou seja, você pode criar um símbolo baseado em uma elipse que será usado dezenas de vezes em seu *movie*.

Para converter qualquer forma em símbolo você deve selecionar a forma desejada e pressionar a tecla F8 do teclado, feito isso o Flash exibirá a seguinte tela:



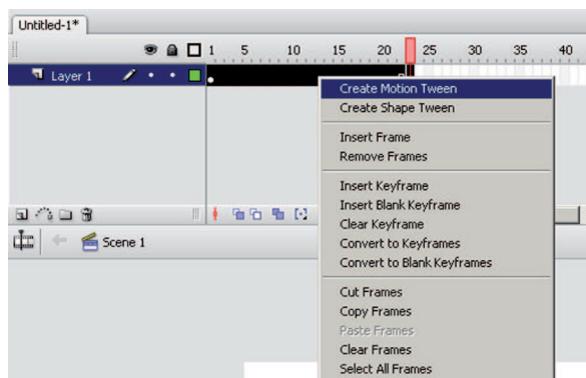
Fonte: Adobe Flash CS3 Professional.

Nessa tela você tem algumas opções para a criação do símbolo:

Opção	Descrição
Name	É o nome que o símbolo terá em sua biblioteca de símbolos.
Behavior	É o comportamento que o símbolo assumirá no Flash, sendo que o símbolo gráfico é o mais comum, usado para proteger formas e criar animações, o <i>button</i> é usado para interações de botão e o <i>movie clip</i> é frequentemente usado para interação de animação e programação dentro do Flash.
Registration	É o ponto de registro do símbolo em relação a sua rotação e disposição na tela.

Agora que você já criou seu símbolo, vamos marcar o frame 20 e pressionar F6. Agora no frame 20, selecione o seu símbolo e na barra de propriedades você tem a opção *color*. Essa caixa tem quatro itens: *Alpha*, *Brightness*, *Tint* e *Advanced*.

Colors	
Efeito	Descrição
Alpha	Define a opacidade (transparência) do objeto que varia de 100% (opaco) até 0% (transparente).
Brightness	Define o brilho aplicado à cor do objeto e varia de -100% (escuro) até +100% (claro).
Tint	Aplica uma camada de cor sobre a cor do objeto sem a necessidade de alterar o próprio objeto, tem faixa variável de 0% (cor natural) até 100% (cor de aplicação). Podemos também escolher qual a cor a ser aplicada em nossa paleta de cores ou por meio da escala RGB.



Fonte: Adobe Flash CS3 Professional.

Após você definir qual o efeito que deseja, selecione um frame no meio do intervalo criado e clique com o botão direito do mouse sobre a *timeline* e selecione o item *Create a Motion Tween*. Feito! Agora é só executar a sua animação.



■ Conhecendo páginas dinâmicas

Uma linguagem usada para deixar as páginas HTML mais dinâmicas seria o JavaScript, mas essa é uma linguagem que não oferece conexão com banco de dados. Portanto, precisamos conhecer mais algumas linguagens para *web*.

Uma página dinâmica tem a capacidade de se ajustar às necessidades do internauta, ou seja, o internauta é capaz de interagir com a página de forma a obter respostas diferentes de acordo com o que ele preencher, clicar ou de acordo com as opções escolhidas em um determinado formulário. Uma página dinâmica oferece opções diferentes para internautas diferentes. E isso traz grandes benefícios para a empresa, pois proporcionará um tratamento personalizado aos seus clientes.

Além de disponibilizar recursos personalizados aos internautas, uma página dinâmica também é capaz de fazer manipulação em banco de dados, que é a sua principal vantagem. Dessa forma, clientes podem fazer pedidos pela internet, consultar o saldo da conta bancária e muito mais.

As linguagens de *web* utilizadas para se fazer manipulação em banco de dados são muitas. Podemos citar algumas: JSP, C#.Net, VB.NET entre outras.

Utilizaremos uma poderosa ferramenta de desenvolvimento, o Visual Studio.NET, bem como conheceremos algumas linguagens de *web* mais usadas no mercado.

Plataforma .NET

Antes de começarmos a falar das linguagens vamos conhecer um pouco mais da Plataforma .NET. Com os recursos oferecidos por essa tecnologia é possível a criação de programas que rodem ou que utilizem a internet como meio de comunicação. Isso fará com que milhares de programadores de aplicativos *desktop* possam reutilizar suas capacidades para criação de programas mais escaláveis, robustos e capazes de se comunicar com outros sistemas, tornando a integração de dados ainda mais fácil.

Para facilitar ainda mais essa interligação entre as aplicações e sistemas, a Plataforma .NET disponibiliza suporte completo à orientação de objetos, estrutura de dados, *namespaces*, linguagem XML etc.

A Plataforma .NET oferece uma grande capacidade de processamento distribuído pela internet. Isso quer dizer que você pode criar aplicações onde seus objetos ficam “distribuídos”, ou seja, separados da aplicação e hospedados em servidores na *web*, por essa característica, eles passam a ser chamados de *web services*. Um *web service* seria uma coleção de objetos que podem ser executados de qualquer aplicação pela internet.

O que é XML?

XML (*Extensible Markup Language*) é uma linguagem utilizada para armazenamento de dados em forma de texto. Tudo que é enviado para as aplicações pelos *web services* é em XML. Como as aplicações podem ser feitas em diferentes linguagens que podem utilizar o mesmo *web service*, foi preciso criar uma maneira de padronizar a comunicação, e a Plataforma .NET utiliza arquivos XML para representar as informações.

Respostas de métodos, cálculos e, até mesmo, tabelas de banco de dados são enviadas pela Plataforma .NET através de XML. Por exemplo, imagine que o gerente de uma empresa solicite para algumas filiais enviarem a lista de clientes que possuem. Essa lista pode vir de várias formas, em forma de arquivo de texto, de tabela etc. Ao receber os dados é necessário entendê-los, já que os campos podem ser diferentes. Se a solicitação é feita todo mês, uma maneira de se facilitar o trabalho seria através de documentos XML, pois os arquivos seriam de texto (forma rápida de envio dos dados) e independente de como os registros estariam organizados no arquivo seria bem simples sua leitura, mesmo porque existem maneiras de se transformar um arquivo XML em uma tabela de banco de dados ou mesmo em uma tabela HTML para que seja exibida ao internauta.

Vamos ver um exemplo de XML. Considere a seguinte tabela:

Código	Nome	Salário
1	Julio Cesar Pereira	1000
2	Leandro Almeida	1200
3	Sheila Alvez	950

Em XML, seria representada da seguinte forma:

```
<EMPRESA>
  <FUNCIONARIO>
    <codigo>1</codigo>
    <nome>Julio Cesar Pereira</nome>
    <salario>1000</salario>
  </FUNCIONARIO>
  <FUNCIONARIO>
    <codigo>2</codigo>
    <nome>Leandro Almeida</nome>
    <salario>1200</salario>
  </FUNCIONARIO>
  <FUNCIONARIO>
    <codigo>3</codigo>
    <nome>Sheila Alvez</nome>
    <salario>950</salario>
  </FUNCIONARIO>
</EMPRESA>
```

XML é um meio de armazenar os dados em memória ou em arquivos de texto de forma simples e eficiente.

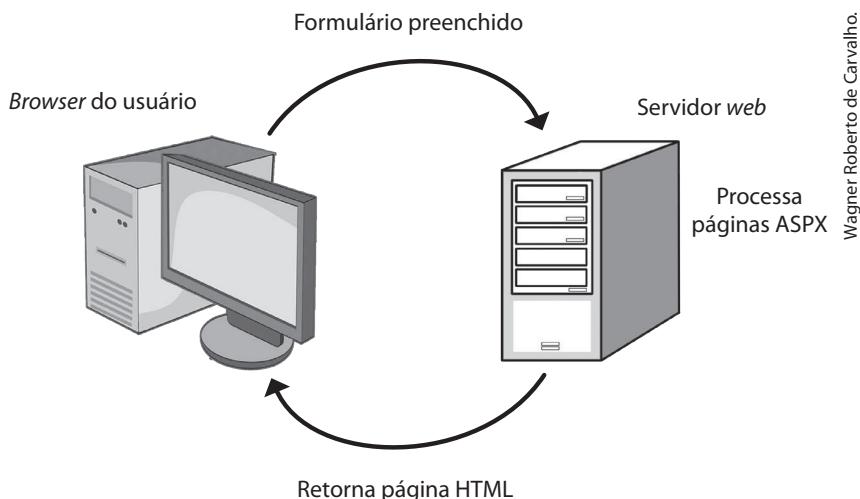
Entendendo o ASP.NET

A linguagem ASP.NET, desenvolvida pela Microsoft, trouxe uma grande facilidade em se montar *Web Form* (formulário para web). Suas novas *tags* auxiliam na criação de formulários dinâmicos. O ASP.NET oferece uma série de componentes úteis para os formulários das aplicações, componentes estes que dificilmente eram vistos nas páginas da internet.

Mas essa não foi a única vantagem que o ASP.NET trouxe. Uma de suas principais características está na capacidade de trabalhar com os dados dos formulários criados utilizando muitas outras linguagens de programação, o que facilita o desenvolvimento e a mudança da plataforma em que seu site está sendo executado.

Podemos dizer que o ASP.NET, por si só, veio para facilitar a montagem dos Web Forms e as linguagens para a Plataforma .NET vieram para manipular os dados dos Web Forms. O ASP.NET suporta uma boa quantidade de linguagens para trabalharmos, o que o torna uma ótima opção para desenvolvimento de aplicações web.

Outra mudança importante está na forma como as páginas ASP.NET são executadas. Quando requisitadas pela primeira vez ao servidor, ela é compilada, e as próximas vezes, o servidor utiliza os recursos já compilados. Mas, a resposta ao internauta é sempre em HTML.



Web Form é um formulário para web, diferentemente de Windows Form. A principal diferença está na maneira como são executados. O Windows Form é um programa compilado, já o Web Form são códigos executados pelo servidor e interpretados pelo *browser*.

Para entendermos melhor como funciona essa poderosa linguagem, vamos ver um exemplo. Se montarmos o seguinte formulário em ASP.NET:



Fonte: Microsoft Internet Explorer 6.

Teríamos como código fonte:

```
<html>
  <head>
    <title>Cadastro</title>
  </head>
  <body>
    <form id="Form1" method="post" runat="server">
      <asp:Label id="Label1" runat="server">
        e-mail:
      </asp:Label>
      <br>
      <asp:TextBox id="TextBox1" runat="server">
      </asp:TextBox>
      <br>
      <asp:Button id="Button1" runat="server" Text="Enviar">
      </asp:Button>
    </form>
  </body>
</html>
```

Perceba que não é difícil identificar o que é ASP.NET, pois podemos notar duas características marcantes:

- A tag <asp: ...> e </asp: ...>.
- A presença do atributo runat = "server".

Requisitos para se executar uma página ASP.NET

O principal requisito para usar o ASP.NET é ter instalado o .NET FrameWork da Microsoft, que pode ser encontrado no próprio site da Microsoft: www.microsoft.com/downloads/en/default.aspx.

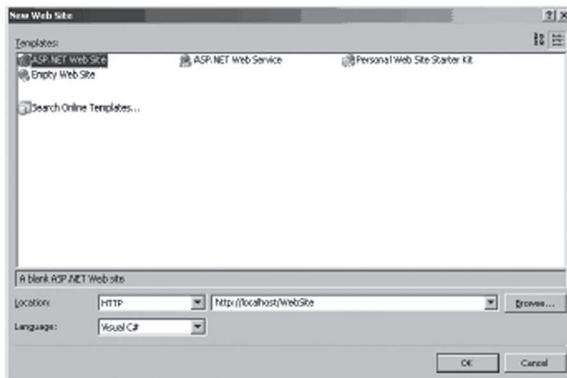
Após instalar o .NET FrameWork, é necessário ter também o IIS (*Internet Information Service*), que pode ser encontrado no próprio CD de instalação do Windows 2000 e posteriores. O IIS oferece recursos de um servidor *web*, e como o ASP.NET é executado no servidor *web*, é necessário ter esse *software* instalado. Após instalar o IIS será gerada uma hierarquia de diretórios no seu computador: C:\InetPub\wwwroot\...

Para saber se está tudo certo, abra o *Browser* e digite o seguinte endereço:
<http://localhost/>

Se o IIS estiver instalado corretamente e em execução, aparecerá uma página falando sobre o *Active Server Pages*. Todas as suas aplicações devem ser salvas dentro do servidor *web*, logo, dentro do diretório "wwwroot". E para acessá-las digite o endereço da seguinte forma: <http://localhost/Sua-Pasta/suapagina.aspx>

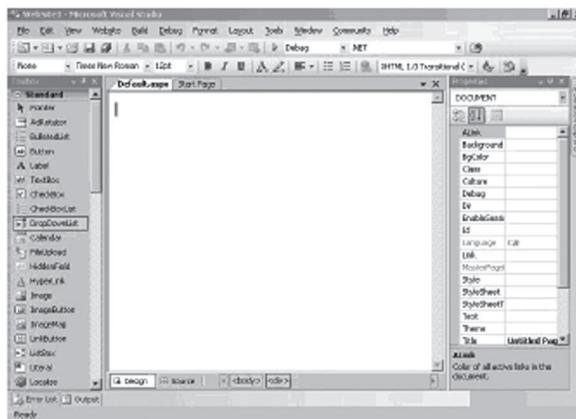
Montando um formulário com ASP.NET

Vamos criar um formulário em ASP.NET usando o VisualStudio.NET da Microsoft. Inicie o VisualStudio.NET, escolha um Novo Projeto (Create - New WebSite), marque na parte de Templates a opção ASP.NET WebSite e escolha a linguagem C# em Language, dê o nome de exemplo ASP ao seu projeto como mostra a figura a seguir.



Fonte: Microsoft Visual Studio.

Clique em Ok. Deverá aparecer a seguinte tela:



Fonte: Microsoft Visual Studio.

Vamos conhecer algumas diferenças entre Web Form e o Windows Form:

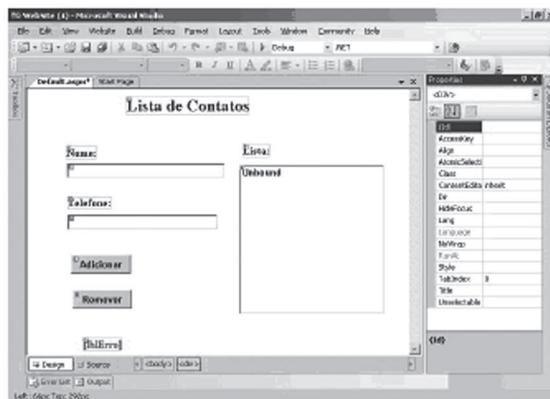
Web Form

- Precisa do *browser* para ser visualizado.
- Os objetos possuem ID.
- Todo código fonte do formulário é em ASP.NET que é interpretado pelo servidor.
- O formulário pode ser visto em qualquer sistema operacional, basta ter um navegador *web*.
- É acessado pela internet.

Windows Form

- Não depende de outros programas para ser exibido.
- Os objetos possuem Name.
- O código-fonte que gera o formulário não é visível.
- O programa final só pode ser executado na plataforma Windows.
- Possui um arquivo executável.

Agora que já conhecemos um pouco sobre os Web Forms, vamos continuar o nosso exemplo. Perceba que existe um ToolBox (Barra de Ferramentas) assim como nos Windows Forms, e também que alguns objetos possuem o mesmo nome. Crie um formulário com a seguinte aparência:



Fonte: Microsoft Visual Studio.

Propriedades dos objetos

WebForm1	
Propriedade	Valor
Title	Lista de Contatos

Label1	
Propriedade	Valor
Text	Lista de Contatos
Font - Bold	True
Font - Size	X-Large

Label2	
Propriedade	Valor
ID	lblNome
Text	Nome:
Font - Bold	True

Label3	
Propriedade	Valor
ID	lblTelefone
Text	Telefone:
Font - Bold	True

Label4	
Propriedade	Valor
ID	lblLista
Text	Lista:
Font - Bold	True

Label5	
Propriedade	Valor
ID	lblErro
Text	
Font - Bold	True
Font - Color	#ff0000

TextBox1	
Propriedade	Valor
ID	txtNome
Font - Bold	True

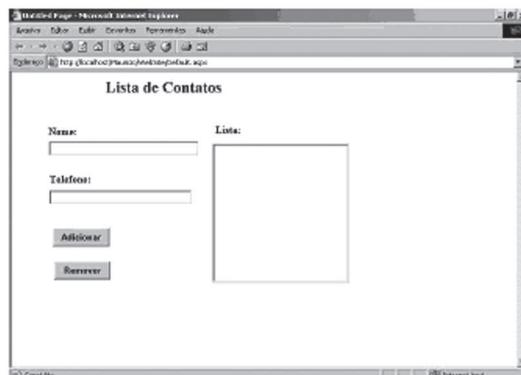
TextBox2	
Propriedade	Valor
ID	txtTelefone
Font - Bold	True

Button1	
Propriedade	Valor
ID	txtAdicionar
Text	Adicionar

Button2	
Propriedade	Valor
ID	txtRemover
Text	Remover
Font - Bold	True

ListBox1	
Propriedade	Valor
ID	IstContatos
Font - Bold	True

Muito bem, agora execute sua aplicação para vermos como ficou o resultado no *browser* e, se necessário, faça alterações.



Fonte: Microsoft Internet Explorer 6.

Para ver o código-fonte do formulário, basta clicar em Source, que se encontra na parte de baixo da tela ao lado esquerdo (no modo Design).

Você pode perceber que os botões não estarão funcionando, pois as codificações são realizadas através da linguagem C# .Net (C Sharp .Net).



Existem várias linguagens de programação que podemos utilizar para manipular os dados dos formulários ASP.NET. Neste capítulo iremos conhecer a linguagem C# (C Sharp).

Escolhemos a linguagem C# pois boa parte das classes do .NET FrameWork foram feitas em C#, o que a torna uma excelente linguagem para desenvolvimento nessa plataforma.

A linguagem C# foi desenvolvida pela Microsoft juntamente com a Plataforma .NET, tornando-se uma linguagem poderosa para seus desenvolvedores. Por ser uma linguagem que surgiu junto com o .NET possui as principais vantagens de outras grandes linguagens, como a linguagem Java e o C++.

Devido a essas influências, muitos programadores de Java podem optar por desenvolver em C# no VisualStudio.NET, já que as linguagens não possuem tantas diferenças. Por essa e outras razões, a Microsoft conseguiu criar uma ferramenta e uma linguagem de desenvolvimento que juntas, oferecem grandes vantagens.

Além da linguagem C# temos outras como a VB.NET e J#, por exemplo.

Características da linguagem C#

Antes de alguns exemplos, precisamos conhecer algumas características importantes da linguagem C#.

- Totalmente orientada a objetos.
- É uma linguagem tipada, ou seja, tudo possui um *data type*.
- *Case-sensitive* (*Nome* é diferente de *nome*).
- Todo o controle de memória é feito através do .NET FrameWork.

Algumas ferramentas para se desenvolver em C#:

- Microsoft Visual Studio.NET.
- ASP.NET WebMatrix.
- Eclipse for C#.

Sintaxe do C#

Como qualquer outra linguagem, o C# possui sua própria sintaxe. Apesar de ser bem parecida com C++ e Java, vamos ver alguns pontos importantes:

Comentários

Para se colocar comentários no código devemos utilizar:

// (barra barra) para comentário em uma linha.

/* (barra asterisco) para comentários em várias linhas */

Em todo fim de instrução deve haver um ";"

Operadores

Operadores aritméticos

Operador	Descrição
^	Exponenciação
*	Multiplicação
/	Divisão
%	Mod
+	Adição
-	Subtração

Operadores relacionais

Operador	Descrição
>	Maior que
<	Menor que
>=	Maior ou Igual
<=	Menor ou Igual
==	Igual
!=	Diferente

Operadores lógicos

Operador	Descrição
&&	AND (E)
	OR (OU)
!	NOT (NÃO)

Operadores especiais

Podemos também fazer operações com incrementos, veja:

`x=x+1`

Para essa situação poderíamos usar a seguinte sintaxe:

`x++`

Esse tipo de notação pode ser usada com qualquer um dos operadores aritméticos.

Estrutura condicional

Vamos conhecer a principal estrutura condicional de uma linguagem de programação, o *SE*.

Exemplo:

Se tivéssemos o seguinte algoritmo:

`SE nota >= 7 ENTÃO situacao = 'APROVADO' SENÃO situacao = 'REPROVADO' FIM SE`

Em C# seria:

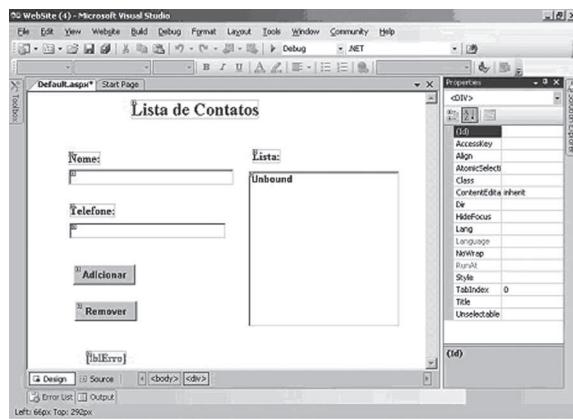
`if(nota >= 7) situacao = "APROVADO"; else situacao = "REPROVADO";`

Para representar um bloco de instruções, devemos utilizar o mesmo que em Java:

`{"}" e "{}"`

Montando um exemplo com ASP.NET e C#

Agora que já conhecemos um pouco do C# vamos ver como as coisas funcionam na prática. Inicie o VisualStudio.NET e abra o formulário “lista de contatos”.



Fonte: Microsoft Visual Studio.

Vamos iniciar a codificação pelo botão Adicionar. Dê um duplo clique no botão adicionar, aparecerá o evento *Click*.

```
protected void btnAdicionar_Click(object sender, EventArgs e)
{
    string item;
    if((txtNome.Text!="") && (txtTelefone.Text!=""))
    {
        item = txtNome.Text + " — " + txtTelefone.Text;
        lstContatos.Items.Add(item);
        txtNome.Text = "";
        txtTelefone.Text = "";
        lblErro.Text = "";
    }
    else
    {
        lblErro.Text = "Preenchimento dos campos obrigatório!";
    }
}
```

Execute sua aplicação e corrija os erros, se necessário. Preste atenção no nome dos objetos, não esqueça que o C# é *case sensitive* e procure respeitar a identação das linhas de código, pois isso facilita a leitura do mesmo.

Com o primeiro botão funcionando, vamos aos códigos do evento *Click* do botão *btnRemover*.

```
protected void btnRemover_Click(object sender, EventArgs e)
{
    if(lstContatos.SelectedIndex >= 0)
    {
        lstContatos.Items.RemoveAt(lstContatos.SelectedIndex);
        lblErro.Text = "";
    }
    else
        lblErro.Text = "Necessário selecionar um item da lista!";
}
```

Vamos ver como ficou o resultado final no *browser*. Perceba que ao tentar remover sem selecionar um item da lista, ocorre um erro, que é exibido na *label* (*lblErro*). No código acima observe como foram usadas as chaves ("{" e "}"). Perceba que para condição verdadeira é preciso ser executado duas linhas de comando, portanto, usa-se chaves, e caso a condição seja falsa, será executado apenas uma, tornando o uso das chaves dispensável.



■ Projeto final

Web service

O *web service* é uma das tecnologias mais utilizadas. Um *web service* é uma coleção de métodos armazenados em um servidor *web* que podem ser acessados de qualquer lugar via internet.

Com um *web service* podemos utilizar os mesmos métodos em vários sites sem a necessidade de refazer os códigos, basta indicar o *web service* como referência. Com isso, muitos códigos são economizados e as aplicações ficam mais “leves”, pois boa parte dos recursos não fica localizada junto às aplicações, e sim em um *web service*.

O que faremos

Como aplicação dos conceitos, desenvolveremos um site de comércio (*E-commerce*), que irá divulgar seus produtos e oferecer um cadastro aos internautas para receberem promoções futuras. Outro recurso disponível no site será a possibilidade do administrador realizar cadastro de novos produtos pelo próprio site, sendo necessário informar *log in* e *senha* de acesso para isso. Para ajudar nos códigos, utilizaremos o *web service* que gerenciará todo o acesso ao banco de dados SQL-Server.

Utilizaremos quatro ferramentas de desenvolvimento para a criação do nosso *E-Commerce*:

- SQL-Server
- VisualStudio.NET
- Flash
- Dreamweaver

A integração de várias ferramentas no desenvolvimento de uma aplicação nos dias de hoje é indispensável. O mercado está exigente e o conhecimento de várias ferramentas e tecnologias é indispensável.

Iniciando

Podemos iniciar a aplicação de várias formas. Poderíamos também ter uma equipe de desenvolvimento onde cada um seria responsável por uma parte do projeto, acelerando a finalização do mesmo.

É muito importante que você dê atenção a cada parte do projeto, procure não se esquecer dos nomes das tabelas, formulários, *layout* das tabelas, nomes dos arquivos e principalmente entender o que será feito em cada um deles.

Para começar, vamos criar as tabelas no banco de dados. Inicie o QueryAnalyser e crie as seguintes tabelas:

```
CREATE TABLE tbContato
(
    codigo INT IDENTITY PRIMARY KEY,
    nome VARCHAR (40) NOT NULL,
    email VARCHAR (50) NOT NULL,
    sexo char (1) NOT NULL
)
go

CREATE TABLE tbCategoria
(
    codigo INT IDENTITY PRIMARY KEY,
    nome VARCHAR (20) NOT NULL,
)
go

CREATE TABLE tbProduto
(
    codigo INT IDENTITY PRIMARY KEY,
```

```
    nome VARCHAR (40) NOT NULL,  
    descricao VARCHAR (50) NOT NULL,  
    valor MONEY NOT NULL,  
    categoria int NOT NULL,  
constraint categoria_fk  
foreign key (categoria)  
references tbcategoria (codigo)  
)  
  
go  
  
CREATE TABLE tbAcesso  
(  
    codigo INT IDENTITY PRIMARY KEY,  
    login VARCHAR (15) NOT NULL,  
    senha VARCHAR (10) NOT NULL  
)
```

Execute o código e corrija os erros, se necessário. Se tudo der certo, aparecerá o seguinte aviso:

The command(s) completed successfully.

Agora vamos inserir alguns valores nas tabelas **tbCategoria** e **tbAcesso**. Salve o *script* acima com o nome de **tabelas** e solicite uma nova *query* usando o atalho **Ctrl+N**.

Digite os seguintes códigos:

```
INSERT INTO tbCategoria(nome)VALUES('TV')  
INSERT INTO tbCategoria(nome)VALUES('DVD')  
INSERT INTO tbCategoria(nome)VALUES('SOM')
```

```
INSERT INTO tbAcesso(login, senha)  
VALUES ('aluno', 'senha')
```

Execute os comandos acima, se tudo estiver correto deverá aparecer quatro vezes o seguinte aviso:

(1 row(s) affected)

Para ver os registros inseridos nas tabelas, digite e execute o comando abaixo:

```
SELECT * FROM <NomeDaTabela>
```

Procure não esquecer o nome das tabelas e a finalidade de cada uma:

tbContato – internautas que visitam o site;

tbProduto – produtos a serem consultados através do site;

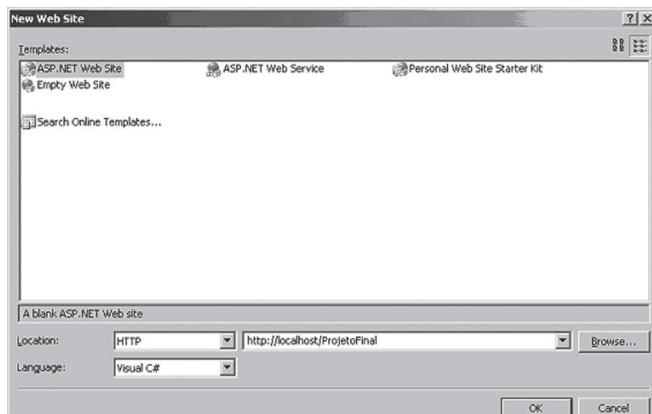
tbCategoria – categorias possíveis para os produtos cadastrados;

tbAcesso – usuários que têm acesso a áreas restritas do site.

Com as tabelas criadas podemos fechar o QueryAnalyser.

Até aqui temos a parte do banco de dados concluída. Podemos partir para a criação das páginas que irão acessar os dados armazenados. Para fazer essa parte de acesso utilizaremos o VisualStudio.NET.

Inicie o VisualStudio.NET e solicite um novo projeto (Create - WebSite). Escolha a linguagem C# e o tipo de projeto – será um ASP.NET Web Site. Nomeie o projeto para ProjetoFinal como mostrado na figura abaixo:



Fonte: Microsoft Visual Studio.

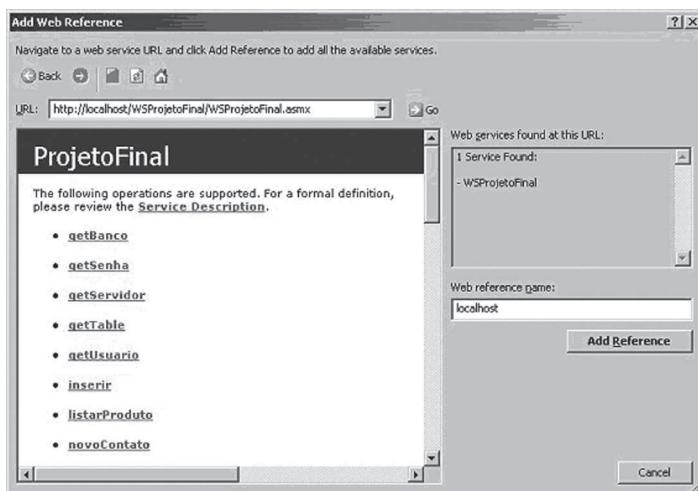
Configurando o web service

Antes de começarmos a desenvolver nossas páginas, vamos adicionar uma referência ao *web service* que utilizaremos no nosso projeto. Para isso vá até o menu web site - Add Web Reference.

Na tela que se abrirá digite o seguinte endereço e pressione Enter:

http://localhost/WSProjetoFinal/WSProjetoFinal.asmx

Deverá aparecer a seguinte tela:



Fonte: Microsoft Visual Studio.

Clique em *Add Reference*.

Agora vamos alterar o nome de nossa referência que está como *localhost*. Para isso vá até o menu *View - Solution Explorer*. No *Solution Explorer* terá uma pasta chamada *Web References*, clique no sinal de mais (+) ao lado dela e você verá a referência que acabamos de adicionar ao projeto.

Clique com o botão direito sobre o *localhost* e escolha a opção para renomear (rename). Altere o nome para *WSProjetoFinal*.

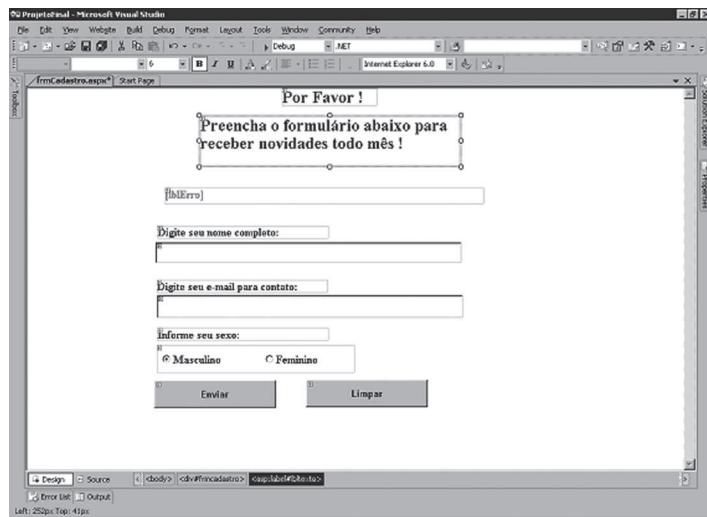


O *web service* já está criado e pronto para ser utilizado pelo projeto.

Página de cadastro – layout

Agora que já temos acesso ao banco de dados através do *web service*, podemos desenvolver as páginas do nosso projeto. Vamos começar pela *Página de Cadastro*.

Veja o *layout* abaixo:



Fonte: Microsoft Visual Studio.

Página de cadastro – propriedades

WebForm1	
Propriedade	Valor
File Name	frmCadastro
Title	Novo Cadastro

Obs.: para alterar o File Name da página vá até o Solution Explorer e renomeie o arquivo.

Label1	
Propriedade	Valor
ID	lblTitulo
Text	Por Favor !
Font - Bold	True
Font - Size	X-Large

Label2	
Propriedade	Valor
ID	lblTexto
Text	Preencha o formulário abaixo para receber novidades todo mês !
Font - Bold	True
Font - Size	X-Large

Label3	
Propriedade	Valor
ID	lblErro
Text	
Font - Bold	True
ForeColor	#ff0000

Label4	
Propriedade	Valor
ID	lblNome
Text	Digite seu nome completo:
Font - Bold	True

Label5	
Propriedade	Valor
ID	lblEmail
Text	Digite seu e-mail para contato:
Font - Bold	True

Label6	
Propriedade	Valor
ID	lblSexo
Text	Informe seu sexo:
Font - Bold	True

TextBox1	
Propriedade	Valor
ID	txtNome
Font - Bold	True

TextBox2	
Propriedade	Valor
ID	txtEmail
Font - Bold	True

RadioButtonList1		
	Propriedade	Valor
ID		rdlSexo
Font – Bold		True
Items	Selected	True
	Text	Masculino
	Value	M
	Selected	False
	Text	Feminino
	Value	F
RepeatDirection		Horizontal

Button1	
Propriedade	Valor
ID	btnEnviar
Font - Bold	True
Text	Enviar

Button2	
Propriedade	Valor
ID	btnLimpar
Font - Bold	True
Text	Limpar

É muito importante que você não esqueça o nome dos objetos e suas respectivas finalidades.

Página de cadastro – código

Evento *Click* do *btnLimpar*:

```
protected void btnLimpar_Click(...)
{
    txtNome.Text = "";
    txtEmail.Text = "";
    lblErro.Text = "";
}
```

Evento *Click* do *btnEnviar*:

```
protected void btnEnviar_Click(...)
{
    string texto = "";
    if (txtNome.Text.Trim() == "")
        lblErro.Text = "Preenchimento do campo nome obrigatório!";
    else
    {
        if (txtEmail.Text.Trim() == "")
            lblErro.Text = "Preenchimento do campo e-mail obrigatório!";
        else
        {
            WSProjetoFinal.ProjetoFinal pf = new WSProjetoFinal.ProjetoFinal();
```

```
pf.novoContato(txtNome.Text, txtEmail.Text, rdlSexo.SelectedItem.  
Value);  
  
lblTitulo.Visible = false;  
  
lblTexto.Visible = false;  
  
lblErro.Visible = false;  
  
lblNome.Visible = false;  
  
lblEmail.Visible = false;  
  
lblSexo.Visible = false;  
  
txtNome.Visible = false;  
  
txtEmail.Visible = false;  
  
rdlSexo.Visible = false;  
  
btnEnviar.Visible = false;  
  
btnLimpar.Visible = false;  
  
texto = "<center><font size = 5>";  
  
texto += "<p>Obrigado</p>";  
  
texto += "<b>" + txtNome.Text + "</b>!<p>";  
  
texto += "Seu cadastro foi realizado</p>";  
  
texto += "com sucesso !!!<br>";  
  
texto += "Em breve estará recebendo<br>";  
  
texto += "novidades no seu e-mail<br>";  
  
texto += "<b>" + txtEmail.Text + "</b>";  
  
texto += "</font></center>";  
  
Response.Write(texto);  
}  
}  
}
```

Preste atenção na *identação do código*, uma boa dica para evitar erros com a *abertura e fechamento das chaves* seria, assim que você abrir uma *chave*, já a feche e coloque um comentário indicando qual estrutura *condicional* ela está finalizando.

Explicação do código

Antes de confirmarmos a inclusão dos dados na tabela, precisamos consistir os dados informados pelo internauta. Isso é feito utilizando algumas condições:

```
if(txtNome.Text.Trim() == "")
    lblErro.Text = "Preenchimento do campo nome obrigatório!";
else
{
    if(txtEmail.Text.Trim() == "")
        lblErro.Text = "Preenchimento do campo e-mail obrigatório!";
    else
{
}
```

Caso algum campo esteja vazio, uma mensagem de erro será exibida na *label* (*lblErro*). Agora, se os campos foram preenchidos, podemos inserir os dados na tabela *tbContato* (criada anteriormente), e para isso vamos executar o método *novoContato* do *web service* (*WSProjetoFinal*). Antes de executar o método devemos declarar um objeto (*pf*) do tipo *ProjetoFinal* (objeto gerenciado pelo *web service*) para que possamos utilizar seus métodos.

```
WSProjetoFinal.ProjetoFinal pf = new WSProjetoFinal.ProjetoFinal();
pf.novoContato(txtNome.Text, txtEmail.Text, rdlSexo.SelectedItem.Value);
```

Agora precisamos enviar o comando SQL para que seja executado pelo banco de dados, e o responsável por fazer isso é o nosso *web service* (*WSProjetoFinal*), onde devemos declarar um objeto (*pf*) do tipo *ProjetoFinal*, e utilizar um de seus métodos para enviar os dados ao banco de dados.

```
WSProjetoFinal.ProjetoFinal pf = new WSProjetoFinal.ProjetoFinal();
pf.inserir(sql);
```

O método *inserir* foi implementado dentro do *web service*. Perceba que não precisamos nos preocupar em abrir e fechar a conexão com o banco de dados, pois o próprio *web service* já faz isso, a única coisa que precisamos fazer é passar o comando SQL a ser executado.

Antes de exibir a mensagem de confirmação vamos “esconder” os objetos do formulário.

```
lblTitulo.Visible = false;
```

```
lblTexto.Visible = false;
```

```
lblErro.Visible = false;
```

```
lblNome.Visible = false;
```

```
lblEmail.Visible = false;
```

```
lblSexo.Visible = false;
```

```
txtNome.Visible = false;
```

```
txtEmail.Visible = false;
```

```
rdlSexo.Visible = false;
```

```
btnEnviar.Visible = false;
```

```
btnLimpar.Visible = false;
```

Após fazermos a inclusão dos dados na tabela, devemos dar uma resposta ao usuário confirmando a mesma, e isso é feito pela variável *texto*, que é exibida na tela usando o *Response.Write(texto)*.

```
texto = "<center><font size = 5>";
```

```
texto += "<p>Obrigado";
```

```
texto += "<b>" + txtNome.Text + "</b> !<p>";
```

```
texto += "Seu cadastro foi realizado";
```

```
texto += "com sucesso !!!<br>";
```

```

texto += "Em breve estará recebendo";

texto += "novidades no seu e-mail<br>";

texto += "<b>" + txtEmail.Text + "</b>";

texto += "</font></center>";

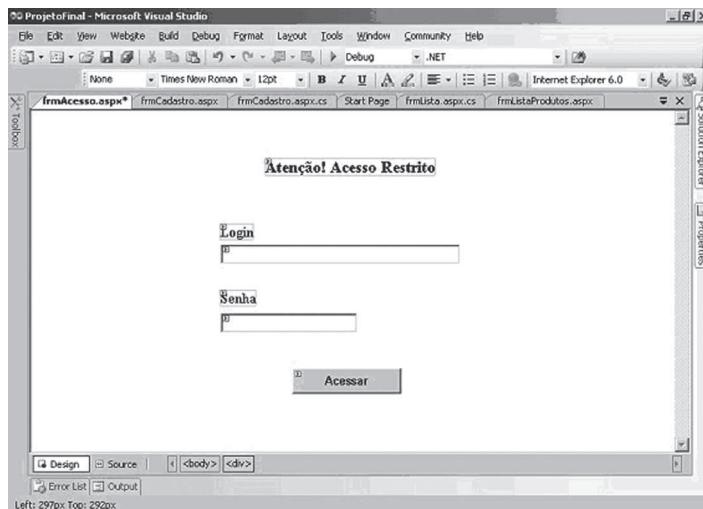
Response.Write(texto);

```

Veja que no código acima temos comandos em HTML para configurar o texto a ser exibido.

Página de acesso – *layout*

Agora vamos criar o formulário de acesso às áreas restritas do nosso site. No nosso caso, teremos uma página para cadastro dos produtos, que só pode ser acessada se os dados (*log in* e *senha*) informados estiverem corretos. Acrescente um novo formulário (*Menu Project - Add Web Form*) ao nosso projeto chamado de frmAcesso.aspx. Veja o *layout* abaixo:



Fonte: Microsoft Visual Studio.

WebForm1	
Propriedade	Valor
Title	Acesso Restrito

Label1	
Propriedade	Valor
Name	lblAviso
Text	Atenção! Acesso restrito!
Font - Bold	True
Font - Size	Large

Label2	
Propriedade	Valor
Text	Login
Font - Bold	True

Label3	
Propriedade	Valor
Text	Senha
Font - Bold	True

TextBox1	
Propriedade	Valor
ID	txtLogin
Text	
Font - Bold	True

TextBox2	
Propriedade	Valor
ID	txtSenha
Text	
Font - Bold	True

Button1	
Propriedade	Valor
ID	btnAcessar
Text	Acessar
Font - Bold	True

Para que ao executar a aplicação esse seja o formulário inicial, vá até o menu *Project – Solution Explorer*. Clique com o botão direito sobre o *frmAcesso* e escolha a opção *Set As Start Page*.

Página de acesso – código

Abra o evento *Click* do objeto *btnAcessar* e digite os códigos abaixo:

```
protected void btnAcessar_Click(...)  
{  
    WSProjetoFinal.ProjetoFinal pf = new WSProjetoFinal.ProjetoFinal();  
    string login = txtLogin.Text.Trim();  
    string senha = txtSenha.Text.Trim();  
    if((login != "")&&(senha != ""))  
    {  
        if (pf.verificarLogin(login, senha))  
            Server.Transfer  
                ("frmCadastroProduto.aspx");  
        else  
        {  
            lblAviso.ForeColor = System.Drawing.Color.Red;  
            lblAviso.Text = "Login e/ou Senha Inválido(s)!";  
        }  
    }  
    else  
    {  
        lblAviso.ForeColor = System.Drawing.Color.Red;  
        lblAviso.Text = "Por Favor! Preencha todos os campos!";  
    }  
}
```

Explicação do código

Aqui estamos criando um objeto do tipo *ProjetoFinal*, usando o *web service*, para que possamos consultar o banco de dados e verificar se o *log in* e *senha* informados estão corretos:

```
WSProjetoFinal.ProjetoFinal pf = new WSProjetoFinal.ProjetoFinal();
```

Ao atribuir o valor dos campos às variáveis, estamos usando o método *Trim()*, que tem a função de retornar o valor da propriedade *Text* tirando o excesso de espaços do lado direito e esquerdo do texto.

```
string login = txtLogin.Text.Trim();
```

```
string senha = txtSenha.Text.Trim();
```

Antes de pedir para o objeto *pf* verificar os dados informados, devemos nos certificar de que os campos foram preenchidos, e exibir aviso, se necessário.

```
if((login != "")&&(senha != ""))
```

Para saber se o *log in* está correto, pedimos para que o objeto *pf* verifique os dados usando o método *verificarLogin()*, no qual devemos passar 2 (dois) parâmetros: *log in* e *senha*. O nome da tabela (*tbAcesso*) está sendo usado dentro do *web service*.

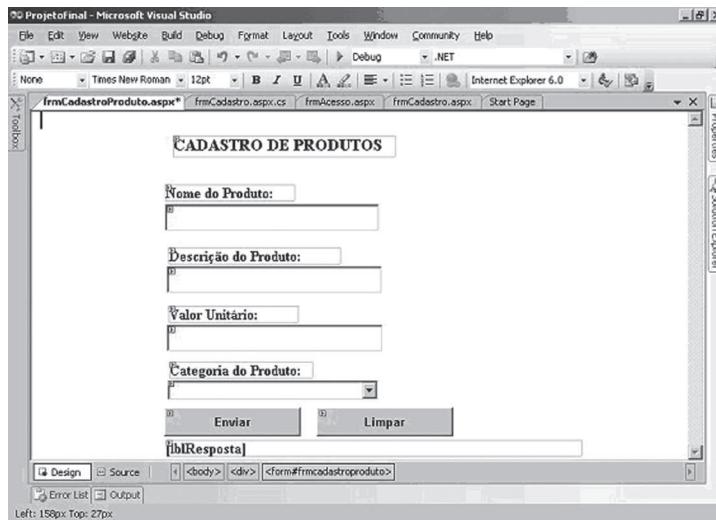
```
if (pf.verificarLogin(login, senha))
```

Caso os dados estejam corretos, chamamos a página *frmCadastroProduto.aspx* (que ainda vamos criar).

```
Server.Transfer("frmCadastroProduto.aspx");
```

Página de cadastro de produto – *layout*

Adicione mais um *Web Form* ao seu projeto com o nome de *frmCadastroProduto.aspx*. Veja o *layout* a seguir:



Fonte: Microsoft Visual Studio.

Página de cadastro de produto – propriedade

WebForm1	
Propriedade	Valor
Title	Cadastro de Produtos

Label1	
Propriedade	Valor
Text	CADASTRO DE PRODUTOS
Font - Bold	True
Font - Size	Large

Label2	
Propriedade	Valor
Text	Nome do Produto:
Font - Bold	True

Label3	
Propriedade	Valor
Text	Descrição do Produto:
Font - Bold	True

Label4	
Propriedade	Valor
Text	Valor Unitário:
Font - Bold	True

Label5	
Propriedade	Valor
Text	Categoria do Produto:
Font - Bold	True

Label6	
Propriedade	Valor
ID	lblResposta
Text	
Font - Bold	True
Font - Size	Medium

TextBox1	
Propriedade	Valor
ID	txtNome
Text	
Font - Bold	True

TextBox2	
Propriedade	Valor
ID	txtDescricao
Text	
Font - Bold	True

TextBox3	
Propriedade	Valor
ID	txtValor
Text	
Font - Bold	True

Button1	
Propriedade	Valor
ID	btnEnviar
Text	Enviar
Font - Bold	True

Button2	
Propriedade	Valor
ID	btnLimpar
Text	Limpar
Font - Bold	True

DropDownList1	
Propriedade	Valor
ID	dplCategoria
Font - Bold	True

Nesse formulário temos um objeto novo. O *DropDownList*, que será usado para listar as categoria da tabela *tbCategoria* (criada no início do projeto) assim que a página for carregada pela primeira vez.

Página de cadastro de produto – código

Vamos começar pelo evento *Page_Load*, para abrir esse evento basta clicar duas vezes na área de *design* da página. Veja os códigos a seguir:

protected void Page_Load(...)

{

if (!Page.IsPostBack)

{

WSProjetoFinal.ProjetoFinal pf = new WSProjetoFinal.ProjetoFinal();

dplCategoria.DataSource = pf.getTable("SELECT nome FROM
tbCategoria");

```
dplCategoria.DataTextField = "nome";  
dplCategoria.DataBind();  
}  
}
```

Explicação do código

Assim que a página for carregada, devemos listar as categorias possíveis no objeto *dplCategoria*, mas antes, devemos verificar se a página está sendo solicitada pela primeira vez, se for, pegamos as categorias na tabela *tbCategoria* e listamos em nosso objeto, caso a página esteja apenas sendo atualizada (F5), não há necessidade de solicitar uma nova lista, portanto, usamos a condição:

```
if (!Page.IsPostBack)
```

O *PostBack* indica se a página já foi carregada antes. O objeto *dplCategoria* possui uma propriedade chamada *DataSource*, que serve para indicarmos qual tabela deverá representar os itens da lista.

Para pegarmos os nomes das categorias de nossa tabela estamos usando o método *getTable* do *web service*, no qual devemos passar o comando *SQL*.

```
dplCategoria.DataSource = pf.getTable("SELECT nome FROM  
tbCategoria");
```

Depois de indicarmos a tabela, devemos indicar qual campo dessa tabela será usado na lista do *DropDownList*. E isso é feito alterando sua propriedade *DataTextField*.

```
dplCategoria.DataTextField = "nome";
```

Por último, executamos o método *DataBind()* do objeto *dplCategoria* que será responsável por adicionar os registros da tabela à lista do *DropDownList*.

```
dplCategoria.DataBind();
```

Muito bem, agora vamos aos outros objetos. Abra o evento *Click* do objeto *btnLimpar* e acrescente os seguintes códigos:

```
protected void btnLimpar_Click(...)
{
    txtNome.Text = "";
    txtDescricao.Text = "";
    txtValor.Text = "";
    lblResposta.Text = "";
    dplCategoria.SelectedIndex = 0;
}
```

Evento **Click** do **btnEnviar**:

```
protected void btnEnviar_Click(...)
{
    lblResposta.ForeColor = System.Drawing.Color.Blue;
    if(txtNome.Text.Trim() != "")
    {
        if(txtDescricao.Text.Trim() != "")
        {
            if(txtValor.Text.Trim() != "")
            {
                try
                {
                    WSProjetoFinal.ProjetoFinal pf = new WSProjetoFinal.ProjetoFinal();
                    pf.novoProduto(txtNome.Text, txtDescricao.Text, txtValor.Text, Convert.ToString(dplCategoria.SelectedIndex+1));
                    txtNome.Text = "";
                    txtDescricao.Text = "";
                }
            }
        }
    }
}
```

```
txtValor.Text = "";
lblResposta.Text = "";
dplCategoria.SelectedIndex = 0;
lblResposta.ForeColor =
    System.Drawing.Color.Green;
lblResposta.Text = "CADASTRO RELIZADO!!!";
}

catch
{
    lblResposta.ForeColor = System.Drawing.Color.Red;
    lblResposta.Text = "VALOR inválido!";
}

}

else
{
    lblResposta.Text = "VALOR obrigatório!";
}

//valor

}

else
{
    lblResposta.Text = "DESCRIÇÃO obrigatória !";
}

//descricao

}

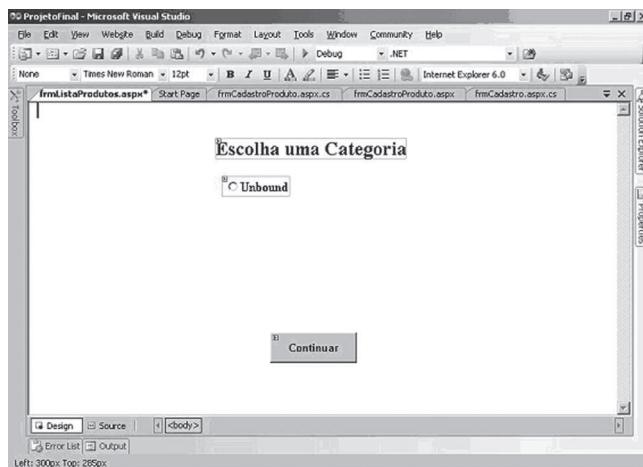
else
{
    lblResposta.Text = "NOME obrigatório!";
}

//nome

}
```

Página lista de produtos – layout

Nessa página iremos oferecer ao internauta a possibilidade de escolher uma das categorias possíveis para que seja exibida uma lista de produtos de acordo com a opção escolhida. Adicione mais uma página ao nosso projeto chamada de *frmListaProdutos.aspx*. Veja o *layout*:



Fonte: Microsoft Visual Studio.

Página lista de produtos – propriedade

WebForm1	
Propriedade	Valor
Title	Lista de Produtos

Label1	
Propriedade	Valor
Text	Escolha uma categoria
Font - Bold	True
Font - Size	X-Large

RadioButtonList1	
Propriedade	Valor
ID	rblCategoria
Font - Bold	True

Button1	
Propriedade	Valor
ID	btnContinuar Continuar
Text	
Font - Bold	True

Página lista de produtos – código

Evento *Load* da Página:

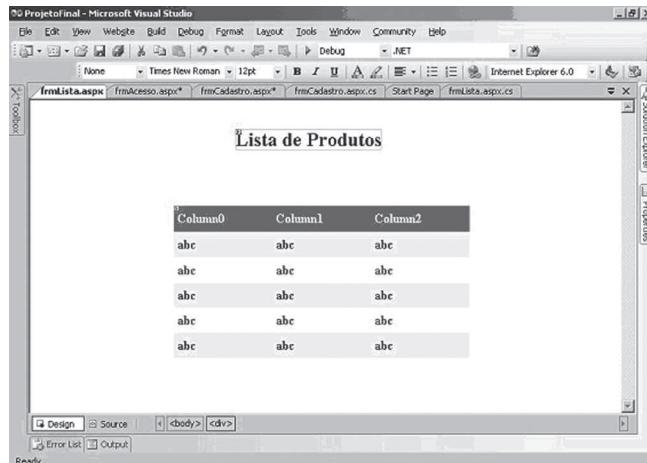
```
protected void Page_Load(...)
{
    if(!Page.IsPostBack)
    {
        WSProjetoFinal.ProjetoFinal pf = new WSProjetoFinal.ProjetoFinal();
        rblCategoria.DataSource = pf.getTable("SELECT nome FROM
        tbCategoria");
        rblCategoria.DataTextField = "nome";
        rblCategoria.DataBind();
    }
}
```

Evento **Click** do botão **btnContinuar**:

```
protected void btnContinuar_Click(...)
{
    Server.Transfer("frmLista.aspx?categoria=" + rblCategoria.SelectedItem.Value);
}
```

Página de lista – *layout*

Nesta página iremos exibir a lista de produtos que o internauta deseja, de acordo com a categoria selecionada na página anterior (frmListaProdutos). Acrescente outra página ao projeto chamada de *frmLista.aspx*. Veja o *layout*:



Fonte: Microsoft Visual Studio.

Página de lista – propriedade

WebForm1	
Propriedade	Valor
Title	Lista de Produtos

Label1	
Propriedade	Valor
Text	Lista de Produtos
Font - Bold	True
Font - Size	X-Large

DataGrid1	
Propriedade	Valor
ID	dtgProdutos
Font - Bold	True
Auto	Format Professional 1

Página de lista – código

Evento *Load* da Página:

```
protected void Page_Load(...)  
{  
    string categoria = Request.QueryString["categoria"];  
    WSProjetoFinal.ProjetoFinal pf = new WSProjetoFinal.ProjetoFinal();  
    dtgProdutos.DataSource = pf.listarProduto(categoria);  
    dtgProdutos.DataBind();  
}
```

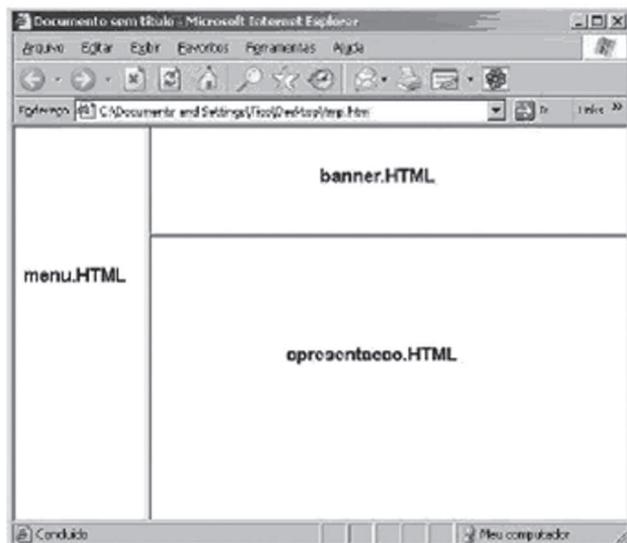
Para ver se os códigos acima estão corretos, execute o projeto começando pela página de Lista de Produtos (**frmListaProdutos**).

Montando o Index.html no Dreamweaver

Concluídas as páginas que permitem acesso ao banco de dados podemos nos preocupar com a parte de *design* do nosso site. Vamos iniciar o Dreamweaver e construir uma página de *links* para as páginas que criamos, e usar um frame para que possamos manter o menu sempre disponível ao internauta.

Com o Dreamweaver aberto, vá até o menu *Site - New Site*, e escolha a página do nosso projeto como sendo o novo site. Chame o novo site de *ProjetoFinal_SeuNome*.

Adicione um frame à página e monte a tela inicial do site. O tipo de fonte, imagem, ou qualquer outro item que achar necessário para a montagem do *design* da página ficará a sua escolha. Apenas procure ter uma página com as seguintes divisões e com os seguintes nomes:



Fonte: Microsoft Visual Studio.

Finalizando

Para ver o resultado final do seu site, abra o *browser* e digite o seguinte endereço:

<http://localhost/ProjetoFinal/index.html>

