Introdução ao Desenvolvimento de Sistemas *Web*

Prof[a]. Ms. Greisse Moser Badalotti

UNIASSELVI
2014

Elaboração:
Profª. Ms. Greisse Moser Badalotti

Revisão, Diagramação e Produção:
Centro Universitário Leonardo da Vinci – UNIASSELVI

Ficha catalográfica elaborada na fonte pela Biblioteca Dante Alighieri
UNIASSELVI – Indaial.

005.133
L237l Badalotti, Greisse Moser
Introdução ao desenvolvimento de sistemas web / Greisse Moser Badalotti. Indaial : Uniasselvi, 2014.
226 p. : il

ISBN 978-85-7830-858-2

1. Linguagens de programação
I. Centro Universitário Leonardo da Vinci.

APRESENTAÇÃO

Prezado(a) acadêmico(a)!

Bem-vindo à Introdução ao Desenvolvimento de Sistemas *WEB*. Termos como CSS, HTML, JAVASCRIPT, Apache, PHP, entre muitos outros, farão parte dos seus estudos nesta disciplina e consequentemente, de seu dia a dia como programador.

O estudo desta disciplina é muito importante para ampliar seus conhecimentos acerca das linguagens de programação *web* e compreender como funcionam.

Com esta disciplina você conhecerá e aprenderá a utilizar a *web* e o conjunto de tecnologias que as forma, além de aprender e aplicar a linguagem HTML.

Este caderno, como próprio nome diz, é uma introdução a linguagens de programação *web*, portanto, estudaremos o básico das linguagens para começarmos a compreender. Além disso, não atentaremos a linguagens de programação *web* avançadas, pois essas serão estudas nas próximas disciplinas.

Aproveitamos esse momento para destacar que os exercícios **NÃO SÃO OPCIONAIS**. O objetivo de cada exercício deste caderno é a fixação de determinado conceito através da prática. É aí que reside a importância da realização de todos. Sugerimos fortemente que em caso de dúvida em algum exercício você entre em contato com seu tutor externo ou com a tutoria da Uniasselvi e que não passe para o exercício seguinte enquanto o atual não estiver completamente entendido e aprendido.

Você precisará de muita determinação e força de vontade, pois a programação vai além de você ler o Caderno de Estudos, porém aqui, forneceremos um início sólido e consistente. Desta forma, passe por esse período de estudos com muita dedicação, pois você que hoje é acadêmico(a), amanhã será um profissional de Tecnologia da Informação com o compromisso de construir.

Este material foi desenvolvido com objetivo de disponibilizar o pontapé inicial às linguagens de programação *web*. A utilização do HTML é um processo de aprendizagem para alavancar as demais linguagens.

Desejo a você sucesso nessa nova busca de informações e principalmente no que tange em ampliar seus conhecimentos!

Bons estudos

Greisse Moser Badalotti

Você já me conhece das outras disciplinas? Não? É calouro? Enfim, tanto para você que está chegando agora à UNIASSELVI quanto para você que já é veterano, há novidades em nosso material.

Na Educação a Distância, o livro impresso, entregue a todos os acadêmicos desde 2005, é o material base da disciplina. A partir de 2017, nossos livros estão de visual novo, com um formato mais prático, que cabe na bolsa e facilita a leitura.

O conteúdo continua na íntegra, mas a estrutura interna foi aperfeiçoada com nova diagramação no texto, aproveitando ao máximo o espaço da página, o que também contribui para diminuir a extração de árvores para produção de folhas de papel, por exemplo.

Assim, a UNIASSELVI, preocupando-se com o impacto de nossas ações sobre o ambiente, apresenta também este livro no formato digital. Assim, você, acadêmico, tem a possibilidade de estudá-lo com versatilidade nas telas do celular, *tablet* ou computador.

Eu mesmo, UNI, ganhei um novo *layout*, você me verá frequentemente e surgirei para apresentar dicas de vídeos e outras fontes de conhecimento que complementam o assunto em questão.

Todos esses ajustes foram pensados a partir de relatos que recebemos nas pesquisas institucionais sobre os materiais impressos, para que você, nossa maior prioridade, possa continuar seus estudos com um material de qualidade.

Aproveito o momento para convidá-lo para um bate-papo sobre o Exame Nacional de Desempenho de Estudantes – ENADE.

Bons estudos!

BATE PAPO SOBRE O ENADE!

Olá, acadêmico!

Você já ouviu falar sobre o **ENADE**?

Se ainda não ouviu falar nada sobre o ENADE, agora você receberá algumas informações sobre o tema.

Ouviu falar? Ótimo, este informativo reforçará o que você já sabe e poderá lhe trazer novidades.

Vamos lá!

Qual é o significado da expressão ENADE?

EXAME NACIONAL DE DESEMPENHO DOS ESTUDANTES

Em algum momento de sua vida acadêmica você precisará fazer a prova ENADE.

Que prova é essa?

É **obrigatória**, organizada pelo INEP – Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira.

Quem determina que esta prova é obrigatória... O **MEC – Ministério da Educação**.

O objetivo do MEC com esta prova é o de avaliar seu desempenho acadêmico assim como a qualidade do seu curso.

Fique atento! Quem não participa da prova fica impedido de se formar e não pode retirar o diploma de conclusão do curso até regularizar sua situação junto ao MEC.

Não se preocupe porque a partir de hoje nós estaremos auxiliando você nesta caminhada.

Você receberá outros informativos como este, complementando as orientações e esclarecendo suas dúvidas.

Você tem uma trilha de aprendizagem do ENADE, receberá e-mails, SMS, seu tutor e os profissionais do polo também estarão orientados.

Participará de webconferências entre outras tantas atividades para que esteja preparado para #mandar bem na prova ENADE.

Nós aqui no NEAD e também a equipe no polo estamos com você para vencermos este desafio.

Conte sempre com a gente, para juntos mandarmos bem no ENADE!

UNIASSELVI

SUMÁRIO

UNIDADE 1

INTRODUÇÃO AO DESENVOLVIMENTO DE SISTEMAS *WEB*

OBJETIVOS DE APRENDIZAGEM

Ao final desta unidade, você será capaz de:

- conhecer a história da internet;
- conhecer os diferentes serviços oferecidos;
- identificar os principais browsers para *web*;
- entender os conceitos básicos sobre Padrões *Web* no uso de *sites* em computadores, celulares e demais mídias de interação com a internet;
- entender os conceitos de usabilidade e acessibilidade para internet;
- identificar as linguagens para desenvolvimento de sistemas *web*.

PLANO DE ESTUDOS

Esta unidade de ensino está dividida em 3 tópicos, sendo que no final de cada um deles, você encontrará atividades que contribuirão para a apropriação dos conteúdos.

TÓPICO 1 – INTRODUÇÃO À INTERNET

TÓPICO 2 – INTRODUÇÃO AO DESENVOLVIMENTO PARA *WEB*

TÓPICO 3 – FERRAMENTAS DE DESENVOLVIMENTO

TÓPICO 1

INTRODUÇÃO À INTERNET

1 INTRODUÇÃO

Como você vê a internet? O que você lembra quando você ouve a frase "um *site* da internet"? O *site* que você está pensando possibilita interação ou traz somente informação?

A internet revolucionou o mundo da informática e comunicações como nada antes. A invenção do telégrafo, telefone, rádio e computador prepararam o terreno para esta integração. A internet é ao mesmo tempo uma capacidade mundial de radiodifusão, um mecanismo de disseminação de informação, e um meio para colaboração e interação entre indivíduos e seus computadores sem levar em conta a localização geográfica. A internet representa um dos mais bem-sucedidos exemplos dos benefícios de um investimento sustentado.

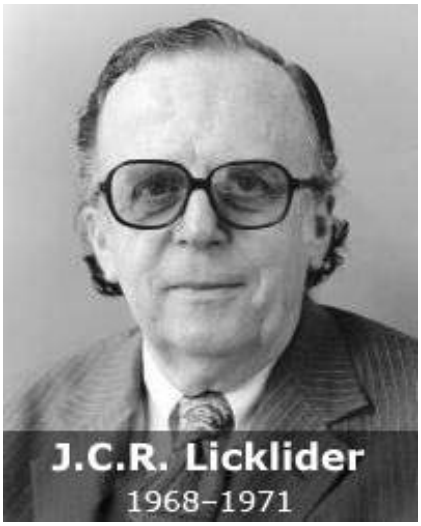
A internet hoje é uma infraestrutura de informação generalizada, o protótipo inicial do que é frequentemente chamado a Infraestrutura Nacional de Informação. Sua história é complexa e envolve muitos aspectos tecnológicos, organizacionais e comunidade. E sua influência atinge não só os campos técnicos das comunicações de computador, mas em toda a sociedade à medida que avançamos em direção à crescente utilização de ferramentas *on-line* para realizar o comércio eletrônico, a aquisição de informações, e as operações da comunidade.

Antes de começarmos a estudar o desenvolvimento de aplicações *web* devemos primeiro compreender um pouco mais sobre a internet, como ela surgiu, como ela funciona para então começarmos a compreender suas aplicabilidades.

2 ORIGENS DA INTERNET

A primeira descrição de registros de interações sociais que poderiam ser realizadas através de redes foi uma série de memorandos escritos por Joseph Carl Robnett Licklider (Figura 1), do MIT (*Massachusetts Institute of Technology*), em agosto de 1962, discutindo o conceito "Rede Galacia". Ele imaginou um conjunto de computadores interconectados globalmente, através do qual todos poderiam acessar rapidamente dados e programas de qualquer local. Em essência, o conceito foi muito parecido com a internet de hoje. Licklider foi o primeiro gerente do programa de pesquisa de computador do DARPA (*Defense Advanced Research Projects Agency*), a partir de outubro de 1962.

FIGURA 1 - JRC



FONTE: Disponível em: <http://mac50.csail.mit.edu/node/6>. Acesso em: 30 set. 2014.

Leonard Kleinrock, do MIT, publicou o primeiro trabalho sobre a teoria de comutação de pacotes em julho de 1961 e o primeiro livro sobre o assunto em 1964, Kleinrock convenceu Roberts da possibilidade teórica das comunicações usando pacotes ao invés de circuitos, o que foi um grande passo ao longo do caminho para os computadores. O outro grande passo foi fazer os computadores se conversarem. Para explorar isto, em 1965, trabalhando com Thomas Merrill, Roberts conectou o computador TX-2 a Q-32 na Califórnia com uma linha telefônica discada de baixa velocidade, criando a primeira (ainda que pequena) rede de computadores em toda a área já construída. O resultado deste experimento foi a comprovação de que os computadores compartilhados poderiam trabalhar bem juntos, rodando programas e recuperando dados quando necessário em máquinas remotas, mas que o circuito comutado com o sistema telefônico era totalmente inadequado para o trabalho.

No final de 1966, Roberts foi para a DARPA para desenvolver o conceito de rede de computadores e rapidamente montar o seu plano para o "ARPANET". Houve também um trabalho sobre um conceito de rede de pacotes do Reino Unido por Donald Davies e Roger Scantlebury da NPL (*National Physical Laboratory*). O grupo tinha escrito um trabalho sobre redes de comutação de pacotes para voz segura quando serviam militarmente em 1964. Aconteceu que o trabalho no MIT (1961-1967), do grupo (1962-1965), e no NPL (1964-1967) teve desenrolar em paralelo sem que nenhum dos pesquisadores soubesse dos outros trabalhos. A palavra "pacote" foi adotada do trabalho desenvolvido no NPL e a velocidade de linha proposta para ser usada no projeto da ARPANET foi atualizada de 2,4 Kb para 50 Kb.

Em agosto de 1968, depois de Roberts e o grupo do DARPA terem refinado a estrutura geral e as especificações para a ARPANET, uma seleção foi dada pela DARPA para o desenvolvimento de um dos componentes-chave, os comutadores de pacotes chamados de *Interface Message Processors* (IMPs) ou interface de processamento de mensagens. A licitação foi vencida em dezembro de 1968 por um grupo liderado por Frank Heart Bolt Beranek e Newman (BBN). Como a equipe BBN trabalhou na interface de processamento de mensagens com Bob Kahn, assumindo um papel vital do desenho arquitetônico da ARPANET, a topologia e economia da rede foi desenvolvida e otimizada por Roberts em conjunto com Howard Frank e seu grupo da *Network Analysis Corporation* (NAC), e o sistema de mensuração da rede foi preparado pela equipe de Kleinrock na UCLA (Universidade da Califórnia em Los Angeles). Devido ao desenvolvimento precoce de Kleinrock da teoria de comutação de pacotes e seu foco em análise, desenho e mensuração, seu Centro de Mensuração de Rede da UCLA foi escolhido para ser o primeiro nó da ARPANET.

Tudo isso aconteceu em setembro de 1969, quando BBN instalou a primeira interface de processamento de mensagens na UCLA e o primeiro servidor de computador foi conectado. O projeto de Doug Engelbart em "Aumento do Intelecto Humano", que incluía um sistema de hipertexto no Stanford Research Institute (SRI), desde um segundo nó. O SRI apoiou o Centro de Informações da Rede, liderada por Elizabeth Feinler e incluindo funções como a manutenção de tabelas de nome *host* para o mapeamento de endereço. Um mês depois, quando o SRI foi conectado à ARPANET, a primeira mensagem *host-to-host* foi enviada do laboratório de Kleinrock para o SRI. Mais dois nódulos foram adicionados na UC Santa Barbara e a Universidade de Utah. Estes dois nós incorporavam projetos de aplicações visuais, com Glen Culler e Burton Fried na UCSB investigando métodos de visualização de funções matemáticas usando o armazenamento monitores para lidar com o problema de atualização sobre a rede e Robert Taylor e Ivan Sutherland em Utah investigando métodos de representações sobre a rede. Assim, até o final de 1969, quatro computadores foram conectados juntos na ARPANET, e a internet foi brotando do chão. Mesmo nesta primeira fase, deve notar-se que a pesquisa de rede incorporada tanto no trabalho de rede subjacente e trabalho sobre como utilizar a rede continua até hoje.

Computadores foram rapidamente adicionados à ARPANET nos anos seguintes, e prosseguiu o preenchimento de um protocolo funcionalmente completo *host-to-host* e outros *softwares* de rede. Em dezembro de 1970, a Rede Grupo de Trabalho (NWG), concluiu o primeiro protocolo servidor a servidor da ARPANET, chamado *Network Control Protocol* (NCP). Com os *sites* da ARPANET deu-se por concluída a implementação NCP durante o período de 1971-1972, os usuários da rede finalmente puderam começar a desenvolver aplicações.

Em outubro de 1972, Kahn organizou uma grande e bem-sucedida demonstração da ARPANET na Conferência Internacional *Computer Communication* (ICCC). Esta foi a primeira demonstração pública da nova tecnologia de rede para o público. Foi também em 1972 que a aplicação do correio eletrônico foi introduzida. Em março Ray Tomlinson, da BBN, escreveu a mensagem básica de *e-mail* de enviar e ler motivado pela necessidade dos desenvolvedores da ARPANET de ter um fácil mecanismo de coordenação. Em julho, Roberts expandiu a utilidade escrevendo o primeiro programa utilitário de *e-mail* para listar, ler seletivamente, arquivar, encaminhar e responder a mensagens. De lá o correio eletrônico se tornou a maior aplicação de rede por mais de uma década. Este foi um prenúncio do tipo de atividade que vemos na *World Wide Web* hoje, ou seja, o enorme crescimento de todos os tipos de tráfego.

A original internet cresceu da ARPANET. A internet foi baseada na ideia de que haveria múltiplas redes independentes de concepção, começando com a ARPANET como rede de comutação de pacotes pioneira, mas logo a incluiu nas redes de satélites, redes de rádio, pacote de pacotes, baseados em terra e outras redes. A internet como a conhecemos hoje incorpora uma ideia técnica subjacente, ou seja, de arquitetura aberta a negócios. Nesta abordagem, a escolha de qualquer tecnologia de rede individual não é ditada por uma arquitetura de rede particular, mas pode ser selecionado livremente por um fornecedor e feito para interagir com as outras. Até aquele momento havia apenas um método geral para redes federadas. Este foi o método de comutação de circuitos tradicionais em que as redes se interligaram com o nível de circuito, passando os *bits* individuais numa base síncrona ao longo de uma porção de um circuito de ponta a ponta entre um par de localizações de extremidade. Lembre-se de que Kleinrock tinha mostrado em 1961 que a troca de pacotes era um método de comutação mais eficiente. Junto com comutação de pacotes, acordos de interconexão entre as redes para fins especiais foram surgindo outras possibilidades.

Em uma rede de arquitetura aberta, as redes individuais podem ser concebidas separadamente e desenvolvidas. Cada uma pode ter sua própria interface na qual pode oferecer aos usuários e/ou em outros fornecedores incluindo outros provedores de internet. Cada rede pode ser projetada de acordo com as necessidades específicas de ambiente e usuários dessa rede. Geralmente não há restrições sobre os tipos de rede que podem ser incluídos, ou em seu âmbito geográfico, apesar de algumas considerações pragmáticas vai ditar o que faz sentido para oferecer.

A ideia de redes de arquitetura aberta foi introduzida pela primeira vez por Kahn, pouco depois de ter chegado a DARPA em 1972. Este trabalho foi originalmente parte do programa de pacotes de rádio, mas depois se tornou um programa separado em seu próprio direito. Na época, o programa foi chamado de "*Internetting*". A chave para fazer funcionar o sistema de rádio de pacotes era um protocolo fim a fim confiável, que poderia manter uma comunicação eficaz em face do bloqueio e outras interferências de rádio, ou suportar apagão intermitente, tais como causado por estar em um túnel ou bloqueado pelo terreno local. Kahn primeiro contemplou o desenvolvimento de um protocolo local apenas para a rede de pacotes de rádio, uma vez que evitaria ter que lidar com o grande número de diferentes sistemas operacionais, e continuar a usar NCP.

No entanto, a NCP não tinha a capacidade de lidar com redes (e máquinas), portanto, algumas mudanças da NCP também seriam necessárias. A suposição era de que a ARPANET não era mutável a este respeito. A NCP dependia da ARPANET para prover confiabilidade. Se todos os pacotes foram perdidos, o protocolo veria uma parada. Neste modelo NCP não tinha controle de erro, uma vez que a ARPANET seria a única rede em existência e ela seria tão confiável que nenhum controle de erro seria necessário por parte dos anfitriões. Assim, Kahn decidiu desenvolver uma nova versão do protocolo que pudesse satisfazer as necessidades de um ambiente de rede de arquitetura aberta. Este protocolo acabaria por ser chamado de *Transmision Control Protocol* / Internet Protocol (TCP/IP). Enquanto NCP tendiam a agir como um *driver* de dispositivo, o novo protocolo seria mais como um protocolo de comunicação.

Quatro regras foram críticas para o pensamento inicial de Kahn:

- Cada rede distinta teria que ficar em sua própria rede e não havia mudança interna que poderia se conectar a qualquer tipo de rede à internet.
- Comunicações seriam a base. Se um pacote não enviá-lo para o destino final em breve seria retransmitido da fonte.
- As caixas pretas seriam usadas para ligar as redes. Estes mais tarde seriam chamados de *gateways* e roteadores. Não haveria nenhuma informação retida pelos *gateways* sobre os fluxos individuais de pacotes que passam através deles, mantendo-os, assim, simples e evitando adaptação e recuperação complicada de vários modos de falha.
- Não haveria controle global no nível de operações.

Outras questões importantes que precisavam ser abordadas foram:

- Algoritmos para evitar perda de pacotes permanentemente de comunicações e permitir que sejam retransmitidos com sucesso a partir da fonte.
- Fornecer para *host-to-host* "pipeline" para que vários pacotes pudessem ser encaminhados da origem ao destino, se as redes intermediárias permitissem.
- Funções de *gateway* para permitir que ele transmita pacotes de forma adequada. Isto incluiu interpretar cabeçalhos IP para roteamento, interfaces de manipulação, quebrando pacotes em pedaços menores, se necessário.

- A necessidade de somas de verificação ponta a ponta, a remontagem de pacotes a partir de fragmentos e detecção de duplicatas, se houver.
- A necessidade de endereçamento global.
- Técnicas de controle de fluxo de *host-to-host*.
- Interface com os vários sistemas operativos.
- Havia também outras preocupações, tais como a eficiência da execução, o desempenho de redes, mas estes eram considerações secundárias.

Kahn começou a trabalhar em uma comunicação orientada definida de um conjunto de princípios do sistema operacional enquanto na BBN orientados e documentados alguns dos seus primeiros pensamentos num memorando interno BBN intitulado "Princípios de Comunicações para Sistemas Operacionais". Nesse ponto, ele percebeu que seria necessário aprender os detalhes de implementação de cada sistema operacional para ter a chance de incorporar quaisquer novos protocolos de uma forma eficiente. Assim, em 1973, depois de iniciar o esforço *internetting*, pediu Vint Cerf (então na Universidade de Stanford) para trabalhar com ele no projeto detalhado do protocolo. Cerf tinha se envolvido intimamente no projeto original NCP e desenvolvimento e já tinha o conhecimento sobre a interface com os sistemas operacionais existentes. Então, armado com abordagem de arquitetura de Kahn para o lado da comunicação e com experiência PCN Cerf, que se uniram para soletrar os detalhes do que se tornou TCP/IP.

O dar e receber foram altamente produtivos na primeira versão escrita da abordagem resultante na qual foi distribuída numa reunião especial do Grupo Rede Internacional de Trabalho (INWG), que tinha sido criada em uma conferência na Universidade de Sussex, em setembro de 1973, Cerf tinha sido convidado para presidir este grupo e usou a ocasião para realizar uma reunião de membros INWG que foram fortemente representados na Conferência de Sussex.

Algumas abordagens básicas emergiram dessa colaboração entre Kahn e Cerf:

- A comunicação entre dois processos logicamente consistem de uma longa corrente de *bytes* (que eles chamavam octetos). A posição de qualquer octeto no fluxo seria usada para identificá-lo.
- O controle de fluxo seria feito usando janelas e reconhecimentos (ACKs). O destino poderia selecionar quando reconhece e cada retornado seria cumulativo para todos os pacotes recebidos a esse ponto.
- Foi deixada em aberto sobre como, exatamente a origem e o destino seriam feitos com os parâmetros das janelas para ser usado. Padrões foram usados inicialmente.
- Embora Ethernet estivesse em desenvolvimento na Xerox PARC naquele tempo, a proliferação de LANs não foi pensada no momento, muito menos PCs e estações de trabalho. O modelo original foi redes em nível nacional como a ARPANET, dos quais se esperava que apenas um número relativamente pequeno existisse. Assim, um endereço IP de 32 *bits* foi usado, dos quais os primeiros 8 *bits* significou a rede e os restantes 24 *bits* designavam o *host* naquela rede. Esta hipótese, que 256 redes seriam suficientes para o futuro

previsível, era claramente visto na necessidade de reconsideração quando LANs começaram a aparecer no final de 1970.

O Cerf e Kahn descreveram um protocolo chamado TCP, que forneceu todos os serviços de transporte e expedição na internet. O esforço inicial para implementar o TCP resultou numa versão que somente permitiu circuitos virtuais. Esse modelo funcionou bem para transferência de arquivos e aplicações de *login* remoto, mas alguns dos primeiros trabalhos sobre aplicações de rede avançadas, nomeadamente pacotes de voz na década de 1970, deixaram claro que, em alguns casos, a perda de pacotes não deve ser corrigida pela TCP, mas deve ser deixado para a aplicação de lidar. Isto levou a uma reorganização do TCP original em dois protocolos, o IP simples, que prevê apenas o endereçamento e o encaminhamento de pacotes individuais e o TCP em separado, que estava preocupado com as características de serviços, tais como controle de fluxo e recuperação de pacotes perdidos. Para aquelas aplicações que não queriam os serviços de TCP, uma alternativa chamada *Datagram Protocol* (UDP) foi adicionada a fim de proporcionar o acesso direto ao serviço básico de IP.

Uma grande motivação inicial para ARPANET e a internet foi o compartilhamento de recursos, permitindo que os usuários nas redes de pacote de rádio acessassem os sistemas de compartilhamento de tempo ligados à ARPANET. Ligar os dois juntos era muito mais econômico que duplicar esses computadores. No entanto, enquanto a transferência de arquivos e *login* remoto (Telnet) foram aplicações muito importantes, o correio eletrônico teve provavelmente o impacto mais significativo das inovações daquela época. O *e-mail* fornecia um novo modelo de como as pessoas podiam se comunicar umas com as outras, e mudou a natureza da colaboração, primeiro na construção da própria internet e, posteriormente, para grande parte da sociedade.

Havia outras aplicações propostas nos primeiros dias da internet, incluindo comunicação de voz baseada em pacotes (o precursor da telefonia via internet), vários modelos de compartilhamento de arquivos e disco, e programas que mostraram o conceito de agentes (e, naturalmente, vírus). Um conceito-chave da internet é que ele não foi projetado para apenas uma aplicação, mas como uma infraestrutura geral em que novas aplicações podem ser concebidas, como ilustrado mais tarde com o surgimento da *World Wide Web*. É a natureza de propósito geral do serviço prestado pelo TCP e IP que torna isso possível.

DARPA deixou três contratos para Stanford (Cerf), BBN (Ray Tomlinson) e UCL (Peter Kirstein) para implementar o TCP/IP (que foi simplesmente chamado TCP no Cerf papel / Kahn, mas que continha ambos os componentes). A equipe de Stanford, liderada por Cerf, produziu a especificação detalhada e dentro de cerca de um ano houve três implementações independentes de TCP que poderiam interoperar.

Este foi o início da experimentação e do desenvolvimento a longo prazo para evoluir e amadurecer os conceitos e a tecnologia da internet. Começando com as três primeiras redes (ARPANET, Packet Radio e Packet Satellite) e

suas comunidades iniciais de pesquisa, o ambiente experimental cresceu para incorporar essencialmente qualquer forma de rede em uma comunidade de pesquisa e desenvolvimento. Com cada expansão vieram novos desafios.

As primeiras implementações de TCP foram feitas por sistemas de compartilhamento de grande duração, como Tenex e TOPS 20. Quando os computadores de mesa apareceram pela primeira vez, foi considerado por alguns que o TCP foi grande e complexo demais para ser executado em um computador pessoal. David Clark e seu grupo de pesquisa do MIT propuseram-se a mostrar que a implementação compacta e simples de TCP era possível. Eles produziram uma implementação, primeiro para o Xerox (a estação de trabalho pessoal desenvolvido no início do Xerox PARC) e depois para o IBM PC. Essa implementação foi adaptada para o conjunto de aplicativos e objetivos do computador pessoal, e mostrou que as estações de trabalho, bem como grandes sistemas de compartilhamento poderiam ser uma parte da internet. Em 1976, Kleinrock publicou o primeiro livro sobre a ARPANET. Ele incluía uma ênfase na complexidade dos protocolos. Este livro foi influente na difusão do conhecimento de redes de comutação de pacotes para uma comunidade muito grande.

Desenvolvimento generalizado de LANs, PCs e estações de trabalho na década de 1980 permitiu que a internet começasse a florescer. A tecnologia Ethernet, desenvolvida por Bob Metcalfe na Xerox PARC em 1973, era agora provavelmente a tecnologia de rede dominante na internet e PCs e estações de trabalho. Esta mudança de ter algumas redes com um número modesto de hospedeiros por um tempo compartilhado (o modelo original ARPANET) para ter muitas redes resultou em uma série de novos conceitos e mudanças na tecnologia subjacente. Em primeiro lugar, isso resultou na definição de três classes de rede (A, B e C) para acomodar o conjunto de redes. Classe A representadas pelas redes em escala nacional grande porte (pequeno número de redes com grande número de *hosts*); Classe B representado pelas redes de escala regional; e Classe C passou a representar redes locais.

A grande mudança ocorreu como resultado do aumento de escala dos seus problemas de gestão associados à internet. Para tornar mais fácil para as pessoas usarem a rede, os anfitriões foram atribuindo nomes, de modo que não era necessário lembrar endereços numéricos. Originalmente, havia um número bastante limitado de hospedeiros, então era possível manter uma tabela única de todos os anfitriões e seus nomes e endereços associados. A mudança para ter um grande número de redes gerenciadas de forma independente (por exemplo, *LANs*) significou ter uma única tabela de anfitriões que não era mais viável, e foi inventado o *Domain Name System* (DNS) por Paul Mockapetris da USC / ISI. O DNS permitiu um mecanismo distribuído escalável para a resolução de nomes de *host* hierárquicos (por exemplo: <www.acm.org>) em um endereço de internet.

O aumento do tamanho da internet também desafiou as capacidades dos roteadores. Originalmente, havia um único algoritmo distribuído para roteamento que foi implementado uniformemente por todos os roteadores na internet. Como o número de redes na internet explodiu, este projeto inicial não poderia se expandir,

se necessário, por isso foi substituído por um modelo hierárquico de roteamento, com um Interior *Gateway Protocol* (IGP) usado dentro de cada região da internet e um Exterior *Gateway Protocol* (EGP) usado para ligar as regiões. Este projeto permitiu diferentes regiões para usar um IGP diferente, de modo que os diferentes requisitos para custo, reconfiguração rápida, robustez e escalas pudessem ser acomodados. Não apenas o algoritmo de encaminhamento, mas os tamanhos das tabelas de endereços salientaram a capacidade dos roteadores. Novas abordagens para agregação de endereço, em particular *Classless Inter-Domain Routing* (CIDR), foram recentemente introduzidas para controlar o tamanho das tabelas de roteamento.

À medida que a internet evoluiu, um dos principais desafios foi como propagar as alterações para o *software*, particularmente o *software host*. DARPA foi apoiada pela UC Berkeley para investigar modificações para o sistema operacional Unix, incluindo incorporação de TCP/IP. Embora Berkeley mais tarde reescreveu o código para atender de forma mais eficiente o sistema Unix e o *kerne*l, a incorporação de TCP/IP para as versões do sistema Unix provou ser um elemento crítico na dispersão dos protocolos para a comunidade de pesquisa. Grande parte da comunidade de pesquisa começou a usar Unix para o seu ambiente de computação do dia a dia. Olhando para trás, a estratégia de incorporar protocolos de internet em um sistema operacional com suporte para a comunidade de pesquisa foi um dos elementos-chave para a adoção generalizada do sucesso da internet.

Um dos desafios mais interessantes foi a transição do protocolo de acolhimento ARPANET de NCP para TCP/IP a partir de 1 de janeiro de 1983. Esta transição foi cuidadosamente planejada dentro da comunidade ao longo de vários anos antes que ela realmente ocorresse e foi surpreendentemente bem.

O TCP/IP foi adotado como um padrão de defesa três anos antes, em 1980, o que permitiu à defesa iniciar o compartilhamento da base de tecnologia de internet DARPA e levou diretamente à eventual divisão do militar e as comunidades não militares. Em 1983, a ARPANET estava sendo usada por um número significativo de defesa e organizações operacionais. A transição da ARPANET de NCP para TCP/IP permitindo que ele seja dividido em um MILNET para apoiar as necessidades operacionais e a ARPANET para as necessidades de investigação e de apoio.

A tecnologia e os conceitos fundamentais utilizados pela internet surgiram dos projetos conduzidos ao longo dos anos com a ARPAnet (ARPA: Advanced Research Projects Agency), um projeto experimental do Departamento de Defesa Norte-Americano, que interligava pesquisadores com centros de computação remotos.

Assim, em 1985, a internet já estava bem estabelecida como uma tecnologia de suporte para uma vasta comunidade de pesquisadores e desenvolvedores, e estava começando a ser usada por outras comunidades para comunicações diárias do computador. O correio eletrônico estava sendo amplamente utilizado em várias comunidades, muitas vezes, com sistemas diferentes, mas a interligação entre os diferentes sistemas de correio foi demonstrando a utilidade das comunicações eletrônicas, com base alargada entre as pessoas.

Ao mesmo tempo que a tecnologia da internet estava sendo experimentalmente validada e amplamente utilizada entre um subconjunto de pesquisadores de ciência da computação, a utilidade das redes de computadores – especialmente correio eletrônico – demonstrada pela DARPA e pelo Departamento de Defesa dos contratantes sobre a ARPANET não foi perdida em outras comunidades e disciplinas, de modo que, em meados da década de 1970 as redes de computadores começaram a surgir onde quer que o financiamento possa ser encontrada. O Departamento de Energia dos EUA (DoE) estabeleceu por seus pesquisadores em energia de fusão magnética, quando então físicos de altas energias do DOE responderam construindo HEPnet. A NASA seguiu com SPAN, e Rick Adrion, David Farber, e Larry Landweber estabelecido CSNET para a comunidade de Ciência da Computação (acadêmica e industrial) com um financiamento inicial da *US National Science Foundation* (NSF) da AT & T, com base em protocolos integrados de comunicação UUCP, e, em 1981, Ira Fuchs e Greydon Freeman inventaram o BITNET, que ligava computadores *mainframe* acadêmicos em um "*e-mail* como imagens de cartão" para o UNIX.

Com exceção da BITNET e USENET, estas primeiras redes (incluindo ARPANET) foram construídas de propósito, ou seja, se destinavam, em grande parte para as restritas comunidades acadêmicas fechadas; houve, portanto, um pouco de pressão para as redes individuais serem compatíveis e, de fato, em grande parte, elas não estavam. Além disso, tecnologias alternativas estavam sendo desenvolvidas no setor comercial, incluindo XNS da Xerox, DECNet e SNA da IBM, à JANET britânico (1984) e US NSFNET (1985) programas que anunciavam explicitamente sua intenção de servir a toda comunidade de ensino superior, independentemente da disciplina. De fato, uma condição para uma universidade dos Estados Unidos para receber o financiamento NSF para uma ligação à internet era de que "... a conexão devia ser disponibilizada a todos os usuários qualificados no campus."

Em 1985, Dennis Jennings veio da Irlanda para passar um ano na NSF liderando o programa da NSFNET. Ele trabalhou com a comunidade para ajudar a NSF a tomar uma decisão crítica – que o TCP/IP seria obrigatório para o programa da NSFNET. Quando Steve Wolff assumiu a NSFNET, em 1986, reconheceu a necessidade de uma ampla área de infraestrutura de rede para apoiar a comunidade acadêmica e de pesquisa em geral, juntamente com a necessidade de desenvolver uma estratégia para o estabelecimento de tais infraestruturas de base, em última instância independente do financiamento. Foram adotadas políticas e estratégias para atingir esse fim.

Além da seleção de TCP/IP para o NSFNET, agências federais implementaram várias outras decisões políticas que moldaram a internet de hoje.

- As agências federais compartilharam o custo da infraestrutura comum, tais como circuitos transoceânicos. Também apoiaram conjuntamente "pontos de interconexão gerenciados" para o tráfego entre agências; bolsas de internet federais construídas para este fim serviram como modelos para os pontos de acesso da rede e instalações que são características proeminentes da arquitetura da internet de hoje.
- Para coordenar essa partilha, um Conselho Federal foi formado. A FNC também colaborou com outras organizações internacionais, através da Comissão de Coordenação sobre *Intercontinental Research Networking*, CCIRN, para coordenar o apoio internet da comunidade de pesquisa em todo o mundo.
- Posteriormente, em um modo semelhante, a NSF encorajou suas redes regionais (inicialmente acadêmicas) da NSFNET a buscar clientes não acadêmicos comerciais, e expandir as suas instalações para servi-los, e explorar as economias de escala resultantes de menores custos de subscrição para todos.
- A NSFNET – NSF aplicou uma "Política de Utilização Aceitável" que proibiu o uso de *Backbone* para fins "não de Apoio à Pesquisa e Educação." O resultado previsível (e intencional) de incentivar o tráfego da rede comercial em nível local e regional, ao negar o acesso ao transporte em escala nacional, foi estimular o surgimento e/ou crescimento de redes competitivas, de longa distância, "privadas" tais como PSI, UUNET, ANS CO + RE, e (mais tarde) outros. Este processo de aumento para fins comerciais foi debatido a partir de 1988, em uma série de conferências NSF-iniciadas na *Kennedy School of Government* da Universidade de Harvard sobre "A comercialização e privatização da internet".
- Em 1988, uma comissão do Conselho Nacional de Pesquisa, presidido por Kleinrock e com Kahn e Clark como membros, produziu um relatório encomendado pela NSF intitulado "Rumo a uma Rede Nacional de Pesquisa". Este relatório marcou o início de redes de alta velocidade que lançaram as bases de rede para obter as informações.
- Em 1994, um relatório do Conselho Nacional de Pesquisa, novamente presidido por Kleinrock (e com Kahn e Clark como membros de novo), A" foi lançado. Este relatório, encomendado pelo NSF, foi o documento no qual um modelo para a evolução da autoestrada da informação foi articulada e teve um efeito prolongado na maneira de pensar sobre a sua evolução. Ele antecipou as questões críticas de direitos de propriedade intelectual, ética, preços, educação, arquitetura e regulamentação para a internet.

O *backbone* fez a transição de uma rede construída de roteadores fora da comunidade de pesquisa para equipamentos comerciais. Em seu tempo de vida 8 e 1/2 anos, a espinha dorsal tinha crescido de seis nós com 56 kbps *links* para 21 nós com múltiplos *links* de 45 Mbps. Ele tinha visto a internet crescer para mais de 50.000 redes em todos os sete continentes e espaço exterior, com cerca de 29.000 redes nos Estados Unidos (INTERNET SOCIETY, 2014).

A chave para o rápido crescimento da internet tem sido o acesso livre e aberto aos documentos básicos, especialmente às especificações dos protocolos.

Os primórdios da ARPANET e da internet na comunidade de pesquisa da universidade promoveu a tradição acadêmica de publicação aberta de ideias e resultados. No entanto, o ciclo normal da publicação acadêmica tradicional era muito formal e muito lento para a troca dinâmica de ideias essenciais para a criação de redes.

Conforme Internet Society (2014), em 1969, um passo importante foi dado por S. Crocker, em que estabelece o *Request for Comments* (ou RFC). Estes memorandos tinham a intenção de ser uma maneira rápida de distribuição informal compartilhando ideias com outros pesquisadores de rede. A princípio, os RFCs eram impressos em papel e distribuídos via correio tradicional. Como o Protocolo de Transferência de Arquivo (FTP) entrou em uso, os RFCs foram preparados como arquivos *on-line* e acessados via FTP. Agora, é claro, os RFCs são facilmente acessados através da *World Wide Web* em dezenas de *sites* ao redor do mundo.

FTP: File Transfer Protocol. Permite a cópia de arquivos e programas que estão na internet para o seu computador ou vice-versa. É a transferência de arquivos, localizados em computadores remotos.

O efeito dos RFCs foi criar um ciclo de *feedback* positivo, com ideias ou propostas apresentadas em um RFC provocando outro RFC com mais ideias, e assim por diante. Quando algum consenso (ou, pelo menos, um conjunto de ideias coerentes) havia reunido um documento de especificação estaria preparado. Tal especificação, então, seria usada como base para implementações por diversas equipes de investigação.

Com o tempo, os RFCs se tornaram mais focados em padrões de protocolo, embora ainda existam RFCs informativos que descrevem abordagens alternativas, ou fornecem informações básicas sobre protocolos e questões de engenharia. Os RFCs são agora vistos como os "documentos de registro" na comunidade de engenharia da internet.

O acesso aberto aos RFCs promove o crescimento da internet, pois permite que as especificações a serem usadas como exemplos em aulas da faculdade e por empreendedores desenvolvendo novos sistemas.

E-mail tem sido um fator significativo em todas as áreas da internet, e isso é certamente verdade no desenvolvimento de especificações de protocolos, padrões técnicos e de engenharia da internet. RFCs frequentemente apresentavam um conjunto de ideias desenvolvidas pelos pesquisadores em um local para o resto da comunidade. Depois que o *e-mail* entrou em uso, o padrão de autoria mudou. RFCs foram apresentados por coautores com vista comum independente de suas localizações.

O uso de listas de discussão de *e-mail* tem sido muito utilizado no desenvolvimento de especificações do protocolo e continua a ser uma ferramenta importante. O IETF (*Internet Engineering Task Force*) tem agora mais de 75 grupos de trabalho, cada um trabalhando em um aspecto diferente da engenharia da internet. Cada um destes grupos de trabalho tem uma lista de discussão para discutir um ou mais projetos de documentos em desenvolvimento.

À medida que a rápida expansão atual da internet é alimentada pela percepção de sua capacidade de promover o compartilhamento de informações, devemos entender que o primeiro papel da rede na partilha de informação estava compartilhando as informações sobre o seu próprio projeto e operação através dos documentos RFC. Este método único para a evolução de novas capacidades da rede continuará a ser fundamental para a evolução futura da internet.

A internet é tanto uma coleção de comunidades como uma coleção de tecnologias, e seu sucesso é atribuído principalmente às necessidades da comunidade, tanto básico como satisfatório, bem como utilizando a comunidade de uma forma eficaz para empurrar a infraestrutura para a frente. Este espírito de comunidade tem uma longa história que começa com o início de ARPANET. Os pesquisadores da ARPANET iniciaram o funcionamento como uma comunidade unida para realizar as manifestações iniciais da tecnologia de comutação de pacotes descritos anteriormente. Da mesma forma, o *Packet Satellite, Packet Radio* e diversos outros programas de pesquisa de ciência da computação da DARPA foram atividades de colaboração multicontratante que foram muito utilizadas em qualquer que seja a disponibilidade de mecanismos, começando com o correio eletrônico e acrescentando compartilhamento de arquivos, acesso remoto, e, eventualmente, a *World Wide Web*. Cada um desses programas formou um grupo de trabalho, começando com o Grupo de Trabalho ARPANET. Devido ao papel único que ARPANET jogado como uma infraestrutura de apoio aos diversos programas de pesquisa, como a internet começou a evoluir.

No final de 1970, reconhecendo que o crescimento da internet foi acompanhado por um crescimento do tamanho da comunidade de pesquisa interessada e, portanto, uma maior necessidade de mecanismos de coordenação, Vint Cerf, então gerente do Programa Internet na DARPA, formou vários órgãos de coordenação – um Conselho de Cooperação Internacional (ICB), presidido por Peter Kirstein da UCL, para coordenar as atividades com alguns países europeus que cooperaram na pesquisa *Packet Satellite*, um grupo de Pesquisa de Internet, que era um grupo inclusive, fornecendo um ambiente para a troca geral de informações, e uma Configuração *Control Board Internet* (ICCB), presidido por Clark.

Em 1983, quando Barry Leiner assumiu a gestão do programa de pesquisa de internet na DARPA, ele e Clark reconheceram que o crescimento contínuo da comunidade da internet exigia uma reestruturação dos mecanismos de coordenação. O ICCB foi dissolvido e em seu lugar uma estrutura de forças-tarefas foi formada, cada uma focada em uma determinada área da tecnologia (por exemplo, roteadores, protocolos etc). O Conselho de Atividades de Internet (IAB) foi formado a partir dos presidentes dos grupos de trabalho.

Como vimos acima, em 1985, houve um tremendo crescimento no lado mais prático e da engenharia da internet. Este crescimento resultou em uma explosão na participação nas reuniões do IETF, e Gross foi obrigado a criar infraestrutura para o IETF na forma de grupos de trabalho.

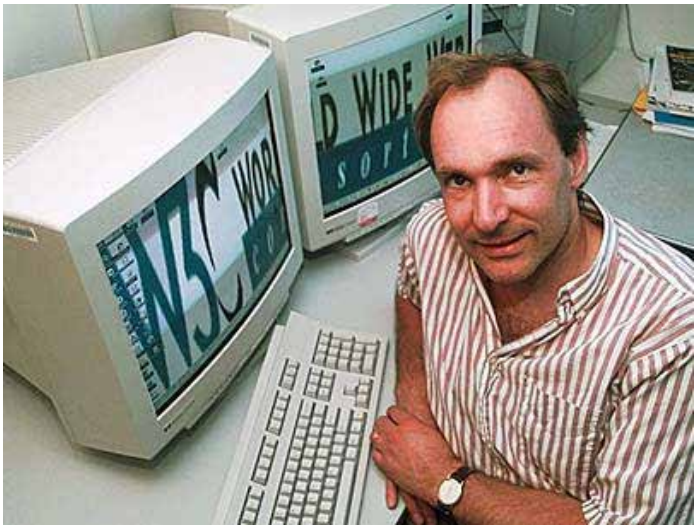
Este crescimento foi complementado por uma grande expansão na comunidade. Já não era a DARPA o único jogador importante no financiamento da internet. Além NSFNet, os EUA, atividades financiadas pelos governos internacionais, o interesse pelo setor comercial começava a crescer.

O crescimento no setor comercial trouxe uma preocupação crescente em relação ao processo das normas. A partir do início dos anos 1980 e continua até hoje, a internet cresceu além de suas raízes de investigação, essencialmente para incluir uma vasta comunidade de utilizadores e aumento da atividade comercial. Maior atenção foi dada para tornar o processo transparente e justo. Isso, combinado com a reconhecida necessidade de apoio da comunidade da internet que levou à formação da Sociedade da Internet, em 1991, sob os auspícios da Corporação de Kahn para o *National Research Initiatives* (CNRI) e a liderança de Cerf, em seguida, com CNRI.

Em 1992, mais uma reorganização ocorreu. Em 1992, o Conselho de Atividades da Internet foi reorganizado e rebatizado de *Internet Architecture Board* que opera sob os auspícios da Sociedade da Internet.

O recente desenvolvimento e implantação generalizada da *World Wide Web* trouxe uma nova comunidade. Muitas pessoas que trabalham na WWW não pensaram em si mesmas, principalmente como pesquisadores e desenvolvedores de rede. A nova organização da coordenação foi formada, a *World Wide Web Consortium* (W3C). Inicialmente levou do Laboratório do MIT de Ciências da Computação por Tim Berners-Lee (Figura 2), o inventor da WWW, e Al Vezza, W3C assumiu a responsabilidade pela evolução dos diversos protocolos e padrões associados com a *web*.

FIGURA 2 - Tim Berners Lee



FONTE: Disponível em: <http://www.aec.com.br/noticias/Perfil/Pages/Pessoas--que-fazem-a-diferen%C3%A7a-Tim-Berners-Lee.aspx>. Acesso em: 20 set. 2014.

Assim, através de mais de duas décadas de atividade na internet, temos visto uma evolução constante de estruturas organizacionais destinadas a apoiar e facilitar uma comunidade cada vez maior do trabalho conjunto sobre as questões da internet.

A comercialização da internet envolveu não apenas o desenvolvimento de serviços de rede privada competitivos, mas também o desenvolvimento de produtos comerciais que aplicam a tecnologia da internet. No início de 1980, dezenas de vendedores foram incorporando o TCP/IP em seus produtos porque viram compradores para a abordagem de redes. Infelizmente, eles não tinham tanta informação real sobre como a tecnologia deveria funcionar e como os clientes planejavam usar essa abordagem para os negócios.

Em 1985, reconhecendo a falta de disponibilidade de informações e formação adequada, Dan Lynch em cooperação com o IAB realizou um *workshop* de três dias para todos os fornecedores para aprender sobre como o TCP/IP funcionava e o que ele ainda não poderia fazer. Os palestrantes vieram principalmente da comunidade de pesquisa da DARPA, que tinham desenvolvido esses protocolos e os usaram no trabalho do dia a dia. Cerca de 250 funcionários do fornecedor vieram para ouvir 50 inventores e pesquisadores. Os resultados foram surpresas em ambos os lados: os vendedores ficaram surpresos ao descobrirem que os inventores eram tão abertos sobre a maneira como as coisas aconteceram (e que ainda não funcionaram) e os inventores tinham o prazer de ouvir novos problemas que não tinham considerado, mas foram sendo descobertos pelos fornecedores no campo.

Após dois anos de conferências, tutoriais, reuniões e oficinas de *design*, um evento especial foi organizado, que convidou os fornecedores para se unirem por três dias para mostrar o quão bem todos eles trabalharam juntos. Em setembro de 1988, a primeira feira Interop nasceu, 50 empresas participaram, 5.000 engenheiros de organizações potenciais e clientes vieram para ver se tudo havia ocorrido como foi prometido. A feira Interop tem crescido imensamente, desde então e hoje é realizada em sete locais ao redor do mundo a cada ano, para um público de mais de 250.000 pessoas que vêm para saber quais produtos trabalhar de uma forma perfeita, aprender sobre os mais recentes produtos, e discutir a tecnologia mais recente. A Interop de 2014 ocorreu em Las Vegas, conforme Figura 3.

FIGURA 3 - LOGO INTEROP 2014

FONTE: Disponível em: <http://www.sparkpresentations.com/wp-contentuploads/2013/12/Interop-Las-Vegas-logo.jpg>. Acesso em 20 set. 2014.

Em paralelo com os esforços de comercialização que foram salientados pelas atividades de interoperabilidade, os vendedores começaram a assistir às reuniões que foram realizadas de três a quatro vezes por ano para discutir novas ideias para extensões do conjunto de protocolos TCP/IP. Começando com algumas centenas de participantes na maior parte do meio acadêmico e pagos pelo governo. Este grupo selecionado evolui a suíte TCP/IP de forma mutuamente cooperativa. A razão é tão útil que ele é composto por todas as partes interessadas: pesquisadores, usuários finais e fornecedores.

Nos últimos anos, temos visto uma nova fase de comercialização. Originalmente, os esforços comerciais são compostos principalmente por fornecedores que fornecem os produtos básicos de rede e provedores de serviços que oferecem a conectividade e serviços básicos da internet. A internet tornou-se um serviço essencial, e grande parte da mais recente atenção foi sobre a utilização desta infraestrutura global de informação para suporte de outros serviços comerciais. Isso tem sido tremendamente acelerado pela adoção rápida

e generalizada dos *browsers* e da tecnologia *World Wide Web*, permitindo aos usuários acesso fácil a informações ligadas durante todo o globo. Os produtos estão disponíveis para facilitar a provisão de que a informação e muitos dos últimos desenvolvimentos em tecnologia têm sido direcionados para a prestação de serviços de informação cada vez mais sofisticados em cima das comunicações básicas de dados da internet.

Em 24 de outubro de 1995 foi aprovada por unanimidade uma resolução definindo o termo internet. Esta definição foi desenvolvida em consulta com os membros das comunidades da internet e os direitos de propriedade intelectual. RESOLUÇÃO: O Conselho Federal Redes (FNC) concorda que o seguinte texto reflete a nossa definição do termo "Internet". "Internet" refere-se ao sistema de informação global que - (i) é logicamente ligado entre si por um espaço de endereço único global baseado nos seus prolongamentos posteriores / *follow-ons* do Internet Protocol (IP) ou; (ii) é capaz de suportar comunicações usando o *Transmission Control Protocol* / Internet Protocol Suite (TCP / IP) ou suas subsequentes extensões / *follow-ons*, e / ou outros protocolos IP compatíveis; e (iii) fornece, utiliza ou torna acessível, público ou privado, serviços de alto nível em camadas sobre as comunicações e infraestrutura relacionadas, aqui descritos (INTERNET SOCIETY, 2014).

A internet mudou muito nas duas décadas desde que entrou em existência. Ela foi concebida na era do compartilhamento de tempo, mas sobreviveu à era dos computadores pessoais, cliente-servidor. Foi desenhado antes da existência das LANs, mas tem acomodado a nova tecnologia de rede. Ela foi concebida como apoiadora de uma série de funções de compartilhamento de arquivos e *login* remoto para compartilhamento e colaboração de recursos, e gerou correio eletrônico e, a *World Wide Web*. Mas o mais importante foi que começou como a criação de um pequeno grupo de pesquisadores dedicados e cresceu para ser um sucesso comercial com bilhões de dólares de investimento anual.

A disponibilidade da rede, ou seja, internet, juntamente com a poderosa computação acessível e comunicações, está possibilitando um novo paradigma. Esta evolução nos trará novas aplicações. Ela está evoluindo para permitir formas mais sofisticadas de preços e de custos, uma exigência talvez dolorosa neste mundo comercial. Ela está mudando para acomodar mais uma geração de tecnologias de rede subjacentes e com diferentes características e exigências. Novos modos de acesso e novas formas de serviço vão gerar novas aplicações, que por sua vez irão conduzir uma maior evolução da própria rede.

A questão mais premente para o futuro da internet não é como a tecnologia muda, mas a forma como o processo de mudança e evolução em si será gerenciado. A arquitetura da internet sempre foi impulsionada por um grupo de *designer*s, mas a forma desse grupo mudou como o número de interessados cresceu.

2.1 UM POUCO DA HISTÓRIA DA INTERNET NO BRASIL

Em 1988, através da comunidade acadêmica de São Paulo – Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), do Rio de Janeiro – Universidade Federal do Rio de Janeiro (UFRJ) e Laboratório Nacional de Computação Científica (LNCC) tivemos a chegada da internet ao Brasil.

Foi criada então, em 1989, uma instituição com os objetivos de iniciar e coordenar o oferecimento de serviços de acesso à internet no Brasil. Essa instituição foi criada pelo Ministério de Ciência e Tecnologia. Inicialmente foi criado um *backbone* que interligava onze estados.

O foco na área comercial iniciou-se apenas em 1994 com projeto piloto da Embratel. Ainda os primeiros acessos à rede no Brasil eram por meio de linhas discadas.

Linhas discadas: modo de estabelecer uma ligação entre o seu computador e um provedor de acesso é através de uma chamada telefônica comum. Neste tipo de ligação, a conexão à internet só existe durante o tempo em que a chamada telefônica ao provedor de acesso estiver ativa.

3 FUNCIONAMENTO DA INTERNET

Agora que conhecemos um pouco do histórico da internet vamos entender como ela funciona. A internet consiste em uma conectividade de redes de tecnologias distintas, isso é possível pelo uso do conjunto de protocolos como o TCP/IP. A sigla TCP/IP significa *Tranmission Control Protocol/Internet Protocol* e sua função é executar a conexão em nível de rede o que permite a comunicação entre aplicações em computadores de redes distintas sem a necessidade de conhecimento da topologia envolvida no processo. A maior característica do TCP/IP é a flexibilidade de adaptação às tecnologias de redes existentes, isso ocorre porque o TCP/IP foi desenvolvido de forma independente das tecnologias de redes.

Na rede TCP/IP, cada equipamento deve ter um endereçamento único capaz de identificá-lo na rede. Este endereçamento é baseado em um identificador que não depende da tecnologia envolvida, e é conhecido como endereço IP, na qual seu formato é o seguinte: 192.168.15.4. Este formato representa um número de 32 *bits*, que está associado a um único equipamento, identificando a rede e o equipamento nesta rede.

O protocolo IP executa uma tarefa chamada roteamento. O roteamento é responsável pela entrega das informações geradas pelas aplicações a seus destinos de forma correta e eficiente. Esta execução é baseada em tabelas de roteamento existentes em todos os equipamentos conectados à internet, onde se encontram as rotas a serem seguidas pelas informações em função do destino. A manutenção destas tabelas se dá de forma estática e de forma dinâmica.

Na forma estática, o administrador da rede pode configurar manualmente rotas para todas as redes a ela ligadas e uma rota *default* apontando para o próximo roteador. Já na forma dinâmica, o administrador da rede pode usar algumas aplicações que implementem protocolos de roteamento que determinam e divulgam essas rotas.

Os endereços na internet são semelhantes aos endereços residenciais e comerciais. Eles auxiliam usuários, correspondências eletrônicas e informações a se deslocarem de um local para outro. A internet utiliza um método conhecido como DNS (*Domain Name System*) para atribuir endereços aos computadores. Os computadores utilizam os endereços numéricos para direcionar os dados pela internet, ao passo que os usuários utilizam endereços em formato de nomes, pois são mais fáceis de lembrar. Para identificar cada um dos usuários, o DNS divide um endereço em três seções, assim como o seu endereço residencial também possui pelo menos três seções: rua, cidade e estado. Observe um exemplo desse formato de endereço:

greisse@uniasselvi.edu.br

Isto significa dizer que:

- greisse: é o nome da pessoa.
- o símbolo @ (arroba): substitui a preposição "em". Todo o conteúdo antes deste símbolo identifica a pessoa e todo o conteúdo após este símbolo identifica a organização a que esta pessoa pertence.
- uniasselvi: é o nome do computador específico onde “greisse” pode ser encontrada.n Trata-se de um computador de uma empresa, UNIASSELVI.
- .edu: é o domínio organizacional – informa que UNIASSELVI é uma organização educacional.
- .br : trata de domínios geográficos, ou seja, indica a que país a UNIASSELVI pertence.

Existem vários domínios organizacionais diferentes, vamos citar aqui os mais conhecidos:

- com = entidades comerciais
- edu = instituições educacionais
- gov = instituições governamentais não militares
- nt = instituições internacionais (NATO)
- mil = instituições militares
- net = recursos de redes de comunicação

Existem vários códigos de domínios geográficos. Veja alguns:

- br – Brasil
- fr – França
- it – Itália
- es – Espanha
- jp – Japão
- ca – Canadá

LEITURA COMPLEMENTAR

A ORIGEM DA INTERNET

A história da rede de computadores criada na Guerra Fria que deu início a Terceira Revolução Industrial.

A internet revolucionou o funcionamento tradicional das sociedades modernas como o fizeram, a seu tempo, a imprensa, a máquina a vapor, a eletricidade ou a telegrafia sem fio (rádio). Hoje parece normal fazer cursos *on-line*, preencher formulários administrativos a distância ou expressar opiniões em fóruns de discussão. Segundo a última Pesquisa Nacional por Amostra de Domicílios (Pnad), realizada em 2009 pelo Instituto Brasileiro de Geografia e Estatística (IBGE), 67,9 milhões de brasileiros estavam conectados à internet, ou seja, o número de domicílios com acesso à internet no Brasil cresceu 71% entre 2005 e 2009. No entanto, poucos conhecem sua história e as razões de sua criação.

De acordo com o dicionário Houaiss, internet é "rede de computadores dispersos por todo o planeta que trocam dados e mensagens utilizando um protocolo comum". Ela nasceu no final dos anos 1960, em plena Guerra Fria, graças à iniciativa do Departamento de Defesa americano, que queria dispor de um conjunto de comunicação militar entre seus diferentes centros. Uma rede que fosse capaz de resistir a uma destruição parcial, provocada, por exemplo, por um ataque nuclear.

Para isso, o pesquisador Paul Baran concebeu um conjunto que teria como base um sistema descentralizado. Esse cientista é considerado um dos principais pioneiros da internet. Ele pensou em uma rede tecida como uma teia de aranha (*web*, em inglês), na qual os dados se movessem buscando a melhor trajetória possível, podendo "esperar" caso as vias estivessem obstruídas. Essa nova tecnologia, sobre a qual também se debruçaram outros grupos de pesquisadores americanos, foi batizada de *packet switching*, "troca de pacotes".

Em 1969, a rede ARPAnet já estava operacional. Ela foi o fruto de pesquisas realizadas pela *Advanced Research Project Agency* (ARPA), um órgão ligado ao Departamento de Defesa americano. A ARPA foi criada pelo presidente Eisenhower em 1957, depois do lançamento do primeiro satélite Sputnik pelos soviéticos, para realizar projetos que garantissem aos Estados Unidos a superioridade científica e técnica sobre seus rivais do leste.

A ARPAnet a princípio conectaria as universidades de Stanford, Los Angeles, Santa Barbara e de Utah. Paralelamente, em 1971, o engenheiro americano Ray Tomlinson criou o correio eletrônico. No ano seguinte, Lawrence G. Roberts desenvolveu um aplicativo que permitia a utilização ordenada dos *e-mails*. As mensagens eletrônicas se tornaram o instrumento mais utilizado da rede. A ARPAnet seguiu sua expansão durante os anos 1970 – a parte de comunicação militar da rede foi isolada e passou a se chamar MILnet.

Outras redes, conectando institutos de pesquisas, foram criadas nos Estados Unidos, Grã-Bretanha e França. Faltava estabelecer uma linguagem comum a todas. Isso foi feito com o protocolo TCP/IP, inventado por Robert Kahnet e Vint Cerf em 1974. A ARPAnet adotou essa padronização em 1976. E assim começou a aventura da *web* com seu primeiro milhar de computadores conectados. O afluxo de usuários engendrou um fenômeno de sobrecarga. Em 1986, uma nova rede foi lançada pela *National Science Foundation*. A ARPAnet se juntou a ela quatro anos mais tarde.

Uma etapa decisiva foi superada em 1990 com a criação, por um pesquisador do Conselho Europeu para a Pesquisa Nuclear em Genebra (Cern), Tim Berners-Lee, do protocolo HTTP (*Hyper Text Transfer Protocol*) e da linguagem HTML (*Hyper Text Markup Language*), que permitem navegar de um *site* a outro, ou de uma página a outra. A *World Wide Web* (www) lançou seu voo, e a internet se abriu ao público, empresas particulares e privadas. Uma multidão de *sites* apareceu.

Com uma infraestrutura de comunicação teoricamente desprovida de autoridade central, a internet, todavia, seria gerida de um contrato com o governo americano, que havia financiado sua criação, e diversos órgãos que assegurariam seu crescimento. Foi o caso da Internet *Assigned Numbers Authority* (IANA), responsável pela gestão dos nomes dos domínios, o DNS (*Domain Name System*). Graças a ele, os endereços IP, constituídos de uma série de códigos (o endereço numérico atribuído a cada computador conectado à rede) são traduzidos em letras que compõem nomes identificáveis e memorizáveis.

Apesar de gerido pela IANA, o DNS sempre esteve sob controle do Departamento de Comércio dos Estados Unidos. Em 1998, sua gestão foi confiada a uma organização californiana de direito privado, a Internet *Corporation for Assigned Names and Numbers* (Icann). Em 2009, os contratos que ligavam a Icann ao Departamento de Comércio americano expiraram, e a empresa passou a ter mais autonomia. Sua missão é assegurar, dos Estados Unidos, a coordenação técnica do sistema de denominação. Deve promover também a concorrência e garantir a representação global das comunidades na internet. Os interessados em política mundial da rede podem participar de seus trabalhos, por meio de fóruns acessíveis em seu *site* na *web*.

Esse controle técnico e administrativo da internet nos Estados Unidos causa, porém, tensões internacionais. Desde 2003, a Organização das Nações Unidas (ONU) reclama uma gestão "multilateral, transparente e democrática, com a plena participação dos Estados, do setor privado, da sociedade civil e das organizações internacionais". Em 2006, em decorrência de tal demanda, foi instituída uma estrutura de cooperação internacional, o *Internet Governance Forum* (IGF). Mas essa instância tem apenas papel consultivo. Ela deve, também, velar pela liberdade de difusão das inovações tecnológicas e ideias. Uma questão essencial, pois a internet se baseia no princípio de neutralidade, que exclui qualquer discriminação da fonte, destinatário ou conteúdo transmitido na rede.

Já existem mais de 2 bilhões de internautas no mundo, ou seja, um terço da população planetária. Os progressos da informática, associados aos do audiovisual e das telecomunicações, permitiram a criação de novos serviços. Depois do desenvolvimento de redes de banda larga com fio (ADSL e fibra óptica) e sem fio (*wifi*, *Bluethooth* e 3G), e da internet móvel (WAP), desenvolveram-se outras tecnologias e produtos da chamada "web 2.0". Essa segunda geração se caracteriza por suas aplicações interativas (*blogs*, *wikis*, *sites* de compartilhamento de fotos e vídeos ou redes sociais), que renovaram a relação entre os usuários e os serviços de internet, criando o princípio de uma cultura compartilhada em rede.

Assim como a dominação americana da regulação técnica é vista por outros Estados como uma ameaça, o estabelecimento de controles nacionais por meio de sistemas que impedem o livre acesso à internet constitui também outro perigo político para as liberdades individuais.

Em janeiro de 2007, o especialista francês Bernard Benhamou anunciou em um artigo sobre as novas questões da governança da internet que a capacidade de fragmentação da rede apresenta riscos em relação ao plano industrial e político. Ele pensava particularmente na China, que tentou criar seu próprio sistema de endereçamento, independente do DNS. Uma maneira eficaz de bloquear a consulta de seus *sites* aos internautas de fora e de interditar à população chinesa o acesso aos *sites* externos.

Isso já é realidade. Há mais de três anos o servidor de nomes de domínios chineses não passa mais pela Icann, para que, de acordo com o governo chinês, seu povo possa aprender os ideogramas, em vez de palavras do alfabeto latino. Essa "sub-rede", de acordo com a expressão do jornalista Hubert Guillaud, do jornal francês Le Monde, que recebe o nome poético de "escudo de ouro", é diretamente controlada pelo governo chinês. Não há dúvidas de que esse tipo de internet do Império do Meio, que associa censura e controle, pode rapidamente ser copiado por nações que não utilizam o alfabeto latino. Isso teria como consequência a fragmentação da internet em múltiplas redes incompatíveis.

A "ciberguerra" que opôs Google e Pequim no início de 2010, quando o maior *site* de buscas do mundo ameaçou deixar o país após ser atacado por *hackers* chineses e constatar a invasão de contas de *e-mai*ls de ativistas de direitos humanos, fez do livre acesso à internet uma prioridade da política externa dos Estados Unidos. A internet se converteu em uma arma política da Casa Branca na luta pela preservação de sua hegemonia comercial e estratégica.

As recentes revoluções na Tunísia e no Egito mostraram o papel determinante da *web*, dos *blogs* e das redes sociais na queda de regimes ditatoriais. A internet se tornou "um Titã que ninguém pode conter", como disse o jornalista tunisiano Taoufik Ben Brik, e essa nova ciber-resistência pode, se não mudar, pelo menos acelerar o curso da história.

FONTE: Disponível em: <http://www2.uol.com.br/historiaviva/reportagens/o_nascimento_da_internet.html>. Acesso em: 20 set. 2014.

RESUMO DO TÓPICO 1

Maravilha! Conseguimos um avanço fantástico ao cumprirmos mais esta tarefa. Você é a pessoa mais interessada em rever um pouco mais daquilo que discutimos neste tópico.

Neste tópico podemos conhecer um pouco da história da internet no mundo e no Brasil.

- Vimos que a ARPAnet foi o fruto de pesquisas realizadas pela ARPA, um órgão ligado ao Departamento de Defesa americano. A ARPAnet, a princípio, conectaria algumas universidades.

- Outras redes, conectando institutos de pesquisas foram criadas. Para estabelecer uma linguagem comum a todas foi criado o protocolo TCP/IP, inventado por Robert Kahnet e Vint Cerf. E assim começou a aventura da *web* com seu primeiro milhar de computadores conectados. Em 1972 foi introduzida a primeira aplicação do correio eletrônico.

- A primeira aparição da internet no Brasil em 1988 por comunidades acadêmicas. Na área comercial apenas em 1994 pela Embratel.

- A WWW é uma teia mundial formada por computadores que são identificados por um protocolo de internet e seus conteúdos são acessados por um documento chamado *Hyper Text Transfer Protocol* – HTTP.

- Cada *site* deve estar depositado em um computador e identificado por um DNS (*Domain Name System*), identificado com domínios organizacionais e domínios geográficos. O funcionamento da internet é outro fator importante para entendermos o seu sucesso.

AUTOATIVIDADE

1 Quais foram os primeiros vestígios da criação da internet e em que época?

2 O que era a ARPANET e como funcionava?

3 Como e quando houve a criação do *e-mail*?

4 Quem foi Tim Berners Lee?

5 Quais são os domínios organizacionais mais conhecidos?

INTRODUÇÃO AO DESENVOLVIMENTO PARA *WEB*

1 INTRODUÇÃO

No tópico anterior estabelecemos as bases para o estudo da introdução à linguagem de programação, abordando o histórico da internet e algumas características que a envolvem.

No Tópico 2 apresentaremos alguns conceitos básicos atuais relacionados à programação para *web*, introduzindo algumas tecnologias e conhecendo os padrões para o desenvolvimento *web*.

2 TECNOLOGIAS ENVOLVIDAS NO DESENVOLVIMENTO PARA *WEB*

Vamos conhecer a seguir algumas tecnologias envolvidas no desenvolvimento para *web*.

2.1 HTML

HTML é a linguagem para descrever a estrutura das páginas *we*b. HTML dá aos autores os meios para publicar documentos *on-line* com títulos, texto, tabelas, listas, fotos etc. Recuperar informações *on-line* através de *links* de hipertexto, com o clique de um botão. Concepção de formulários para a realização de transações com serviços remotos, para uso em busca de informações, reservas, encomenda de produtos etc. Incluir de folhas de difusão, videoclipes, clipes de som e outras aplicações diretamente em seus documentos. Com HTML, os autores descrevem a estrutura das páginas usando a marcação. Os elementos das peças, rótulo, linguagem de conteúdo, como "ponto", "lista", "mesa", e assim por diante.

2.2 XHTML

XHTML é uma variante do HTML que usa a sintaxe XML, *Extensible Markup Language*. XHTML tem todos os mesmos elementos (para parágrafos etc.) como a variante de HTML, mas a sintaxe é um pouco diferente. Porque XHTML é uma aplicação XML, você pode usar outras ferramentas XML com ele (como XSLT, uma linguagem para transformar o conteúdo XML).

2.3 CSS

CSS é a linguagem para descrever a apresentação das páginas da *web*, incluindo cores, *layou*t e fontes. Ele permite se adaptar à apresentação de diferentes tipos de dispositivos, tais como telas grandes, telas pequenas, ou impressoras. CSS é independente do HTML e pode ser usado com qualquer linguagem de marcação baseada em XML. A separação do código HTML CSS faz com que seja mais fácil de manter locais, folhas de estilo entre páginas e páginas personalizadas para diferentes ambientes. Isto é referido como a separação da estrutura (ou: o conteúdo) da apresentação.

2.4 WEBFONTS

WebFonts é uma tecnologia que permite às pessoas usar fontes sob demanda pela *web*, sem necessidade de instalação no sistema operacional. Até recentemente, as fontes não têm sido comuns na *web*, devido à falta de um formato de fonte interoperável. O esforço *webfonts* pretende abordar que através da criação de, um formato de fonte aberta suportada pela indústria para a *web* (chamado de "WOFF").

2.5 EXEMPLO HTML

O seguinte exemplo é de uma parte de um documento HTML ilustra como criar uma ligação dentro de um número. Quando prestados na tela o texto do *link* será "relatório final"; quando alguém ativa o *link*, o navegador irá recuperar o recurso identificado por "http://www.exemplo.com.br":

```
<p class = "maisinfo"> Para mais informações, consulte o
<a href=" http://www.exemplo.com.br"> </a> relatório final. </ p>
```

O atributo de classe no início *tag* do parágrafo ("<p>") pode ser usado, entre outras coisas, para adicionar estilo. Por exemplo, para colocar em itálico o texto de todos os parágrafos com uma turma de "maisinfo", pode-se escrever, em CSS:

```
p.maisinfo {font-style: italic}
```

Ao colocar essa regra em um arquivo separado, o estilo pode ser partilhado por qualquer número de documentos HTML.

2.6 JAVASCRIPT WEB APIS

Um *script* é um código de programa que não precisa de pré-processamento (por exemplo, compilação), antes de ser executado. No contexto de um navegador da *web*, *scripting* normalmente se refere ao código do programa escrito em JavaScript que é executado pelo navegador quando uma página é baixada, ou em resposta a um evento acionado pelo usuário.

Scripting pode fazer páginas da *web* mais dinâmica. Por exemplo, sem ter que recarregar uma nova versão de uma página pode permitir modificações para o conteúdo da página, ou permitir que o conteúdo a ser adicionado ou enviado a partir dessa página. O primeiro foi chamado DHTML (*Dynamic* HTML), e o último AJAX (*Asynchronous JavaScript and* XML).

Além disso, os *scripts* permitem cada vez mais desenvolvedores para criar uma ponte entre o navegador e a plataforma a ser usada, o que torna possível, por exemplo, para criar páginas da *web* que incorporam informações do ambiente do usuário, tais como localização atual, detalhes do catálogo de endereços etc.

Esta interatividade adicional faz que páginas da *web* se comportem como um aplicativo de *software* tradicional. Essas páginas da *web* são frequentemente chamados de aplicativos da *web* e podem ser disponibilizadas diretamente no navegador como uma página *web*.

Enquanto *scripting* oferecem uma grande oportunidade para desenvolver novas interfaces e experimentar novas interações com o usuário, ao longo do tempo uma série destas adições beneficiam uma abordagem mais declarativa.

2.7 WEB STANDARDS OU PADRÕES DA *WEB*

Web Standards é um termo muito discutido, mas talvez você esteja se perguntando: qual é realmente o significado? *Standards* ou padrões da *web* são um conjunto de normas, diretrizes, recomendações, notas, artigos, tutoriais e afins de caráter técnico, produzidos pela W3C. É destinado a orientar fabricantes, desenvolvedores e projetistas para o uso de práticas que possibilitem a criação de uma *web* acessível a todos.

Você pode acessar o W3C através do site: <http://www.w3.org/>.

Os padrões na W3C definem uma plataforma da *web* aberta para o desenvolvimento de aplicações que têm o potencial sem precedentes para permitir aos desenvolvedores criar experiências interativas, alimentados por vastas quantidades de dados, que estão disponíveis para qualquer dispositivo. Embora os limites da plataforma continuem a evoluir, os líderes da indústria falam quase que unicamente sobre como HTML5 será a pedra angular para esta plataforma. Mas a força total da plataforma se baseia em muitas outras tecnologias que a W3C e os seus parceiros estão criando, incluindo CSS, SVG, WOFF, XML e uma variedade de APIs.

A W3C desenvolve essas especificações técnicas e orientações através de um processo projetado para maximizar a consenso sobre o conteúdo de um relatório técnico, para garantir a alta qualidade técnica e editorial, e ganhar aprovação pelo W3C e a comunidade em geral.

O *World Wide Web Consortium* (W3C) é uma comunidade internacional em que as organizações associadas, profissionais de tempo integral, bem como a obra pública em conjunto para desenvolver padrões *web*. Liderado pelo inventor da *Web* Tim Berners-Lee e o CEO Jeffrey Jaffe, a missão do W3C é levar a *web* ao seu potencial máximo (W3C, 2014).

O W3C possui um escritório aqui no Brasil e caso você queira acompanhar o trabalho deles ou até mesmo participar destes trabalhos, basta acompanhar a página do W3C Brasil: <http://www.w3c.br/Home/WebHome>.

2.7.1 Acessibilidade

A W3C publicou *Accessibility Guidelines* (WCAG) para ajudar os autores criarem conteúdo que seja acessível a pessoas com deficiência.

A *web* é fundamentalmente concebida para funcionar para todas as pessoas, independentemente do seu *hardware*, *software*, língua, cultura, localização ou condição física ou mental. Quando a *web* atende a esse objetivo é acessível a pessoas com uma gama diversificada de audição, movimento, visão e capacidade cognitiva.

Assim, o impacto da deficiência é radicalmente alterado na *web* porque a *web* remove barreiras à comunicação e interação que muitas pessoas enfrentam no mundo físico. No entanto, quando *sites*, tecnologias *web*, ou ferramentas da *web* são mal concebidos, eles podem criar barreiras que excluem as pessoas de usar a internet.

A missão da *Web Accessibility Initiative* (WAI) é levar a *web* ao seu potencial para ser acessível, permitindo que as pessoas com deficiência participem igualmente na *web*.

É essencial que a *web* seja acessível a fim de proporcionar igualdade de acesso e oportunidades iguais para pessoas com diversas habilidades. De fato,

a Convenção das Nações Unidas sobre os Direitos das Pessoas com Deficiência reconhece o acesso às tecnologias de informação e comunicação, incluindo a *web*, como um direito humano básico.

Acessibilidade apoia a inclusão social de pessoas com deficiência, bem como outros, tais como os idosos, as pessoas em áreas rurais, e as pessoas nos países em desenvolvimento.

Há também um caso de negócios forte para a acessibilidade. Acessibilidade sobreposições com outras práticas, tais como *design* para *web* móvel, independência do dispositivo, interação multimodal, usabilidade e *design* para os usuários mais velhos. Os estudos de caso mostram que *sites* acessíveis têm melhores resultados de busca, custos de manutenção reduzidos e maior alcance público, entre outros benefícios.

Devidamente projetados *sites* e ferramentas da *web* podem ser usados por pessoas com deficiência. No entanto, atualmente muitos *sites* e ferramentas são desenvolvidos com barreiras de acessibilidade que dificultam ou impossibilitam para algumas pessoas a usá-los. A seguir estão apenas alguns exemplos:

- Texto alternativo para imagens: o texto alternativo é o exemplo clássico. As imagens devem incluir texto alternativo equivalente na marcação/código. Se o texto não é fornecido para as imagens, a informação da imagem é inacessível, por exemplo, para pessoas que não podem ver e usam um leitor de tela que lê em voz alta as informações em uma página, incluindo o texto para a imagem visual. Quando o texto equivalente à imagem é fornecido, a informação está disponível para todas as pessoas cegas, bem como para as pessoas que desligam imagens em seu telefone móvel por terem menor largura de banda ou encargos, as pessoas em uma área rural com baixa largura de banda que desligando imagens para velocidade de *download* e outros. Também está disponível tecnologias que não podem ver a imagem.
- Apenas teclado: algumas pessoas não conseguem usar um *mouse*, incluindo muitos usuários mais velhos. Um *site* acessível não conta somente com o *mouse*, ele fornece todas as funcionalidades através de um teclado. Então, as pessoas com deficiência podem utilizar tecnologias assistivas que imitam o teclado, como a entrada de voz.
- Transcrições para *Podcasts*: assim como as imagens não estão disponíveis para as pessoas que não podem ver, os arquivos de áudio não estão disponíveis para as pessoas que não podem ouvir. Fornecer uma transcrição de texto faz com que a informação de áudio seja acessível a pessoas surdas ou com deficiência auditiva, bem como aos mecanismos de busca e outras tecnologias que não podem ouvir (W3C, 2014).

O W3C *Web Accessibility Initiative* (WAI) reúne pessoas da indústria, organizações de deficientes, governo e laboratórios de pesquisa de todo o mundo para desenvolver diretrizes e recursos para ajudar a tornar a *web* acessível a pessoas com deficiência, incluindo auditivo, cognitivo, neurológico, físico, fala e deficiência visual.

A cobertura de acessibilidade da *web* WAI inclui "o conteúdo da *web* (*sites* e aplicações *web*), ferramentas de autoria (como sistemas de gerenciamento de conteúdo (CMS) e *software* de *blog*), navegadores e outros" agentes de usuário, e as especificações técnicas do W3C (W3C, 2014).

2.7.2 *Web* para móvel

A implantação generalizada de dispositivos móveis habilitados para a *web* (como telefones) torna-se um alvo de escolha para criadores de conteúdo. Compreendendo seus pontos fortes e suas limitações, e utilização de tecnologias que se encaixam nestas condições são fundamentais para criar o sucesso de conteúdo.

Idealmente, os autores do *site* seriam capazes de atender à crescente demanda por uma experiência móvel de qualidade sem alterar uma linha de código. Mas a realidade é que um *site* projetado especificamente com mobilidade em mente sempre vai proporcionar uma melhor experiência de usuário para usuários móveis, mesmo quando eles estão equipados com o dispositivo.

As razões para isso incluem os desafios colocados pelos custos de rede e atrasos, limitações de memória e CPU, teclado e diferentes dispositivos. Tão importante quanto isso, eles apresentam um conjunto crescente de vantagens com a sua natureza pessoal e sempre disponível, e as suas capacidades sensíveis ao contexto cada vez mais.

Como resultado, a experiência móvel frequentemente merece seu próprio conjunto de considerações de projeto, como discutido em um crescente corpo de literatura, incluindo a criação da *web* móvel do W3C e de dispositivas e diretrizes de autoria independentes. Os usuários móveis operam em um contexto de uso muito diferente do que os usuários de PC, e proporciona-lhes uma experiência personalizada para suas necessidades. É provável que seja o melhor serviço a eles.

Uma série de barreiras que os usuários móveis enfrentam são semelhantes às vividas por pessoas com deficiência. Estas semelhanças tornam-se naturais a visarem ao desenvolvimento de *sites* que são acessíveis, tanto para pessoas com deficiência quanto para dispositivos móveis. Os dois documentos de referência neste espaço, as Diretrizes de Conteúdo *Web* (WCAG) e Acessibilidade e as Melhores Práticas em *Web* Móvel, têm, assim, uma série de *links* e semelhanças.

Nos últimos anos, o W3C desenvolveu uma série de tecnologias *web* que explicitamente levam em conta as especificidades dos dispositivos móveis:

- CSS *Mobile*, um perfil de linguagem de folhas de estilo em cascata que corresponde à necessidade dos autores de *web* móvel.

- SVG Tiny, um perfil de escalável formato de gráficos vetoriais da *web* bem adaptada às capacidades de dispositivos móveis.

- XHTML para dispositivos móveis, que define um subconjunto do XHTML para dispositivos móveis.

A última geração de navegadores móveis é capaz de usar as tecnologias da *web* mais avançados, incluindo recursos de HTML5, CSS 2.1 e 3, uma série de APIs JavaScript, abrindo o caminho para aplicações móveis baseadas na *web* (incluindo os *widgets*).

A iniciativa *Mobile Web* criou um programa de treinamento para ajudar os *webdesigners* e produtores de conteúdo que já estão familiarizados com o mundo do *desktop* para se familiarizar com a *web* como entregue em dispositivos móveis.

Normas para aplicações *web* no celular proporcionam uma visão completa de todas as tecnologias em desenvolvimento no W3C que ajudam a tornar a *web* melhor plataforma para desenvolvimento de aplicações *web* (W3C, 2014).

2.8 *BROWSERS* PARA *WEB*

As pessoas utilizam diversos *browser*s para navegar na internet. Por isso é importante conhecermos alguns, pois cada um possui suas características e desconfortos no momento de acessar uma página. Além disso, velocidade, segurança e acessibilidade são importantes. Problemas podem aparecer ao se utilizar linguagens que são processadas diretamente, pois nem todas as versões ou tipos de navegadores reconhecem todos os recursos de uma linguagem.

Desde os primeiros navegadores gráficos, como o Mosaic e o Netscape até o Internet Explorer, Mozilla Firefox, Opera, Apple Safari e Google Chrome. A seguir, vamos ver considerações de alguns deles.

O Windows Internet Explorer, também conhecido pela sigla IE, é o navegador de internet da Microsoft, que foi lançado como componente integrado desde a versão do Windows 98. Veja na figura a seguir a tela principal do Internet Explorer e suas principais funcionalidades.

FIGURA 4 – INTERNET EXPLORER

FONTE: A autora

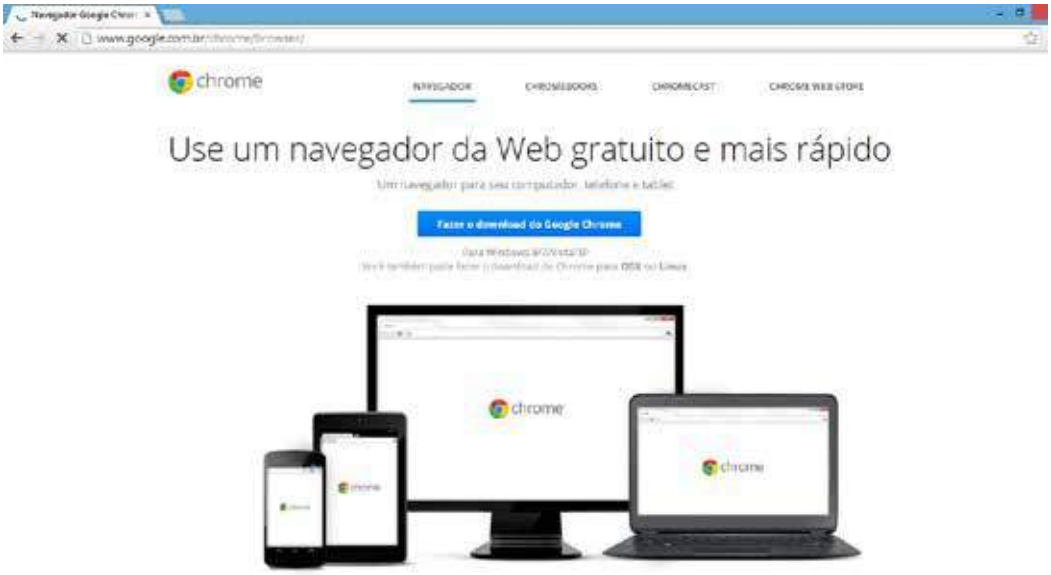
O Firefox é um navegador livre e multiplataforma, desenvolvido com ajuda de centenas de colaboradores ao redor do mundo. Observe a figura a seguir e perceba que as funcionalidades são basicamente as mesmas para todos os navegadores.

FIGURA 5 – MONZILA FIREFOX

FONTE: A autora

Assim como os exemplos anteriores, o navegador Chrome, desenvolvido pela Google, possui funcionalidades similares e que são características dos navegadores. Confira a figura a seguir:

FIGURA 6 – GOOGLE CHROME



FONTE: A autora

O navegador norueguês Opera traz recursos para marcar páginas favoritas, organizar sua leitura na internet e manter fixas as abas de navegação consideradas importantes. O aplicativo permite que as páginas favoritas sejam afixadas na barra e consegue realizar buscas por meio de *tags*. Dá também para personalizar a tela com temas e planos de fundo.

FIGURA 7 - OPERA

FONTE: A autora

O Safari traz em sua última edição o botão "*Links* compartilhados" onde guarda um histórico de *links* de amigos das redes sociais e as páginas por eles compartilhadas. Além disso, possui a opção de navegação privada na qual garante que nenhum histórico seja salvo.

FIGURA 8 - SAFARI

FONTE: A autora

LEITURA COMPLEMENTAR

COISAS PARA VERIFICAR ANTES DE PUBLICAR SEU *SITE*.

Finalmente, seu *site* está pronto para decolar! Este é um momento importante, e esperado para muitas pessoas. Embora você tenha talvez passado semanas ou meses olhando *link* por *lin*k, página por página, lendo e relendo o conteúdo, é sempre importante ter uma lista de coisas para se verificar. Afinal seria terrível logo nos primeiros dias, seus visitantes acharem uma palavra escrita errada, ou um *link* quebrado, ou uma imagem faltando.

Aqui segue uma lista de coisas que você deve olhar antes de lançar seu *site* definitivamente na *web*:

Trabalho com *web* desde 1998. Já faz quase 10 anos, e quase nunca vejo pessoas ou empresas falando que gostam de fazer um Controle de Qualidade (CDQ). Fazer um CDQ não é difícil, basta fazer uma lista do que deve ser verificado em cada página. Depois verificar cada item de cada página e pronto. Tudo questão de organização.

Aqui vão algumas dicas que deveriam estar inclusas em sua lista:

1- Veja o que está acima da **DOBRA**. A dobra da página é a linha horizontal imaginária que existe no final da sua tela. Quando você recém entra em uma página, tudo que há em cima desta linha está acima da dobra. Coloque o seu conteúdo principal e imagens acima desta dobra para obter máxima exposição. Lembre-se de que o propósito da página, e o tipo de informação contida nesta página precisa estar bem claro para quem está acessando.

2- Verifique **CADA *LINK***. Você pode ter editado, reorganizado e finalizado cada página, mesmo com todo seu trabalho duro é possível haver *links* quebrados. Cada vez que um usuário clica um *link* ele espera encontrar exatamente o que está procurando, e um *link* quebrado ou uma página de erro vão causar frustração e desconfiança na mente do seu visitante. Portanto, não custa nada verificar que cada *link* esteja de fato enviando a pessoa para o destino certo. Existem programas que podem fazer isto por você (noutro *pos*t eu falo de alguns, mas podem me cobrar se eu demorar).

Faça **correção ortográfica** (eu admito, nem sempre faço isso). O *DreamWeaver*, o *FrontPag*e e maioria dos editores de páginas *web* tem corretores de idioma, talvez você tenha que procurar um *add-on* ou *plugin* mas sempre é

possível ter algum tipo de corretor ortográfico instalado para assegurar que os textos estão escritos corretamente em seu *website*. Lembre-se de que durante o desenvolvimento do *site*, você formatou e editou o texto diversas vezes, então vale a pena dar uma última verificada.

4- Esteja certo de que a **navegação** do seu *site* é **fácil** de compreender. Ter os tópicos, ou os *menus* do seu *site* fácil de compreender, com títulos descritivos, e sem muita firula ajuda ao visitante encontrar o que está buscando.

5- Diferenciar cabeçalhos, botões, índices, datas e texto ajuda ao visitante para identificar cada elemento do seu *site*, e assim ele consegue rapidamente navegar por suas páginas.

6- Esteja certo de que a ilustração que você veicular seja relevante ao conteúdo. Imagens bonitas que reforçam a ideia do título ou da apresentação do conteúdo ajudam a definir a decisão do visitante para acessar aquele determinado conteúdo.
7- Verifique os **formulários** de envio. A última coisa que você precisa, é que um visitante queira se comunicar com você e não consiga. Verifique principalmente formulários de Fale Conosco, Atendimento, Pedidos etc.
8- Receber de braços abertos às sugestões, nem sempre o seu olhar solitário tem condições de ver todos os aspectos do seu *site*, portanto, é sempre bom mostrar seu projeto para alguns amigos, para aquele camarada viciado na *we*b, e pedir sugestões, avaliações, e depois filtrar de acordo com suas necessidades. Um desconhecido ao site, navegando pela primeira vez, sempre é interessante para verificar como será com seus visitantes. Tendo isso dito, qualquer sugestão é muito bem-vinda.

Claro que há outras regras, que podem ser importantes, existe inclusive milhares de ferramentas que são utilizadas por *webmaster* pelo mundo todo para efetuar a verificação de diversos aspectos do seu *site*. Desde ferramentas que te mostram quantas vezes cada palavra se repete pela extensão do seu *site*, para determinar qual é a palavra mais encontrada, até validadores de código XML, HTTP, CSS etc.

Dependendo do *site* que você está publicando, haverá verificações adicionais que devem ser feitas.

O principal é você sempre se lembrar de que o seu *site* deve ser visto pelos seus visitantes como uma ferramenta, um meio de informação, um meio de encontrar ou obter o que estão buscando.

É uma maneira muito importante de expressar suas ideias, seus produtos, sua informação, e o *site* serve como um exemplo de tudo isso, portanto, você precisa zelar por ele assim como faz com sua empresa ou negócio e imagem.

FONTE: Disponível em: <http://www.revistainternet.com.br/2007/12/20/coisas-para-verificar-antes-de-publicar-seu-site/>. Acesso em: 20 set. 2014.

RESUMO DO TÓPICO 2

- HTML é a linguagem para descrever a estrutura das páginas *web*.
- XHTML é uma variante do HTML que usa a sintaxe XML.
- CSS é a linguagem para descrever a apresentação das páginas da *web*.
- *WebFonts* é uma tecnologia que permite às pessoas usar fontes sob demanda pela *web*, sem necessidade de instalação no sistema operacional.
- Um *script* é um código de programa que não precisa de pré-processamento (por exemplo, compilação), antes de ser executado.
- *Standards* ou padrões da *web* são um conjunto de normas, diretrizes, recomendações, notas, artigos, tutoriais e afins de caráter técnico, produzidos pela W3C.
- Vimos a importância da acessibilidade e da programação para aplicativos móveis.
- Além disso, vimos alguns *browsers* também importantes para o desenvolvimento *web*.

AUTOATIVIDADE

1 Sobre o HTML analise as sentenças a seguir:

I- É uma linguagem para descrever estrutura das páginas *web*.
II- Através dele podemos publicar documentos, textos, tabelas etc.
III- Podemos conceber formulários.
IV- Foi desenvolvido apenas para *webdesigners*.

Agora assinale a alternativa na qual as sentenças estão corretas:

a) I, II e IV.
b) I, III e IV.
c) I, II e III.
d) Todas as alternativas estão corretas.

2 Sobre o CSS analise as sentenças a seguir:

I- É uma linguagem para descrever a apresentação das páginas da *web*.
II- CSS é dependente do HTML.
III- Permite se adaptar à apresentação de diferentes tipos de dispositivos.
IV- Pode ser usado com qualquer linguagem de marcação baseada em XML.

Agora assinale a alternativa em que as sentenças estão corretas:

a) I, II e IV.
b) I, III e IV.
c) I, II e III.
d) Todas as alternativas estão corretas.

3 A partir do conceito de *script* assinale V para as sentenças verdadeiras e F para as falsas.

a) ( ) É um código de programa que não necessita de pré-processamento.
b) ( ) Geralmente se refere ao código do programa escrito em JavaScript.
c) ( ) Pode fazer páginas mais dinâmicas.
d) ( ) O primeiro foi chamado de AJAX.

Agora, assinale a sequência CORRETA:

a) ( ) F – V – F – V.
b) ( ) V – V – V – F.
c) ( ) F – F – V – V.
d) ( ) F – V – V – F.

4 Vimos que para desenvolver uma página de qualidade é importante seguirmos padrões específicos para a *WEB*. Para auxiliarmos nestes padrões temos como nosso aliado a W3C. Descreva o que é a W3C.

5 Cite os *browsers* apresentados.

FERRAMENTAS DE DESENVOLVIMENTO

1 INTRODUÇÃO

As ferramentas de desenvolvimento aplicadas ao desenvolvimento *web* disponíveis são as mais diversas possíveis que vão desde *software*s geradores de HTML, JavaScript, CSS até os que trazem um servidor de *scripts* de páginas dinâmicas.

Ao escolher uma forma de desenvolvimento é necessário lembrar que um *site* não será visto somente pelos usuários na máquina em que foi desenvolvido, mas será visto por usuários em outras máquinas que às vezes não possuem recursos disponíveis. As implementações em HTML.

Além disso, *software*s geradores de códigos não são reconhecidos por todos os servidores. Uma saída é conhecer os recursos existentes, seus limites, vantagens, sua relação custo-benefício e testar os resultados dos *scripts* na máquina servidora antes de disponibilizar um *site* ao público.

Podemos desenvolver *sites* de duas maneiras uma delas é utilizando *softwares* geradores de códigos a outra maneira é digitando os comandos em um editor de texto e visualizando o resultado através do navegador.

2 ALGUMAS FERRAMENTAS PARA DESENVOLVIMENTO WEB

Veremos a seguir algumas ferramentas para desenvolvimento *WEB*:

- Notepad ++ é uma ferramenta livre. O editor de código-fonte é a substituição do bloco de notas que suporta várias línguas. Em execução no ambiente Windows, o seu uso é regido pela licença GPL. Baseado no poderoso componente de edição Scintilla, Notepad ++ é escrito em C ++ e usa API Win32 puro e STL, que garante uma maior velocidade de execução e menor tamanho do programa. Ao aperfeiçoar tantas rotinas possíveis, sem perder a facilidade de utilização.

FIGURA 9 – Notepad++

FONTE: A autora

- NetBeans: o IDE oficial para Java 8 com seus editores, analisadores de código, e conversores, podem rapidamente e facilmente atualizar seus aplicativos para usar as novas construções de linguagem Java 8, como operações funcionais e referências de métodos. Analisadores e conversores de lote são fornecidos para pesquisa através de múltiplas aplicações ao mesmo tempo, os padrões de conversão correspondente a novas construções de linguagem Java 8. Com seu constante aperfeiçoamento Java Editor possui muitas características ricas e uma ampla gama de ferramentas, modelos e exemplos, o NetBeans IDE define o padrão para o desenvolvimento com tecnologias de ponta. O editor suporta várias linguagens de Java, C / C ++, XML e HTML, para PHP, Groovy, Javadoc, JavaScript e JSP. Porque o editor é extensível, você pode conectar o suporte para muitas outras línguas.

Mantendo uma visão clara de grandes aplicações, com milhares de pastas e arquivos, e milhões de linhas de código. NetBeans IDE oferece diferentes pontos de vista de seus dados, de várias janelas do projeto para ferramentas úteis para a criação de seus aplicativos e gerenciá-los de forma eficiente, permitindo-lhe aprofundar em seus dados de forma rápida e facilmente, dando-lhe ferramentas de controle.

O NetBeans fornece ferramentas de análise estática, especialmente a integração com a ferramenta amplamente utilizadas para identificar e corrigir problemas comuns em código Java. Além disso, o depurador do NetBeans permite que você coloque pontos de interrupção em seu código-fonte, adicione inspeções de campo, depure seu código, executando métodos, tirar fotos e acompanhar a execução como ele ocorre.

O NetBeans fornece assistência especializada para otimizar a velocidade e uso de memória do seu aplicativo, e torna mais fácil para construir aplicações confiáveis e escaláveis para Java. NetBeans IDE inclui um depurador visual para aplicativos Java, permitindo que você depure interfaces de usuário, sem olhar no código-fonte (NETBEANS, 2014).

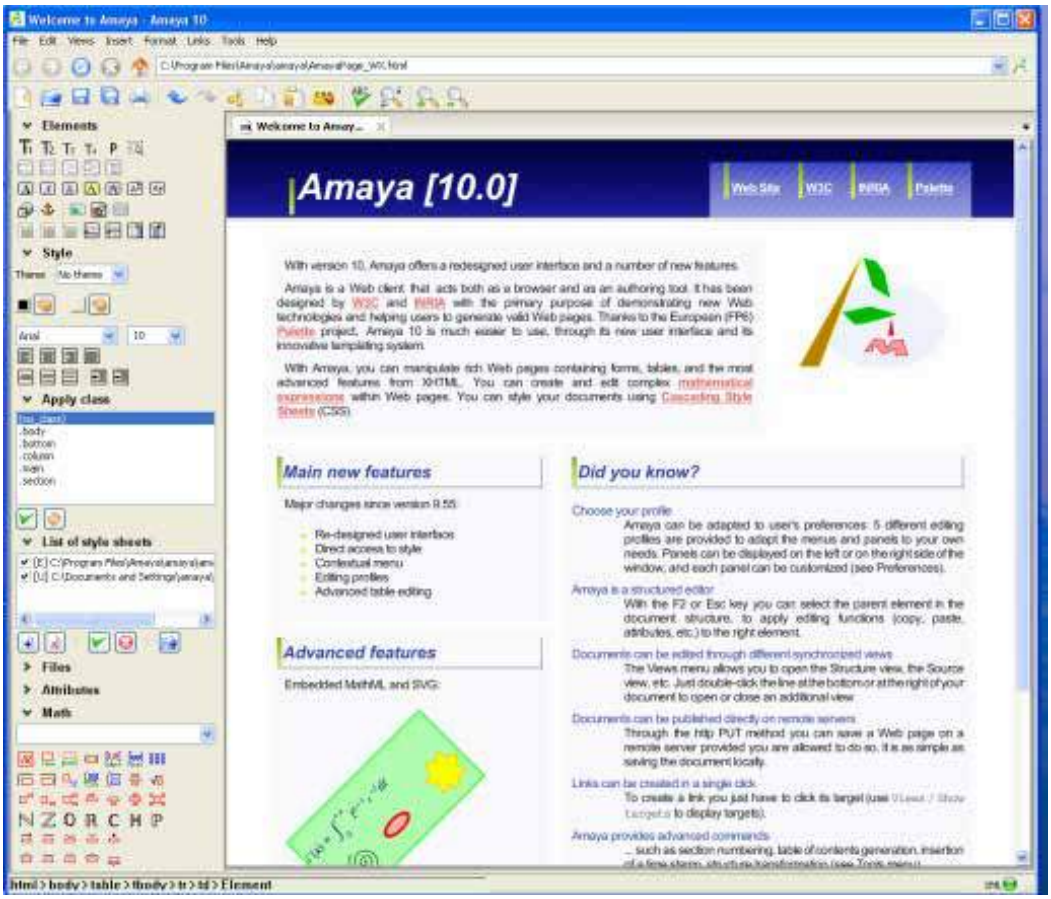
FIGURA 10 – NetBeans

FONTE: A autora

- Amaya: é um editor para *web*, ou seja, uma ferramenta usada para criar e atualizar documentos diretamente na *web*. Recursos de navegação são perfeitamente integrados com os recursos de edição e acesso remoto em um ambiente uniforme. Isso segue a visão original da *web* como um espaço de colaboração e não apenas um meio de publicação de uma via. A Amaya começou no W3C em 1996 para mostrar as tecnologias da *web* em um cliente da *web* cheio de recursos. A principal motivação para o desenvolvimento de Amaya era fornecer uma estrutura que podia integrar tantas tecnologias do W3C possível. Ele é usado para demonstrar estas tecnologias em ação, aproveitando a sua combinação em um único ambiente, consistente. Amaya começou como um editor de folhas de estilo CSS HTML +. Desde aquela época, foi estendido para suportar XML e um número crescente de aplicações XML, tais como a família XHTML, MathML e SVG. Ele permite que todos os vocabulários possam ser editados simultaneamente em documentos compostos. Amaya inclui um aplicativo de anotação colaborativa baseada em *Resource Description Framework* (RDF), XLink e XPointer. Amaya é um projeto de *software* de código aberto oferecido pelo W3C. Você pode contribuir de várias formas (documentação, tradução, escrita de código, corrigir *bugs*, portar para outras plataformas etc. O *software* Amaya é escrito em C e está disponível para plataformas Windows, Unix e MacOS X. Amaya está na versão 11.4.4. Ele suporta HTML 4.01, XHTML 1.0, XHTML Basic, XHTML 1.1, HTTP 1.1, MathML 2.0, muitos CSS 2 características, e SVG e agora inclui um editor de SVG (para um subconjunto

da língua). Você pode exibir e editar documentos XML parcialmente. É uma aplicação internacionalizada. Ele fornece uma interface de usuário avançada com menus contextuais, um conjunto personalizável de *menus* e ferramentas, temas pré-definidos. As distribuições estão disponíveis para Linux, Windows e MacOS X agora PowerPC e Intel. Esta versão oferece um modelo de apoio parcialmente financiado pelo 6º Programa Quadro da Comissão Europeia no âmbito do projeto Palette (W3C, 2014).

FIGURA 11 – Amaya



FONTE: A autora

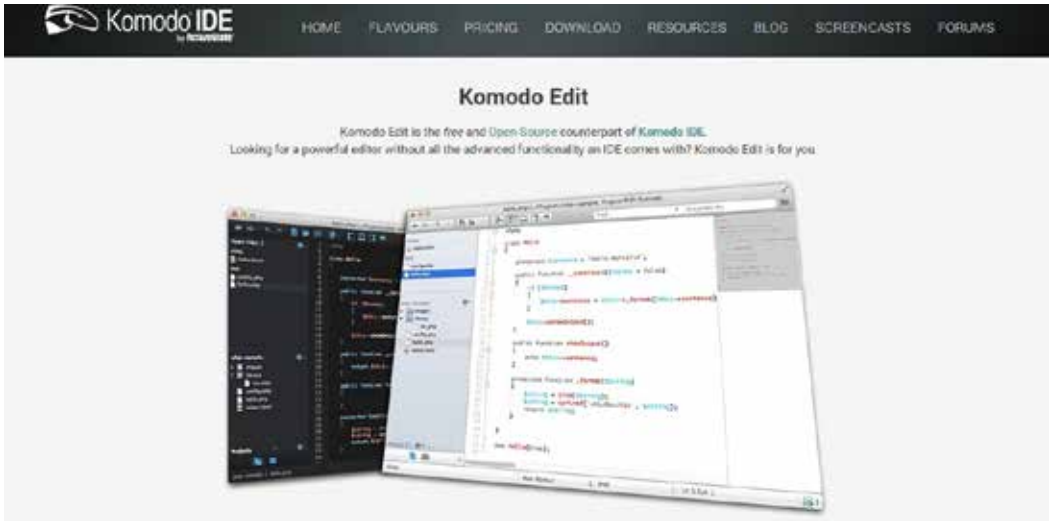
- Aptana Studio 3: é uma ferramenta profissional de desenvolvimento *open source* para a *web*. Pode-se desenvolver e testar toda a sua aplicação *web* utilizando um único ambiente. Com suporte para as mais recentes especificações de tecnologia de navegador, como HTML5, CSS3, JavaScript, Ruby, Rails, PHP e Python. Inclui informações sobre o nível de suporte para cada elemento nos principais navegadores. Define pontos de interrupção, inspeciona variáveis e execução de controle. Os depuradores integrados de Ruby e Rails e JavaScript ajuda a resolver os *bugs*.

FIGURA 12 – Aptana

FONTE: A autora

- Komodo: é um editor de texto para escrever e editar código em diversas sintaxes. Suporta as linguagens Python, PHP, Ruby, Perl, Tcl, NodeJS, HTML, CSS e JavaScript. O programa conta com identação automática e destaque para diferentes termos da sintaxe, conforme a linguagem. Este programa também traz alguns exemplos de código em algumas linguagens de programação suportados pela ferramenta.

FIGURA 13 – Komodo IDE

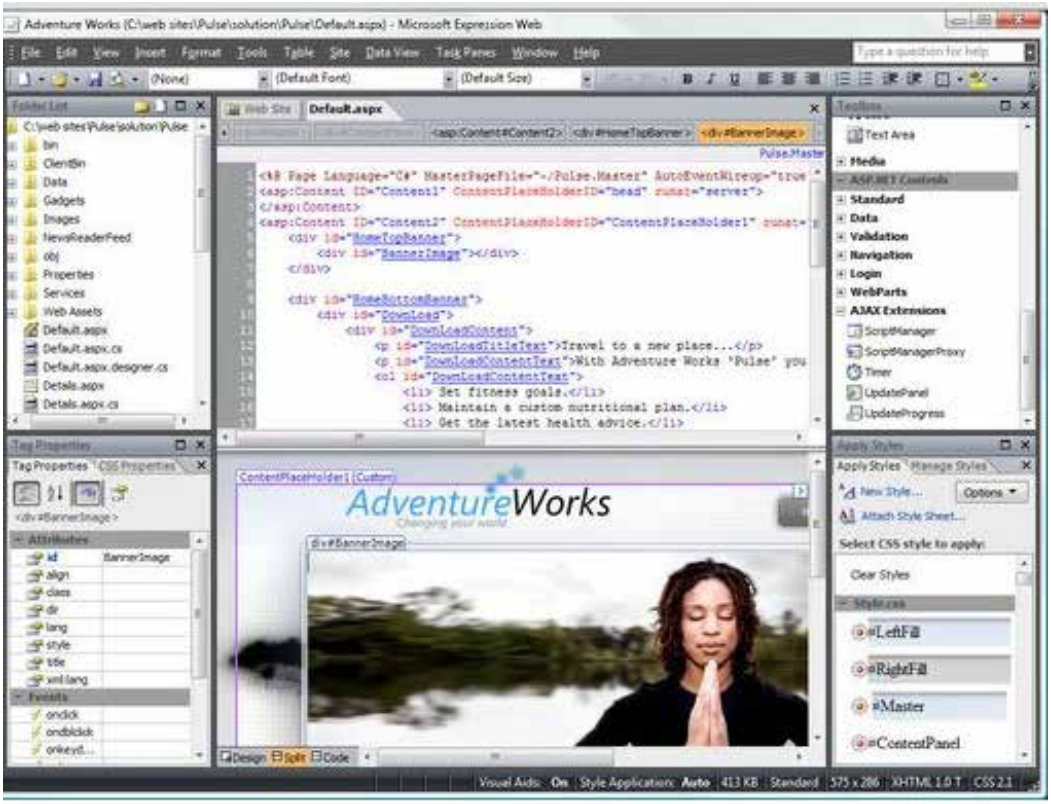


FONTE: A autora

- MS *Expression Web*: A proliferação de ricas aplicações interativas *webs* através da nuvem e dispositivos móveis continua a criar novas oportunidades para a concepção e desenvolvimento criativo. À medida que essas tecnologias evoluem, a Microsoft está comprometida em fornecer ferramentas para criar

aplicativos modernos. Em apoio a estas tendências da indústria a Microsoft consolidou um projeto de desenvolvimento para oferecer aos clientes uma solução unificada que reúne o melhor da *web* e padrões de desenvolvimento modernos. Esta mistura para o Visual Studio 2013 oferece um ambiente rico centrado no *design* para a criação de aplicativos para a Windows Store, Windows Phone, WPF e Silverlight.

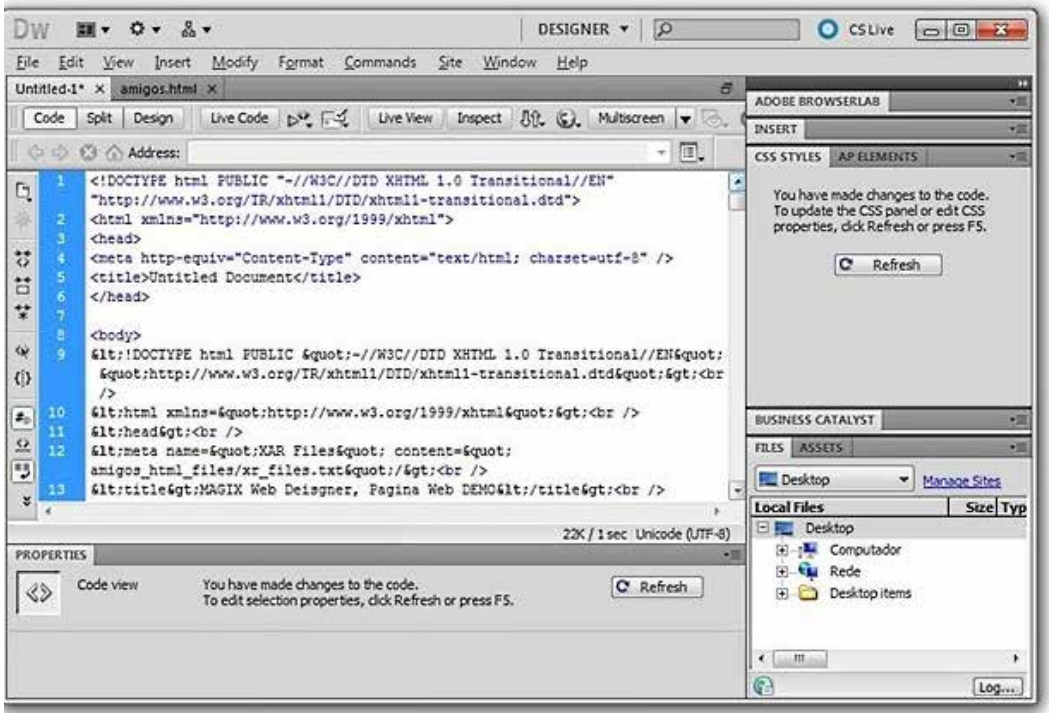
FIGURA 14 – MS Expression Web



FONTE: A autora

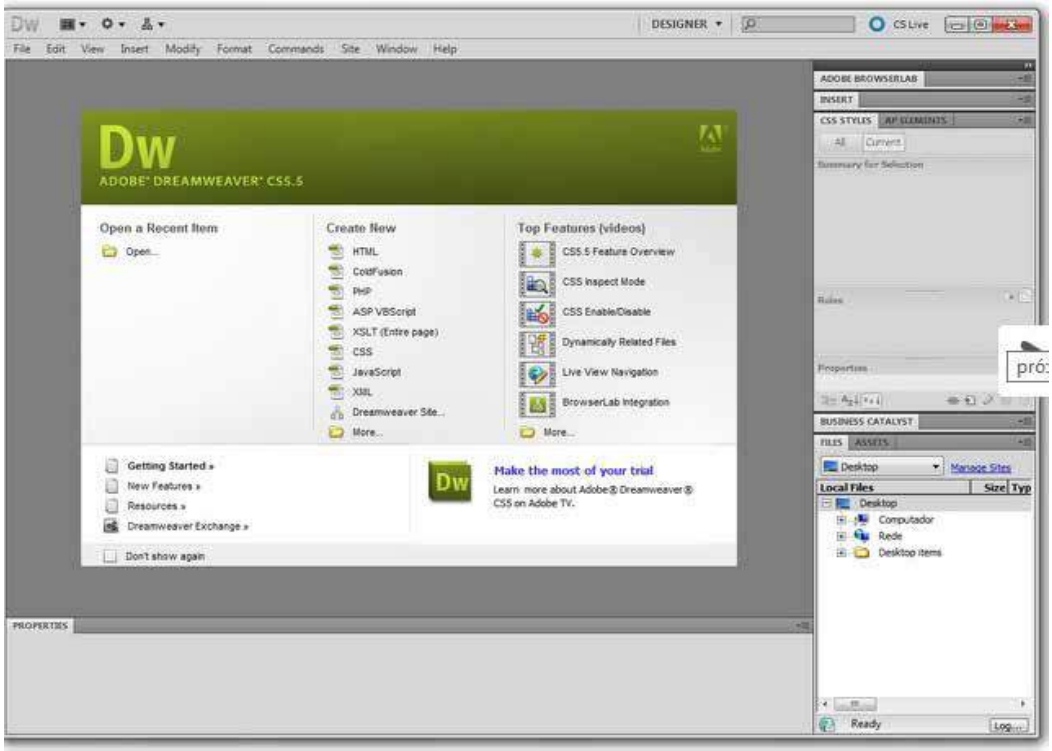
- *Dreamweaver*: é um dos programas mais populares de *web design* disponível. Ele possui vários recursos. É um editor de código e editor WYSIWYG para Windows e Macintosh. Editor WYSIWYG são ferramentas que disponibilizam ao desenvolvedor o sistema de "arrastar e soltar" para adicionar controles, imagens, formulários e outra infinidade de controles, tudo de forma visual. Com o *Dreamweaver* pode-se programar em HTML, CSS, JSP, XML, PHP, JavaScript e outros. Inclui um sistema de rede para fazer *layouts* de resposta baseado em *grid* para três tamanhos de dispositivos diferentes ao mesmo tempo. Além disso, o *Dreamweaver* oferece uma série de ferramentas para fazer o desenvolvimento de *web* móvel, incluindo a criação de aplicativos nativos para dispositivos iOS e Android.

FIGURA 15 – TELA PRINCIPAL *DREAMWEAVER*



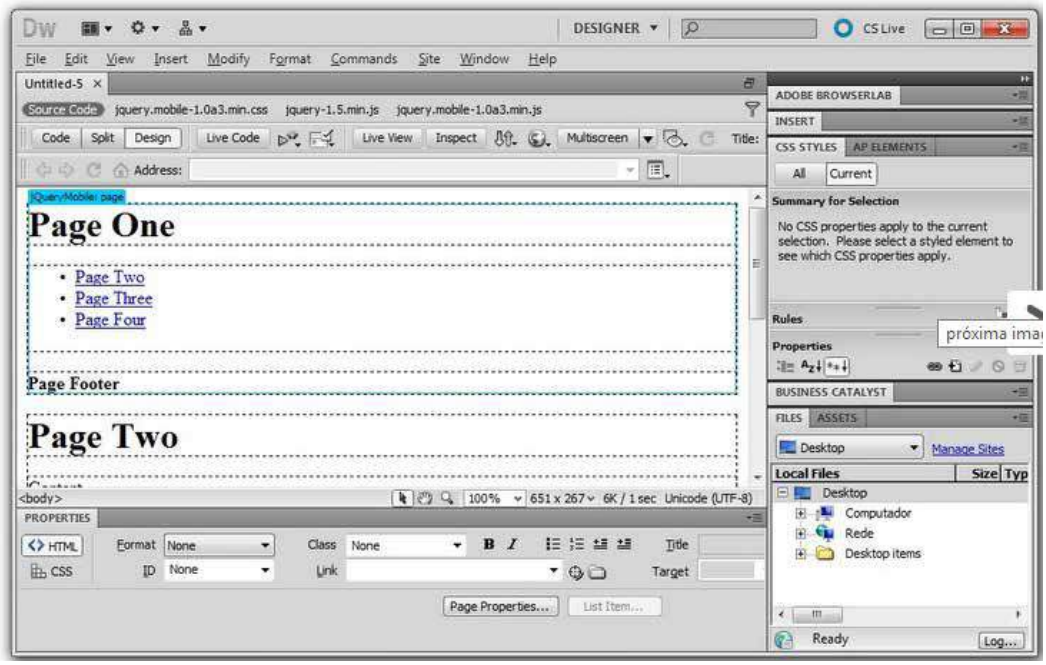
FONTE: A autora

FIGURA 16 – ESCOLHENDO A LINGUAGEM NO *DREAMWEAVER*



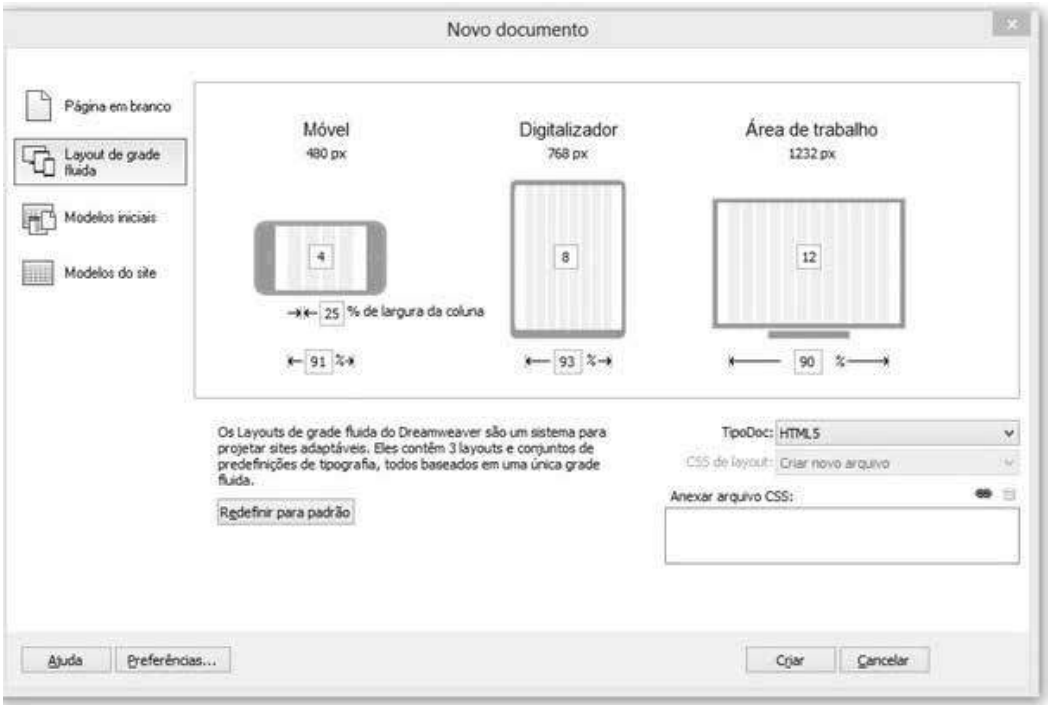
FONTE: A autora

FIGURA 17 – DESENVOLVENDO PARA MÓVEL



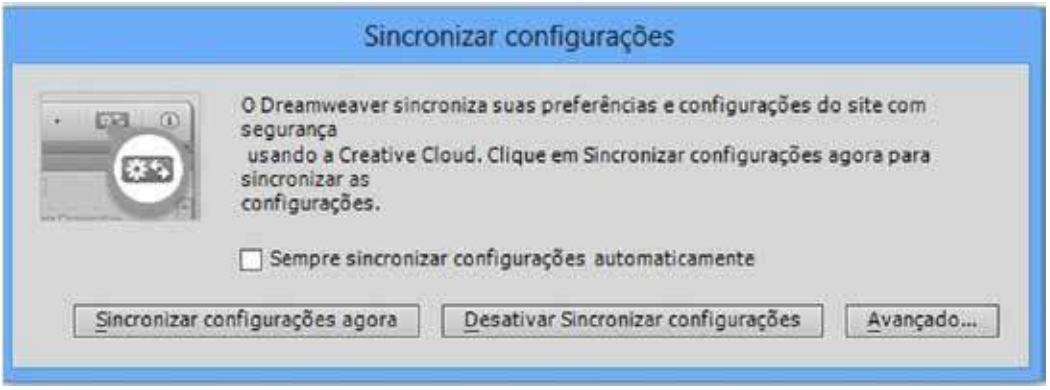
FONTE: A autora

FIGURA 18 – ESCOLHENDO O *LAYOT*



FONTE: A autora

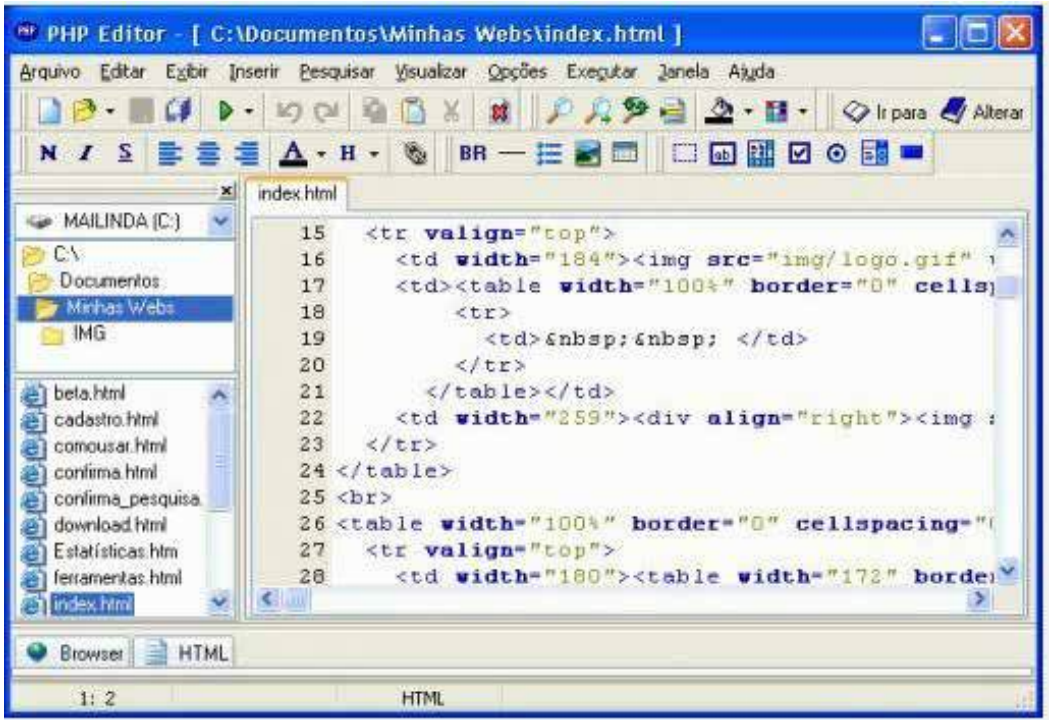
FIGURA 19 – SINCRONIZANDO CONFIGURAÇÕES



FONTE: A autora

- PHP Editor: é um programa para desenvolvimento em PHP, porém também é possível programar linguagens como HTML, SQL, Java, JavaScript, XML, Perl, C++, Python, CSS, e outros. Além de ser um programa gratuito, permite trabalhar com vários projetos ao mesmo tempo, inclui um visualizador de páginas e correção de erros.

FIGURA 20 – PHP EDITOR



FONTE: A autora

3 LINGUAGENS DE PROGRAMAÇÃO

Conforme Dzendzik (2005), podemos utilizar programação para páginas dinâmicas através de consulta de banco de dados. Vamos conhecer alguns recursos disponíveis na área de desenvolvimento de aplicações de páginas dinâmicas para a *web*, como ASP, PHP e JSP.

3.1 ASP

ASP (*Active Server Pages*) é um conjunto de ferramentas de desenvolvimento *web* oferecidas pela Microsoft. Programas como o *Visual Studio* e *Visual Web Developer* permitem que os desenvolvedores da *web* criem *sites* dinâmicos usando uma interface visual. É claro que os programadores podem escrever seus próprios códigos e *scripts* e incorporá-lo em *sites* ASP.NET. Embora muitas vezes é visto como um sucessor à tecnologia de programação da Microsoft ASP, ASP.NET também suporta Basic.NET Visual, JScript e linguagens de código aberto, como Python e Perl.

ASP.NET é construído sobre a estrutura do NET, que fornece uma interface de programação de aplicativo (API) para programadores de *software*. As ferramentas de desenvolvimento NET podem ser usadas para criar aplicativos para o sistema operacional Windows e Internet. Programas como o Visual Studio fornece uma interface visual para desenvolvedores criarem suas aplicações, o que torna NET uma escolha razoável para *design* de interfaces baseadas na *web* também.

Para que um *site* ASP.NET funcione corretamente, ele deve ser publicado em um servidor *web* que suporta aplicações ASP.NET. Servidor da *Web* do *Microsoft Internet Information Services* (IIS) é, de longe, a plataforma mais comum para *sites* ASP.NET. Enquanto há algumas opções de código aberto disponíveis para sistemas baseados em Linux, essas alternativas costumam oferecer menos suporte completo para aplicações ASP.NET.

Vejamos um exemplo simples em ASP:

```
<html>
<body>

<%
response.write("Meu primeiro script em ASP!")
%>

</body>
</html>
```

O resultado será:
Meu primeiro *script* em ASP!

3.2 PHP

Esta linguagem teve início em 1995, foi criada pelo sueco Rasmus Lerdorf. Logicamente que naquela época o PHP não era da forma que o conhecemos hoje. Rasmus criou vários *scripts* em Perl para serem usados em seu *sit*e, onde mostrava seu currículo. Além disso este *script* tinha um contador de visitas. Como eram apenas ferramentas ele nomeou de PHP (*Personal Home Page tools*).

Com o passar o tempo as visitas aumentaram e Rasmus começou a desenvolver ferramentas mais poderosas e para isso usou a linguagem C. Ele liberou as ferramentas para quem quisesse usar e aprimorar. Como recebeu muitas contribuições, ele lançou a versão 2 e se aliou a mais dois programadores que melhoraram para uma linguagem *script* para criação de conteúdo dinâmico.

O PHP é uma linguagem *script* onde sua execução é feita no lado servidor, ou seja, não necessita ser compilado pra ser executado. Assim, é necessário um servidor que o interprete. Todo código PHP é transformado em código HTML para ser enviado ao usuário que fez a solicitação.

PHP é um *software* livre, em que qualquer pessoa pode ver seu código e modificá-lo (COSTA; TODESCHINI, 2006).

Vejamos um exemplo de PHP:

```
<html>
<body>

<? php
echo "Meu primeiro script PHP!";
?>

</ body>
</ html>
```

O resultado será:
Meu primeiro *script* PHP!

3.3 JSP

Java Server Pages (JSP) é uma tecnologia de programação do lado do servidor que permite a criação de método dinâmico, independentemente da plataforma para criação de aplicativos baseados na *web*. JSP tem acesso a toda a família de APIs Java, incluindo a API JDBC para acessar bancos de dados corporativos.

A tecnologia JSP é usada para criar aplicativos *web* como tecnologia *Servlet*. Ele pode ser pensado como uma extensão para o *servlet*, pois fornece mais funcionalidade que *servlet* como linguagem de expressão, JSTL etc. Uma página JSP consiste de *tags* HTML e *tags* JSP. As páginas JSP são mais fáceis de manter do que *servlet* porque podemos separar concepção e desenvolvimento.

Vejamos um exemplo:

```
<html>
<body>

<%
String mensagem = "Meu primeiro script JSP!";
%>
<% out.println(mensagem); %>
%>
</body>
</html>
```

O resultado será:
Meu primeiro *script* JSP!

4 LINGUAGENS DE PROGRAMAÇÃO *CLIENT SIDE*

Conforme Dzendzik (2005):

> As linguagens de programação para a *web* são os *scripts* que compõem os códigos ou sequências de códigos, que formam os arquivos existentes em um *website*. Algumas linguagens são interpretadas na máquina cliente (*Client Side*) onde o conteúdo gerado é exibido conforme os recursos disponíveis em cada navegador.

Vejamos a seguir algumas linguagens de programação cliente *side*:

4.1 CSS

CSS foi desenvolvido pela primeira vez em 1997, como uma maneira para os desenvolvedores da *web* para definir a aparência de suas páginas *web*. Destinava-se a permitir que os desenvolvedores, para separar o conteúdo de projeto de modo que o HTML pudesse executar mais de uma função sem se preocupar com o *design* e *layout*.

Um arquivo CSS (*Cascading Style Sheet*) permite que você separe seus *sites* do conteúdo HTML. Você usa seu arquivo HTML para organizar o conteúdo, mas toda a apresentação (fontes, cores de fundo, bordas, formatação de texto, efeitos de ligação e assim por diante...) são realizadas dentro de um CSS.

Primeiro vamos explorar o método interno. Desta forma, você está simplesmente colocando o código CSS dentro do <head> </ head> de cada arquivo HTML que você deseja estilo com o CSS. As instruções de CSS são inseridas entre as *tags* <STYLE > e </STYLE>. Basta que se insira uma vez a *tag* no código para que toda a página responda às instruções. O formato para isso é mostrado no exemplo a seguir.

```
<head>
<title> <title>
<style type = "text / css">
Conteúdo CSS vai aqui
</ style>
</ head>
<body>
```

Com este método cada arquivo HTML contém o código CSS necessário para o estilo da página. O que significa que todas as mudanças que você quer fazer para uma página, terão que serem feitas para todos. Este método pode ser bom se você precisa de estilo para apenas uma página, ou se você quiser páginas diferentes para ter estilos variados.

4.2 HTML

Hyper Text Markup Language (HTML) – Linguagem de formatação de hipertexto é uma linguagem *script* muito utilizada para programação *web*. HTML é uma linguagem de marcação para descrever documentos *web* (páginas *web*). O HTML não precisa ser compilado, sendo seu código interpretado no momento em que o arquivo é aberto. Para trabalhar com o HTM se utiliza um editor de texto e um navegador (COSTA; TODESCHINI, 2006).

Vejamos um exemplo:

```
<! DOCTYPE html>
<html>
<body>

<h1> Meu primeiro título </ h1>
<p> O meu primeiro parágrafo. </ p>
</ body>
</ html>
```

Explicado o exemplo:
A declaração DOCTYPE define o tipo de documento.
O texto entre <html> e </ html> descreve o documento *web.*
O texto entre <body> e </ body> descreve o conteúdo da página visível.
O texto entre <h1> e </ h1> descreve um título.
O texto entre <p> e </ p> descreve parágrafo.

Usando a descrição, um navegador pode exibir um documento com um título e um parágrafo.

4.3 JAVASCRIPT

JavaScript é uma linguagem de objeto *script* da *Netscape*. Desenvolvida e usada em milhões de páginas *web* e aplicativos de servidor em todo o mundo.

Contrário à crença popular, JavaScript não é "Interpretativo de Java". Em poucas palavras, JavaScript é uma linguagem de *script* dinâmico baseada na construção do objeto. A sintaxe básica é intencionalmente semelhante ao Java e C ++ para reduzir o número de novos conceitos necessários para aprender a língua. Construções de linguagem, tais como *if*, *for* e *while* e *switch* entre outros funcionam da mesma forma que em outras linguagens (ou quase isso.)

JavaScript pode funcionar tanto como um procedimento e uma linguagem orientada a objetos. Os objetos são criados por meio de programação em JavaScript, anexando métodos e propriedades aos objetos de outro modo vazios em tempo de execução, ao contrário das definições de classes sintáticas comuns em linguagens compiladas como C ++ e Java. Uma vez que um objeto tenha sido construído, pode ser utilizado como um modelo (ou protótipo) para criar objetos semelhantes.

Capacidades dinâmicas do JavaScript incluem construção de objeto de tempo de execução, listas de parâmetros variáveis, variáveis de função, criação de roteiro dinâmico, objeto introspecção, e recuperação de código-fonte (programas em JavaScript podem decompor corpos das funções de volta em seu texto de origem).

5 SERVIDORES *WEB*

Servidor *web* é um programa executado em uma máquina classificada como servidora capaz de interpretar requisições recebidas de máquinas clientes. Os clientes fazem conexões com estes servidores através de uma porta preestabelecida usando um navegador. O servidor *web* recebe essas requisições, interpreta e devolve um resultado aos clientes.

5.1 APACHE

Conforme Apache (2014), em fevereiro de 1995, o *software* de servidor mais popular da *web* foi desenvolvido por Rob McCool, no Centro Nacional de Aplicações de Supercomputação da Universidade de Illinois, Urbana-Champaign. No entanto, o desenvolvimento desse httpd tinha parado depois de Rob deixar o NCSA em meados de 1994, e muitos *webmasters* desenvolveram suas próprias extensões e correções de *bugs* que estavam na necessidade de uma distribuição

comum. Um pequeno grupo desses *webmasters*, contatado via *e-mail* particular, se reuniram com a finalidade de coordenar as suas alterações. Brian Behlendorf e Cliff Skolnick montaram uma lista de discussão, espaço de informação compartilhada e *logins* para os desenvolvedores do núcleo em uma máquina na área da baía da Califórnia, com largura de banda doados por *HotWired*. Até o final de fevereiro, oito importantes colaboradores, formaram a base do grupo original Apache:

Brian Behlendorf
Roy T. Fielding
Rob Hartill
David Robinson
Cliff Skolnick
Randy TerBush
Robert S. Thau
Andrew Wilson
com contribuições adicionais de:

Eric Hagberg
Frank Peters
Nicolas Pioch

Usando NCSA httpd 1.3 como base, foram adicionadas todas as correções publicadas e melhorias, testaram o resultado em seus próprios servidores, e fizeram o primeiro lançamento público oficial (0.6.2) do servidor Apache em Abril de 1995. Por coincidência, NCSA havia reiniciado o seu próprio desenvolvimento, no mesmo período, e Brandon Long e Beth Frank da equipe de desenvolvimento do servidor NCSA entraram para a lista em março como membros honorários para que os dois projetos pudessem compartilhar ideias e correções.

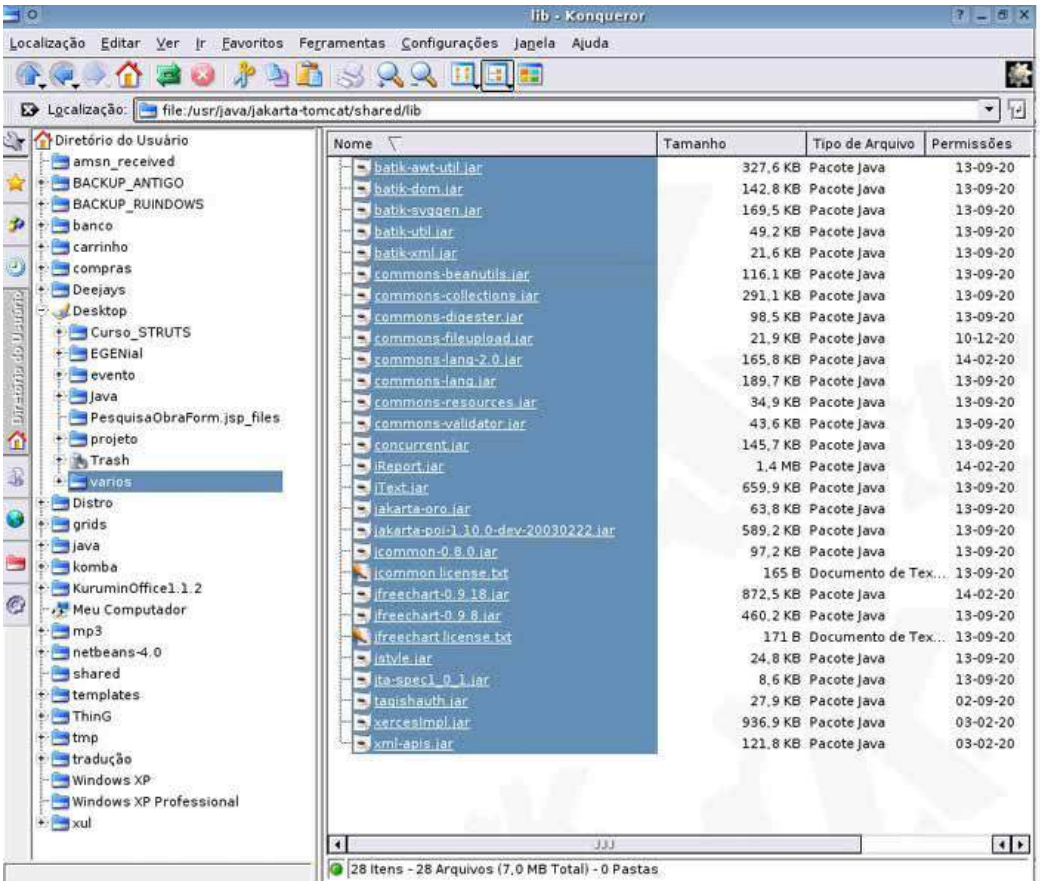
Durante maio e junho de 1995, enquanto Rob Hartill e o resto do grupo centrado na implementação de novas funcionalidades para 0.7.x apoiaram o rápido crescimento da comunidade de usuários do Apache, Robert Thau projetou uma nova arquitetura de servidor, que incluiu uma estrutura modular e API para uma melhor capacidade de extensão, alocação de memória, e um modelo adaptativo do processo de pré-bifurcação. O grupo ligado a esta nova base de servidores em julho adicionou as características de 0.7.x, resultando em Apache 0.8.8 (e seus irmãos) em agosto.

Após extensos testes beta, muitos portos para obscurecer as plataformas, um novo conjunto de documentação (por David Robinson), e a adição de muitos recursos, sob a forma de módulos padrão, o Apache 1.0 foi lançado em 1º de dezembro de 1995.

Menos de um ano depois que o grupo foi formado, o servidor Apache passou da NCSA como o servidor # 1 na internet e de acordo com a pesquisa da *Netcraft*, mantém essa posição hoje.

Em 1999, os membros do Grupo Apache formaram a Apache *Software Foundation* para fornecer suporte organizacional, legal e financeiro para o Apache HTTP *Server*. A fundação colocou o *software* em uma base sólida para o desenvolvimento futuro, e expandiu enormemente o número de projetos de *software* de código aberto, que estão sob o guarda-chuva da Fundação.

FIGURA 21 – TELA APACHE



FONTE: A autora

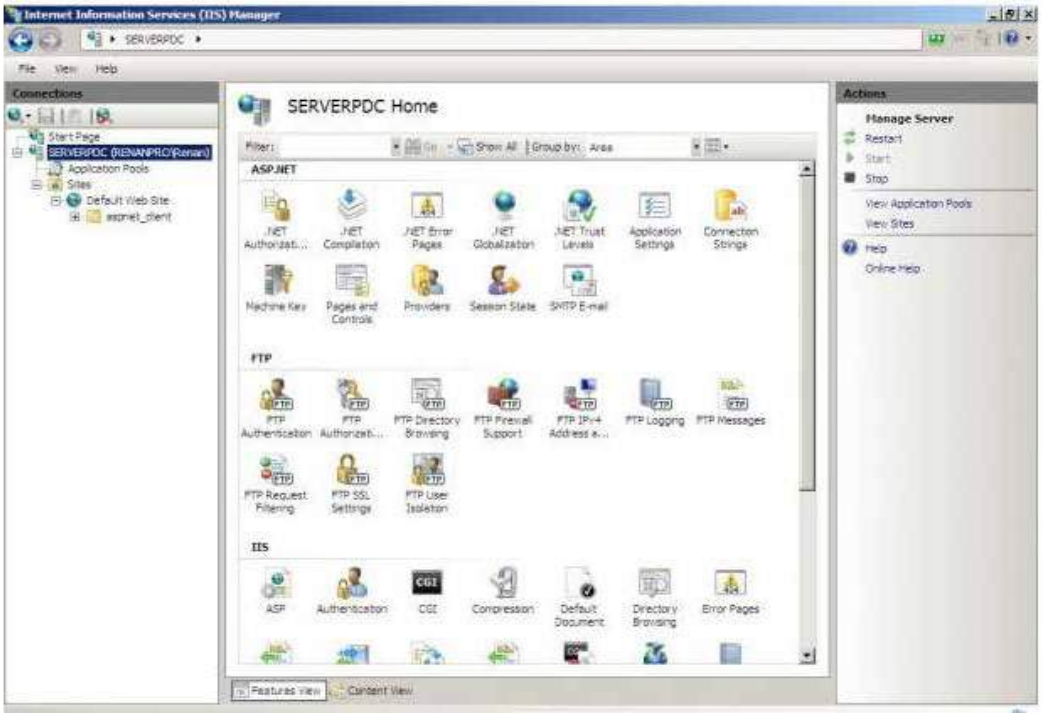
5.2 IIS

Internet Information Services (IIS) para o *Windows Server* é um servidor *web* flexível, seguro e gerenciável para a realização de qualquer coisa na *web*. De *streaming* de mídia para aplicações *web*, arquitetura aberta e escalável do IIS está pronto para lidar com as tarefas mais exigentes. Algumas características do IIS:

- implantar e gerenciar *sites* da *web* e aplicações em grandes servidores *web* a partir de um local central;
- gerenciamento remoto;
- configuração e gerenciamento usando um conjunto de linha de comando;

- interface do usuário e APIs programáticas;
- melhora o desempenho, permitindo o armazenamento em *cache* dinâmico de alta velocidade e compressão;
- encontra e corrige problemas de forma rápida e fácil com poderosas ferramentas de diagnóstico;
- publica conteúdo na *web* com mais segurança usando protocolos baseados em padrões;
- proteje o servidor de acesso não autorizado; e
- desenvolve e implementa aplicações ASP.NET e PHP no mesmo servidor (Microsoft, 2014).

FIGURA 22 – TELA IIS



FONTE: A autora

LEITURA COMPLEMENTAR

DESENVOLVENDO JOGOS PARA CELULARES COM O *GAME* DO *NETBEANS*

Ricardo Ogliari

Atua no mundo *mobile* há 7 anos. Bacharel em Ciência da Computação. Pós-Graduado em *web*: Estraté©gias de Inovação e Tecnologia.

Robison Cris Brito.

Mestre em Engenharia Elétrica e Informática Industrial pela Universidade Tecnológica Federal do Paraná, onde ministra aulas sobre tecnologia Java, Computação Móvel e Sistemas Distribuídos.

De que trata o artigo: O artigo apresenta o *Game Builder*, uma ferramenta do *Netbeans* que permite desenvolver jogos complexos para celular utilizando a tecnologia JavaME de forma rápida e fácil. Para um melhor entendimento, esse artigo também apresenta os conceitos básicos sobre criação de jogos para a plataforma MIDP 2.0.

Para que serve: O artigo pode ser utilizado por desenvolvedores de jogos e por programadores JavaME que desejam aprimorar seus conhecimentos sobre desenvolvimento de jogos para celulares.

Em que situação o tema é útil: Na construção rápida de jogos com Java ME para aparelhos celulares.

Hoje em dia é comum a utilização de IDEs para agilizar o desenvolvimento de aplicativos utilizando a linguagem de programação Java. Essas ferramentas agilizam a criação de *software* para várias plataformas, tais como internet, *desktop* ou *mobile*, automatizando processos básicos para a construção e distribuição de aplicativos, como a codificação de rotinas básicas, a formatação de códigos, sugestões para correções de erros etc.

Nesse contexto, duas IDEs livres se destacam, são elas: a IDE Eclipse (www.eclipse.org) e a IDE *NetBeans* (www.netbeans.org), sendo o *Netbeans* uma das IDEs preferidas pelos iniciantes no desenvolvimento de aplicativos móveis, pois possui o *Mobility Pack*, sendo este um conjunto de recursos que auxilia no desenvolvimento de aplicativos para dispositivos móveis. O *Mobility Pack* foi inserido na IDE *Netbeans* a partir da versão 5.0.

Com ele, é possível utilizar o *Sun Wireless Toolkit* (*kit* de desenvolvimento para celular fornecido pela *Sun)* no próprio ambiente do *Netbeans*, sendo possível também clicar e arrastar componentes visuais na tela do celular e realizar a depuração do projeto.

Com o passar das versões do *Netbeans*, novos recursos foram acoplados ao *Mobility Pack*, sendo que um dos mais recentes e que chama bastante a atenção é o *Game Builder*. Este recurso permite a construção da interface gráfica de jogo para dispositivos móveis de forma visual. Dessa maneira, o programador pode interagir em alto nível com seus cenários e personagens, ou seja, graficamente e através de *wizards*. Assim é possível dedicar mais tempo na lógica do jogo que continua sendo desenvolvida através das linhas de código.

Para o desenvolvimento da interface gráfica, o *Game Builder* utiliza o MIDP 2.0, o qual possui uma biblioteca específica para o desenvolvimento de jogos, chamada *Game* API. Para a utilização do *Game Builder*, é desejável um prévio conhecimento sobre as APIs MIDP/CLDC, bem como alguma experiência no desenvolvimento de aplicativos móveis com o *Mobility Pack* do *Netbeans*, lembrando que o *Game Builder* está disponível a partir da versão 6.0 do *Netbeans*.

O objetivo desse artigo é apresentar brevemente a *Game* API e as vantagens de sua utilização. Posteriormente será apresentado o *Game Builder*, suas telas e recursos e, ao final do texto, será apresentado um exemplo didático de construção de jogo para aparelhos celulares. O *game* a ser construído apresenta somente duas fases de um jogo fictício, onde um personagem deve circular por dois cenários diferentes.

Devido ao objetivo didático sobre a *Game Builder*, não vamos detalhar o jogo e programar todos os agentes envolvidos. Então, primeiramente o jogador se encontra no primeiro estágio (chamado *Forest*), quando o personagem atingir o canto inferior direito, será enviado ao segundo estágio, que seria o cenário chamado *Scene Desert*.

Jogos em JavaME e a *Game* API

Desde as primeiras versões da plataforma JavaME, era possível utilizar os recursos do MIDP/CLDC para desenvolver jogos para celulares, como por exemplo jogos de arcade, de estratégia, *puzzle*, RPG, esportes etc. Porém, a MIDP 1.0, perfil que era utilizado nas primeiras versões do JavaME, possuía muitas deficiências para a manipulação avançada da interface gráfica do celular. Por exemplo, os personagens de um jogo eram desenvolvidos através da codificação pura, não existindo classes que implementassem características comuns, como o tamanho do personagem, a imagem do mesmo, sua posição na tela, dentre outras. Com isso, alguns fabricantes, como a Nokia, por exemplo, criaram suas próprias bibliotecas de jogos para suprir estas falhas. Essa ação resultou na perda de portabilidade dos aplicativos, sendo que aplicativos desenvolvidos com as APIS de um fabricante funcionariam apenas em aparelhos compatíveis (da mesma marca).

A *Game* API surgiu para suprir essa deficiência. Ela foi disponibilizada a partir da versão 2.0 do MIDP e possui uma biblioteca de classes poderosa para desenvolvimento de jogos. Além disso, com as padronizações advindas com a *Game* API, até o tamanho dos jogos gerados pelo JavaME ficaram menores, pois alguns códigos, como por exemplo os que gerenciam os personagens e cenários, estão inclusos na biblioteca de classe, que são implementadas nos dispositivos.

Dentre os novos recursos oferecidos pela *Game* API, destacam-se:

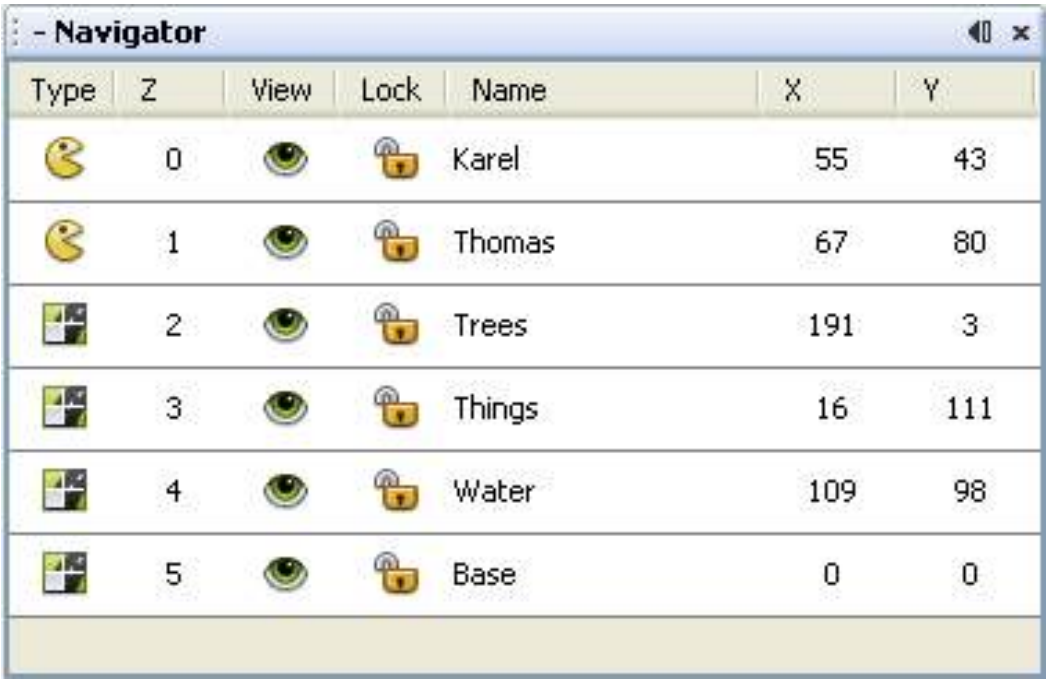
- GameCanvas: É uma evolução da classe Canvas, sendo projetada especificamente para o desenvolvimento de jogos. Dentre as melhorias, podemos destacar que com GameCanvas é possível controlar todo o jogo com apenas uma *Thread*, e existem algumas facilidades para captura de evento de teclado. Outra vantagem da GameCanvas está na atualização da tela do jogo, que só irá acontecer após a chamada do método flushGraphics(). Essa técnica faz com que toda a tela do jogo seja atualizada de uma só vez, e não a cada alteração da tela (ex.: primeiro se altera a posição de todos os personagens do jogo e depois executa o método flushGraphics(), que apresentará a nova tela). Dessa maneira, as imagens são apresentadas de forma mais suave para o usuário. Outra mudança significativa é a captura de eventos de teclado. Na MIDP 1.0 isso era feito em uma *Thread* separada. Na *Game* API isso é feito através do método getKeyStates. Além disso, este método permite a captura de teclas simultâneas (mais de uma tecla pressionada ao mesmo tempo) e de teclas que são mantidas pressionadas.

- Sprite: A classe *Sprite* é um elemento gráfico do jogo, geralmente utilizada para os personagens. Essa classe possui vários métodos que facilitam o trabalho do desenvolvedor, como para verificar colisões, métodos para transformação de uma imagem e para visualização de uma sequência de imagens (ideia de animação). A Figura 1 apresenta os dois *Sprites* que serão utilizados pelo programa de exemplo desse artigo.

FIGURA 1 - SPRITES KAREL E THOMAS, PERSONAGENS DO APLICATIVO A SER DESENVOLVIDO NESSE ARTIGO

- LayerManager: Essa classe é de fundamental importância para a criação dos cenários. É ela quem obtém e gerencia todos os componentes visuais do jogo, organizando estes elementos em camadas (*layers*). Ao ser inserido no LayerManager, o componente visual recebe uma prioridade (representada pela letra z), assim os elementos gráficos com maior prioridade poderão sobrescrever os com menor prioridade se necessário. A Figura 2 apresenta a organização dos elementos do jogo no LayerManager. Nessa figura (extraída do *NetBeans*) são apresentados os tipos dos componentes (Sprite ou Tiled Layer), a prioridade (z), se o mesmo será apresentado na tela, se está bloqueado para movimentação (evitar movimentações acidentais do componente na tela), o nome e as posições x e y.

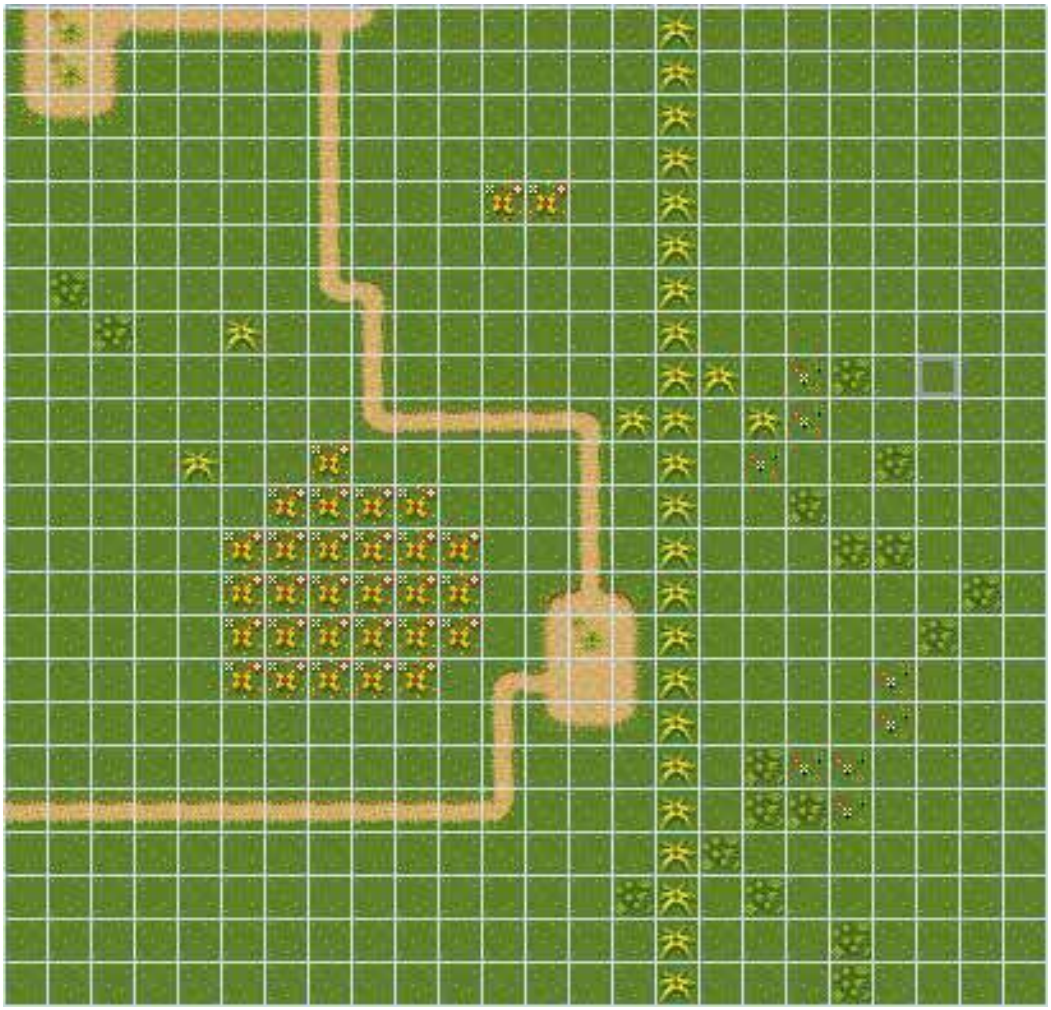
FIGURA 2 - REPRESENTAÇÃO GRÁFICA DO LAYERMANAGER, APRESENTANDO OS PERSONAGENS E ALGUMAS CARACTERÍSTICAS DELES



- TiledLayer: A classe TiledLayer é usada para construção de cenários de jogos. Basicamente é utilizada para montar imagens a partir de subimagens, ou seja, através de uma repetição de pequenas imagens é possível criar um cenário maior e mais complexo. A Figura 3 apresenta o TiledLayer referente ao fundo do primeiro estágio do jogo.

[abrir imagem em janela]

FIGURA 3 - TILEDLAYER DA PRIMEIRA FASE DO JOGO DESENVOLVIDO NO ARTIGO

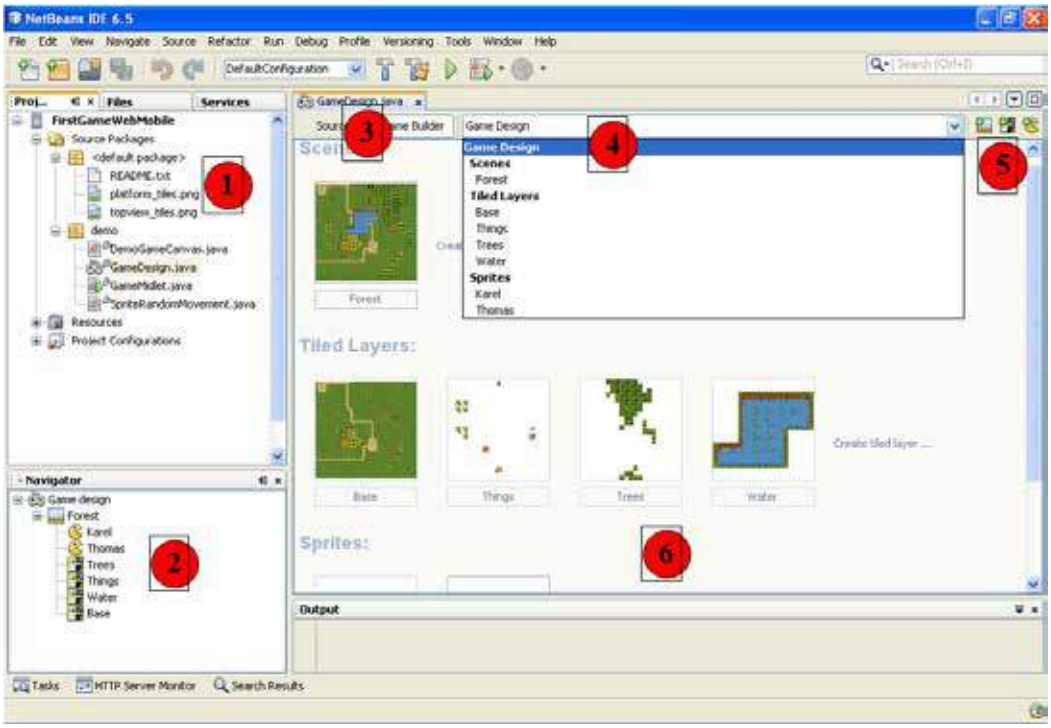


Utilizando o *Game Builder*

Para iniciar na utilização do *Game Buider*, será utilizado um exemplo pronto oferecido pela própria IDE *Netbeans*. Para isso, deve-se selecionar o menu File -> New Project... Deve-se selecionar a categoria Java ME (MIDP), que é uma subcategoria de *Samples*. Em seguida, em *Projects* devemos selecionar *Simple Game Created With Game Builder*. Ao pressionar o botão *Next*, será solicitado o nome do projeto (para o exemplo foi utilizadoFirstGameWebMobile) e o local onde o mesmo será armazenado. Novamente, ao se pressionar o botão *Next* será apresentada a última tela do *Wizard*, onde serão solicitadas informações como a versão do MIDP/CLDC desejada. Para finalizar o projeto, deve-se clicar no botão de *Finish*. Uma tela semelhante à Figura 4 é apresentada.

[abrir imagem em janela]

FIGURA 4 - TELA PRINCIPAL DE UM PROJETO CRIADO COM O *GAME BUILDER*



À esquerda (1), é apresentada a paleta *Projects*, onde estão disponíveis os arquivos do projeto. Como pode ser visualizado, o projeto é composto por dois arquivos de imagens (plataform_tiles.png e topview_tiles.png) e quatro classes que se encontram dentro do pacote demo. A classe responsável pelo desenvolvimento da interface visual do jogo é a GameDesign. Dando dois cliques sobre esse arquivo, é exibido o ambiente de desenvolvimento visual (6). Para alternar entre o desenvolvimento visual e o código gerado pelo *Game Builder*, são utilizados os botões *Source* e *GameBuilder* (3).

Também é apresentada uma janela de navegação (2), na qual é possível alternar entre os elementos visuais do jogo. Esse exemplo de jogo gerado pelo *Netbeans* possui dois personagens (*Sprites*) com nomes Karel e Thomas, e quatro TiledLayers, os quais representam partes dos cenários do jogo.

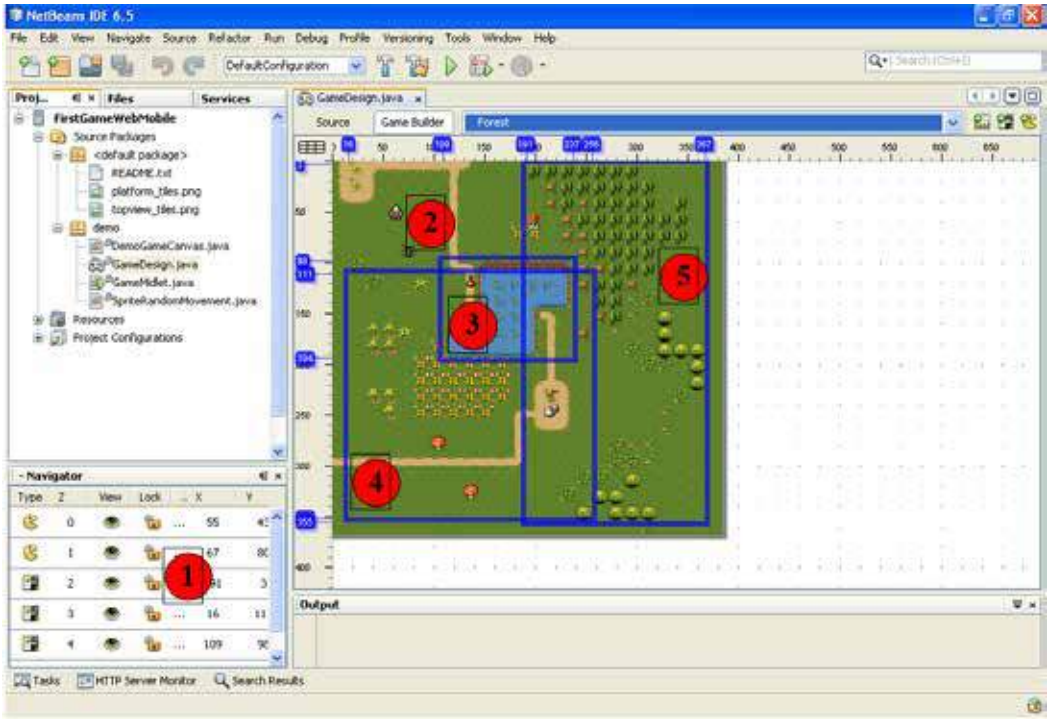
O *Base* é o cenário propriamente dito, *Things* são os objetos fixos existentes no cenário, *Threes* representam as árvores que serão adicionadas ao cenário e *Water* representa um lago. Basicamente, os TiledLayesr, com exceção da base, são os obstáculos que não poderão ser ultrapassados pelos personagens. Nesse exemplo, todos os TiledLayers fazem parte de um cenário, que pode ser considerado um estágio do jogo. Para facilitar a navegação, pode também ser utilizado o ComboBox da parte superior (4), possuindo ao seu lado botões que

permitem adicionar novos cenários, TiledLayers e Sprites (5). A janela central (6) também apresenta uma lista com todos os componentes do jogo. Ao clicar em cima de um componente, é apresentada a tela de edição, podendo retornar à tela inicial escolhendo a opção GameDesign no ComboBox (4).

Ao selecionar a cena *Forest* (primeira fase do jogo – Figura 5), temos uma instância do *Layer Manager* (1) com todos os componentes visuais da tela (conforme exibido na Figura 2).

[abrir imagem em janela]

FIGURA 5 - *LAYOUT MANAGER* DO CENÁRIO *FOREST*



Ao centro, visualiza-se a posição inicial dos Sprites (2), bem como dos TiledLayers (3) (4) (5). Como existem réguas horizontal e vertical, visualiza-se a posição inicial e final em pixel de cada componente, bem como o tamanho do cenário. Como a tela é maior que a resolução da maioria dos celulares (384x368), é evidenciada a utilização de paginação no cenário.

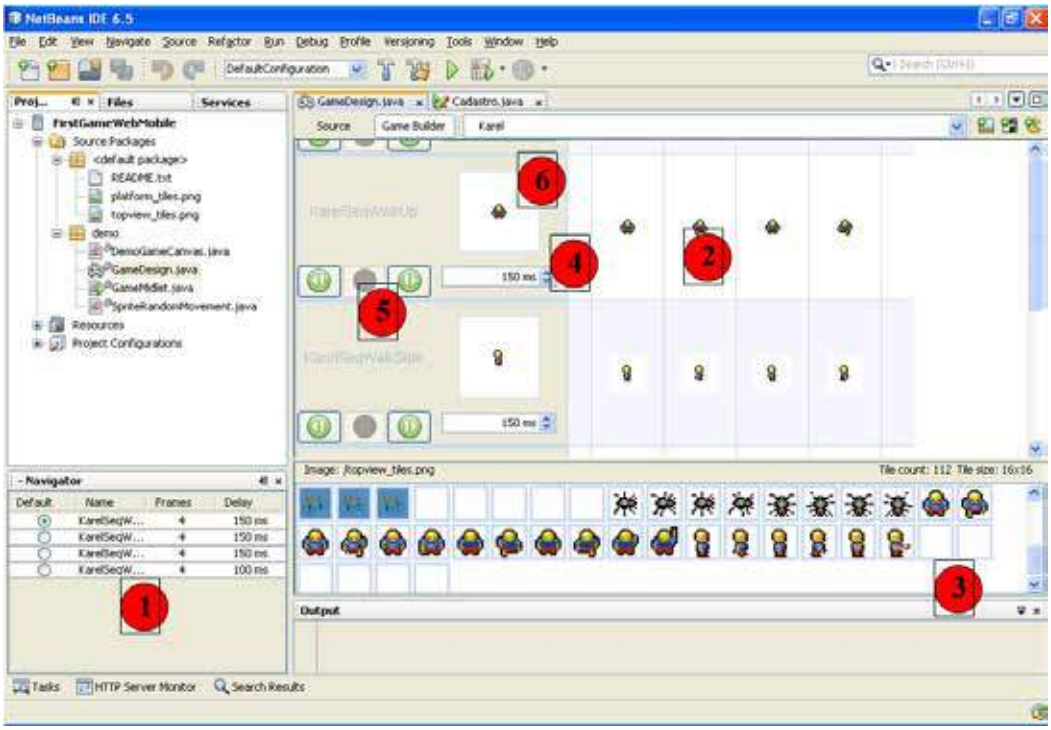
É possível também, na tela de edição de cenários, adicionar novos elementos visuais (TiledLayers e Sprites) clicando com o botão direito sobre a tela e escolhendo a opção addSprite ou addTiledLayer. É possível adicionar apenas uma instância de cada componente visual na tela (por exemplo, um Sprite do Karel e um Sprite do Thomas). Ao retornar à tela principal (*Game Design*) e clicar sobre o Sprite Karel, é apresentada a tela de edição de Sprites (Figura 6).

O Sprite é criado a partir de uma imagem ou um conjunto de imagens. Nesse exemplo, as imagens de Karel são recuperadas do arquivo topviw-tiles. png, sendo que este personagem é criado a partir de uma sequência de imagens, as quais formarão uma animação. Para essa animação dá-se o nome de sequência.

Um Sprite pode possuir uma ou mais sequências. No exemplo do Sprite Karel, existe uma que representa a movimentação do personagem quando ele sobe no cenário, outra que representa sua movimentação para baixo e outras duas que representam as movimentações laterais. As listas de sequência disponíveis são apresentadas no Navigator (1), sendo apresentada a sequência padrão atribuída para o Sprite, o nome das sequências, a quantidade de *frames* (imagens) e o *delay* (tempo de espera) entre um frame e outro. O tempo do *delay* é atribuído via interface visual (4), sendo possível visualizar a animação (5). Caso seja necessário, novas imagens podem ser adicionadas à sequência, clicando na imagem desejada (3) e arrastando até a sequência desejada (2). Para criar uma nova sequência, devemos clicar com o botão direito sobre a área central (2) e escolher a opção Create Sequence (ver Nota 1).

[abrir imagem em janela]

FIGURA 6 - TELA DE CONFIGURAÇÃO DO SPRITE KAREL



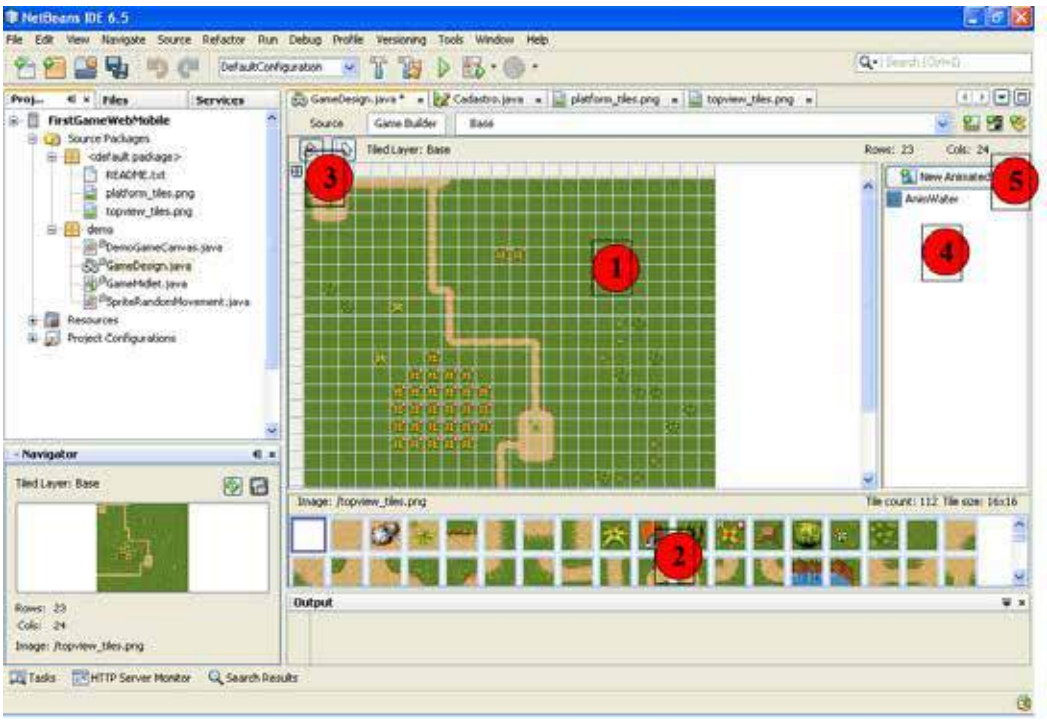
Nota: Ao ser iniciada a lista com as imagens - Figura 6 (3) – possui altura zero, logo deve ser redimensionada para apresentar as imagens disponíveis.

Um TiledLayer é formado a partir de uma matriz de "subimagens", as quais são agrupadas com o objetivo de formar uma imagem maior. No exemplo da Figura 7, temos um Tiled Layercom 23 linhas e 24 colunas, sendo que cada célula da matriz possui 16x16 pixels.

A área central (1) é utilizada para montar a tela, sendo possível clicar e arrastar novas imagens de (2). Na parte superior (3) é possível alternar entre as opções Paint Mode, o qual permite adicionar a imagem selecionada em (2) quando se seleciona uma célula em (1), e a opção *Selection Mode*, que permite selecionar células em (1), aplicando sobre elas operações de duplicação, deleção etc. (essas operações são obtidas através do clique com o botão direito sobre a célula selecionada) (Ver Nota). Outro recurso interessante no TiledLayer são os AnimatedTiles (4). Estes são animações que podem ser adicionadas à tela. Para criar um novo AnimatedTile, devemos clicar no botão *New Animated Tiled* (5). Será apresentada uma tela semelhante à criação de Sprites, como pode ser observado na Figura 8.

[abrir imagem em janela]

FIGURA 7 - *TILED LAYER BASE*



[abrir imagem em janela]

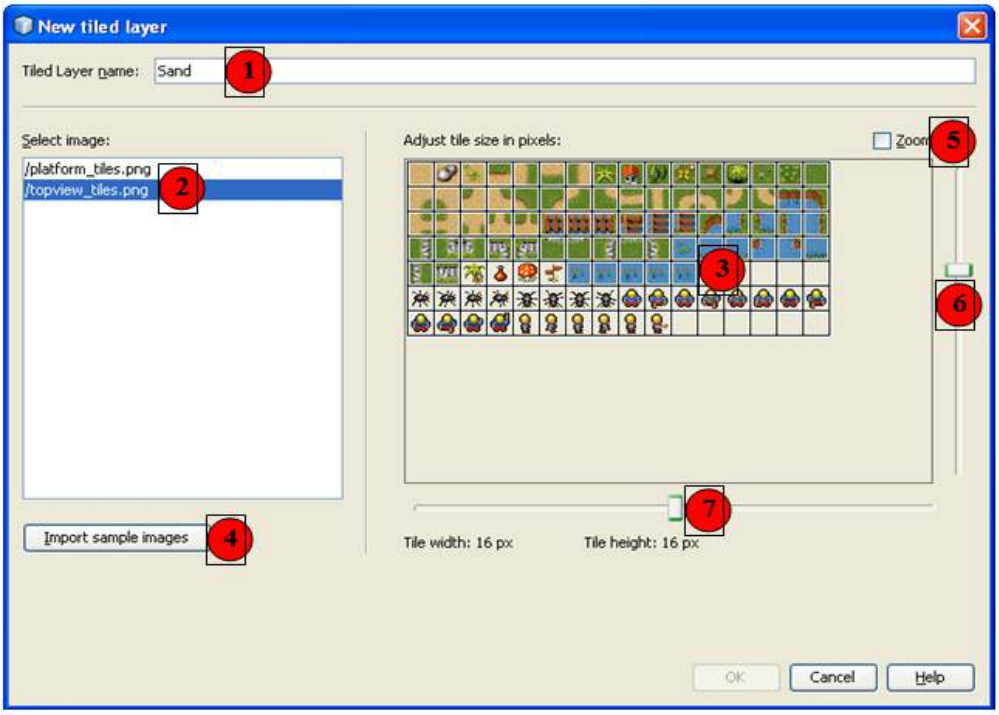
FIGURA 8 - EDIÇÃO DE UM *ANIMATEDTILED*

Nota: Para preencher uma grande área com a mesma imagem, basta selecionar a área que se deseja preencher (a seleção pode ser feita clicando sobre os cabeçalhos de linhas e colunas em (1)) e após clicar/arrastar a imagem que se deseja no preenchimento (2), assim toda a área selecionada será preenchida com a mesma imagem. Assim como os Sprites, um AnimatedTiles possui uma sequência, a qual pode ser formada por uma ou mais imagens. Para se criar um novo elemento com o *Game Builder* (Scene, TiledLayer ou Sprite), devemos escolher a opção *Create* correspondente ao recurso na tela representada pela Figura 4. Como exemplo, será iniciado o desenvolvimento de uma nova TiledLayer. Portanto, devemos selecionar Create Tiled Layer. A tela da Figura 9 é apresentada ao programador. Para o exemplo, foi dado o nome de *Sand* para o novo TiledLayer (1), sendo o mesmo criado a partir da imagem topview_tiles. png. Essa imagem é formada por um conjunto de subimagens, as quais podem ser visualizadas em (3). Se for necessário utilizar uma imagem que não está no projeto, podemos escolher a opção *Import sample images* (4).

É possível escolher a opção *zoom* (5) para aumentar as imagens em (3), e também definir um novo tamanho para as subimagens deslizando os *slider*s (6) e (7). Após clicar em OK será exibida uma tela em branco para o desenvolvimento no novo TiledLayer. Como sugestão, segue a tela apresentada na Figura 10.

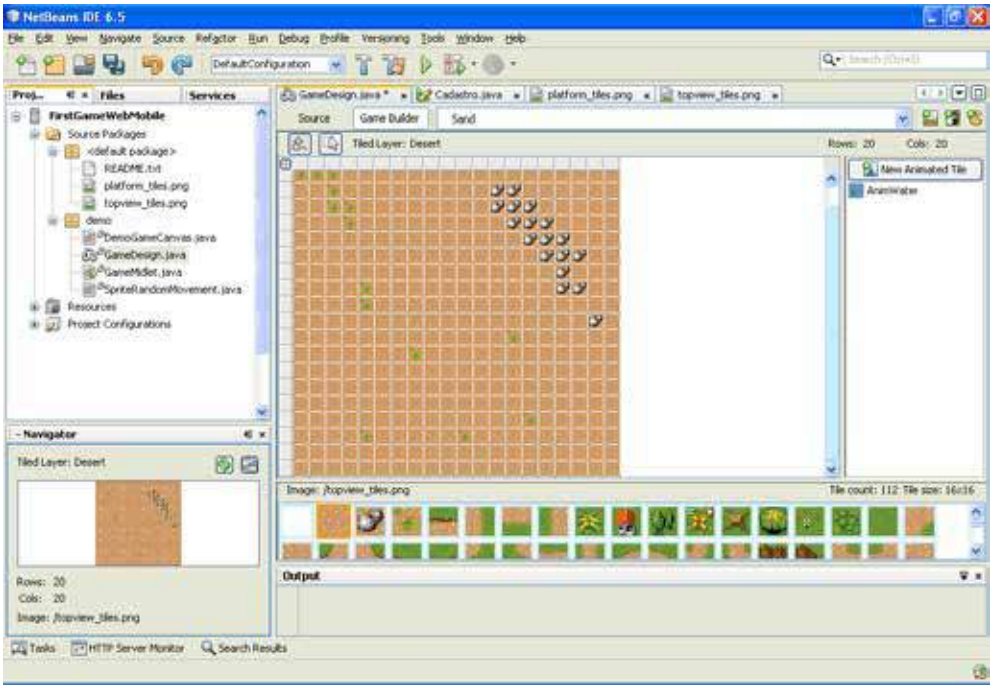
[abrir imagem em janela]

FIGURA 9 - CRIAÇÃO DE UM *TILEDLAYER*



[abrir imagem em janela]

FIGURA 10 - EDIÇÃO DO NOVO *TILEDLAYER*



Com o novo TiledLayer, será desenvolvido um novo estágio para o jogo. Então na tela principal de *Game Design* devemos escolher a opção *Create Scene*... e na tela seguinte será solicitado um nome para a fase. Para esse exemplo será utilizado o nome *Desert*. Será apresentada a tela para desenvolvimento da *Scene* (Figura 11). Nessa tela podemos adicionar TiledLayers clicando com o botão direito e escolhendo a opção Add TiledLayer (1). Também é possível adicionar Sprites, escolhendo a opção Add Sprite. Conforme se adicionam elementos visuais ao estágio do jogo, os mesmos são apresentados em Navigator (3), sendo possível mudar a prioridade dos mesmos (z) e visualizar a posição inicial na tela (x e y). Agora para finalizar o exemplo, a *Scene Desert* será apresentada quando o personagem Karel, esse controlado pelo usuário, atingir o canto inferior direito do primeiro estágio (Forest). Para isso, devemos adicionar os primeiros códigos ao jogo desenvolvido. Como já foi observado, o projeto possui quatro classes, sendo elas:

- GameMidlet.java - Esse é o MIDlet do jogo. Essa classe é responsável por controlar o ciclo de vida do aplicativo e também por apresentar a tela inicial do jogo.

- GameDesign.java - Classe que possui o código de todos os elementos visuais do jogo (TiledLayer, Sprite etc.). Essa classe é criada e mantida automaticamente pelo ambiente visual do *Game Builder*.

- SpriteRandomMoviment - Essa classe é responsável pelo movimento dos personagens que são controlados pela máquina (celular). Nesse exemplo, o personagem Thomas, que é uma aranha, se move constantemente pelo cenário e ao encontrar um obstáculo (nesses exemplos os únicos obstáculos são as bordas do cenário), ele muda de direção automaticamente.

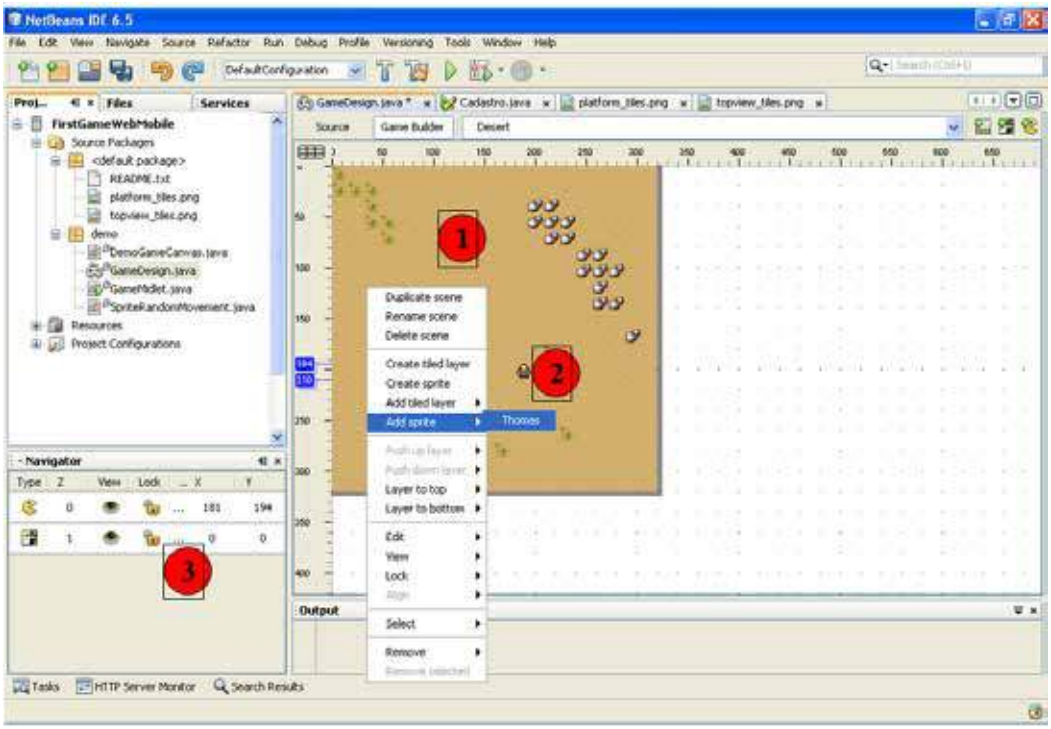
- DemoGameCanvas - Essa é a classe responsável pela lógica do jogo. É a classe principal e seu código está dividido em:

* Método Construtor - public DemoGameCanvas(): Esse método é utilizado para iniciar o jogo. Algumas configurações, como deixar o jogo ocupando toda a tela do celular, podem ser realizadas nesse método.

* Método private void init(): Nesse método são instanciados os componentes visuais da tela, são iniciadas as animações (Sprites e AnimatedTile). Esse método também pode ser utilizado para iniciar os *Scores* do jogo, por exemplo.

[abrir imagem em janela]

FIGURA 11 - TELA DE CONSTRUÇÃO DE UM NOVO ESTÁGIO PARA O JOGO



* Método public boolean spriteCollides (Sprite sprite): É responsabilidade desse método detectar a colisão entre os Sprites passados por parâmetro e os componentes visuais da tela. Inicialmente são detectadas apenas as colisões do Sprite com as bordas da interface, porém outras colisões podem ser implementadas no código-fonte.

* Método private void adjustViewport(int x, int y): Esse método é utilizado para a paginação na interface visual. Quando o Sprite controlado pelo usuário sai da área de visão (tamanho do display) esse método faz o deslocamento da tela.

* Método public void run(): Método mais importante do jogo. É responsável pela lógica do aplicativo. Dentro desse método que é chamado o spriteCollides para identificar colisões e é atualizada a posição da tela com o método adjustViewport.

* Método public void stop(): Método responsável por parar a lógica do jogo. Chamado quando se recebe uma ligação, por exemplo. Além dos métodos citados anteriormente, a classe DemoGameCanvas possui duas classes privadas, sendo elas:

- private class TileAnimationTask extends TimerTask: Classe responsável pelo efeito de animação dos *Animated Tiles*.

- private class SpriteAnimationTask extends TimerTask: Classe responsável pelo efeito de animação dos Sprite.

A lógica para mudar de fase é apresentada na Listagem 1, onde é apresentado parte da lógica do método run.

Listagem 1. Lógica para mudar de fase no DemoGameCanvas.java

```
01.    public void run() {
02.  Graphics g = getGraphics();
03.
04.  while (!this.interrupted) {
05.
06.   if (this.spriteKarel.getX() >= 340 &&
07.    this.spriteKarel.getY() >= 340) {
08.
09.    try {
10.
11.     while (this.lm.getSize() > 0) {
12.       javax.microedition.lcdui.
          game.Layer l =
13.           this.lm.getLayerAt(0);
14.
15.       this.lm.remove(l);
16.     }
17.
18.       this.gameDesign.
          updateLayerManagerForDesert
          (this.lm);
19.     } catch (IOException e) {
20.     e.printStackTrace();
21.    }
22.
23.    this.viewPortX = 0;
24.    this.viewPortY = 0;
```

```
25.      this.lm.setViewWindow
           ( this.viewPortX, this.viewPortY,
26.         this.getWidth(), this.getHeight());
27.
28.      this.lastDirection = -1;
29.     }
30.
31.     //check for user input
32.     int keyState = getKeyStates();
33.
34.     //if user is pressing the left button
35.     if ((keyState & LEFT_PRESSED) != 0)  {
36.     ...
37.     ...
38.   }
```

Na listagem anterior, o método run() inicialmente recupera a interface gráfica do celular através do comando getGraphics() (linha 02) para então iniciar o *looping* de renderização da tela.

Dentro do *looping* será verificada a posição do Sprite Karel através dos comandos getX() eget(Y), verificando se as coordenadas são maiores que 340 (linhas 6 e 7), sendo este o tamanho em pixel aproximado da *Scene Forest*. Caso positivo, é realizado um *looping* pelo objeto lm, o qual representa o LayerManager, sendo cada elemento do objeto recuperado (linhas 12 e 13) e excluído (linha 15), afim de limpar da tela os componentes visuais do primeiro estágio.

Na linha 18 é utilizado o comando updateLayerManagerForDesert(), o qual possui a função de adicionar à tela o segundo cenário.

Para finalizar a lógica, são reiniciadas as variáveis para o controle de paginação na tela (linhas 23 e 24), definida a área visível do segundo cenário (linhas 25 e 26) e reinicializada a variável que representa a última movimentação do Sprite Karel (linha 28). Em seguida, a lógica continua conforme o arquivo original, recuperando a tecla pressionada pelo usuário (linha 34) e fazendo a verificação da mesma (linha 37).

As telas do jogo apresentadas no emulador são apresentadas na Figura 12.

FIGURA 12 - TELAS DO JOGO. À ESQUERDA O PRIMEIRO ESTÁGIO E À DIREITA O SEGUNDO ESTÁGIO DESENVOLVIDO

Conclusão

A criação de jogos é um dos principais ramos de atuação da plataforma JavaME, e a última versão de seu perfil MIDP traz melhorias significativas na construção deste tipo de aplicativo. Para ajudar ainda mais o desenvolvedor, a IDE *NetBeans* traz a ferramenta *Game Builder*, que faz aumentar significativamente a produtividade no desenvolvimento de jogos. Agora é soltar a imaginação e utilizar todos os recursos dessa ferramenta.

FONTE: Disponível em: <http://www.devmedia.com.br/websys.5/webreader.asp?cat=5&artigo=1823&revista=webmobile_26#a-1823>. Acesso em: 20 set. 2014.

RESUMO DO TÓPICO 3

- Conhecemos algumas ferramentas de desenvolvimento como Notepad++, Netbeans, Amaya, Aptana, komodo, MS Expression Web, Dreamweaver e PHP Editor.

- Conhecemos algumas linguagens de programação dinâmicas como ASP, PHP e JSP.

- Vimos que linguagens de programação *Client Side Web* são os *scripts* que compõem os códigos ou sequências de códigos, que formam os arquivos existentes em um *website*.

- Conhecemos algumas linguagens de programação *Client Side* como CSS, HTML e Javascript.

- Vimos que Servidor *web* é um programa executado em uma máquina classificada como servidora capaz de interpretar requisições recebidas de máquinas clientes.

- Conhecemos alguns servidores *web* como APACHE e IIS.

AUTOATIVIDADE

1 Sobre o *NetBeans* analise as afirmações a seguir:

I- O editor suporta várias linguagens.
II- Fornece ferramentas de análise estática.
III- É uma linguagem de programação.

Agora assinale a alternativa correta:

a) As afirmativas I e II estão corretas.
b) As afirmativas I e III estão corretas.
c) As afirmativas II e III estão corretas.
d) Todas as alternativas estão corretas.

2 Cite as linguagens de programação para páginas dinâmicas.

3 Descreva a função do CSS.

UNIDADE 2

HTML – HYPER TEXT MARKUP LANGUAGE

OBJETIVOS DE APRENDIZAGEM

Ao final desta unidade, você será capaz de:

- conhecer a linguagem HTML;
- estruturar uma página HTML;
- entender o funcionamento do Notepad++;
- compreender os elementos básicos do HTML;
- entender o funcionamento das tags em HTML.

PLANO DE ESTUDOS

Esta unidade de ensino está dividida em três tópicos. No final de cada um deles, você encontrará atividades que contribuirão para a apropriação dos conteúdos.

TÓPICO 1 – FUNDAMENTOS DE HTML

TÓPICO 2 – FORMATAÇÕES

TÓPICO 3 – TORNANDO A PÁGINA MAIS INTERATIVA

FUNDAMENTOS DE HTML

1 INTRODUÇÃO

HTML – Hyper Text Markup Language é uma linguagem de formatação hipertexto para descrever documentos *web* (páginas *web*). É uma linguagem de marcação que usa um conjunto de *tags* de marcação.

Desde os primeiros dias da *we*b, têm havido muitas versões de HTML, conforme tabela a seguir:

TABELA 1 – VERSÕES DE HTML

Versão	Ano
HTML	1991
HTML+	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2012

Nesta unidade, você vai aprender primeiro os conceitos básicos de HTML para então ensiná-lo a programar. A redundância é intencional e importante porque queremos que você realmente entenda. Os exemplos são o mais simples possível para que você aprenda de forma clara e simples.

2 INSTALANDO O NOTEPAD++

Para trabalharmos esta disciplina utilizaremos o programa Notepad++. A instalação da ferramenta é extremamente simples. Ela está disponível gratuitamente. Basta acessar a página <http://notepad-plus-plus.org/> e clicar em *download* conforme figura a seguir:

FIGURA 23 – *SITE* PARA *DOWLOAD* NOTEPAD++

FONTE: A autora

A próxima tela se abrirá:

FIGURA 24 – PÁGINA DO *DOWLOAD* DO NOTEPAD++

FONTE: A autora

Clique em *download* e ele irá baixar o executável. Após baixar o executável clique no mesmo e aparecerá a seguinte tela:

FIGURA 25 – ESCOLHENDO A LÍNGUA NOTEPAD++

FONTE: A autora

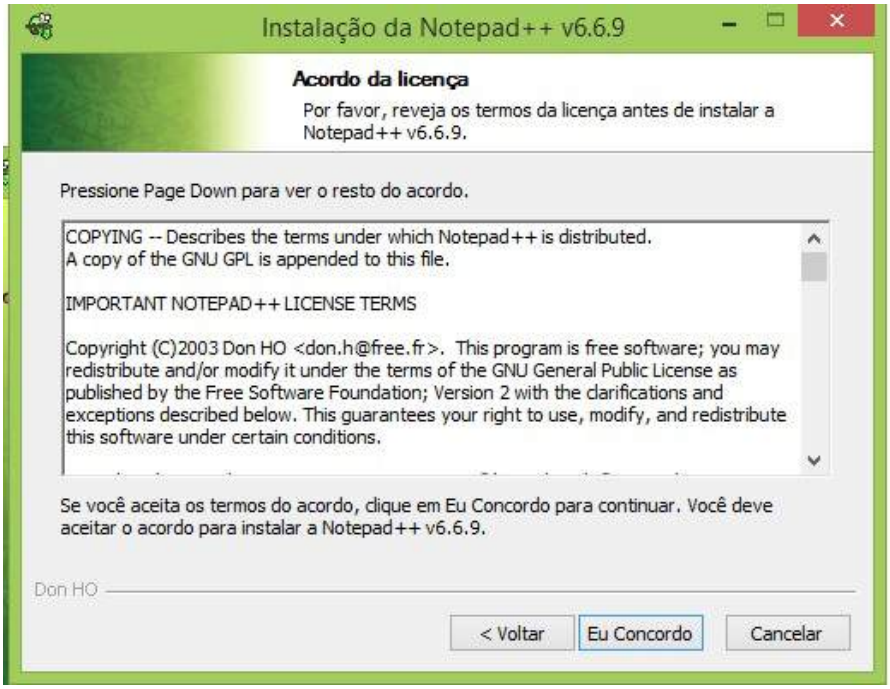
Você clica em ok e aparecerá a próxima tela:

FIGURA 26 – BEM-VINDO AO ASSISTENTE

FONTE: A autora

Clicando em próximo aparecerá a seguinte tela:

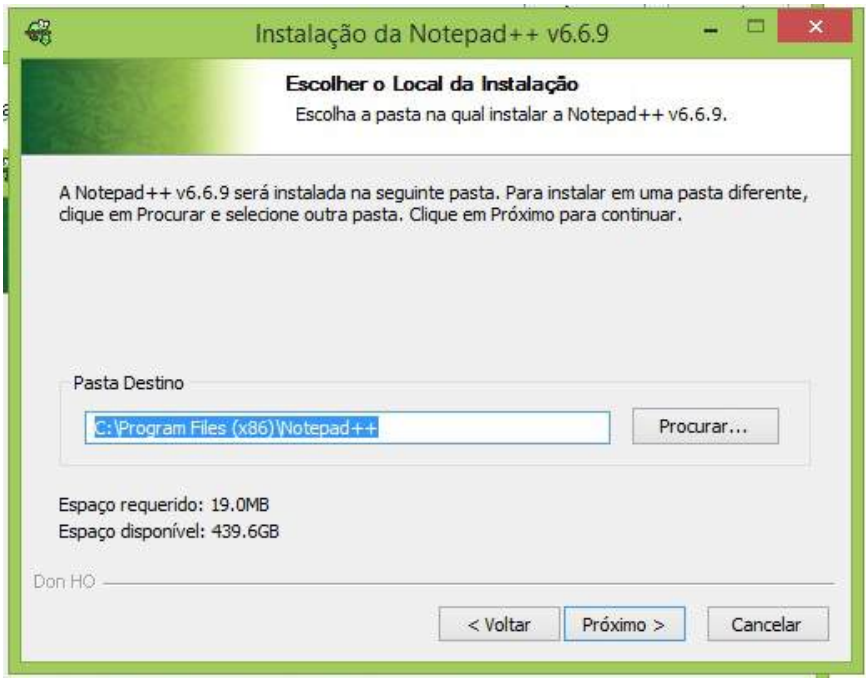
FIGURA 27 – ACORDO DA LICENÇA



FONTE: A autora

Clicando em Eu Concordo aparecerá a seguinte tela:

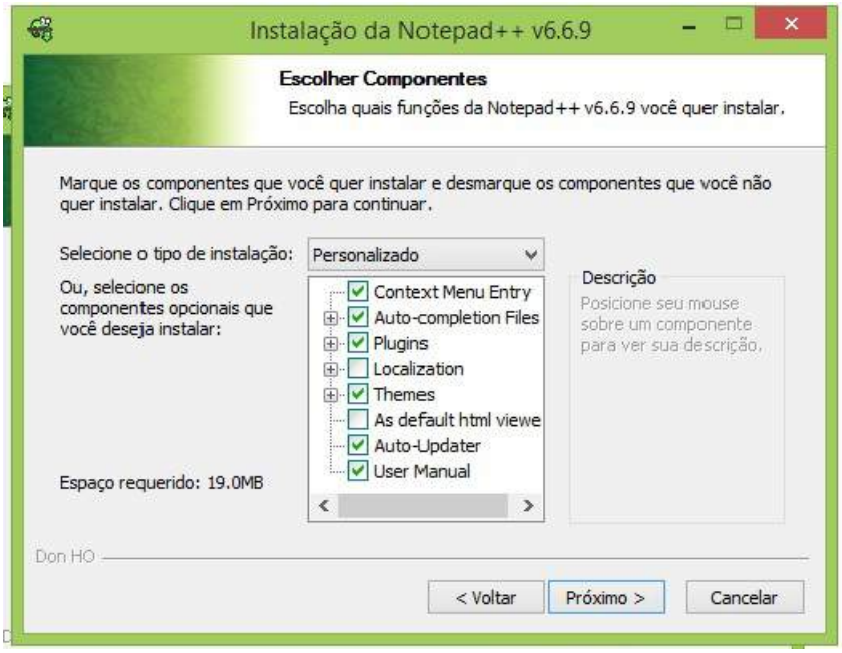
FIGURA 28 – ESCOLHENDO O LOCAL DE INSTALAÇÃO



FONTE: A autora

Clicando em Próximo a seguinte tela aparecerá:

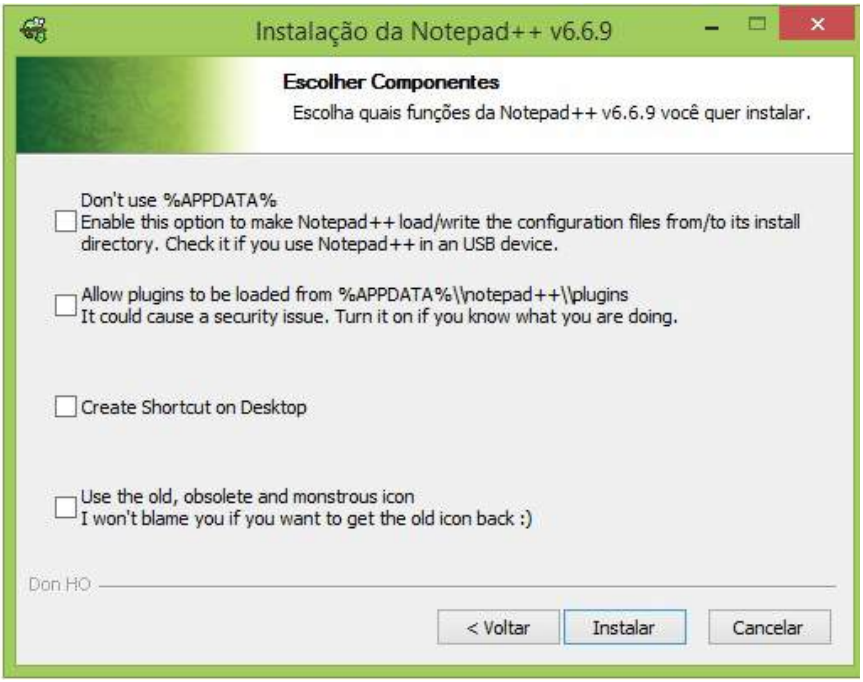
FIGURA 29 – ESCOLHENDO COMPONENTES



FONTE: A autora

Clicando em Próximo a seguinte tela aparecerá:

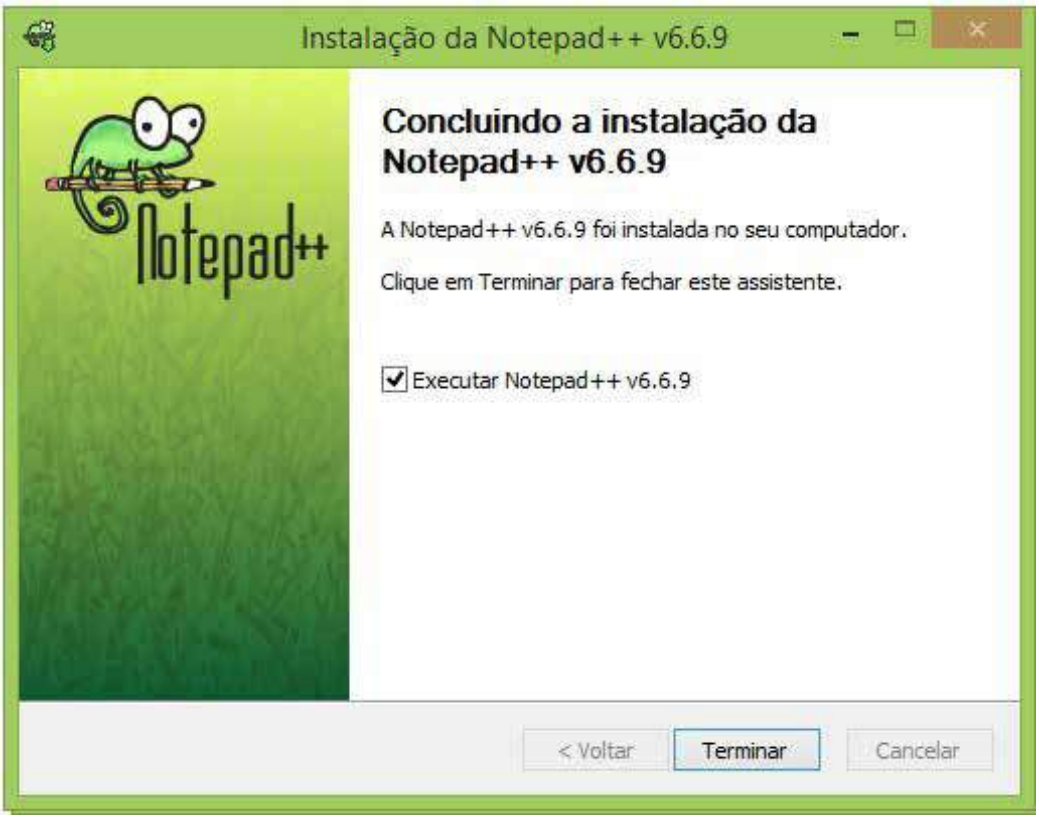
FIGURA 30 – ESCOLHENDO MAIS COMPONENTES



FONTE: A autora

Agora apenas *click* em Instalar. Após a instalação aparecerá a seguinte tela:

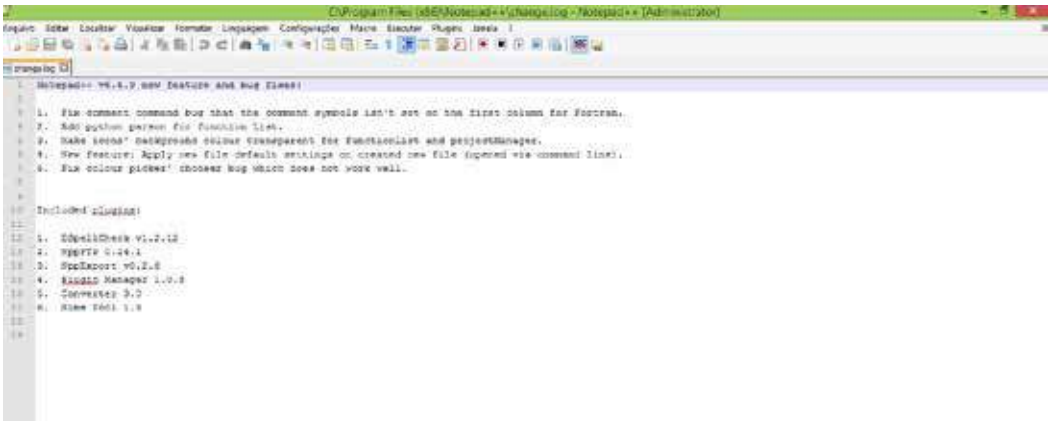
FIGURA 31 – CONCLUINDO A INSTALAÇÃO



FONTE: A autora

Apenas *click* em Terminar e pronto você está com o Notepad++ instalado conforme tela a seguir:

FIGURA 32 – TELA NOTEPAD++



FONTE: A autora

Com este programa você poderá programar em várias linguagens, para utilizar com HTML você deverá clicar na aba Linguagem e escolher HTML.

3 UTILIZANDO TAGS HTML

Primeiramente precisamos compreender o que são *tags* para depois aprender a utilizá-las. As *tags* são comandos de formatação de hipertextos específicos e distintos, colocados em um documento html para que o browser possa interpretá-lo.

São representadas pelo sinal menor que <, em sequência o nome da tag, finalizando com o sinal de maior que >.

Na maioria das vezes as tags são usadas em pares e a tag que finaliza possui uma barra / após o sinal de menor que<. Exemplo <html> e </html>.

Nem todas as tags trabalham em pares, algumas fazem apenas o papel de fechamento como, por exemplo, a tag
 que significa quebra de linha (COSTA; TODESCHINI, 2006).

A marca de início é muitas vezes chamada de tag de abertura. A tag final é muitas vezes chamada de tag de fechamento.

Vejamos o primeiro exemplo HTML já sendo utilizada no programa Notepad+

QUADRO 1 – HTML UTILIZADA NO PROGRAMA NOTEPAD+

```
1  <! DOCTYPE html>
2  <html>
3  <body>
4
5  <h1> Meu primeiro título </ h1>
6
7  <p> Meu primeiro parágrafo. </ p>
8
9  </ body>
10 </ html>
```

FONTE: A autora

Explicando o exemplo

- A declaração DOCTYPE define o tipo de documento
- O texto entre <html> e </ html> descreve o documento *web*
- O texto entre <body> e </ body> descreve o conteúdo da página visível
- O texto entre <h1> e </ h1> descreve um título
- O texto entre <p> e </ p> descreve parágrafo

Usando a descrição, um navegador pode exibir um documento com um título e um parágrafo.

Agora utilize o exemplo no Notepad++.

Após digitar o exemplo no programa você deve salvá-lo para obter o resultado no seu navegador. Você deve ir até o local onde salvou o arquivo e dar um duplo clique e se abrirá no seu navegador o exemplo feito por você. Veja como ficará no navegador:

FIGURA 33 – TELA DO EXEMPLO

FONTE: A autora

Lembre-se de que o propósito de um navegador (Chrome, IE, Firefox, Safari) é ler documentos HTML e exibi-los. O navegador não exibe as tags HTML, mas as usa para determinar como exibir o documento.

4 ESTRUTURA PÁGINA HTML E COMANDOS BÁSICOS

A seguir está uma visualização de uma estrutura de página HTML. Não esperamos que você conheça o HTML, isto é apenas uma mostra de como é uma estrutura HTML.

QUADRO 2 – ESTRUTURA HTML

```
1 <html>
2 <body>
3 <h1> Este é um título </ h1>
4 <p> Este é um parágrafo. </ p>
5 <p> Este é outro parágrafo. </ p>
6 </ body> </ html>
```

FONTE: A autora

4.1 <! DOCTYPE> DECLARAÇÃO

O <! DOCTYPE> declaração ajuda o navegador a exibir uma página *web* corretamente. Há muitos documentos diferentes na *web*, e um navegador só pode exibir uma página HTML corretamente se ele conhece a versão em HTML e o seu tipo.

Todos os documentos HTML devem começar com uma declaração do tipo: <! DOCTYPE html>.

O documento HTML em si começa com <html> e termina com </ html>.

A parte visível do documento HTML é entre <body> e </ body>.

Exemplo:

QUADRO 3 – PARTE VISÍVEL DO DOCUMENTO HTML

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h1> Meu primeiro título </ h1>
5 <p> O meu primeiro parágrafo. </ p>
6 </body>
7 </html>
```

FONTE: A autora

4.2 HTML HEADINGS

Cabeçalhos HTML são definidos com as *tags* <h1> e <> h6. <h1> define o título maior e <h6> define o título menor.

Exemplo:

QUADRO 4 – DEFINIÇÃO DE CABEÇALHOS HTML

```
<h1> Este é um título </ h1>
<h2> Este é um título </ h2>
<h3> Este é um título </ h3>
```

FONTE: A autora

4.3 HTML PARÁGRAFOS

Parágrafos HTML são definidos com a tag <p>.

Exemplo:

QUADRO 5 – PARÁGRAFOS HTML DEFINIDOS COM A TAG <p>

```
<p> Este é um parágrafo. </p>
<p> Este é outro parágrafo. </p>
```

FONTE: A autora

4.4 LINKS EM HTML

Links em HTML são definidos com a tag <a>:

Exemplo:

QUADRO 6 – *LINKS* EM HTML

```
<a href="http://www.nead.com.br"> Este é um link </a
```

FONTE: A autora

O endereço do *link* é especificado no atributo href. Os atributos são usados para fornecer informações adicionais sobre os elementos HTML.

4.5 IMAGENS EM HTML

Imagens HTML são definidas com a tag <img>. O arquivo de origem (src), um texto alternativo (alt) e tamanho (largura e altura) são fornecidos como atributos:

Exemplo:

<img src = "uniasselvi.jpg" alt = "Logo Uniasselvi" width = "104" height = "142">

4.6 ELEMENTOS HTML

Elementos HTML são escritos com uma tag de início, com uma tag final, com o conteúdo no meio:

<tagname> conteúdo </ tagname>

O elemento HTML é tudo, desde a tag de início até a tag de fim:

 O meu primeiro parágrafo. </ p>

Início da tag	Elemento de conteúdo	Fim da tag
<h1>	Primeiro título	</h1>
	Primeiro parágrafo	

Alguns elementos HTML não possuem uma tag final.

Elementos HTML podem ser aninhados, ou seja, elementos podem conter elementos. Todos os documentos HTML consistem em elementos HTML aninhados. Este exemplo contém quatro elementos HTML. Exemplo:

```
<! DOCTYPE html>
<html>
<body>
   <h1> Meu primeiro título </ h1>
  <p> O meu primeiro parágrafo. </ p>
</ body>
</ html>
```

Explicando o exemplo HTML:

A <html> define todo o documento. Ele tem uma tag de início <html> e uma tag final </ html>. O conteúdo do elemento é outro elemento HTML (o elemento <body>).

```
<html>
<body>
   <h1> Meu primeiro título </h1>
  <p> O meu primeiro parágrafo. </p>
</body>
</html>
```

O elemento <body> define o corpo do documento. Ele tem uma tag de início <body> e uma tag final </body>. O conteúdo do elemento é de dois outros elementos HTML (<h1> e <p>).

```
<body>
   <h1> Meu primeiro título </ h1>
```

O <p> define um parágrafo. Ele tem uma tag de início <p> e uma tag final </ p>.

```
  <p> O meu primeiro parágrafo. </ p>
</ body>
```

O <h1> define um título. Ele também tem uma tag de início <h1> e uma tag final </ h1>. O conteúdo do elemento é: Meu primeiro título.

```
<h1> Meu primeiro título </ h1>
```

O conteúdo do elemento é: Meu primeiro parágrafo.

```
<p> O meu primeiro parágrafo. </ p>
```

Não se esqueça da tag final. Alguns elementos HTML serão exibidos corretamente, mesmo se você esquecer a tag final. Exemplo:

<html>
<body>
<p> Este é um parágrafo
<p> Este é um parágrafo
</ body>
</ html>

O exemplo acima funciona em todos os navegadores, pois a tag de fechamento é considerada opcional. Nunca confie nisso. Pode produzir resultados e/ou erros inesperados se você esquecer da tag final.

Elementos HTML sem conteúdo são chamados de elementos vazios.

 é um elemento vazio sem uma tag de fechamento (a tag
 define uma quebra de linha).

Elemento vazio pode ser "fechado" na tag de abertura assim:
.
Elementos HTML podem ter atributos. Atributos fornecem informações adicionais sobre um elemento. Atributos são escritos dentro da tag, seguidos por um sinal de igual e depois entre aspas são declaradas as informações do atributo. Vejamos a seguir alguns atributos:

O HTML5 não requer elementos vazios a serem fechados. Mas se você precisa de validação mais rigorosa, e trouxer o seu documento lido por analisadores XML, feche todos os elementos HTML.

- Atributo title: parágrafos HTML são definidos com a tag <p>. Neste exemplo, o elemento <p> tem um atributo título. O valor do atributo é "Conhecendo o HTML". Exemplo:

QUADRO 7 – CONHECENDO O HTML

```
<p title = "Conhecendo o HTML">
O HTML é uma linguagem de formatação hipertexto para
descrever documentos web (páginas web)., é o primeiro
passo para programar em JavaScript, XML, SQL, PHP, ASP, etc
</p>
```

FONTE: A autora

- O atributo href: *links* em HTML são definidos com a tag <a>. O endereço do *link* é especificado no atributo href.

  Exemplo:

   Este é um link

- Atributos de tamanho: imagens HTML são definidas com a tag <img>. O ficheiro de fonte (src), e o tamanho da imagem (largura e altura) são fornecidos como atributos.

  Exemplo:

  <img src = "uniasselvi.jpg" = "104" height = "142">

O tamanho da imagem é especificado em pixels: width = "104" significa que 104 pixels de tela grande.

A tag <hr> cria uma linha horizontal em uma página HTML. O elemento hr pode ser usado para separar o conteúdo.

Exemplo:

```
<p> Este é um parágrafo. </ p>
<hr>
<p> Este é um parágrafo. </ p>
<hr>
<p> Este é um parágrafo. </ p>
```

A seguir está uma lista alfabética de alguns atributos frequentemente utilizados em HTML:

TABELA 1 – ATRIBUTOS DE HTML

Atributo	Descrição
Alt	Especifica um texto alternativo para uma imagem.
Disabled	Especifica que um elemento de entrada deve ser desativado.
Href	Especifica a URL (endereço *web*) de um *link*.
Id	Especifica um id único para um elemento
Src	Especifica a URL (endereço *web*) para uma imagem.
Style	Especifica um estilo CSS em linha para um elemento.
Title	Especifica informação extra sobre um elemento (exibido como uma dica de ferramenta).
Value	Especifica o valor (conteúdo de texto) para um elemento de entrada.

FONTE: W3schools (2014)

A seguir uma tabela de tags HTML ordenadas alfabeticamente:

TABELA 2 – TAGS HTML

Tag	Descrição
<! --...-->	Define um comentário.
<! DOCTYPE>	Define o tipo de documento.
<a>	Define uma hiperligação.
<abbr>	Define uma abreviação.
<area>	Define uma área dentro de um mapa-imagem.
<article>	Define um artigo.
<audio>	Define o conteúdo de som.
<b>	Define o texto em negrito.
<body>	Define o corpo do documento
 	Define uma única quebra de linha
<button>	Define um botão clicável.
<col>	Especifica as propriedades da coluna para cada coluna dentro de um elemento <colgroup>.
<colgroup>	Especifica um grupo de uma ou mais colunas de uma tabela para formatar.
<datalist>	Especifica uma lista de opções pré-definidas para controles de entrada.
<dialog>	Define uma caixa de diálogo ou janela.
<dir>	Não suportado em HTML5. Use . Define uma lista de diretórios.
	Define uma seção em um documento.
<dl>	Define uma lista descrição.
<dt>	Define um termo / nome na lista de inscrição.
<font>	Não suportado em HTML5. Use CSS em seu lugar. Define fonte, cor e tamanho do texto.
<footer>	Define um rodapé de um documento ou seção.
<form>	Define um formulário HTML para entrada do usuário.
<frame>	Não suportado em HTML5. Define uma janela (um quadro) em um conjunto de quadros.
<frameset>	Não suportado em HTML5. Define um conjunto de quadros.
<h1> e <h6>	Define cabeçalhos HTML.
<head>	Define informações sobre o documento.
<header>	Define um cabeçalho para um documento ou seção.
<hgroup>	Define um grupo de cabeçalhos.
<hr>	Define uma mudança temática no conteúdo.
<html>	Define a raiz de um documento HTML.

<img>	Define uma imagem.
<input>	Define um controle de entrada.
<ins>	Define um texto que foi inserido em um documento.
<kbd>	Define a entrada de teclado.
<keygen>	Define um campo gerador de par de chaves (para formulários).
<label>	Define um rótulo para um elemento <input>.
	Define um item de lista.
<link>	Define a relação entre um documento e um recurso externo (o mais usado para conectar-se a folhas de estilo).
<main>	Especifica o conteúdo principal de um documento.
<map>	Define um client-side image-map.
<mark>	Define marcada / realçado texto.
<menu>	Define uma lista / menu de comandos.
<menuitem>	Define um item de comando / menu que o usuário pode selecionar a partir de um menu pop-up.
<meta>	Define metadados sobre um documento HTML.
<nav>	Define *links* de navegação.
	Define uma lista ordenada.
<output>	Define o resultado de um cálculo.
<p>	Define um parágrafo.
<param>	Define um parâmetro para um objeto.
<pre>	Define texto pré-formatado.
<progress>	Representa o progresso de uma tarefa.
<q>	Define uma citação curta.
<section>	Define uma seção em um documento.
<small>	Define texto menor.
<source>	Define vários recursos de mídia para elementos de mídia (<video> e <audio>).
<strike>	Não suportado em HTML5. Use <del> vez. Define texto tachado.
<style>	Define as informações de estilo para um documento.
<sub>	Define texto subscrito.
<table>	Define uma tabela.
<tbody>	Grupos do conteúdo do corpo em uma tabela.
<td>	Define uma célula em uma tabela.
<textarea>	Define um controle de entrada de várias linhas (área de texto).
<tfoot>	Grupos do conteúdo do rodapé em uma tabela.

<th>	Define uma célula de cabeçalho em uma tabela.
<thead>	Grupos do conteúdo do cabeçalho em uma tabela.
<time>	Define uma data / horário.
<title>	Define um título para o documento.
<tr>	Define uma linha em uma tabela.
<u>	Define o texto que deve ser estilisticamente diferente do texto normal.
	Define uma lista não ordenada.
<var>	Define uma variável.
<video>	Define um vídeo ou filme.
<wbr>	Define uma possível quebra de linha.

FONTE: W3schools (2014)

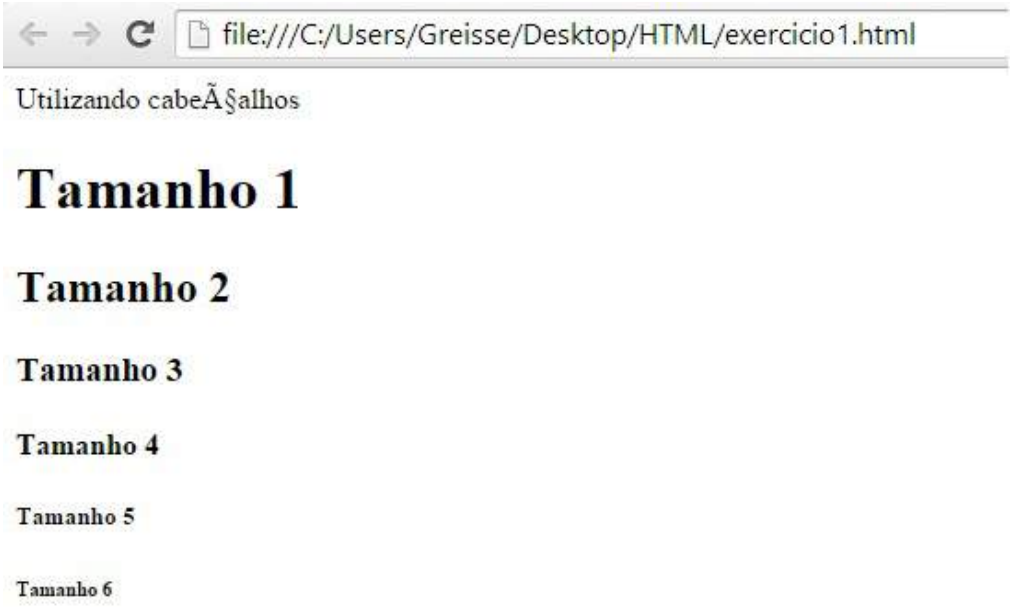
RESUMO DO TÓPICO 1

Neste tópico você viu:

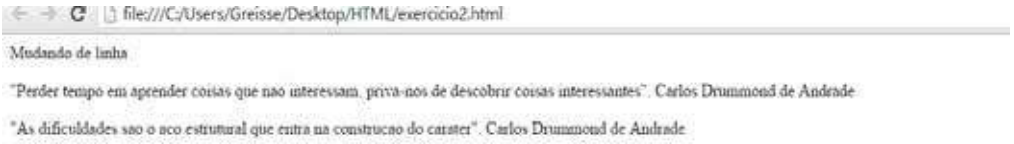
- As várias versões de HTML.
- Como instalar o NOTEPAD++.
- A utilizar tags em HTML.
- A estrutura do HTML.
- Os elementos HTML podem ter atributos.

AUTOATIVIDADE

1 Com os comandos que você aprendeu faça uma página HTML com a aparência semelhante ao anexo e salve-a com o nome de exercicio1.



2 Crie um documento HTML com a aparência semelhante à tela a seguir e salve-o com o nome de exercicio2.



3 Crie um documento HTML com a aparência semelhante à tela a seguir e salve-o com o nome de exercicio3.

FORMATAÇÕES

1 INTRODUÇÃO

Até aqui você viu como montar uma estrutura básica HTML, agora é o momento de dar estilo a ela. Neste tópico iremos aprender a manipular os parágrafos, adicionar estilos e cores.

2 HTML PARÁGRAFOS

Documentos HTML são divididos em parágrafos. O HTML <p> define um parágrafo. Exemplo:

<p> Este é um parágrafo </ p>
<p> Este é outro parágrafo </ p>

Browsers adicionam automaticamente uma linha em branco antes e depois de um parágrafo.

Com HTML, você não pode alterar a saída, adicionando espaços extras ou linhas extras em seu código HTML. O navegador irá remover espaços extras e linhas extras quando a página é exibida. Qualquer número de vagas, e qualquer número de novas linhas são considerados um único espaço (W3SCHOOL, 2014).

Exemplo:

<p>
este parágrafo
contém uma grande quantidade de linhas
no código-fonte,
mas o navegador
ignora.
</ p>

<p>
este parágrafo
contém uma grande quantidade de espaços
no código-fonte,
mas o navegador
ignora.
</ p>

2.1 QUEBRAS DE LINHA HTML

O elemento HTML
 define uma quebra de linha. Use
 se você quiser uma quebra de linha (uma nova linha) sem iniciar um novo parágrafo.

Exemplo:

<p> Este é um
 para
 gráfico com quebras de linha </ p>

No HTML os espaços e as novas linhas são ignorados.
Exemplo:

<p> Este poema foi exibido como uma linha: </ p>

<p>

Minha bola encontra-se sobre o oceano.

Minha bola encontra-se sobre o mar.

Minha bola encontra-se sobre o oceano.

Oh, traga de volta minha bola para mim.

</ p>

Veja como o browser exibirá:

FIGURA 35 – LINHAS

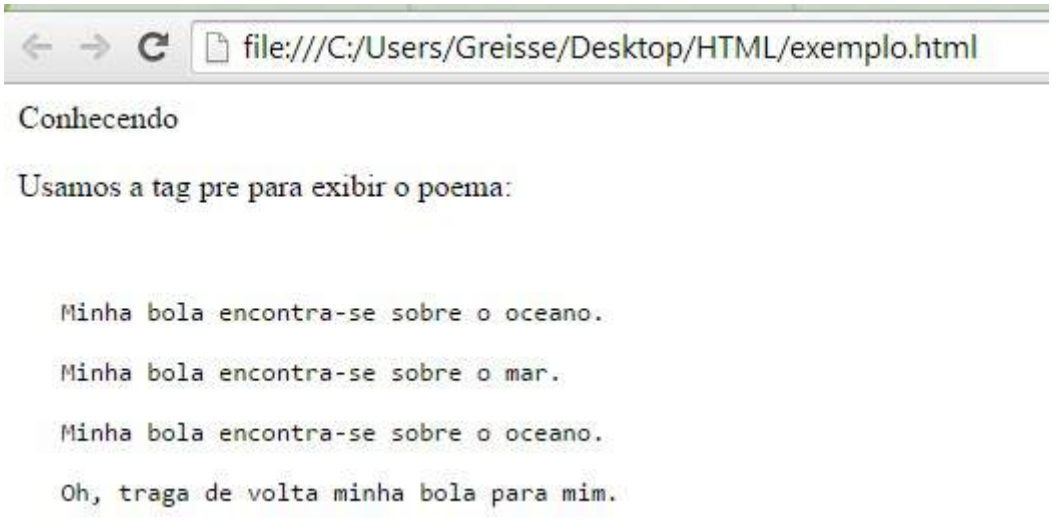
Para resolvermos isto o HTML possui um elemento chamado <pre>. O HTML <pre> define um bloco de texto pré-formatado, com espaços e linhas estruturadas. Para exibir qualquer coisa, com espaçamento entre direita e quebras de linha, você deve quebrar o texto em um elemento <pre>.

Exemplo:

```
<p> Usamos a tag pre para exibir o poema:</ p>

<pre>

  Minha bola encontra-se sobre o oceano.

  Minha bola encontra-se sobre o mar.

  Minha bola encontra-se sobre o oceano.

  Oh, traga de volta minha bola para mim.

</ pre>
```

Veja como o browser exibirá:

FIGURA 36 – LINHAS



3 ESTILOS HTML

Cada elemento HTML tem um estilo padrão (cor de fundo é branco, cor do texto é preto, de tamanho texto é 12px entre outros). Mudar o estilo padrão de um elemento HTML, pode ser feito com o atributo de style. Este exemplo altera a cor de fundo padrão do branco ao cinza.

Exemplo:

<style body = "background-color: lightgrey">

```
1  <!DOCTYPE html>
2  <html>
3  <body style="background-color:lightgrey">
4
5  <h1>Este e um cabecalho</h1>
6
7  <p>Este e um paragrafo.</p>
8
9  </body>
10 </html>
```

Veja como o browser exibirá:

FIGURA 37 – ATRIBUTO STYLE

FONTE: A autora

O estilo HTML possui atributos. O atributo do estilo HTML tem a seguinte sintaxe:

style = "propriedade: valor" .

A propriedade é uma propriedade CSS. O valor é um valor CSS.

Veremos mais sobre CSS adiante.

3.1 CORES DE TEXTO EM HTML

A propriedade color define a cor do texto a ser usada para um elemento HTML.

Exemplo:

```
<! DOCTYPE html>
<html>
<body>
     <h1 style = "color: blue"> Este e um cabecalho </h1>
     <p style = "color: red">. Este e um paragrafo </p>
</body>
</html>
```

Veja como o browser exibirá:

FIGURA 38 – CORES EM HTML

3.2 FONTE DE TEXTO HTML

A propriedade font-family define a fonte a ser usada para um elemento HTML.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="font-family:verdana">
Usando fonte Verdana</h1>

<p style="font-family:courier">
Usando fonte courier.</p>

</body>
```

Veja como o browser exibirá:

FIGURA 39 – FONTE EM HTML

FONTE: A autora

3.3 TAMANHO DO TEXTO EM HTML

A propriedade font-size define o tamanho do texto a ser usado para um elemento HTML. Exemplo:

```
<! DOCTYPE html>
<html>
<body>
   <style h1 = "font-size: 300%"> Tamanho da fonte maior </ h1>
  <p style = "font-size: 160%">. Tamanho da fonte menor </ p>
</ body>
</ html>
```

Já a propriedade text-align define o alinhamento do texto horizontal para um elemento HTML.

Exemplo:

```
<! DOCTYPE html>
<html>
<body>
   <style h1 = "text-align: center"> título centralizado </ h1>
   <p> Testando o alinhamento. </ p>
</ body>

</ html>
```

4 ELEMENTOS DE FORMATAÇÃO HTML

No capítulo anterior, você aprendeu sobre estilo HTML, usando o atributo de estilo HTML. O HTML também define elementos especiais, para definir o texto com um significado especial. O HTML utiliza elementos como <b> e <i> para a formatação de saída, como texto em negrito ou itálico. Elementos de formatação foram projetados para exibir tipos especiais de texto:

- texto em negrito;
- texto importante;
- texto em itálico;
- texto enfatizado;
- texto marcado;
- texto pequeno;
- texto excluído;
- texto inserido;
- sobescrito;
- subscrito. (FREEMAN; FREEMAN, 2008).

4.1 FORMATAÇÃO HTML NEGRITO E ÊNFASE

O HTML <b> define o texto em negrito, sem qualquer importância extra.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>
<p>Texto normal.</p>
<p><b>Texto em negrito.</b></p>
</body>
</html>
```

O *browser* exibirá da seguinte forma:

FIGURA 40 – TEXTO NEGRITO

Texto normal.

Texto em negrito.

FONTE: A autora

O HTML define texto com ênfase.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Texto normal.</p>

<p><strong>Texto com ênfase. </strong></p>

</body>
</html>
```

4.2 HTML ITÁLICO E FORMATAÇÃO ENFATIZADO

O HTML <i> define o texto em itálico, sem qualquer importância extra.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Texto normal.</p>

<p><i>Texto em itálico.</i></p>

</body>
</html>
```

Veja o resultado no *browser*:

FIGURA 41 – TEXTO ITÁLICO

FONTE: A autora

O HTML define texto enfatizado, com importância semântica.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Texto normal.</p>

<p><em>Texto enfatizado.</em></p>

</body>
</html>
```

Os navegadores exibem como <b>, e como <i>. No entanto, há uma diferença no significado dessas tags: <b> e <i> define o texto em negrito e itálico, mas e significa que o texto é "importante".

4.3 OUTRAS FORMATAÇÕES

O HTML <small> define pequeno texto.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<h2>HTML <small>Small</small> Formato</h2>

</body>
</html>
```

O HTML <mark> define marcado ou texto destacado.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<h2>HTML <mark>Formato</mark> destacado</h2>

</body>
</html>
```

O HTML <del> define excluída (retirada) de texto.

Exemplo

```
<!DOCTYPE html>
<html>
<body>

<p>Minha cor favorita é <del>vermelho</del> azul.</p>

</body>
</html>
```

O browser exibirá da seguinte maneira:

FIGURA 42 – TEXTO

Minha cor favorita e ~~vermelho~~ azul.

FONTE: A autora

O HTML <ins> define sublinhado no texto inserido.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Eu estudo na <ins>UNIASSEVI</ins> . </p>

</body>
</html>
```

O HTML <sub> define o texto subscrito.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Texto <sub>subscrito</sub>.</p>

</body>
</html>
```

O HTML <sup> define o texto sobrescrito.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Texto <sup>sobrescrito</sup>.</p>

</body>
</html>
```

5 COMENTÁRIOS EM HTML

No HTML podemos inserir comentários dentro do código-fonte, o qual não resultará em nenhum comando a ser interpretado pelo browser. Tags de comentário <! - E -> usados para inserir comentários em HTML (COSTA; TODESSCHINI, 2006). Você pode adicionar comentários a sua fonte HTML usando a seguinte sintaxe.

Exemplo:

<! – Escreva aqui seus comentários ->

Comentários são importantes, pois ajudam o programador e melhor organizar os seus códigos.

Observe que há um ponto de exclamação na tag de abertura, mas não na tag de fechamento (!). Com os comentários você pode colocar notificações e lembretes em seu HTML. Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<!-- Este é um comentário -->
<p>Estudo na UNIASSELVI.</p>
<!-- Este comentário não aparecerá em seu browser -->

</body>
</html>
```

Veja que o browser não exibe os comentários:

Comentários também são ótimos para a depuração HTML, porque você pode comentar as linhas do código HTML, um de cada vez, para procurar erros.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<!-- Não exibir isto neste momento
<img border="0" src="uniasselvi.jpg" alt="UNIASSELVI">
-->

</body>
</html>
```

Depuração: reanalisar o código-fonte fazendo um passo a passo.

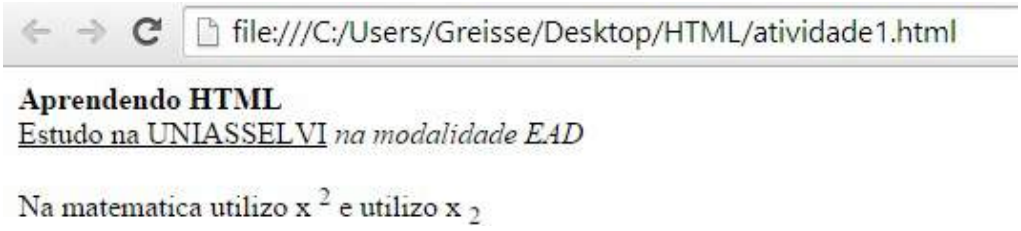
RESUMO DO TÓPICO 2

Neste tópico, você aprendeu formatações para páginas HTML. Vejamos alguns itens importantes:

- <p> define um parágrafo.
-
 define uma quebra de linha.
- Para mudar um elemento padrão do HTML usa-se o atributo style.
- Utiliza-se a propriedade font-size para definir o tamanho do texto.
- O HTML possui elementos especiais, para definir o texto com um significado especial.
- O HTML possui forma de colocar comentários no código-fonte, algo muito importante para o programador.

AUTOATIVIDADE

1 Com os comandos que você aprendeu faça uma página HTML com a aparência semelhante ao anexo e salve-a com o nome de atividade1.



2 Com os comandos que você aprendeu faça uma página HTML com a aparência semelhante ao anexo e salve-a com o nome de atividade2.

3 Todas as liguagens de programação disponibilizam a ferramenta comentário. Diante desta afirmativa descreva a importânica de utilizar comentários no código-fonte HTML.

TORNANDO A PÁGINA MAIS INTERATIVA

1 INTRODUÇÃO

O que veremos agora é como fazermos uma ligação de um arquivo a outro, o modo de apresentar várias páginas em uma mesma tela. Explicaremos também neste tópico a criação de tabelas, a inserção de imagens, a criação de formulários e listas em HTML.

2 LINKS E FRAMES

Os *links* são encontrados em quase todas as páginas da web. *Links* permitem aos usuários clicar em seu caminho a partir de uma página para outra. *Links* em HTML são *hiperlinks*. Um *hiperlink* é um elemento, um texto ou uma imagem que você pode clicar e ir para outro documento. Os *links* em HTML possuem uma sintaxe. Em HTML, os *links* são definidos com a tag <a>. *Link* Sintaxe: texto do link .

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>
<a href="http://www.nead.com.br">Visite a pagina</a>
</p>

</body>
</html>
```

O atributo href especifica o endereço de destino (<http://www.nead.com.br/>). O texto do *link* é a parte visível (Visite a página). Ao clicar no texto do *link*, vai mandar você para o endereço especificado.

O texto do *link* não precisa ser um texto. Ele pode ser uma imagem de HTML ou qualquer outro elemento HTML.

O exemplo acima foi utilizado uma URL absoluta, ou seja, um endereço web completo. Uma ligação local (*link* para o mesmo *site*) é especificado com uma URL relativa, sem http: // www....

Exemplo:

```
<!DOCTYPE html>
<html>
<body>
<p>
<a href="exemplo8.html">Imagem HTML</a> este é um link para
página deste site.
</p>
<p>
<a href="http://www.nead.com.br/">NEAD</a> is a link para um
site na web.
</p>
</body>
</html>
```

Quando você move o cursor do *mouse* sobre um *link*, duas coisas vão acontecer normalmente, a seta do *mouse* se transformará em uma mão apontando e a cor do elemento de ligação irá mudar. Por padrão, os *links* aparecerão como este em todos os navegadores:

- Um *link* não visitado é sublinhado e azul
- Um *link* visitado é sublinhado e roxo
- Um *link* ativo é sublinhado e vermelho

Você pode alterar os padrões, usando estilos.

Exemplo:

```
<! DOCTYPE html>
<html>
<head>
<style>
a: link {
      color: #000000;
      background-color: transparent;
      text-decoration: none;
}
a: visited {
      color: # 000000;
      background-color: transparent;
      text-decoration: none;
}
a: hover {
      color: # ff0000;
      background-color: transparent;
      text-decoration: underline;
}
a: active {
      color: # ff0000;
      background-color: transparent;
      text-decoration: underline;
}
</style>
</head>

<body>
<p> Você pode alterar as cores padrão de ligações </p>
<a href="uniasselvi.jpg" target="_blank"> Imagens HTML </a>
</body>
</html>
```

Nos *links* em HTML temos o atributo alvo. Este atributo de destino especifica onde abrir o documento vinculado. Este exemplo irá abrir o documento vinculado em uma nova janela do navegador ou em uma nova aba.

Exemplo:

```
<! DOCTYPE html>
<html>
<body>

<a href="http:www.nead.com.br" target="_blank"> Visite o site! </a>

<p> Se você definir o atributo target para "_blank", o link irá abrir em uma nova
janela do navegador. </ p>

</ body>
</ html>
```

A seguir os valores do atributo com sua descrição:

TABELA 3 – VALORES E ATRIBUTOS

Valor	Descrição
_blank	Abre o documento vinculado em uma nova janela ou aba.
_self	Abre o documento vinculado no mesmo quadro, uma vez que foi clicado (este é o padrão).
_parent	Abre o documento vinculado no quadro pai.
_top	Abre o documento vinculado no corpo inteiro da janela.
Framename	Abre o documento vinculado em um quadro chamado.

FONTE: W3schools (2014)

Se a sua página é bloqueada em um quadro, você pode usar target = "_top" para sair do quadro.

Exemplo:

```
<! DOCTYPE html>
<html>

<body>

<p>
Abrindo o documento vinculado no corpo inteiro da janela <a href="www.nead.com.br" target="_top">
Clique aqui! </a>
</p>

</body>
</html>
```

Também podemos utilizar *links* em HTML usando uma imagem como *link*. É comum o uso de imagens como *links*.

Exemplo:

```
<! DOCTYPE html>
<html>
<body>

<p> A imagem é um link. Você pode clicar sobre ela. </ P>

<a href="default.asp">
<img src = alt = estilo "uniasselvi.jpeg" "tutorial HTML" = "width: 42px; height: 42px; border:0">
</a>

<p>
Nós adicionamos "border: 0" para impedir que o IE9 (e anteriores) exibam uma borda ao redor
da imagem.
</p>

</body>
</html>
```

Border: 0 é adicionado para evitar IE9 (e anteriores) de exibir uma borda ao redor da imagem.

O atributo id em *links* pode ser usado para criar marcadores dentro de documentos HTML. Marcadores não são exibidos em qualquer forma especial. Eles são invisíveis para o leitor. Exemplo:

```
<! DOCTYPE html>
<html>
<body>

<p>
<a href="#C4"> Veja também Unidade 4. </a>
</p>

<h2> Unidade 1 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 2 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 3 </h2>
<p> Esta unidade explica sobre... </p>

<h2> <a id="C4"> Unidade 4 </a> </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 5 </h2>
<p> Esta unidade explica sobre... </p>
```

```
<h2> Unidade 6 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 7 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 8 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 9 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 10 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 11 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 12 </h2>
<p> Esta unidade explica sobre... </p>
```

```
<h2> Unidade </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 14 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 15 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 16 </h2>
<p> Esta unidade explica sobre... </p>

<h2> Unidade 17 </h2>
<p>Esta unidade explica sobre... </p>

</body>
</html>
```

Frames e frameset em html são utilizadas para apresentar várias páginas em uma mesma tela do browser. As frames, também conhecidas como quadros, delimitam a página com arquivos distintos (COSTA; TODESCHINI, 2006).

A tag <frame> define uma determinada janela (frame) dentro de um <frameset>. Cada <frame> em <frameset> pode ter diferentes atributos, como fronteira, a rolagem, a capacidade de redimensionar etc. Em HTML, a tag <frame> não tem tag final. Em XHTML, o tag <frame> deve estar devidamente fechada. Vejamos a seguir um exemplo de frame, lembre-se que você terá que ter salvo os demais arquivos html na mesma pasta:

```
<!DOCTYPE html>
<html>

<frameset cols="25%,*,25%">
  <frame src="framea.html">
  <frame src="frameb.html">
  <frame src="framec.html">
</frameset>

</html>
```

Vejamos a seguir os atributos com a descrição para frame:

TABELA 4 – ATRIBUTOS PARA FRAME

Atributo	Valor	Descrição
frameborder	0 ou 1	Especifica se deve ou não exibir uma borda em torno de um quadro.
longdesc	URL	Especifica uma página que contém uma longa descrição do conteúdo de um quadro.
marginheight	pixels	Especifica as margens superior e inferior de uma moldura.
marginwidth	pixels	Especifica as margens esquerda e direita de um quadro.
name	texto	Especifica o nome de um quadro.
noresize	noresize	Especifica que um quadro não é redimensionável.
scrolling	yes ou no ou auto	Especifica se deve ou não exibir barras de rolagem em um quadro.
Src	URL	Especifica o URL do documento para mostrar em um quadro.

FONTE: W3School (2014)

3 TABELAS

Tabelas servem para organizar textos, figuras, formulários dentro de colunas e linhas que formam as células. Com as tabelas o sistema ficará mais fácil de visualizar (COSTA; TODESCHINI, 2006). Vejamos o primeiro exemplo de tabela:

```
<!DOCTYPE html>
<html>
<body>
<table style="width: 100%">
    <tr>
      <td> Produto </td>
      <td> Valor </td>
      <td> Quantidade </td>
    </tr>
    <tr>
      <td> Lápis </td>
      <td> R$1,00 </td>
      <td> 10 </td>
    </tr>
    <tr>
      <td> Caneta </td>
      <td> R$2,50 </td>
      <td> 20 </td>
    </tr>
</table>
</body>
</html>
```

Entendendo o exemplo:

- As tabelas são definidas com a tag <table>.
- As tabelas são divididas em linhas da tabela com a tag <tr>.
- As linhas da tabela são divididas em dados da tabela com a tag <td>.

- A linha da tabela também pode ser dividida em títulos da tabela com a tag <th>.

Se você não especificar uma borda para a tabela, ela será exibida sem divisões. A divisão pode ser adicionada usando o atributo border.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>
<table border="1" style="width: 100%">
    <tr>
      <td> Produto </td>
      <td> Valor </td>
      <td> Quantidade </td>
    </tr>
    <tr>
      <td> Lápis </td>
      <td> R$1,00 </td>
      <td> 10 </td>
    </tr>
    <tr>
      <td> Caneta </td>
      <td> R$2,50 </td>
      <td> 20 </td>
    </tr>
</table>
</body>
</html>
```

4 IMAGENS

Como inserimos textos em páginas HTML também podemos inserir imagens. Entre os vários formatos de imagem os mais utilizados em documentos HTML são gif, BMP e JPG, lembre-se de que o formato da imagem pode deixar seu *site* mais lento.

Em HTML, imagens são definidas com a tag <img>. A tag <img> é vazia, ela contém apenas atributos e não tem uma tag de fechamento. O atributo src define a URL (endereço web) da imagem.

```
<!DOCTYPE html>
<html>
<body>

<h2>Imagem UNIASSELVI</h2>
<img src="uniasselvi.jpg" alt="Logo" style="width:304px;height:228px">

</body>
</html>
```

O atributo alt especifica um texto alternativo para a imagem, sem ele a imagem não pode ser exibida. O valor do atributo alt deve descrever a imagem em palavras.

Exemplo:

<img src = "uniasselvi. jpg" alt = "Imagem da Uniasselvi">

O atributo alt é necessário. A página da *web* não irá validar corretamente sem ele. Ainda podemos utilizar leitores de tela que são programas de *softwares* que podem ler o que é exibido em uma tela. Usado na *web*, os leitores de tela podem "reproduzir" HTML como texto-para-fala, ícones de som ou saída em *braile*. Os leitores de tela são utilizados por pessoas cegas, com deficiência visual ou dificuldade de aprendizagem.

Você pode usar o atributo de estilo para especificar a largura e a altura de uma imagem. Os valores são especificados em pixels.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<img src="imagem.gif" alt="Imagem" style="width:128px;height:128px">

</body>
</html>
```

Alternativamente, você pode usar os atributos de largura e altura. Os valores são especificados em pixels.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<img src="imagem.gif" alt="Imagem" width="128" height="128">

</body>
</html>
```

Tanto a largura, a altura e os atributos de estilo são válidos no último padrão HTML5. Nós sugerimos que você use o atributo de estilo. Ele impede folhas, estilos de alterar o tamanho padrão de imagens. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
    width:100%;
}
</style>
</head>

<body>

<p>Melhor utilizar o atributo de estilo (em vez do atributo largura e altura), porque impede
folhas de estilos internos ou externos alterarem o tamanho padrão de uma imagem:</p>

<img src="imagem.jpg" alt="Imagem HTML" style="width:128px;height:128px">
<img src="imagem.jpg" alt="Imagem HTML" width="128" height="128">

</body>
</html>
```

Veja a exibição no browser das duas maneiras:

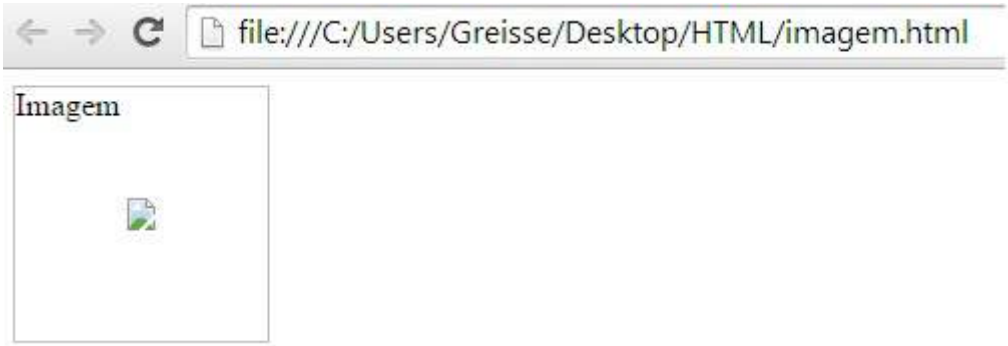
FIGURA 19 – IMAGENS

FONTE: A autora

Se não for especificado, o navegador espera encontrar a imagem na mesma pasta que a página *web*. No entanto, é comum na *web*, para armazenar imagens em uma subpasta e se referem à pasta no nome da imagem.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<img src="/imagens/imagem.jpg" alt="Imagem" style="width:128px;height:128px">

</body>
</html>
```



Alguns *sites* armazenam suas imagens em servidores de imagem. Na verdade, você pode acessar as imagens a partir de qualquer endereço da *web* no mundo. Ainda podemos utilizar imagens animadas de GIF.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>
O padrão GIF permite imagens em movimento.
</p>

<img src="gifanimado.gif" alt="gif animado" style="width:48px;height:48px">

</body>
</html>
```

Tente você também. Você encontra gifs animados no *site* <https://liveunderconstruction.wordpress.com/tag/gif-animado/>.

Note que a sintaxe de inserção de imagens animadas não é diferente de imagens não animadas.

É comum o uso de imagens como *links*.

Exemplo:

```
<! DOCTYPE html>
<html>
<body>

<p> A imagem é um link. Você pode clicar sobre ela. </ P>

<a href="paginainicial.html">
<img src="uniasselvi.jpg" alt="Logo" style="width:42px;height:42px;border:0">
</a>

</body>
</html>
```

Para uma imagem você pode criar um mapa de imagem, com áreas clicáveis.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Clique no sol ou em um dos planetas para vê-lo mais de perto: </p>

<img src="planetass.gif" alt="Planetas" usemap="#planetmap" style="width:145px;height:126px">

<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" alt="Sun" href="sol.html">
  <area shape="circle" coords="90,58,3" alt="Mercury" href="mercururio.html">
  <area shape="circle" coords="124,58,8" alt="Venus" href="venus.html">
</map>

</body>
</html>
```

5 FORMULÁRIOS

Formulários HTML são usados para selecionar diferentes tipos de entrada do usuário. Um formulário HTML pode conter elementos de entrada, como campos de texto, caixas de seleção, botões de rádio, apresentar botões e mais. Um formulário também pode conter listas de seleção, área de texto, legenda e elementos de rotulagem.

A tag <form> é usado para criar um formulário HTML:
<form>
.
Insira os elementos
.
</form>

O elemento mais importante forma o elemento <input>. O elemento <input> é usado para selecionar as informações do usuário. Um elemento <input> pode variar de muitas formas, dependendo do tipo de atributo. Um elemento <input> pode ser do tipo de campo de texto, caixa de seleção, senha, botão de rádio, botão de enviar e muito mais. Os tipos mais comuns de entrada encontram-se descritos a seguir.

- Campo de texto:

<input type = "text"> define um campo de entrada de uma linha que um usuário pode inserir texto:

```
<!DOCTYPE html>
<html>
<body>

<form>
Nome: <input type = "text" name = "nome"> <br>
Sobrenome: <input type = "text" name = "sobrenome">
</ form>

</body>
</html>
```

Como o código HTML acima aparece no navegador:

FIGURA 20 – FORMULÁRIOS

FONTE: A autora

A forma em si não é visível. Observe também que a largura padrão de um campo de texto é de 20 caracteres.

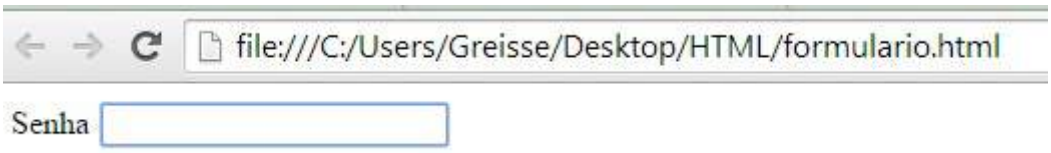
- Campo de senha

<input type = "password"> define um campo de senha:

```
<!DOCTYPE html>
<html>
<body>

<form>
Senha <input type="password" name="pwd">
</form>

</body>
</html>
```

Como o código HTML acima aparece no navegador:

FIGURA 21 – CAMPO DE SENHA



FONTE: A autora

Os caracteres em um campo de senha são mascarados (mostrados como asteriscos ou círculos).

- Botões de rádio

<input type = "radio"> define um botão de rádio. Os botões de rádio permitem que um usuário selecione apenas uma opção de um número limitado de opções:

```
<!DOCTYPE html>
<html>
<body>

<form>
<input type = "radio" name = "sexo" value = "masculino"> Homem <br>
<input type = "radio" name = "sexo" value = "feminino"> Mulher
</form>

</body>
</html>
```

Como o código HTML acima aparece no navegador:

FIGURA 22 – BOTÕES

FONTE: A autora

- Caixas de seleção

<type = "checkbox" input> define uma caixa de seleção. As caixas de seleção permitem que o usuário selecione zero ou mais opções de um número limitado de opções:

```
<!DOCTYPE html>
<html>
<body>

<form>
<input type = "checkbox" name = value "Eletronico" = "TV"> Eu tenho uma TV <br>
<input type = "checkbox" name = value "Eletronico" = "Computador"> Eu tenho um computador
</form>

</body>
</html>
```

Como o código acima aparece no navegador:

FIGURA 23 – CAIXAS

FONTE: A autora

- Botão enviar

<input type = "submit"> define um botão de envio. Um botão enviar é usado para enviar dados de formulário para um servidor. Os dados são enviados para a página especificada no atributo action do formulário. O arquivo definido no atributo action geralmente faz algo com a entrada recebida:

```
<!DOCTYPE html>
<html>
<body>

<form name="input" action="demo_form_acao.php" method="get">
Nome: <input type="text" name="user">
<input type="submit" value="Enviar">
</form>

</body>
</html>
```

Como o HTML acima aparece no navegador:

FIGURA 24 – BOTÃO ENVIAR

FONTE: A autora

Se você digitar alguns caracteres no campo de texto acima e clicar no botão "Enviar", o navegador irá enviar o seu contributo para uma página chamada "demo_form_acao.php". A página irá mostrar-lhe a entrada recebida.

Vejamos a seguir tags para formulários em HTML:

TABELA 5 – TAGS PARA FORMULÁRIOS HTML

Tag	Descrição
<form>	Define um formulário HTML para entrada do usuário.
<input>	Define um controle de entrada.
<textarea>	Define um controle de entrada de várias linhas (área de texto).
<label>	Define um rótulo para um elemento <input>.
<fieldset>	Grupos relacionados a elementos em um formulário.
<legend>	Define uma legenda para um elemento <fieldset>.
<select>	Define uma lista drop-down.
<optgroup>	Define um grupo de opções relacionadas em uma lista drop-down.
<option>	Define uma opção em uma lista suspensa
<button>	Define um botão clicável.

FONTE: W3School (2014)

6 LISTAS EM HTML

HTML pode ter listas não ordenadas, listas ordenadas, ou descrição Listas:

Lista HTML não ordenada:

- primeiro item;
- segundo item;
- terceiro item;
- quarto item;

Lista Ordenada HTML

1) primeiro item;
2) segundo item;
3) terceiro item;
4) quarto item;

Descrição de Lista HTML

Primeiro item
Descrição do item
Segundo item
Descrição do item

Uma lista não ordenada começa com a tag <ul>. Cada item da lista começa com a tag <li>. Os itens da lista serão marcados com bolas, ou seja, pequenos círculos pretos.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<h2>Aprendendo listas ordenadas</h2>

<ul>
 <li>Smarthphone</li>
 <li>Computador</li>
 <li>Tablet</li>
 <li>TV</li>
</ul>

</body>
</html>
```

O navegador exibirá da seguinte maneira:

FIGURA 25 – LISTAS ORDENADAS1

FONTE: A autora

Um atributo de estilo pode ser adicionado a uma lista não ordenada, para definir o estilo do marcador. Vejamos os estilos:

TABELA 6 – ESTILO DE LISTAS

Estilo	Descrição
list-style-type:disc	Os itens da lista serão marcados com bolas.
list-style-type: circle	Os itens da lista serão marcados com círculos.
list-style-type:square	Os itens da lista serão marcados com quadrados.
list-style-type:none	Nenhum dos itens da lista não será marcado.

FONTE: W3school (2014)

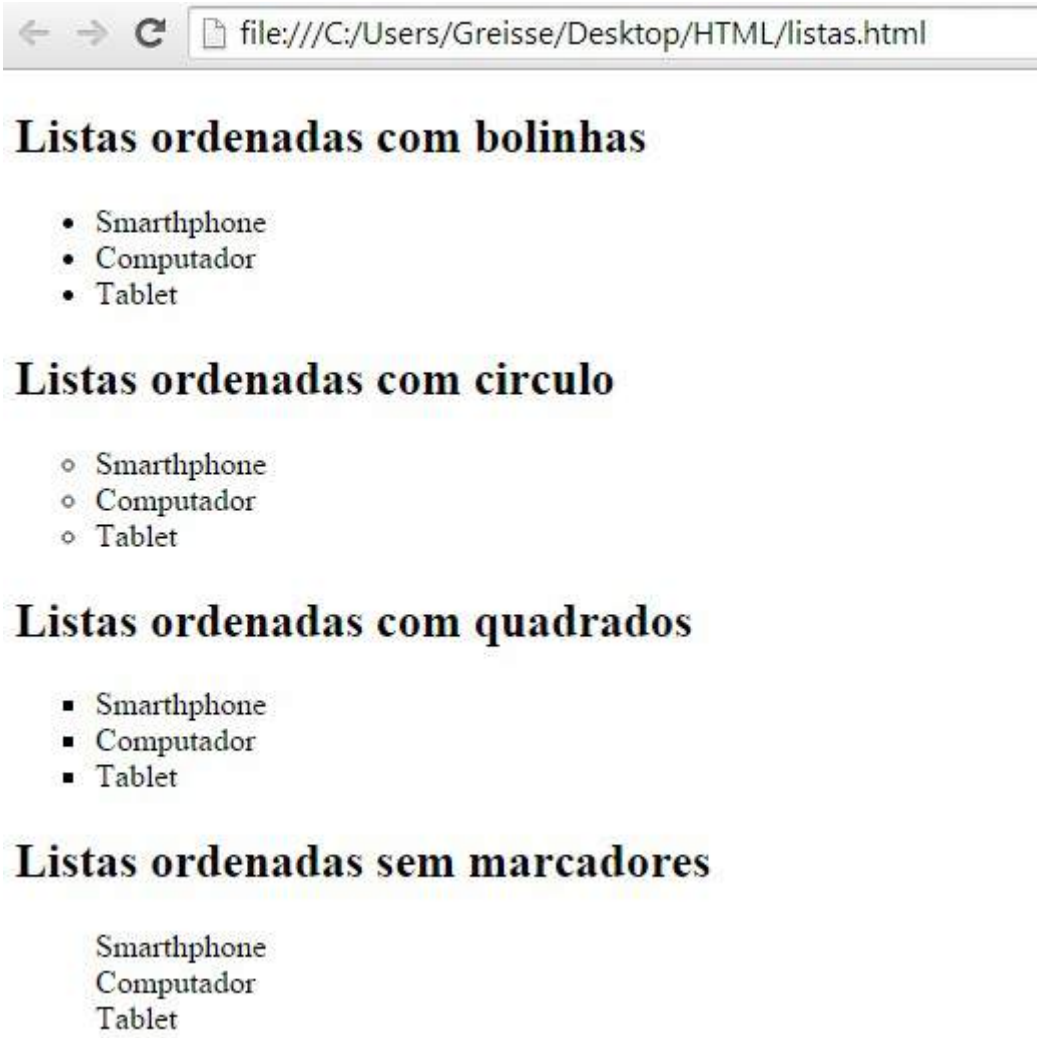
Exemplos:

```
<!DOCTYPE html>
<html>
<body>

<h2>Listas ordenadas com bolinhas</h2>

<ul>
 <li>Smarthphone</li>
 <li>Computador</li>
 <li>Tablet</li>
 </ul>

<h2>Listas ordenadas com circulo</h2>
<ul style="list-style-type:circle">
  <li>Smarthphone</li>
  <li>Computador </li>
  <li>Tablet</li>
</ul>
```

```
<h2>Listas ordenadas com quadrados</h2>
<ul style="list-style-type:square">
  <li>Smarthphone</li>
  <li>Computador </li>
  <li>Tablet</li>
</ul>

<h2>Listas ordenadas sem marcadores</h2>
<ul style="list-style-type:none">
  <li>Smarthphone</li>
  <li>Computador </li>
  <li>Tablet</li>
</ul>

</body>
</html>
```

O navegador exibirá da seguinte maneira:

FIGURA 43 – LISTAS ORDENADAS2



FONTE: A autora

Uma lista ordenada começa com a tag <ol>. Cada item da lista começa com a tag <li>. Um tipo de atributo pode ser adicionado a uma lista ordenada, para definir o tipo de marcador. Vejamos os tipos:

TABELA 7 – LISTA ORDENADA

Tipo	Descrição
type = "1"	Os itens da lista serão numerados com números (padrão).
type = "A"	Os itens da lista serão numerados com letras maiúsculas.
type = "a"	Os itens da lista serão numerados com letras minúsculas.
type = "I"	Os itens da lista serão numerados com números romanos em maiúsculo.
type = "i"	Os itens da lista serão numerados com números romanos em letra minúscula.

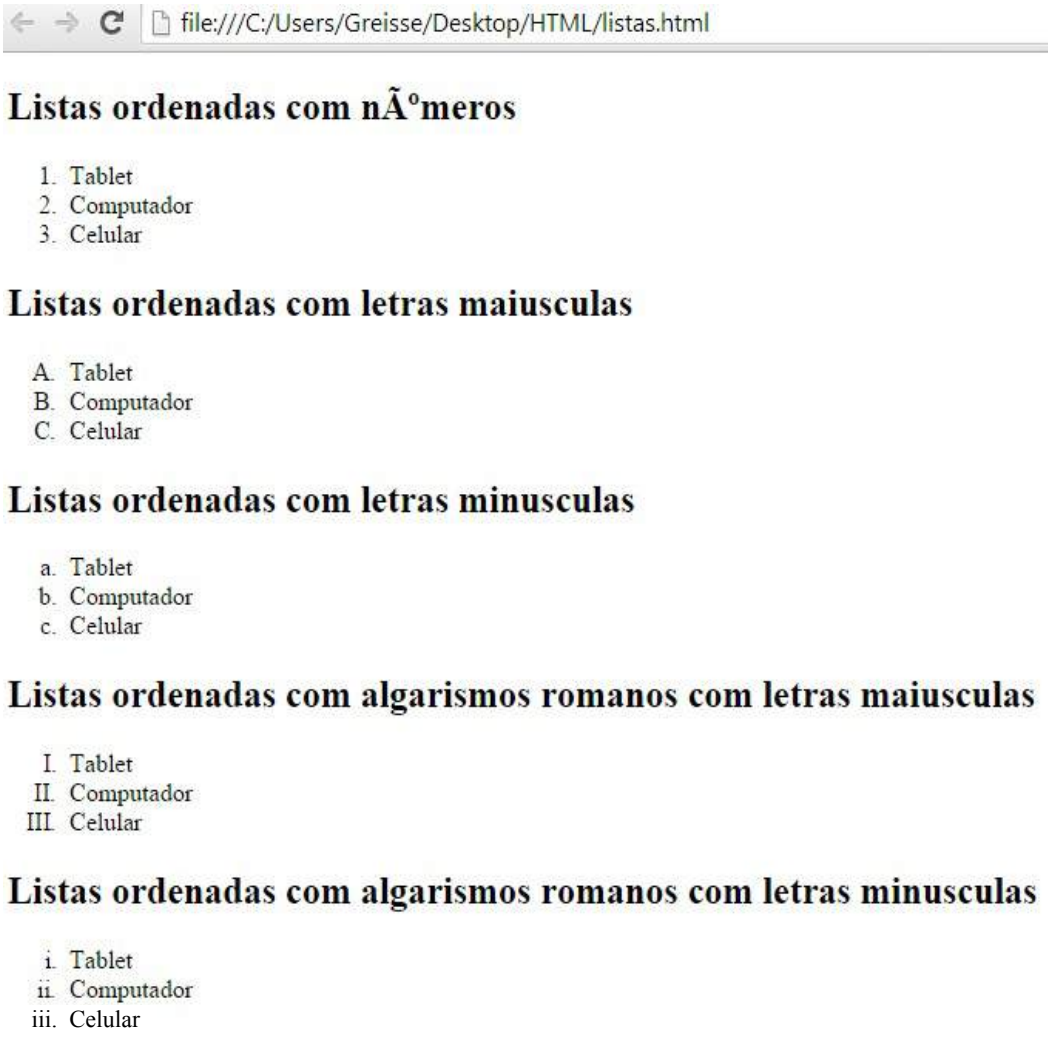
FONTE: W3Schools (2014)

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<h2>Listas ordenadas com números</h2>

<ol type="1">
  <li>Tablet</li>
  <li>Computador </li>
  <li>Celular</li>
</ol>

<h2>Listas ordenadas com letras maiusculas</h2>
<ol type="A">
  <li>Tablet</li>
  <li>Computador </li>
  <li>Celular</li>
</ol>

<h2>Listas ordenadas com letras minusculas</h2>
<ol type="a">
  <li>Tablet</li>
  <li>Computador </li>
  <li>Celular</li>
</ol>

<h2>Listas ordenadas com algarismos romanos com letras maiusculas</h2>
<ol type="I">
  <li>Tablet</li>
  <li>Computador </li>
  <li>Celular</li>
</ol>

<h2>Listas ordenadas com algarismos romanos com letras minusculas</h2>
<ol type="i">
  <li>Tablet</li>
  <li>Computador </li>
  <li>Celular</li>
</ol>

</body>
</html>
```

O navegador exibirá da seguinte maneira:

FIGURA 44 – LISTAS ORDENADAS3



FONTE: A autora

LEITURA COMPLEMENTAR

Usando Geolocalização com HTML5

Veja nesse artigo como usar a API de Geolocalização disponível na versão 5 do HTML e veja também como exibir a localização do usuário em um Mapa do Google.

Ricardo Arrigoni

Olá pessoal! No artigo de hoje iremos ver como trabalhar com a API Geolocation do HTML5.

Primeiro de tudo devemos saber que existem três maneiras de se descobrir a posição de alguma coisa no globo terrestre que são as mais usadas, são elas:

- Geolocalização IP
- Triangulação GPRS
- GPS

***Nota**: Lembrando que para funcionar a geolocalização é necessário que o navegador do usuário tenha suporte a essa tecnologia.*

Utilizando o método getCurrentPosition() é possível pegar a posição do usuário, como podemos ver no exemplo a seguir:

Listagem 1: Obtendo posição de latitude e longitude

```
1   <!DOCTYPE html>
2   <html>
3   <body>
4   <p id="demo">Clique no botão para receber sua localização em latitude e longitude:</p>
5   <button onclick="getLocation()">Clique Aqui</button>
6   <script>
7   var x=document.getElementById("demo");
8   function getLocation()
9     {
10    if (navigator.geolocation)
11      {
12      navigator.geolocation.getCurrentPosition(showPosition);
13      }
14    else{x.innerHTML="O seu navegador não suporta Geolocalização.";}
15    }
16  function showPosition(position)
17    {
18    x.innerHTML="Latitude: " + position.coords.latitude +
19    "<br>Longitude: " + position.coords.longitude;
20    }
21  </script>
22  </body>
23  </html>
```

O código acima irá exibir no seu navegador as suas coordenadas de localização.

Basicamente o que fizemos foi:

- Verificamos se a Geolocalização é suportada pelo browser, caso positivo executamos o método getCurrentPosition(), caso contrário, exibimos uma mensagem de erro para o usuário.
- Após isso, se o método getCurrentPosition() for executado com sucesso, ele irá retornar as coordenadas para a função específica no parâmetro (showPosition).
- A função showPosition() exibe a Latitude e a Longitude para o usuário.

No exemplo acima não tivemos nenhum tratamento de um possível erro que possa acontecer, no exemplo abaixo iremos ver como tratar esses erros.

Listagem 2: Obtendo Geolocalização com tratamento de erros

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Clique no botão para receber as coordenadas:</p>
<button onclick="getLocation()">Clique Aqui</button>
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition,showError);
    }
  else{x.innerHTML="Seu browser não suporta Geolocalização.";}
  }
function showPosition(position)
  {
  x.innerHTML="Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
  }
function showError(error)
  {
  switch(error.code)
    {
    case error.PERMISSION_DENIED:
      x.innerHTML="Usuário rejeitou a solicitação de Geolocalização."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML="Localização indisponível."
      break;
    case error.TIMEOUT:
      x.innerHTML="A requisição expirou."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML="Algum erro desconhecido aconteceu."
      break;
    }
  }
</script>
</body>
</html>
```

A seguir veremos o que quer dizer cada um dos códigos de erro:

- **Permission denied** – O usuário não permitiu o uso de Geolocalização.
- **Position unavailable** – Não foi possível obter a localização.
- **Timeout** – O tempo de resposta expirou.

Exibindo a localização em um Mapa

Caso você não queria apenas exibir as coordenadas de latitude e longitude e queira exibir a posição do usuário em um mapa, basta utilizar o código abaixo.

Listagem 3: Exibindo a localização em um mapa com imagem estática.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Clique no botão para obter sua localização:</p>
<button onclick="getLocation()">Clique aqui</button>
<div id="mapholder"></div>
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition,showError);
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";}
  }

function showPosition(position)
  {
  var latlon=position.coords.latitude+","+position.coords.longitude;

  var img_url="http://maps.googleapis.com/maps/api/staticmap?center="
  +latlon+"&zoom=14&size=400x300&sensor=false";
  document.getElementById("mapholder").innerHTML="<img src='"+img_url+"'>";
  }

function showError(error)
  {
  switch(error.code)
    {
    case error.PERMISSION_DENIED:
      x.innerHTML="Usuário rejeitou a solicitação de Geolocalização."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML="Localização indisponível."
      break;
    case error.TIMEOUT:
      x.innerHTML="O tempo da requisição expirou."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML="Algum erro desconhecido aconteceu."
      break;
    }
  }
</script>
</body>
</html>
```

Dessa forma o que será exibido ao usuário será a localização dele no mapa do Google Maps.

No exemplo da listagem 3 é exibido uma imagem estática do mapa, como se fosse um printscreen, mas é possível fazer com que seja exibido o mapa em si, onde o usuário pode editar e manipular dentro da API do Google Maps.

Listagem 4: Obtendo geolocalização e exibindo um mapa do google maps.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Clique no botão para obter sua localização:</p>
<button onclick="getLocation()">Clique aqui</button>
<div id="mapholder"></div>
<script src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition,showError);
    }
  else{x.innerHTML="Geolocalização não é suportada nesse browser.";}
  }

function showPosition(position)
  {
  lat=position.coords.latitude;
  lon=position.coords.longitude;
  latlon=new google.maps.LatLng(lat, lon)
  mapholder=document.getElementById('mapholder')
  mapholder.style.height='250px';
  mapholder.style.width='500px';

  var myOptions={
  center:latlon,zoom:14,
  mapTypeId:google.maps.MapTypeId.ROADMAP,
  mapTypeControl:false,
  navigationControlOptions:{style:google.maps.NavigationControlStyle.SMALL}
  };
  var map=new google.maps.Map(document.
getElementById("mapholder"),myOptions);
  var marker=new google.maps.Marker({position:latlon,map:map,title:"Você está
Aqui!"});
  }

function showError(error)
  {
  switch(error.code)
    {
    case error.PERMISSION_DENIED:
      x.innerHTML="Usuário rejeitou a solicitação de Geolocalização."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML="Localização indisponível."
      break;
    case error.TIMEOUT:
      x.innerHTML="O tempo da requisição expirou."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML="Algum erro desconhecido aconteceu."
      break;
    }
  }

</script>
</body>
</html>
```

Conclusão

Existem outras maneiras de se usar a API de Geolocalização, nesse artigo mostrei apenas as formas mais comuns de se trabalhar com elas.

FONTE: Disponível em: <http://www.linhadecodigo.com.br/artigo/3653/usando-geolocalizacao-com-html5.aspx>. Acesso em: out. 2014.

RESUMO DO TÓPICO 3

Neste tópico, você aprendeu que:

- *Links* são encontrados em praticamente todas as páginas HTML e que você pode "linkar" textos, imagens utilizando a tag href e demais atributos.
- Frames são utilizadas para apresentar várias páginas HTML em uma página, sempre utilizando a tag frame e framaset.
- Tabelas são importantes para organizar textos, figuras, formulários e sua principal tag é table.
- Podemos inserir diversos tipos de imagens e ainda podemos utiliza-as como *links*. A tag principal de inserção de imagens é img src.
- Formulários são importantes para a comunicação entre a empresa e o usuário. A tag para esta função é <form>.
- Ainda podemos trabalhar com listas ordenadas de várias formas ou números.

AUTOATIVIDADE

1 Com os comandos que você aprendeu faça uma página HTML com a aparência semelhante ao anexo e salve-a com o nome de Tabela.

Nome	Nota 1	Nota 2
Augusto	8	10
Maria	7	8.5

2 Faça uma página em html para exibir o seguinte resultado:

Lista de filmes

1. FROZEN - UMA AVENTURA CONGELANTE
2. O TEOREMA ZERO
3. PLANETA DOS MACACOS: O CONFRONTO

3 Faça uma página HTML conforme a imagem:

UNIDADE 3

CSS E JAVASCRIPT

OBJETIVOS DE APRENDIZAGEM

Ao final desta unidade, você será capaz de:

- conhecer o CSS;
- trabalhar com CSS;
- conhecer o JavaScript.

PLANO DE ESTUDOS

Esta unidade de ensino está dividida em três tópicos, sendo que no final de cada um deles, você encontrará atividades que contribuirão para a apropriação dos conteúdos.

TÓPICO 1 – CSS

TÓPICO 2 – OUTRAS FORMATAÇÕES DE CSS

TÓPICO 3 – JAVASCRIPT

CSS

1 INTRODUÇÃO

Na unidade anterior, você aprendeu o HTML e como ele pode criar estruturas de página HTML. Porém, sobre estilos vimos superficialmente. Com o CSS, você pode controlar o estilo e *layout* de várias páginas da *web* de uma única vez.

Vejamos nosso primeiro exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: #d0e4fe;
}

h1 {
    color: orange;
    text-align: center;
}

p {
    font-family: "Times New Roman";
    font-size: 20px;
}
</style>
</head>
<body>

<h1>Exemplo CSS</h1>
<p>Seja bem vindo(a)!</p>

</body>
</html>
```

Você visualizará em seu *browser* da seguinte maneira:

2 CONHECENDO O CSS

CSS, sigla em inglês que significa Cascading Style Sheets, traduzindo para o português, Folhas de Estilo em Cascata. Os estilos definem como exibir elementos HTML. O HTML nunca foi destinado a conter tags para a formatação de um documento. Quando tags como <font> e atributos de cor foram adicionados à especificação HTML 3.2, começou um pesadelo para os desenvolvedores *web*. Desenvolvimento de grandes *sites*, onde fontes, cores e informações foram adicionadas para cada página, tornou-se um processo longo e caro. Para resolver este problema, o *World Wide Web Consortium* (W3C) criou o CSS (SILVA, 2012).

CSS possui um conjunto de regras que consistem em um seletor e um bloco de declaração:

FIGURA 45 – REGRA CSS

FONTE: Silva, 2012

Definindo cada componente do CSS:

- Seletor: é o alvo da regra CSS.
- Declaração: determina os parâmetros.
- Propriedade: define a característica do seletor.
- Valor: quantifica ou qualifica a propriedade.

O bloco de declaração contém uma ou mais declarações separadas por ponto e vírgula. Cada declaração inclui um nome de propriedade e um valor, separados por dois pontos (SILVA, 2012).

A declaração CSS sempre termina com um ponto e vírgula, e os grupos de declaração são cercados por chaves:

```
p {color: red; text-align: center;}
```

Para tornar o código mais legível em CSS, você pode colocar uma declaração em cada linha, como este exemplo:

```
p {
    color: red;
    text-align: center;
}
```

2.1 COMENTÁRIOS

Os comentários são usados para explicar seu código, e pode ajudá-lo quando você editar o código-fonte em uma data posterior. Os comentários são ignorados pelos navegadores. Um comentário CSS começa com / * e termina com * /. Os comentários também podem abranger várias linhas. Exemplo:

```
p {
    color: red;
    / * Este é um comentário de uma única linha * /
    text-align: center;
}

/ * Este é
um comentário
de várias linhas * /
```

2.2 SELETORES

Seletores CSS permitem selecionar e manipular elementos HTML. Seletores CSS são usados para "encontrar" (ou selecionar) elementos HTML com base no ID, classes, tipos, atributos, valores de atributos e muito mais.

O seletor de elemento seleciona elementos com base no nome do elemento. Você pode selecionar todos os elementos <p> em uma página, como o exemplo a seguir. Todos os elementos <p> serão alinhados ao centro, com uma cor de texto em vermelho.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<p>Vejamos o estilo.</p>
<p id="para1">UNIASSELVI</p>
<p>NEAD</p>

</body>
</html>
```

O seletor id usa o atributo id de uma tag HTML para encontrar o elemento específico. Uma id deve ser única dentro de uma página, por isso você deve usar o seletor de id quando você quiser encontrar um único elemento. Para encontrar um elemento com um id específico, escreve-se uma tag, seguido do elemento id.

Nunca inicie um nome de ID com um número.

O seletor de classe encontra elementos com a classe específica. O seletor de classe usa o atributo de classe HTML. Para encontrar elementos com uma classe específica, escreva um ponto, seguido do nome da classe.

No exemplo a seguir, todos os elementos HTML com class = "center" serão alinhados ao centro e terão cor da fonte em vermelho.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<p>Vejamos o estilo.</p>
<p id="para1">UNIASSELVI</p>
<p>NEAD</p>

</body>
</html>
```

Você também pode especificar quais elementos HTML serão afetados por uma classe.

No exemplo a seguir, apenas os elementos p class = "center" terão texto alinhado e a cor da fonte em vermelho:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<h1 class="center">Texto normal.</h1>
<p class="center">Texto afetado pela class.</p>

</body>
</html>
```

Seu *browser* exibirá da seguinte maneira:

Nunca inicie uma class com um número!

Em folhas de estilo muitas vezes há elementos com o mesmo estilo. Exemplo:

```
h1 {
    text-align: center;
    color: red;
}

h2 {
    text-align: center;
    color: red;
}

p {
    text-align: center;
    color: red;
}
```

Podemos minimizar o código agrupando seletores. Para seletores de grupo, deve-se separar cada seletor com uma vírgula. No exemplo a seguir temos agrupados os seletores do código acima:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<h1>Bom dia!</h1>
<h2>Boa tarde!</h2>
<p>Boa noite!.</p>

</body>
</html>
```

2.3 FORMAS DE INSERIR FOLHA DE ESTILO

Quando um navegador lê uma folha de estilo, ele vai formatar o documento de acordo com as informações contidas na folha de estilo. Há três maneiras de inserir uma folha de estilo:

- Folha de estilo externa.
- Folha de estilo interna ou incorporada.
- Estilo *inline*.

"As folhas de estilo podem ser escritas no próprio documento HTML ao qual serão aplicadas ou ser arquivos externos independentes, gravados com a extensão .css, como, por exemplo, um arquivo chamado de main.css, e lincados ao documento" (SILVA, 2012, p. 18).

2.3.1 Folha de estilo externa

Uma folha de estilo externa é ideal quando o estilo é aplicado a muitas páginas. Com uma folha de estilo externa, você pode mudar a aparência de um *site* inteiro, alterando apenas um arquivo. Cada página deve incluir um *link* para a folha de estilo com a tag <link>. A tag <link> vai dentro da seção head:

```
<head>
<link rel="stylesheet" type="text/css" href="estilo.css">
</head>
```

Uma folha de estilo externa pode ser escrita em qualquer editor de texto. O arquivo não deve conter tags HTML. O arquivo de folha de estilo deve ser salvo com a extensão .css. Um exemplo de um ficheiro de folha de estilo chamado "estilo.css", é mostrado a seguir:

```
body {
    background-color: lightblue;
}
h1 {
    color: navy;
    margin-left: 20px;
}
```

2.3.2 Folha de estilo interna

Uma folha de estilo interna deve ser utilizada quando um único documento tem um estilo único. Você define estilos internos na seção head de uma página HTML, dentro da tag <style>, como segue exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: linen;
}
h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
<body>

<h1>UNIASSELVI
<p>NEAD</p>

</body>
</html>
```

A vantagem desse método sobre o anterior é que, agora, localizamos com mais facilidade a folha de estilo, mas há a desvantagem de colocar a folha de estilos dentro do próprio documento. Não seria sensato vincular uma mesma folha de estilo a vários documentos empregando esse método. Toda vez que for preciso alterar a apresentação, será necessário abrir o documento ou centenas de documentos, se o *site* for grande, e fazer a mesma alteração de estilo em todos eles. Uma boa escolha para uso desse método seria para o caso de aplicação de estilos específicos a um documento do *site*. (SILVA, 2012, p. 19).

2.3.3 Estilos inline

Um estilo inline perde muitas das vantagens de uma folha de estilo (através da mistura de conteúdo com apresentação). Utilize este método com moderação!

Para usar estilos inline, adicione o atributo de estilo diretamente dentro da tag de abertura. O atributo de estilo pode conter qualquer propriedade CSS. O exemplo mostra como mudar a cor e a margem esquerda de um elemento h1:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;margin-left:30px;">UNIASSELVI.</h1>
<p>NEAD.</p>

</body>
</html>
```

2.3.4 Múltiplas folhas de estilo

Se algumas propriedades foram definidas para o mesmo seletor em diferentes folhas de estilo, os valores serão herdados da folha de estilo mais específico. Por exemplo, suponha que uma folha de estilo externa tem as seguintes propriedades para o seletor h1:

```
h1 {
    color: navy;
    margin-left: 20px;
}
```

Suponhamos que uma folha de estilo interna também tem a seguinte propriedade para o seletor h1:

```
h1 {
    color: orange;
}
```

E ainda, se a página com a folha de estilo interna também tiver *links* para a folha de estilo externa as propriedades do elemento h1 será:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="estilo.css">
<style>
h1 {
    color: orange;
}
</style>
</head>
<body>

<h1>UNIASSELVI</h1>
<p>NEAD</p>

</body>
</html>
```

A margem esquerda é herdada da folha de estilo externa e a cor passa a ter a folha de estilo interna.

2.3.5 Múltiplos estilos em cascata

Os estilos podem ser especificados:

- dentro de um elemento HTML;
- dentro da seção head de uma página HTML;
- em um arquivo CSS externo.

Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="estilo.css">
<style>
body {background-color: linen;}
</style>
</head>
<body style="background-color: lightcyan">

<h1>Vários estilos em cascata</h1>
<p>Neste exemplo, a cor de fundo é definida inline, em uma folha de estilo interno, em uma folha de estilo externa.</p>
<p> Tente experimentar através da remoção de estilos para ver como as folhas de estilo em cascata trabalham.
(tente remover a linha em primeiro lugar, em seguida, o interno, então a externa) </p>

</body>
</html>
```

Assim, um estilo inline (dentro de um elemento HTML) tem a maior prioridade, o que significa que ele irá substituir um estilo definido dentro da tag <head>, ou em uma folha de estilo externa, ou em um navegador (um valor padrão).

2.4 FUNDO (BACKGROUND) CSS

Propriedades de fundo CSS são utilizadas para definir os efeitos de um elemento de fundo. Propriedades CSS utilizados para efeitos de fundo:

- background-color;
- background-image;
- background-repeat;
- background-attachment;
- background-position;

2.4.1 Background Color

A propriedade background-color especifica a cor de fundo de um elemento. A cor de fundo de uma página é definida no seletor de corpo. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: #b0c4de;
}
</style>
</head>
<body>

<h1>Olá acadêmico(a).</h1>
<p>Bem vindo(a) a UNIASSELVI.</p>

</body>
</html>
```

Com CSS, uma cor é frequentemente especificada por:

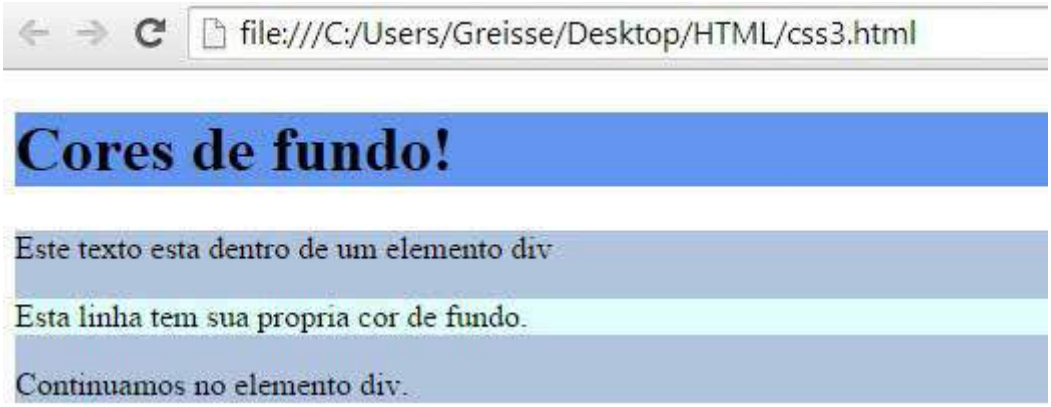
- um valor HEX - como "# ff0000"
- um valor RGB - como "rgb (255,0,0)"
- um nome de cor - como "red"

Veja a tabela completa de cores em <http://erikasarti.net/html/tabela-cores-seguras-web-safe/>.

No exemplo a seguir, o H1, p, e elementos div têm diferentes cores de fundo:

```
<html>
<head>
<style>
h1 {
    background-color: #6495ed;
}

p {
    background-color: #e0ffff;
}

div {
    background-color: #b0c4de;
}
</style>
</head>
<body>

<h1>Cores de fundo!</h1>
<div>
Este texto está dentro de um elemento div
<p>Esta linha tem sua própria cor de fundo.</p>
Continuamos no elemento div.
</div>

</body>
</html>
```

Vejamos como o navegador exibirá:



2.4.2 Imagem de fundo

A propriedade background-image especifica uma imagem para usar como plano de fundo de um elemento. Por padrão, a imagem é repetida para que ela cubra todo o elemento.

A imagem de fundo de uma página pode ser definida assim:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("imagem1.jpg");
}
</style>
</head>
<body>

<h1>Seja bem-vindo!</h1>

</body>
</html>
```

Lembre-se que a imagem que for utilizada deve estar salva na mesma pasta do arquivo html. O navegador exibirá da seguinte forma:

A seguir está um exemplo de uma má combinação de imagem de fundo e texto. O texto quase é ilegível:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("tecnologia1.jpg");
}
</style>
</head>
<body>

<h1>Bem-vindo(a)!</h1>
<p>Com essa imagem de fundo fica dificil a leitura.</p>

</body>
</html>
```

O navegador exibirá da seguinte maneira:

2.4.3 Imagem de fundo – repetir horizontalmente ou verticalmente

Por padrão, a propriedade background-image repete uma imagem horizontal e verticalmente. Algumas imagens devem ser repetidas apenas horizontal ou verticalmente ou vão parecer estranhas, como a seguir. Código-fonte:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("imagem1.jpg");
}
</style>
</head>
<body>

<h1>Seja bem-vindo(a)!</h1>

</body>
</html>
```

Exibição no navegador:

Se a imagem é repetida apenas horizontalmente (repeat-x), o fundo vai ficar melhor. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("imagem1.jpg");
    background-repeat: repeat-x;
}
</style>
</head>
<body>

<h1>Bem-vindo(a)!</h1>

</body>
</html>
```

2.4.4 Definindo a posição da imagem de fundo

Ao usar uma imagem de fundo, use uma imagem que não perturbe o texto.

Mostrando a imagem apenas uma vez é especificado pela propriedade background-repeat property. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("LixeiraELixo.jpg");
    background-repeat: no-repeat;
}
</style>
</head>
<body>

<h1>Seja Bem-Vindo(a)!</h1>
<p>Seja consciente.</p>
<p>Recicle e estaremos salvando o mundo.</p>

</body>
</html>
```

O navegador exibirá da seguinte forma:

No exemplo acima, a imagem de fundo é mostrada no mesmo lugar que o texto. Queremos mudar a posição da imagem, para que não perturbe o texto. A posição da imagem é especificada pela propriedade background-position. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("LixeiraELixo.jpg");
    background-repeat: no-repeat;
     background-position: right top;
     margin-right: 200px;
}
</style>
</head>
<body>

<h1>Seja Bem-Vindo(a)!</h1>
<p>Seja consciente.</p>
<p>Recicle e estaremos salvando o mundo.</p>

</body>
</html>
```

O navegador exibirá da seguinte forma:

Como você pode ver nos exemplos acima, existem muitas propriedades a considerar quando se trata de fundos. Para encurtar o código, também é possível especificar todas as propriedades em uma única propriedade. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background: url("LixeiraELixo.jpg") no-repeat right top;
     margin-right: 200px;
}
</style>
</head>
<body>

<h1>Seja Bem-Vindo(a)!</h1>
<p>Seja consciente.</p>
<p>Recicle e estaremos salvando o mundo.</p>

</body>
</html>
```

Ao usar a propriedade exibida acima existe uma ordem dos valores de propriedade:

- background-color;
- background-image;
- background-repeat;
- background-attachment;
- background-position.

Não importa se um dos valores de propriedade não estiver presentes, os que estão presentes deverão estar nesta ordem.

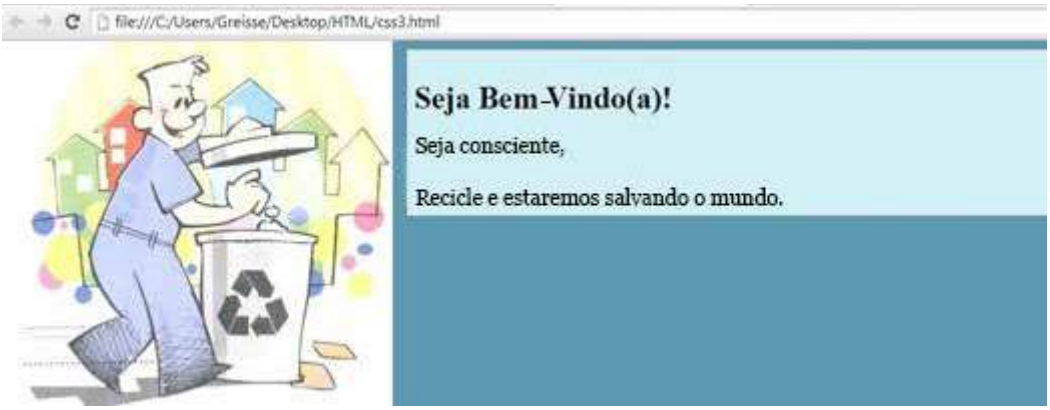
Vejamos um exemplo mais avançado utilizando CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    margin-left: 200px;
    background: #5d9ab2 url("LixeiraELixo.jpg") no-repeat top left;
}

.alinhatexto {
    text-align: center;
}
.centralizatabela {
    border: 1px solid gray;
    margin-left: auto;
    margin-right: auto;
    width: 60%;
    background-color: #d0f0f6;
    text-align: left;
    padding: 8px;
}
</style>
</head>
<body>


<div class="alinhatexto">
  <div class="centralizatabela">
  <h1>Seja Bem-Vindo(a)!</h1>
  <p>Seja consciente.</p>
  <p>Recicle e estaremos salvando o mundo.</p>
  </div>
</div>

</body>
</html>
```

Explicando o exemplo:

- Dentro da estrutura style estamos alinhando o CSS.
- Alinhatexto e cetralizatabela são as classes com as configurações que utilizaremos na estrutura html.

Vejamos como será exibido pelo navegador:



Vejamos todas as propriedades de fundo do CSS:

Propriedade	Descrição
Background	Define todas as propriedades do fundo em uma declaração
background-attachment	Define se uma imagem de fundo é fixa
background-color	Define a cor de fundo de um elemento
background-image	Define a imagem de fundo para um elemento
background-position	Define a posição inicial de uma imagem de fundo
background-repeat	Define como uma imagem de fundo será repetida

RESUMO DO TÓPICO 1

Neste primeiro tópico, você viu que:

- O CSS pode controlar o estilo e o *layout* de várias páginas.
- CSS é uma sigla em inglês que em português significa folhas de estilo em cascata.
- Para termos um CSS necessitamos da estrutura: seletor {propriedade: valor; }.
- Podemos ter uma ou mais declarações.
- Como no HTML também utilizamos comentários no CSS para explicar seu código.
- Os seletores permitem selecionar e manipular elementos HTML.
- Podemos inserir folhas de estilo externa, interna e inline.
- Quando quisermos inserir propriedades de fundo em uma página, seja cor ou imagem, utilizamos background.

AUTOATIVIDADE

1 Observe o código a seguir apresentado. Execute o programa e verifique o seu resultado.

```
<HTML>
<STYLE TYPE="text/css">
<!--
H1 {color: navy; font-size: 30px; font-family: impact}
P {text-indent: 1cm; background: yellow; font-family: arial}
-->
</STYLE>
<HEAD>
<TITLE> Estilos</TITLE>
</HEAD>
<BODY>
<H1>Utilizando estilos</H1>
<P>Agora estou aprendendo estilos</P>
</BODY>
</HTML>
```

2 Observe o seguinte trecho de código:

```
B: {color: DimGray}
<B>Minhas páginas agora usarão <I> CSS </I> </B>
```

Agora diga a que grupo de cores pertence, qual é seu código hexadecimal e seu código RGB.

3 Crie um documento CSS que permita melhorar a apresentação de uma página. Trabalhe o fundo, o texto, o posicionamento da imagem. Após isso, crie um documento html, que *link* o documento CSS.

OUTRAS FORMATAÇÕES DE CSS

1 INTRODUÇÃO

Neste tópico daremos sequência aos estilos em CSS. Veremos a importância e modo de aplicar estilos em textos, fontes, *links*, listas e tabelas. Lembre-se de quão importante é você tentar exercitar os exemplos de cada item.

2 FORMATAÇÕES DE TEXTO

O texto de um *site* pode ser configurado com algumas das propriedades de formatação de texto. Pode-se utilizar texto alinhado, propriedade de cor, parágrafo recuado, alinhado e o espaço entre caracteres é especificado.

2.1 COR DO TEXTO

A propriedade de cor é usada para definir a cor do texto. Como na cor de fundo, o CSS frequentemente especificada uma cor por:

- um valor HEX - como "# ff0000"
- um valor RGB - como "rgb (255,0,0)"
- um nome de cor - como "red"

Veja a tabela completa de cores em <http://erikasarti.net/html/tabela-cores-seguras-web-safe/>.

A cor padrão para uma página é definida no seletor body. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    color: red;
}

h1 {
    color: #00ff00;
}

p.ex {
    color: rgb(0,0,255);
}
</style>
</head>
<body>

<h1>Boa noite!</h1>
<p>Texto em vermelho</p>
<p class="ex">Texto em azul</p>

</body>
</html>
```

O navegador exibirá da seguinte forma:

Boa noite!

Texto em vermelho

Texto em azul

2.2 ALINHAMENTO DE TEXTO

A propriedade text-align é usado para definir o alinhamento horizontal de um texto. O texto pode ser centralizado ou alinhado à esquerda ou à direita ou justificado.

Quando a propriedade text-align está definida para "justificar", cada linha é esticada de modo que cada linha tenha largura igual, e as margens esquerda e direita são retas (como em revistas e jornais). Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    text-align: center;
}

p.date {
    text-align: right;
}

p.main {
    text-align: justify;
}
</style>
</head>
<body>

<h1>Exemplo de alinhamento</h1>
<p class="date">Novembro, 2014</p>
<p class="main">"Mentes pequenas são controladas pela desventura e submissas a ela. Grandes mentes
crescem acima delas" Washington Irving'</p>

</body>
</html>
```

Agora tente executar o exemplo acima e veja o resultado em seu navegador.

2.3 DECORAÇÃO DE TEXTO

A propriedade text-decoration é usada para definir ou remover decorações de texto.

A propriedade text-decoration é usada principalmente para remover sublinhados de ligações para fins de projeto. O comando segue a seguinte sintaxe:

text-decoration:tipo

Também pode ser utilizado para decorar texto. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    text-decoration: overline;
}

h2 {
    text-decoration: line-through;
}

h3 {
    text-decoration: underline;
}
</style>
</head>
<body>

<h1>Texto 1</h1>
<h2>Texto 2</h2>
<h3>Texto 3</h3>

</body>
</html>
```

2.4 TRANSFORMAÇÃO DE TEXTO

A propriedade text-transform é utilizada para especificar letras maiúsculas e minúsculas em um texto. Ela pode ser usada para transformar tudo em letras maiúsculas ou minúsculas, ou capitalizar a primeira letra de cada palavra. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.uppercase {
    text-transform: uppercase;
}

p.lowercase {
    text-transform: lowercase;
}

p.capitalize {
    text-transform: capitalize;
}
</style>
</head>
<body>

<p class="uppercase">Texto em letras maiusculas.</p>
<p class="lowercase">Texto em letras minusculas.</p>
<p class="capitalize">Iniciais de cada palavra em letras maiusculas.</p>

</body>
</html>
```

O text-transform: uppercase transforma todas as palavras em letras maiúsculas. O text-transform: lowercase transforma todas as palavras em letras minúsculas e o text-transform: capitaliza e transforma as letras iniciais das palavras em maiúsculas.

2.5 RECUO DO TEXTO

A propriedade text-indent é usada para especificar o recuo da primeira linha de um texto. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    text-indent: 50px;
}
</style>
</head>
<body>

<p>Recuando o texto</p>

</body>
</html>
```

TABELA 8 – PROPRIEDADES DE TEXTO PARA CSS

Propriedade	Descrição
Color	Define a cor do texto
Direction	Especifica a direção do texto
letter-spacing	Aumenta ou diminui o espaço entre os caracteres de um texto
line-height	Define a altura da linha
text-align	Especifica o alinhamento horizontal do texto
text-decoration	Especifica a decoração adicionada ao texto
text-indent	Especifica o recuo da primeira linha em um bloco de texto
text-shadow	Especifica o efeito de sombra adicionado ao texto
text-transform	Controla maiúscula e minúscula de texto
vertical-align	Define o alinhamento vertical de um elemento

FONTE: W3Schools (2014)

3 FONTES EM CSS

A propriedade font em CSS define a família de fontes, o tamanho e estilo de um texto.

3.1 FAMÍLIAS CSS FONT

Em CSS, existem dois tipos de nomes de família de fonte:

- família genérica - um grupo de famílias ou de fontes com um olhar similar (como "Serif" ou "Monospace");
- família da fonte - uma família de fontes específicas (como "Times New Roman" ou "Arial").

3.1.1 Família de fontes

A família de fontes de um texto definido com a propriedade font-family.

A propriedade font-family deve conter vários nomes de fontes como um sistema de "fallback". Se o navegador não suporta a primeira fonte, ele tenta a próxima fonte.

Comece com o tipo de letra que você quer, e termine com uma família genérica, para deixar o navegador escolher uma fonte semelhante na família genérica, se não houver outras fontes disponíveis.

Se o nome de uma família de fontes é mais do que uma palavra, ele deve estar entre aspas, como: "Times New Roman".

Mais do que uma família de fontes é especificada em uma lista separada por vírgula. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.serif {
    font-family: "Times New Roman", Times, serif;
}

p.sansserif {
    font-family: Arial, Helvetica, sans-serif;
}
</style>
</head>
<body>

<h1>CSS font-family</h1>
<p class="serif">Fonte Times New Roman.</p>
<p class="sansserif">Fonte Arial.</p>

</body>
</html>
```

3.2 ESTILO DA FONTE

A propriedade font-style é maioritariamente utilizado para especificar o texto em itálico. Esta propriedade tem três valores:

- normal - O texto é exibido normalmente
- italic - O texto é exibido em itálico
- oblique - O texto é "inclinar-se" (oblíqua é muito semelhante ao itálico, mas menos deitado). Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
    font-style: normal;
}

p.italic {
    font-style: italic;
}

p.oblique {
    font-style: oblique;
}
</style>
</head>
<body>

<p class="normal">Texto normal.</p>
<p class="italic">Texto em itálico.</p>
<p class="oblique">Texto obliquo.</p>

</body>
</html>
```

Tente fazer o código acima e veja o resultado em seu navegador.

3.3 TAMANHO DA FONTE

A propriedade font-size define o tamanho do texto.

Ser capaz de gerenciar o tamanho do texto é importante em *web design*. No entanto, você não deve usar os ajustes de tamanho de fonte para fazer parágrafos ou títulos ou posições de texto que parecem com parágrafos.

Use sempre as tags HTML adequadas, como o <h1> - <h6> para títulos e <p> para parágrafos.

O valor de font-size pode ser um tamanho absoluto ou relativo.

Tamanho absoluto:

- Define o texto a um tamanho especificado.
- Não permite que um usuário altere o tamanho do texto em todos os navegadores (ruim por questões de acessibilidade).
- Tamanho absoluto é útil quando o tamanho físico da saída é conhecido.

Tamanho relativo:

- Define o tamanho em relação a elementos circundantes.
- Permite que o usuário altere o tamanho do texto em navegadores.

FONTE: W3School (2014)

3.3.1 Definir tamanho da fonte com pixels

Definir o tamanho do texto com pixels lhe dá controle total sobre o tamanho do texto. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    font-size: 40px;
}

h2 {
    font-size: 30px;
}

p {
    font-size: 14px;
}
</style>
</head>
<body>

<h1>Tamanho 40px</h1>
<h2>Tamanho 30px</h2>
<p>Tamanho 14px</p>

</body>
</html>
```

3.3.2 Definir tamanho da fonte com o EM

Para permitir que os usuários redimensionem o texto (no menu do navegador), muitos desenvolvedores usam *em* no lugar de pixels. A unidade em tamanho é recomendada pela W3C.

1em é igual ao tamanho da fonte corrente. O tamanho de texto padrão em navegadores é 16px. Assim, o tamanho padrão de 1em é 16px.

O tamanho pode ser calculado em pixels utilizando esta fórmula: pixels / 16 = EM. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    font-size: 2.5em; /* 40px/16=2.5em */
}

h2 {
    font-size: 1.875em; /* 30px/16=1.875em */
 }

p {
    font-size: 0.875em; /* 14px/16=0.875em */
}
</style>
</head>
<body>

<h1>Tamanho 2.5em</h1>
<h2>Tamanho1.875em</h2>
<p>Tamanho 0.875em</p>

</body>
</html>
```

No exemplo acima, o tamanho do texto é o mesmo do exemplo anterior em pixels. No entanto, com o tamanho em é possível ajustar o tamanho de texto em todos os navegadores.

Infelizmente, ainda há um problema com versões mais antigas do Internet Explorer. O texto torna-se maior do que deveria quando feito maior, e menor do que deveria quando feita menor.

A solução que funciona em todos os navegadores, é definir um font-size padrão em porcentagem para o elemento <body>. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    font-size: 100%;
}

h1 {
    font-size: 2.5em;
}

h2 {
    font-size: 1.875em;
}

p {
    font-size: 0.875em;
}
</style>
</head>
<body>
<h1>Tamanho 2.5em</h1>
<h2>Tamanho 1.875em</h2>
<p>Tamanho 0.875em</p>
</body>
</html>
```

Nosso código agora funciona muito bem! Ele mostra o mesmo tamanho de texto em todos os navegadores, e permite que todos os navegadores façam zoom ou redimensionem o texto!

4 LINKS

Os *links* podem ser decorados com qualquer propriedade CSS (por exemplo cor, font-family, fundo etc.). São chamados de *styling links*, que traduzindo significa estilos de *links*. Os quatro tipos de *link* são:

- a: link – um *link* normal
- a: visited - um *link* que o usuário tenha visitado;
- a: hover - um *link* quando o usuário passa o *mouse* sobre ele;
- a: active - um *link* no momento em que for clicado. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
/* link normal */
a:link {
    color: #FF0000;
}
/* link visitado*/
a:visited {
    color: #00FF00;
}
/* mouse sobre o link */
a:hover {
    color: #FF00FF;
}
/*link selecionado */
a:active {
    color: #0000FF;
}
</style>
</head>
<body>

<p><b><a href="pagina1.asp" target="_blank">Link</a></b></p>
</body>
</html>
```

Ao definir o estilo para vários estados de ligação, existem algumas regras de ordem: a: hover deve vir após a: *link* e a: visited e a: active deve vir após a: hover.

No exemplo acima, o *link* muda de cor dependendo do estado em que se encontra. Vamos passar por algumas outras maneiras comuns para colocar estilos nos *links*.

4.1 DECORAÇÃO DE TEXTO

A propriedade text-decoration é usada principalmente para remover sublinhados de *links*. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
a:link {
    text-decoration: none;
}

a:visited {
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

a:active {
    text-decoration: underline;
}
</style>
</head>
<body>

<p><b><a href="pagina.asp" target="_blank">Link</a></b></p>

</body>
</html>
```

4.2 COR DE FUNDO PARA *LINKS*

A propriedade background-color especifica a cor de fundo para *links*. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
a:link {
    background-color: #B2FF99;
}

a:visited {
    background-color: #FFFF85;
}

a:hover {
    background-color: #FF704D;
}

a:active {
    background-color: #FF704D;
}
</style>
</head>
<body>

<p><b><a href="pagina.asp" target="_blank">Link</a></b></p>

</body>
</html>
```

5 LISTAS EM CSS

As propriedades da lista de CSS permitem que você:

- Defina diferentes marcadores de item de lista para listas ordenadas.
- Defina diferentes marcadores de item de lista para as listas não ordenadas.
- Defina uma imagem como marcador item da lista.

Em HTML, existem dois tipos de listas:

- Listas desordenadas - os itens da lista são marcados com bolas.
- Listas ordenadas - os itens da lista são marcados com números ou letras.

Com CSS, as listas podem ir mais adiante, e as imagens também podem ser utilizadas como o marcador de lista.

O tipo de marcador da lista é especificado com a propriedade list-style-type. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.a {
    list-style-type: circle;
}
ul.b {
    list-style-type: square;
}
ol.c {
    list-style-type: upper-roman;
}
ol.d {
    list-style-type: lower-alpha;
}
</style>
</head>
<body>

<p>Exemplo de listas não ordenada usando CSS</p>
<ul class="a">
  <li>Caneta</li>
  <li>Papel</li>
  <li>Caderno</li>
</ul>

<ul class="b">
 <li>Caneta</li>
  <li>Papel</li>
  <li>Caderno</li>
</ul>

<p>Examplo de lista ordenada usando CSS</p>
<ol class="c">
 <li>Caneta</li>
  <li>Papel</li>
  <li>Caderno</li>
</ol>

<ol class="d">
 <li>Caneta</li>
  <li>Papel</li>
  <li>Caderno</li>
</ol>

</body>
</html>
```

Para especificar uma imagem como marcador de item de lista, use a propriedade list-style-image. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
    list-style: square inside url("logo.jpg");
}
</style>
</head>
<body>

<ul>
  <li>Caderno</li>
  <li>Caneta</li>
  <li>Papel</li>
</ul>

</body>
</html>
```

O exemplo acima não exibe igualmente em todos os *browsers*. IE e Opera irão exibir o marcador de imagem um pouco maior do que o Firefox, Chrome e Safari. Se você quiser que o marcador de imagem seja colocado igualmente em todos os navegadores, uma solução é a propriedade crossbrowser.

Crossbrowser: habilidade de um site suportar múltiplos navegadores indiferente da linguagem.

O exemplo a seguir exibe o marcador de imagem igualmente em todos os navegadores:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
    list-style-type: none;
    padding: 0px;
    margin: 0px;
}

ul li {
    background-image: url(logo.jpg);
    background-repeat: no-repeat;
    background-position: 0px center;
    padding-left: 15px;
}
</style>
</head>
<body>

<ul>
  <li>Caneta</li>
  <li>Caderno</li>
  <li>Papel</li>
</ul>
</body>
</html>
```

Explicando o exemplo:

Para ul foi definido o list-style-type: none para remover o marcador da lista. Foi definido tanto preenchimento e margem para 0px (para compatibilidade cross-browser).

Para todos os ul li foi definido a URL da imagem, e mostrada apenas uma vez (no-repeat).

A imagem pode ser exibida onde quiser (0px esquerda e valor vertical: centro). O texto pode ser posicionado com a propriedade padding-left.

A propriedade list-style é uma propriedade usada para definir todas as propriedades da lista em uma declaração. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
    list-style: square inside url("logo.jpg");
}
</style>
</head>
<body>

<ul>
  <li>Caderno</li>
  <li>Caneta</li>
  <li>Papel</li>
</ul>

</body>
</html>
```

6 TABELAS EM CSS

A visualização de uma tabela HTML pode ser muito melhorada com CSS.

Para especificar as bordas da tabela em CSS, use a propriedade border. O exemplo a seguir especifica uma borda preta para os elementos th e td:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
    border: 1px solid black;
}
</style>
</head>
<body>

<table>
  <tr>
    <th>Produto</th>
    <th>Valor</th>
  </tr>
  <tr>
    <td>Caneta</td>
    <td>R$1,50</td>
  </tr>
  <tr>
    <td>Caderno</td>
    <td>R$6,80</td>
  </tr>
</table>
</body>
</html>
```

Observe que a tabela no exemplo acima tem bordas duplas. Isso porque tanto a tabela como os elementos th/td têm bordas separadas. Para exibir uma única borda para a tabela, use a propriedade border-collapse.

A propriedade border-collapse define se as bordas da tabela são únicas ou separadas. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
    border-collapse: collapse;
}
table, td, th {
    border: 1px solid black;
}
</style>
</head>

<body>
<table>
  <tr>
    <th>Produto</th>
    <th>Valor</th>
  </tr>
  <tr>
    <td>Caneta</td>
    <td>R$1,50</td>
  </tr>
  <tr>
    <td>Caderno</td>
    <td>R$6,80</td>
  </tr>
</table>
</body>
</html>
```

Largura e altura de uma tabela são definidas pelas propriedades width e height. O exemplo que segue define a largura da tabela para 100%, e a altura dos elementos th para 50px:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
    border: 1px solid black;
}

table {
    width: 100%;
}

th {
    height: 50px;
}
</style>
</head>
<body>

<table>
  <tr>
    <th>Produto</th>
    <th>Quantidade</th>
    <th>Valor</th>
  </tr>
  <tr>
    <td>Caneta</td>
    <td>10</td>
    <td>R$1,50</td>
  </tr>
  <tr>
    <td>Caderno</td>
    <td>15</td>
    <td>R$6,80</td>
  </tr>
  <tr>
    <td>Papel</td>
    <td>50</td>
    <td>R$0,35</td>
  </tr>
  <tr>
    <td>Cartolina</td>
    <td>20</td>
    <td>R$1,20</td>
</tr>
```

RESUMO DO TÓPICO 2

Neste tópico, você viu que:

- Em CSS, o texto pode ser alinhado, possuir cor, o parágrafo pode ser recuado, alinhado e/ou espaço entre caracteres.
- Que a propriedade font define a fonte de um texto.
- Que a propriedade font-family deve conter vários nomes de fontes para garantir que/se o navegador não possuir a primeira fonte possa carregar a segunda e assim sucessivamente.
- A propriedade font-size define o tamanho do texto.
- Os *links* podem ser decorados com qualquer propriedade CSS.
- As listas diferentes de HTML também podem conter figuras como marcador de lista.
- Que com CSS as visualizações das tabelas podem ser mais sofisticadas.

AUTOATIVIDADE

1 Observe o trecho de código-fonte CSS a seguir:

p{background-image: url(“azul.jpg”);}

a) Somente os elementos que utilizarem a classe p ficarão com a imagem de fundo azul.jpg
b) Somente os elementos que utilizarem a classe p ficarão com o fundo da cor azul.
c) Todos os parágrafos ficarão com a cor azul.
d) Todos os parágrafos ficarão com a imagem de fundo azul.jpg.

2 Hoje é um efeito comum em páginas *web* os *links* serem sublinhados apenas quando o internauta posicionar o ponteiro do *mouse* sobre eles. A propriedade CSS que permite retirar ou colocar o sublinhado nos *links* é:

a) link-style.
b) text-decoration.
c) underline.
d) link-style-type.

3 Com a utilização de Cascading Style Sheets (CSS), é possível configurar que a fonte de um elemento HTML seja exibida em itálico, essa configuração é feita com a utilização da propriedade:

a) font-weight.
b) font-variant.
c) font-stretch.
d) font-style.

TÓPICO 3

JAVASCRIPT

1 INTRODUÇÃO

Neste tópico você conhecerá um pouco da linguagem JavaScript.

O JavaScript é utilizado em documentos HTML sob a forma de funções também conhecidas por applets, as quais são chamadas em determinadas situações ou em respostas em determinados eventos. As funções podem estar localizadas em qualquer parte do código HTML. (COSTA, TODESCHINI, 2006).

Além disso, JavaScript é uma das três línguas que todos os desenvolvedores *web* devem conhecer. O HTML foi visto na Unidade 2 para definir o conteúdo de páginas da *web*. O CSS visto nos tópicos anteriores para especificar o *layout* de páginas da *web* e o JavaScript para programar o comportamento de páginas da *web*.

2 CONHECENDO O JAVASCRIPT

JavaScript é uma linguagem para criação de páginas *web*. As funções podem ser embutidas dentro do HTML. Porém Java e JavaScript são duas linguagens diferentes. Java é uma linguagem de programação e JavaScript é uma linguagem de hipertexto (COSTA, TODESCHINI, 2006).

Quando trabalhamos JavaScript em HTML a declaração deve iniciar com <SCRIPT> e finalizar com </SCRIPT>. Não se preocupe veremos adiante exemplos.

3 VARIÁVEIS EM JAVASCRIPT

Conforme Costa e Todeschini (2006), as variáveis dinâmicas podem ser criadas e iniciadas sem declarações formais. As variáveis dividem-se em globais e locais.

As globais são declaradas fora de uma função e podem ser acessadas de qualquer parte do programa.

Já as variáveis locais são declaradas dentro de uma função e somente podem ser utilizadas dentro da função onde foram criadas e precisam ser definidas com a instrução Var.

As variáveis devem sempre começar por uma letra ou por um caractere sublinhado "_", o restante da definição pode conter qualquer letra ou número. Lembre-se de que uma variável, como exemplo "Codigo" é diferente da variável "codigo" que por sua vez também é diferente da variável "CODIGO", portanto, tome cuidado quando definir o nome das variáveis, utilize sempre o mesmo padrão para defini-las. Podem existir variáveis globais com o mesmo nome de variáveis locais, porém isto não é recomendado (COSTA, TODESCHINI, 2006).

Existem três tipos de variáveis. As numéricas, como seu nome já diz, armazenam números. As booleanas que armazenam valores lógicos (true/false) e as strings que armazenam sequências de caracteres.

As strings podem ser delimitadas por aspas simples ou duplas, a única regra é se começar com aspas simples deve terminar com as mesmas e assim sucessivamente (COSTA, TODESCHINI, 2006).

4 CONHECENDO NA PRÁTICA

JavaScript pode ser colocado no <body> e <head> em seções de uma página HTML. Em HTML, o código JavaScript deve ser inserido entre <script> e </ script> tags. Exemplo**:**

```
<script>
document.getElementById ("demo") innerHTML = "Conhecendo JavaScript.";
</ script>
```

A função JavaScript é um bloco de código JavaScript, que pode ser executado quando você "pedir" para parar. Por exemplo, uma função pode ser executada quando ocorre um evento, como quando o usuário clica em um botão.

Você pode colocar qualquer número de scripts em um documento HTML. Scripts podem ser colocados no <body>, ou na seção <head> de uma página HTML, ou em ambos.

Neste exemplo, uma função JavaScript é colocado na seção <head> de uma página HTML. A função é chamada quando um botão é clicado:

```
<!DOCTYPE html>
<html>
<head>
<script>
function altera() {
    document.getElementById("teste").innerHTML = "Boa Tarde!";
}
</script>
</head>

<body>

<h1>Bem-vindo!</h1>

<p id="teste">Bom dia!</p>

<button type="button" onclick="altera()">Clique</button>

</body>
</html>
```

Neste exemplo criamos uma função (function) com o nome de altera. Dentro desta função podemos observar que temos o comando document. getElementById () na qual sua função é retornar o objeto de qualquer elemento na página que tenha um id único. Mais adiante temos o comando <button type="button" onclick="altera()">Clique</button> no qual estamos criando um botão com o nome de clique na qual, ao ser clicado chamará a função altera. Para compreendermos melhor veja como o navegador exibirá:

Quando clicarmos no botão Clique aparecerá a seguinte tela:

Lembre-se de que este mesmo exemplo pode estar dentro de <body>. Tente utilizar este exemplo e veja o resultado.

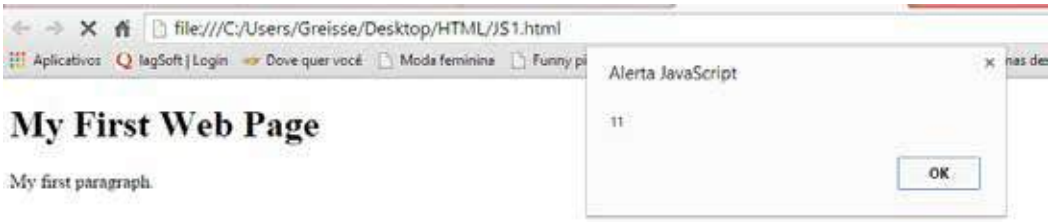
O JavaScript não possui nenhuma função de visualização embutidas. Ele apenas consegue exibir utilizando as seguintes maneiras:

- Escrevendo em uma caixa de alerta, usando window.alert ().
- Escrevendo para a saída HTML usando document.write ().
- Escrevendo em um elemento HTML, usando innerHTML.
- Escrevendo para o console do navegador, usando console.log ().

Vejamos um exemplo utilizando a caixa de alerta:

```
<!DOCTYPE html>
<html>
<body>

<h1>Bem-Vindo!</h1>
<p>Vejamos o exemplo.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

Este código, quando executado, exibirá uma janela com a soma de 5 + 6. Vejamos a seguir:



4.1 SINTAXE

As regras, como uma língua é construída, são chamadas de sintaxe da linguagem. As sentenças em uma linguagem de programação são chamadas declarações de programa. Um programa de computador é uma sequência de "comandos executáveis" chamados declarações. As instruções JavaScript são separadas por ponto e vírgula.

Instruções JavaScript são compostas por valores, variáveis, operadores, expressões, palavras-chave e comentários. Com o JavaScript, as regras mais importantes para escrever valores são:

- Os números são escritos com ou sem casas decimais, e com ou sem notação científica (e). Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Escrevendo números.</p>

<p id="teste"></p>

<script>
document.getElementById("teste").innerHTML = 10.50;
</script>

</body>
</html>
```

- Strings são escritas com aspas duplas ou simples. Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Utilizando strings.</p>

<p id="teste"></p>

<script>
document.getElementById("teste").innerHTML = 'UNIASSELVI';
</script>

</body>
</html>
```

- Expressões de valor como +,-,/,*. Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Utilizando expressões.</p>

<p id="teste"></p>

<script>
document.getElementById("teste").innerHTML = 5 * 10;
</script>

</body>
</html>
```

Em uma linguagem de programação, as variáveis são utilizadas para armazenar valores. JavaScript usa a palavra-chave var para definir variáveis. Um sinal de igual é usado para atribuir valores a variáveis.

Neste exemplo, x é definido como uma variável. Em seguida, é atribuído a x (o valor de 6):

```
<!DOCTYPE html>
<html>
<body>

<p>Exemplo de variável:</p>

<p id="teste"></p>

<script>
var x;
x = 6;
document.getElementById("teste").innerHTML = x;
</script>

</body>
</html>
```

Todas as variáveis Javascript (e funções de JavaScript) devem ser identificadas com nomes exclusivos. Estes nomes originais são chamados identificadores. Os identificadores podem ser nomes curtos (como x e y), ou nomes mais descritivos (idade, soma, totalVolume). As regras gerais para a construção de nomes para variáveis (identificadores únicos) são:

- Os nomes devem começar com uma letra.
- Nomes também podem começar com $ e _ (mas não vamos usá-lo).
- Os nomes podem conter letras, números, sublinhados e cifrões.
- Os nomes maiúsculos são diferentes de minúsculos (y e Y são variáveis diferentes).

As palavras reservadas (como palavras-chave JavaScript) não podem ser utilizadas como nomes. Nem todas as instruções JavaScript são "comandos executáveis". Qualquer coisa depois de barras duplas // é tratado como um comentário. Os comentários são ignorados, e não será executado. Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p id="teste"></p>

<script>
var x = 1;
// var x = 5 + 6; Este comando não será executado.
var y = x * 10;
document.getElementById("teste").innerHTML = y;
</script>

</body>
</html>
```

Em HTML, JavaScript declarações são "linhas de comando" executados pelo navegador da *web*. O objetivo das declarações é dizer ao navegador o que fazer.

Esta declaração diz ao navegador para escrever "Olá Acadêmico" dentro de um elemento HTML identificado com id = "teste": Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p id="teste"></p>

<script>
document.getElementById("teste").innerHTML = "Olá Acadêmico.";
</script>

</body>
</html>
```

Instruções JavaScript, muitas vezes, começam com uma palavra-chave para identificar a ação JavaScript para ser realizada. Palavras-chave são palavras reservadas, que não podem ser utilizadas como nomes de variáveis (ou quaisquer outras coisas). Abaixo está uma lista de algumas das palavras-chave que você deve conhecer:

TABELA 9 – PALAVRAS-CHAVE

Palavra-Chave	Descrição
Break	Encerra um loop
Catch	Marca o bloco de instruções a serem executadas quando ocorre um erro
continue	Salta fora de um ciclo e começa no topo
debugger	Para a execução de JavaScript, e chamada (se disponível) a função de depuração
do ... while	Executa um bloco de instruções, e repete o bloco, enquanto a condição for verdadeira
For	Marca um bloco de instruções a ser executado, desde que uma condição é verdadeira
for ... in	Marca um bloco de instruções a serem executadas, para cada elemento de um objeto (ou array)
function	Declara uma função
if ... else	Marca um bloco de instruções a serem executadas, dependendo de uma condição
Return	Sai de uma função
Switch	Marca um bloco de instruções a serem executadas, dependendo casos diferentes
Throw	Lança (gera) um erro
Try	Implementa o tratamento de erros para um bloco de instruções
var	Declara uma variável
While	Marca um bloco de instruções a serem executadas, enquanto uma condição for verdadeira

FONTE: W3SCHOOLS (2014)

4.2 FUNÇÕES

Uma função JavaScript é definida com a palavra-chave function, seguida por um nome, seguida de parênteses (). Nomes de funções podem conter letras, números, sublinhados e cifrões (mesmas regras que as variáveis). Os parênteses podem incluir nomes de parâmetros separados por vírgulas: (parametro1, parametro2, ...). O código a ser executado, pela função, é colocado dentro de colchetes: {}

```
functionName (parametro1, parametro2, parametro3) {
    código a ser executado
}
```

Os parâmetros da função são os nomes listados na definição da função. Argumentos da função são os valores reais recebidos pela função quando é invocada. Dentro da função, os argumentos são usados como variáveis locais. A função é a mesma coisa que um procedimento ou uma sub-rotina, em outras linguagens de programação.

O código dentro da função será executado quando chamada a função:

- Quando ocorre um evento (quando um usuário clica em um botão).
- Quando é invocado (chamado) a partir do código JavaScript.
- Automaticamente (autoinvocado).

Quando JavaScript atinge uma instrução de retorno, a função irá parar a execução. Se a função foi chamada a partir de um comunicado, JavaScript vai "voltar" para executar o código depois que a instrução for chamada. Funções frequentemente calculam um valor de retorno. O valor de retorno é "devolvido" de volta para o "chamador".

4.3 DECLARAÇÕES CONDICIONAIS

Declarações condicionais são usadas para executar ações diferentes com base em diferentes condições. Muitas vezes, quando você escrever código, você quer executar ações diferentes para diferentes decisões. Você pode usar declarações condicionais em seu código para fazer isso. Em JavaScript temos as seguintes declarações condicionais:

- Utilize-se para especificar um bloco de código a ser executado, se uma condição especificada for verdadeira.
- Use-se para especificar um bloco de código a ser executado, se a mesma condição é falsa.
- Usa-se else if para especificar uma nova condição para testar, se a primeira condição é falsa.
- Use-se a opção de especificar muitos blocos alternativos de código a ser executado.

4.3.1 Instrução if

Use a instrução if para especificar um bloco de código JavaScript a ser executado se a condição for verdadeira. Sintaxe:

```
if (condição) {
    bloco de código a ser executado se a condição for verdadeira
}
```

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Quando clicar no botão aparecerá Boa tarde, somente se a hora for menor que 20:00</p>

<button onclick="hora()">Try it</button>

<p id="teste"></p>

<script>
function hora() {
    if (new Date().getHours() < 20) {
        document.getElementById("teste").innerHTML = "Boa tarde.";
    }
}
</script>

</body>
</html>
```

4.3.2 Declaração else

Use a instrução else para especificar um bloco de código a ser executado se a condição for falsa.

```
if (condição) {
    bloco de código a ser executado se a condição for verdadeira
} Else {
    bloco de código a ser executado se a condição for falsa
}
```

Exemplo: Se a hora for menor que 20:00, criar uma saudação "Boa tarde", senão, "Boa noite":

```
<!DOCTYPE html>
<html>
<body>

<p>Clique no botão para disparar a saudação</p>

<button onclick="hora()">Clique</button>

<p id="teste"></p>

<script>
function hora() {
    var saudacacao;
    if (new Date().getHours() < 20) {
        saudacao = "Boa tarde.";
    } else {
        saudacao = "Boa noite.";
    }
    document.getElementById("teste").innerHTML = saudacao;
}
</script>

</body>
</html>
```

4.3.3 Loops

Loops podem executar um bloco de código de acordo com o número de vezes declarado. São úteis, se você deseja executar o mesmo código repetidas vezes, cada vez com um valor diferente. Muitas vezes, este é o caso quando se trabalha com agregados. Em vez de escrever:

```
texto + = carros [0] + "<br>";
texto + = carros [1] + "<br>";
texto + = carros [2] + "<br>";
texto + = carros [3] + "<br>";
texto + = carros [4] + "<br>";
texto + = carros [5] + "<br>";
```

Você pode escrever:

```
for (i = 0; i <carro; i ++) {
    texto + = carros [i] + "<br>";
}
```

Existem diferentes tipos de *loops*:

for - percorre um bloco de código um número de vezes;
for / in - percorre as propriedades de um objeto;
while - percorre um bloco de código enquanto uma condição especificada for verdadeira;

do/while - também percorre um bloco de código enquanto uma condição especificada for verdadeira.

O loop for é muitas vezes a ferramenta que você vai usar quando você quer criar um *loop*. O loop for tem a seguinte sintaxe:

for (declaração 1; declaração 2; afirmação 3) {
bloco de código a ser executado
}
Declaração 1 é executado antes do loop (o bloco de código) começa.

Declaração 2 define a condição para a execução do loop (o bloco de código).

Declaração de 3 é executado cada vez após o loop (o bloco de código) foi executado.

Exemplo:

```
<!DOCTYPE html>
<html>
<body>

<p>Clique no botão.</p>

<button onclick="numero()">Clique aqui</button>

<p id="teste"></p>

<script>
function numero() {
    var text = "";
    var i;
    for (i = 0; i < 5; i++) {
        text += "O número é " + i + "<br>";
    }
    document.getElementById("teste").innerHTML = text;
}
</script>

</body>
</html>
```

A partir do exemplo acima, você pode ler que:

- Declaração 1 define uma variável antes do laço começar (var i = 0).
- Declaração 2 define a condição para o loop para executar (i deve ser inferior a 5).
- Declaração 3 aumenta um valor (i ++) cada vez que o bloco de código no loop foi executado.

Normalmente, você vai usar uma declaração para iniciar a variável usada no loop (var i = 0). Isso nem sempre é o caso, no JavaScript a Declaração 1 é opcional. Você pode iniciar muitos valores na declaração 1 (separados por vírgula).

Você também pode omitir declaração 1 (como quando seus valores são definidos antes do início do ciclo).

Muitas vezes, a Declaração 2 é usada para avaliar a condição da variável inicial. A declaração 2 também é opcional. Se Declaração 2 retornar verdadeira, o ciclo vai começar tudo de novo, se ela retornar falsa, o ciclo vai terminar. Se você omitir Declaração 2, você deve fornecer uma ruptura dentro do *loop*. Caso contrário, o laço não terá fim. Isto irá travar o seu *browser.*

Muitas vezes, a Declaração 3 aumenta a variável inicial. Como as demais declarações, está também é opcional. A Declaração 3 pode fazer qualquer coisa como incremento negativo (i--), ou incremento positivo (i = i + 15), ou qualquer outra coisa.

O loop while percorre um bloco de código, desde que uma condição especificada for verdadeira. Sintaxe:

```
while (condição) {
    bloco de código a ser executado
}
```

No exemplo a seguir, o código do ciclo será executado, repetidas vezes, enquanto uma variável (i) é inferior a 10:

```
<!DOCTYPE html>
<html>
<body>

<p>Clique no botão para contar até 9.</p>

<button onclick="contagem()">Clique aqui</button>

<p id="teste"></p>

<script>
function contagem() {
    var text = "";
    var i = 0;
    while (i < 10) {
        text += "<br>O número é: " + i;
        i++;
    }
    document.getElementById("teste").innerHTML = text;
}
</script>

</body>
</html>
```

O do / while é uma variante do loop while. Este *loop* executará o bloco de código uma vez, antes de verificar se a condição for verdadeira, então ele vai repetir o ciclo enquanto a condição for verdadeira. Sintaxe:

```
do {
   bloco de código a ser executado
}
while (condição);
```

O exemplo a seguir usa um loop do/while. O circuito será sempre executado pelo menos uma vez, mesmo que a condição é falsa, porque o bloco de código é executado antes que a condição seja testada:

```
<!DOCTYPE html>
<html>
<body>

<p>Clique no botão para executar o loop.</p>

<button onclick="contagem()">Clique aqui</button>

<p id="teste"></p>

<script>
function contagem() {
    var text = ""
    var i = 0;
    do {
        text += "<br>O número é " + i;
        i++;
    }
    while (i < 10)
    document.getElementById("teste").innerHTML = text;
}
</script>

</body>
</html>
```

LEITURA COMPLEMENTAR

HTML + JAVASCRIPT
Explorando o Composite Pattern JavaScript

10 de novembro de 2014 por Pedro Araujo

Design Patterns é um assunto bem comum em todas as linguagens de programação, e o mais importante quando se trata em manutenção de código. Um pattern é uma solução reutilizável que pode ser aplicada em um projeto de desenvolvimento de *software*. Hoje, vamos explorar o Composite pattern com implementação JavaScript.

O composite pattern diz que um grupo de objetos pode ser tratado da mesma maneira que um objeto individual desse grupo.

Um uso bem comum de Composite Pattern que você provavelmente já tenha visto é o sistema de diretórios de um Sistema Operacional que utiliza estrutura de dados em árvore (considere que temos uma pasta e dentro dessa pasta temos várias outras pastas, e que dentro de cada uma dessas outras pastas temos mais algumas pastas, e assim por diante).

Nesse modelo, temos "N" objetos que possui "N" filhos que também pode ter mais "N" filhos. Entretanto, a quantidade de objetos não importa, porque a implementação é a mesma para todos eles. Ok? Vamos ver isso melhor na prática.

Exemplo: Construindo uma cidade com Composite Pattern

Vamos colocar a mão na massa e criar nosso próprio exemplo modificando um pouco o modelo apresentado acima, mas ainda seguindo a estrutura em árvore. No nosso exemplo vamos construir duas cidades que possuem bairros que, por sua vez, possuem algumas casas (lembre-se da estrutura em árvore). Algo bem simples e que vai ficar parecido com isto:

```
| cities
|
|– São Paulo/
| |– Liberdade
| |– Casa 1
| |– Casa 2
| |– Ipiranga
| |– Casa 3
|
```

|– Rio de Janeiro/
| |– Leblon
| |– Casa 4
| |– Lapa
| |– Casa 5
| |– Casa 6
|

O objeto possuirá os seguintes métodos específicos do nosso composite:
add: adiciona um novo filho para o objeto;
remove: remove um determinado filho do objeto;
getChild: retorna um filho do objeto;

Método auxiliar:
getElement: retorna o elemento HTML do objeto específico;

Criamos o objeto "Cidade" que terá o formato composite:

```
1  var City = function (title, id,
   className) {
2      this.children = [];

3      this.element = document.
   createElement('div');
4      var h2 = document.
   createElement('h2');

5
6      this.element.id =
   id;

7      if (className) this.element.className =
   className;
8
9      h2.textContent = title;
10                 this.element.
               appendChild(h2);

11
12  }

13
14   City.prototype =
     {
```

```
15      add: function (child) {
16          this.children.
    push(child);
19
        remove: function (child)
20  {
            for (var node, i = 0; node = this.getChild(i);
21  i++) {
22              if (node == child) {
23                  this.children.splice(i, 1);
                    this.element.removeChild(child.
24  getElement());
                    return
25  true;
26              }
27          }
28
            return
29  false;
30      },
31
        getChild: function (i)
32  {
            return this.
33  children[i];
34      },
35
        getElement: function
36  () {
            return this.
37  element;
38      }
39  }
```

Instanciamos os objetos e montamos a estrutura

```
1  var cities = new City('', 'cities');
2  var saoPaulo = new City('São Paulo', 'sao-
   paulo');
3  var rioDeJaneiro = new City('Rio de Janeiro', 'rio-de-
   janeiro');
4  var liberdade = new City('Liberdade', 'liberdade');
5  var ipiranga = new City('Ipiranga',
   'ipiranga');
6  var lapa = new City('Lapa', 'lapa');
7  var leblon = new City('Leblon', 'leblon');
8  var casa1 = new City('Casa 1', 'casa-1', 'composite-
   house');
9  var casa2 = new City('Casa 2', 'casa-2', 'composite-
   house');
10                 var casa3 = new City('Casa 3', 'casa-3',
                   'composite-house');
11 var casa4 = new City('Casa 4', 'casa-4', 'composite-
   house');
12 var casa5 = new City('Casa 5', 'casa-5', 'composite-
   house');
13 var casa6 = new City('Casa 6', 'casa-6', 'composite-
   house');
14 var casaRemover7 = new City('Casa remover 7', 'casa-
   remover-7', 'composite-house');
15
16 liberdade.
   add(casa1);
17 liberdade.
   add(casa2);
18
19 ipiranga.add(casa3);
20 ipiranga.
   add(casaRemover7);
```

```
21
22  ipiranga.remove(casaRemover7); //
    Removido
23
24  lapa.
    add(casa4);
25
26  leblon.
    add(casa5);
27  leblon.
    add(casa6);
28
29  saoPaulo.
    add(liberdade);
30  saoPaulo.
    add(ipiranga);
31
32  rioDeJaneiro.
    add(lapa);
33  rioDeJaneiro.
    add(leblon);
34
35  cities.add(saoPaulo);
36  cities.
    add(rioDeJaneiro);
37
38  document.body.appendChild(cities.
    getElement());
```

FONTE: Disponível em: <http://imasters.com.br/front-end/javascript/explorando-o-composite--pattern-javascript/>. Acesso em: 20 out. 2014.

RESUMO DO TÓPICO 3

Neste tópico, você estudou os seguintes conteúdos referentes à JavaScript:

- As variáveis em JavaScript e como devemos aplicá-las no código-fonte.
- Como o JavaScript é aplicado dentro do HTML.
- Conhecemos as regras, ou seja, a sintaxe utilizada dentro de JavaScript.
- Vimos exemplos teóricos e práticos de como aplicar as funções.
- As declarações condicionais como if/else.
- Os loops for e do/while.

AUTOATIVIDADE

1 Na linguagem Javascript, uma variável declarada que possa ser acessada a qualquer momento é denominada variável:

a) De método.
b) De script.
c) Global.
d) Local.
e) De função.

2 Em JavaScript, as estruturas de repetição utilizadas podem ser:

a) for, in e do while.
b) for, if e do while.
c) if, while e do while.
d) for, while e if.
e) for, if e else.

3 É necessário que as variáveis tenham a definição de um tipo antes de serem utilizadas em um programa JavaScript. Analise a afirmação e responda:

( ) Verdadeiro ( ) Falso

REFERÊNCIAS

APACHE. ***HTTP Server Project***. Disponível em: <http://httpd.apache.org/ABOUT_APACHE.html>. Acesso em: 20 out. 2014.

BRASIL, Cyclades. **Guia internet de conectividade**. São Paulo: Editora SENAC, 2001.

COSTA, Ramon Gomes; TODESCHINI, Leonardo. ***WEB*. Como programar usando ferramentas livres**. Alta Books: Rio de Janeiro, 2006.

DZENDZIK, Isolete Teresinha. **Processo de desenvolvimento de *websites* com recursos da UML**. Dissertação de Mestrado do Curso de Pós-graduação em Computação Aplicada. INPE: São José dos Campos, 2005. Disponível em: <http://www.unafiscobrasilia.org.br/unafisco/atos/53.pdf>. Acesso em: 20 set. 2014.

Internet Society. ***Brief History of the internet***. Disponível em: <http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet#JCRL62>. Acesso em: 10 ago. 2014.

MICROSOFT. ***A flexible & easy-to-manage webserver***. Disponível em: <http://www.iis.net/>. Acesso em: 20 out. 2014.

NETBEANS. ***NetBeans IDE Features***. Disponível em: <https://netbeans.org/features/index.html>. Acesso em: 20 set. 2014.

W3C. ***Web design and applications***. Disponível em: <http://www.w3.org/standards/webdesign/>. Acesso em: 20 set. 2014.

W3C. ***Welcome to Amaya***. Disponível em: <http://www.w3.org/Amaya/>. Acesso em: 20 set. 2014.

COSTA, Ramon Gomes; TODESCHINI, Leonardo. ***WEB***. Como programar usando ferramentas livres. Alta Books: Rio de Janeiro, 2006.

FREEMAN, Elisabeth; FREEMAN, Eric. **Use a cabeça**. HTML com CSS e XHTML. Rio de Janeiro: Alta Books, 2008.

W3SCHOOL. ***The world´s largest web development site***. Disponível em: <http://www.w3schools.com/>. Acesso em: out. 2014.

COSTA, Ramon Gomes; TODESCHINI, Leonardo. **WEB. Como programar usando ferramentas livres.** Alta Books: Rio de Janeiro, 2006.

SILVA, Mauricio Samy. **CSS3**: desenvolva aplicações profissionais com o uso dos poderosos recursos de estilização das CSS3. NOVATEC: São Paulo, 2012.

W3SCHOOL. **The world´s largest web development site.** Disponível em: <http://www.w3schools.com/>. Acesso em: 20 out. 2014.