Tecnologias para Web e para Dispositivos Móveis

Tecnologias para web e para dispositivos móveis

Kleber Ricardi Rovai
Ruy Flávio de Oliveira
Thiago Salhab Alves
Vanessa Cadasn Scheffer
Roque Maitino Neto

Dados Internacionais de Catalogação na Publicação (CIP)

Rovai, Kleber Ricardi
R873t Tecnologias para web e para dispositivos móveis / Kleber Ricardi Rovai, et al. – Londrina : Editora e Distribuidora Educacional S.A., 2018.
192 p.

ISBN 978-85-522-0732-0

1. Tecnologia. 2. Web. 3. Dispositivos móveis. I. Rovai, Kleber Ricardi. II. Título.

CDD 600

Thamiris Mantovani CRB-8/9491

2018
Editora e Distribuidora Educacional S.A.
Avenida Paris, 675 – Parque Residencial João Piza
CEP: 86041-100 — Londrina — PR
e-mail: editora.educacional@kroton.com.br
Homepage: http://www.kroton.com.br

Sumário

Palavras do autor

Caro aluno, seja bem-vindo à disciplina de tecnologias para web e para dispositivos móveis. Você já parou para pensar na importância que as tecnologias para Web e para dispositivos móveis têm em nossas vidas e em como elas são, hoje em dia, diferentes das vidas que levávamos antes dessa tecnologia estar disponível em larga escala? Pois é, nestas últimas duas décadas a Web vem provocando uma enorme transformação, facilitando o acesso à informação e permeando inúmeras atividades em nosso dia a dia. Bancos, notícias, entretenimento e compras nunca mais foram os mesmos depois da introdução dessa tecnologia, lá no início da década de 1990. É importante observar que a Internet já vinha unindo computadores e disponibilizando informações desde o fim da década de 1960, mas foi apenas com a chegada da Web que a Internet passou a revolucionar nossas vidas em larga escala.

E tem mais: sabe a revolução que os dispositivos móveis (*smartphones* e *tablets*) vêm provocando em nossas vidas, permitindo que carreguemos as facilidades da Internet em nossos bolsos e mochilas para onde quer que queiramos ir? Pois então: esta chamada "revolução móvel" tem seu alicerce na Web. Foi sobre a infraestrutura, a filosofia de disponibilização de informações de forma simples e por meio de código aberto que os fabricantes de hardware e software móveis criaram suas plataformas que dão ênfase à mobilidade. Os chamados "apps", que tanto facilitam nossas vidas hoje em dia, já não se utilizam tanto das linguagens e protocolos da Web, mas só puderam emergir e se tornar o que são hoje porque lá no começo — no "longínquo" fim da década de 2000 — puderam utilizar as peças criadas pela Web.

Então, para aproveitar melhor tudo isso, nesta unidade curricular você vai conhecer e compreender os principais fundamentos, ferramentas e linguagens relacionadas ao desenvolvimento Web. Afinal de contas, o mercado está cheio de oportunidades para quem se dispõe a aprender como criar e manter *sites*.

Na primeira unidade de estudo, você entrará em contato com os elementos de design de um Website. Entenderá melhor as cores, as fontes, as disposições de conteúdo e tudo aquilo que

torna a experiência do usuário na Web realmente agradável (ou horrivelmente desagradável, se não for feito direito).

Na segunda unidade, você entrará em contato com as principais peças do quebra-cabeça que compõe o conteúdo da Web: as linguagens de marcação, de programação e de *script* por meio das quais o conteúdo escrito, e também o conteúdo multimídia, são construídos e disponibilizados.

Na terceira unidade do nosso curso, você será imerso na tecnologia de dispositivos móveis, onde será trabalhada a evolução da tecnologia móvel, assim como os seus tipos e sistemas operacionais para o seu funcionamento.

Vamos terminar o curso aplicando ferramentas e técnicas de programação para criação de aplicativos para os dispositivos móveis.

Persista no seu aprendizado e boa sorte.

Elementos de desenvolvimento para tecnologias web

Convite ao estudo

Atualmente é perceptível a evolução da Web em relação à sua disponibilização inicial no começo da década de 90, mas o fato é que as coisas ficaram mais fáceis, mais intuitivas e mais agradáveis quando navegamos e realizamos as ações necessárias ao nosso dia a dia na Web. Se você está tendo dificuldade de visualizar o quanto evoluímos neste sentido, veja o *site* {404} Page Found (2016) (disponível em <http://www.404pagefound.com>) que disponibiliza acesso a websites antigos que ainda estão funcionando. São *sites* que desde (pasme!) 1993 estão no ar, e uma breve navegada por alguns deles te dará uma ideia do quanto já caminhamos desde então. Comparar um daqueles primeiros sites com o que temos à nossa disposição hoje em dia é como comparar uma carroça medieval com o último modelo de um carro de luxo, não é mesmo?

Pois bem, as facilidades de navegação que temos à nossa disposição hoje em dia são exatamente o assunto desta primeira unidade de nossos estudos. Aliás, aqui você vai conhecer e compreender os principais fundamentos, ferramentas e linguagens relacionadas ao desenvolvimento Web. Afinal de contas, não é porque temos muita tecnologia à nossa disposição para criar websites que todos eles serão de boa qualidade. É necessário prestar muita atenção ao equilíbrio de cores, à programação visual da página, às fontes tipográficas que são utilizadas para disponibilizar as informações e à arquitetura do site e de suas páginas. Afinal de contas, com mais de um bilhão de sites disponíveis na Web hoje em dia, segundo a Internet Live Stats (2016), quem não oferecer uma experiência agradável ao usuário, verá seus visitantes optando por navegar pelos sites dos concorrentes.

Os objetivos de aprendizado da presente unidade são:

1. Entender como usar adequadamente os vários elementos de composição de uma página (cores, tipografia, imagens, vídeos, responsividade);
2. Aprender o que é Gestalt aplicada à Web, sua importância, e como utilizá-la para conseguir páginas e websites que atendam às necessidades de seus usuários;
3. Entender o que é a arquitetura de websites e páginas, e de como projetar melhor a interface de comunicação com o usuário em um website;
4. Entender o que é otimização de máquinas de busca e como organizar as informações de forma a dar mais visibilidade ao website.

Carla trabalha há bastante tempo em uma grande editora de livros didáticos. Por sua formação ser na área de marketing, ela supervisiona todas as publicações desta área, e seus colegas cuidam, cada um dentro de sua especialidade, das publicações em diversas outras áreas de conhecimento: línguas, administração e economia, engenharia, ciências exatas e biológicas, dentre outras. Vendo-se cercada por tanto conhecimento e, principalmente, por tanta gente capacitada, ela teve uma ideia: por que não criar um portal de cultura, com blogs e páginas criadas e mantidas pelos colegas, cada um em sua área de especialidade? Certa de que esta é uma ideia com bom potencial, mas sem conhecer nada acerca da criação e manutenção de websites, ela não demora a procurar você, que tem bons conhecimentos na área, e sabe como ajudá-la a concretizar sua ideia.

Será que só o fato de Carla ter um time de bons e capacitados escritores é suficiente para que o Website seja um sucesso? Que elementos devem ser planejados, e como deve ser este planejamento para que o site seja um sucesso?

Nossa aventura pelas tecnologias para web está começando.

Vamos em frente!

Seção 1.1

Ergonomia de recursos para tecnologias web

Diálogo aberto

Já reparou que às vezes você navega por sites que, apesar de serem de extrema importância para seu dia a dia, são difíceis de usar, lembrando-nos mais de uma punição do que de uma tarefa prazerosa ou mesmo simples? Em outros casos, o site é tão intuitivo, familiar e agradável que, apesar de ser a primeira vez que você navega por ele, parece que já é frequentador há muitos anos, de tão fácil que é obter os resultados que procura.

O que faz com que alguns sites sejam agradáveis e intuitivos, bonitos de se ver e fáceis de navegar, enquanto outros nos despertam frustração, ansiedade, impondo-nos dificuldades a cada passo do caminho? A resposta está em vários aspectos do website e de suas páginas, com os quais você vai entrar em contato a partir de agora. Cores, tipografia, imagens, sons e, principalmente, a organização destes elementos nas várias páginas de um site, compõem aquilo que podemos chamar (e que em seguida definiremos mais formalmente) de "ergonomia do website". É importante observar que a ergonomia deve ser mantida em qualquer dispositivo que tenha acesso à Web e que, portanto, possa acessar o site em questão. Sim, pois hoje em dia não somente computadores e desktops, mas também tablets e smartphones com os mais variados tamanhos e resoluções de tela, permitem acesso à Web.

É justamente nestes aspectos, isto é, sobre a ergonomia do site sendo criado por Carla, que você deverá se concentrar nesta fase do projeto. Como definir os elementos gráficos e tipográficos do site de forma a criar um ambiente agradável e adequado aos propósitos de Carla para seu projeto? Elementos audiovisuais serão necessários? Se sim, em que proporção e em que posições ao longo das páginas? Qual a disposição das informações e quais as paletas de cores mais adequadas para o portal de cultura de Carla? Sabendo que o público-alvo que Carla tem em mente é composto por profissionais e estudantes de nível superior interessados em discussões e ensaios

sobre aspectos culturais da sociedade contemporânea, como organizar o site de forma que este público se identifique com o site e decida tornar suas visitas uma atividade periódica e contínua? Como garantir que o site terá uma disposição adequada de suas informações em telas de vários tamanhos e resoluções (computadores, laptops, tablets e smartphones)? Você deverá entender todos esses aspectos e explaná-los à Carla, de forma que ela possa decidir sobre como proceder no tocante ao design de seu site.

Pronto para esse desafio?

Então, vamos lá!

Não pode faltar

A Web é uma das mais úteis ferramentas já desenvolvidas pelo homem, porém nem sempre foi assim. Antes dela, já nos comunicávamos via Internet e muito desenvolvimento teve que ocorrer para que as coisas fossem "fáceis" como são nos dias de hoje.

Para iniciarmos nossos estudos sobre tecnologias para web, é importante fazermos uma rápida passagem pelo caminho que percorremos até chegarmos aos dias de hoje.

Um breve histórico: da Guerra Fria, passando pela Internet, e chegando à Web

A Internet não é nada nova: como afirma Ryan (2010), ela foi implementada pela primeira vez como conceito na segunda metade da década de 1960, pelas forças armadas dos EUA, que temiam ataques nucleares por parte dos Soviéticos durante a Guerra Fria e buscavam meios de comunicação que permitissem que a rede continuasse no ar mesmo que algumas de suas partes fossem destruídas. Tendo demonstrado que o conceito de comunicação descentralizada funcionava, e adotado muitos de seus princípios em suas redes particulares, os militares publicaram seus estudos, e despertaram o interesse das universidades norte-americanas, que rapidamente se interessaram por desenvolver os protocolos de comunicação nascentes para, assim, interligar seus computadores e centros de pesquisa. Já no início da década de 70, vários dos protocolos de acesso remoto e compartilhamento de arquivos da Internet estavam em pleno funcionamento: FTP (transferência de arquivos), Telnet (acesso a sistemas e recursos remotos, execução de programas, etc.) e SMTP (correio eletrônico) eram os principais deles (RYAN, 2010).

Durante mais de duas décadas — do início da década de 1970 até o início da década de 1990 — a Internet cresceu e floresceu como ferramenta de comunicação entre entidades de ensino e, mesmo, entre várias empresas do setor de tecnologia (a IBM e a Digital Corporation estavam na frente dessa fila). Contudo, o acesso do grande público a este crescente manancial de informações ainda não havia acontecido, e Ryan (2010) especula que a ausência de facilidades das ferramentas de comunicação aliada ao alto custo de acesso colaborou para este isolamento.

Essa situação começou a mudar a partir de março de 1989, quando Tim Berners-Lee, cientista do CERN (Centre Européenne pour la Recherche Nucléaire, ou Centro Europeu de Pesquisas Nucleares) criou e, mais tarde, no mesmo ano, implementou a comunicação entre computadores utilizando um novo protocolo que previa a marcação e editoração de texto, inclusão de figuras, e ligação de elementos externos. O protocolo se chamava HTTP (*Hypertext Transfer Protocol*) e a linguagem de marcação era o HTML (*Hypertext Mark-up Language*). Era o nascimento da *World Wide Web* (Rede de Alcance Mundial) e o início da massificação da Internet (RYAN, 2010).

A Web hoje: foco no usuário

Diferentemente das primeiras duas décadas da internet, e mesmo dos primórdios da Web, o foco nos dias de hoje é o usuário. Se nos primórdios da Web, como nos afirma Ryan (2010), o foco foi a disseminação das informações e o crescimento da base de usuários com acesso à infraestrutura (inclusão digital), nos dias de hoje as preocupações são bastante diferentes. Em 2015, segundo a Internet Live Stats (2016), ultrapassamos 3,1 bilhões de usuários da Web, ou seja, o acesso já é garantido a quase metade da população do planeta. Como, ainda segundo a Internet Live Stats (2016), já ultrapassamos um bilhão de *sites* ativos, também é visível que a disponibilização de informações é um sucesso.

Mas então, qual é o foco agora? O curioso é que a enorme quantidade de *sites* disponíveis gera um problema para qualquer um que tenha intenção de disponibilizar informações por meio da Web: há escolhas demais para o usuário e, a quantidade de escolhas só cresce a cada dia que passa.

Travis (2013) afirma que a tecnologia nos permite inúmeras formas de apresentar o conteúdo, mas nem sempre os *sites* mais

"tecnológicos" são os que fazem mais sucesso: faz mais sucesso aquele *site* que põe a experiência do usuário em primeiro lugar.

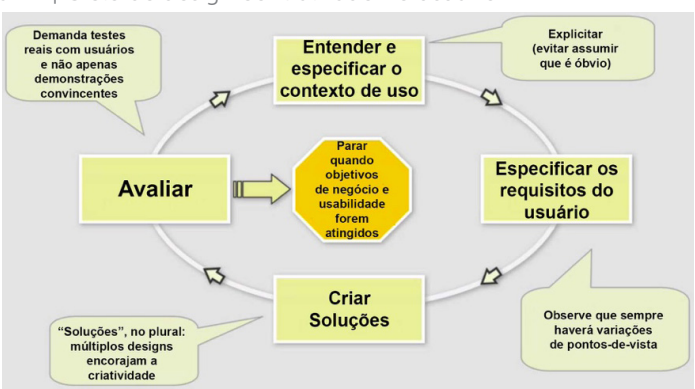
O assunto é tão importante que, no início de década de 2000, virou foco da ISO (*International Standards Organization*, ou Organização Internacional de Padronização) que publicou a família ISO 9241, que se preocupa especificamente em padronizar a experiência do usuário com os sistemas com os quais interage. Essa norma foi traduzida para o português pela ABNT, e hoje é conhecida como ISO/NBR 9241 (ABNT, 2000). Em especial, a ISO/NBR 9241-11, que leva o título de "Requisitos Ergonômicos para Trabalho de Escritórios com Computadores Parte 11 – Orientações sobre Usabilidade" é de interesse para quem quer que se disponha a desenvolver conteúdo para a Web.

A ISO/NBR 9241 postula três princípios de usabilidade, ou de centralização do processo de design no usuário. São eles (ABNT, 2000):

1. O foco do processo deve ser o usuário e as tarefas que este usuário precisa desempenhar, e esse foco deve começar cedo e ser contínuo.
2. O comportamento do usuário deve ser mensurado empiricamente ao longo de todo o processo.
3. O design é um processo iterativo, isto é, deve seguir um fluxo de implementação, mensuração e correção, repetindo-se estes três estágios até que os resultados esperados sejam atingidos.

A Figura 1.1, a seguir, ilustra este ciclo de design centralizado no usuário:

Figura 1.1 | Ciclo de design centralizado no usuário



Fonte: adaptado de ABNT (2000)

O objetivo principal dos três princípios de usabilidade é, obviamente, maximizar a usabilidade do website (ou do recurso, de maneira mais generalizada) sendo desenvolvido. Mas o que é usabilidade? Os autores Hartson e Pyla (2012) definem usabilidade como sendo o conjunto de componentes mais pragmáticos da experiência do usuário, conjunto este composto por elementos que podem ser observados e constatados:

- Eficiência; Produtividade; Facilidade de uso; Facilidade de aprendizado; Facilidade de retenção do conteúdo aprendido; Satisfação do usuário.

Cores

Alguns dos aspectos de usabilidade levantados por Hartson e Pyla (2012) são objetivamente mensuráveis. Por exemplo, a mudança na disposição de informações de um website pode aumentar a produtividade daqueles que o frequentam, e esta variação na produtividade pode ser mensurada. Outros aspectos são mais subjetivos e, entre eles, Miller (2011, p. 202) cita as cores como sendo o elemento mais poderoso para "guiar, direcionar e persuadir o usuário".

O esquema de cores em um website se encaixa naquilo que Hartson e Pyla (2012) definem como "a perspectiva emocional do design". Essa perspectiva, segundo os autores, se refere ao impacto emocional do design, bem como aos aspectos do design que dependem dos valores pessoais do indivíduo que o presencia. Esse aspecto tem implicações culturais e sociais, e se relaciona diretamente aos aspectos de estética e de quanto seu uso é agradável a quem o utiliza.

As respostas psicológicas que apresentamos quando submetidos às cores, segundo Bleicher (2012) são de duas naturezas:

- **Respostas inatas** – são efeitos da evolução e da seleção natural. Todos nós reagimos psicologicamente da mesma forma, por exemplo, à combinação de amarelo e preto: é uma combinação que sinaliza perigo, que demanda atenção, que nos põe em alerta. Um indivíduo pode nunca ter sido picado por uma abelha, mas ao ver um destes insetos, vai reagir com cautela, e isto é em função de sua combinação de cores;
- **Respostas aprendidas** – são efeitos que aprendemos ao longo da vida, e são dependentes da cultura em que estamos

inserimos. Na cultura ocidental, por exemplo, azul é uma cor ligada ao gênero masculino, enquanto rosa é uma cor ligada ao gênero feminino. Um exemplo de dependência da cultura é a cor branca, que no ocidente é a cor do vestido da noiva em seu casamento, enquanto na cultura indiana, a cor vermelha é a cor do vestido da noiva.

É justamente sobre estas respostas psicológicas às cores que surgem todas as técnicas de design que são usadas hoje em dia. Entender o significado das cores e as respostas psicológicas que provocam no indivíduo não é tarefa simples, mas é fundamental para todo bom designer de websites.

Pesquise mais

O designer David Arty publicou recentemente no *site* Chief of Design um excelente artigo intitulado "Guia Sobre Cores – Teoria das Cores", no qual aborda desde aspectos físicos e biológicos das cores até os aspectos mais psicológicos, comportamentais e de marketing associados às cores.

ARTY, D. **Guia Sobre Cores – Teoria das Cores**, 2014, Chief of Design. Disponível em: <http://chiefofdesign.com.br/teoria-das-cores/>. Acesso em: 10 abr. 2016.

Tipografia

A autora Krystin Cullen (2012) define tipografia como sendo o processo que torna a linguagem visível aos olhos, usando como ferramenta a formatação do texto por meio de tipos (fontes) que lhe atribuem fluência e (ainda segundo a autora) vida. Trata-se de um conjunto de técnicas para se organizar textos de forma que tenham (CULLEN, 2012):

- **Legibilidade** – característica do texto que pode ser identificado, decifrado e compreendido pelo leitor;
- **Atratividade textual** – Um texto ser "legível" significa que ele pode ser interpretado, e só. Afirmar que um texto é legível equivale a dizer que o capim é comestível: você até pode comer capim, mas certamente não vai fazê-lo a não ser em situações extremas. Da mesma forma, um texto tem que ter mais do que legibilidade. Mais do que poder ler um texto, como afirma Santa Maria (2014), é preciso querer lê-lo. E para tanto, é preciso que ele seja atrativo: é preciso que tenha a fonte certa, o espaçamento certo e a disposição certa para o conteúdo que se quer transmitir. Isso é atratividade textual;

- **Beleza estética** – A atratividade textual está ligada ao conforto, à ergonomia do texto, enquanto a beleza estética é um aspecto mais artístico, que diz respeito à beleza em si do conjunto do que se lê. É um conceito que cabe em alguns tipos de texto, mas não em outros.

Exemplificando

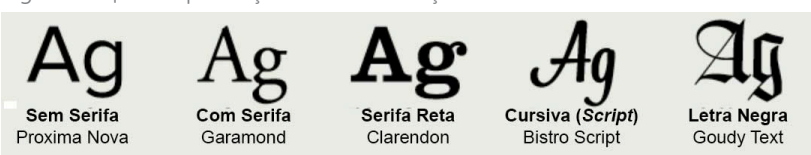
No cartaz de uma festa, ou no livreto de uma peça teatral, cabe o conceito de beleza estética no que concerne à tipografia desses textos, contudo, em um processo judicial, este conceito é desnecessário, e poderia se tornar uma distração prejudicial se estivesse presente.

As fontes tipográficas diferem enormemente umas das outras. E como são literalmente milhares delas à disposição dos designers, algum tipo de classificação se torna necessária. Santa Maria (2014) afirma corretamente que há inúmeras formas de se classificar as fontes, uma vez que qualquer classificação é arbitrária. Contudo, sugere a classificação mais utilizada entre os tipógrafos. São cinco as famílias de fontes (SANTA MARIA, 2014):

- **Sem serifa** (ou fontes não serifadas) – fontes sem apêndices nas extremidades dos caracteres;
- **Com serifa** (ou serifadas) – fontes com apêndices levemente curvos ou angulados nas extremidades dos caracteres;
- **Com serifa reta ou retangular** (do inglês *slab serif*) – fontes com apêndices retos nas extremidades dos caracteres;
- **Cursivas** (do inglês script) – fontes que imitam a escrita à mão;
- **Letras negras** (do inglês *Blackletter*) – fontes em estilo gótico antigo, incidentalmente, as mais antigas fontes tipográficas de que se tem notícia.

A Figura 1.2, a seguir, ilustra a classificação das fontes:

Figura 1.2 | Exemplificação da classificação de fontes



Fonte: adaptado de Santa Maria (2012).

Quando um designer se depara com essa imensidão de opções, é natural que pergunte "OK, quando uso esta ou aquela fonte? Quais são as regras de uso de fontes?", ao que Santa Maria corretamente responde (SANTA MARIA, 2012): "não existem regras em tipografia".

Há sim, segundo o autor, princípios, melhores práticas e métodos que funcionam na maior parte do tempo, mas como o autor observa, não existe algo que funcione o tempo todo.

A professora Denise Aristimunha de Lima (2012) oferece algumas regras de utilização para sua classificação de fontes:

- **Com serifa** – indicadas para textos longos pela legibilidade e atratividade textual que oferecem;
- **Sem serifa** – Indicadas para textos curtos, frases isoladas e títulos;
- **Com serifa reta** – indicadas para textos casuais, textos curtos e títulos;
- **Cursivas** – Textos estilizados, curtos e convites
- **Letras negras** – Indicadas para títulos de documentos oficiais tradicionais ou religiosos (diplomas, documentos cerimoniais, etc.);

Essas indicações são, de fato, tomadas a título de melhores práticas, e são bastante indicadas quando compomos websites e páginas da Web. Afinal de contas, estes documentos são exemplos de textos escritos, e o que vale para o texto impresso não tem porque não funcionar para o texto na tela do computador.

Pesquise mais

A designer Dani Guerrato (2012) escreveu um artigo bastante completo acerca da utilização adequada de fontes em páginas Web. Vale a pena explorar esse recurso para aprender mais acerca deste assunto.

GUERRATO, D. **Um guia completo de tipografia para a Web**, Tableless, 2012. Disponível em: <http://tableless.com.br/um-guia-completo-de-tipografia-para-a-web>. Acesso em: 10 abr. 2016.

Imagem e vídeo

Miller (2011) afirma que os usuários não leem detalhadamente uma página, mas sim dão uma "passada de olhos" pelas páginas (o autor usa o termo inglês *scan*). Neste sentido, vale a máxima de que uma imagem (desde que bem escolhida e usada no contexto correto) vale mais de que mil palavras. A imagem certa no lugar

certo pode adicionar informações importantes ao usuário da página, e pode, também, aumentar o interesse deste usuário pelo conteúdo escrito. O uso de imagens, enfatiza Miller (2011), deve ser deliberado, sem casualidade, com muito planejamento, e deve também buscar contribuir com a marca do *site* (*branding*) ou com a mensagem que a página em si esteja transmitindo a quem a visita. É importante observar que imagens, ao contrário de textos, adicionam mais volume de dados trafegados — volumes, aliás, sempre ordens de grandeza maiores —, ou seja, usar imagens só por usar imagens pode, no fim das contas, jogar contra a experiência do usuário que se deseja atingir. O mesmo vale para vídeos, com uma diferença: adicione uma ou duas ordens de grandeza ao volume de dados que trafegam para a montagem das páginas que contêm vídeos.

Para imagens e vídeos, alguns elementos devem ser observados (MILLER, 2011):

- **Escala** – A variação de tamanho dos elementos gráficos (imagens e vídeos) é uma das formas de se adicionar "dramaticidade" às páginas. Um elemento dominante (maior em escala que os demais elementos gráficos da página) naturalmente cria uma hierarquia visual, atraindo a atenção do usuário primeiramente para si. A Figura 1.3 mostra um exemplo de um elemento hierarquicamente dominante na página (tablet); imagens enormes em pano de fundo (*background*) podem ser usadas com moderação e emprestam um ar de exclusividade à página, mas repetições de pequenos quadros (*tiles*) não são mais aconselháveis (melhor deixa-las em seu lugar: na década de 1990);
- **Profundidade** – imagens em perspectiva, especialmente aquelas em que superfícies grandes se encontram em ângulo, dão a impressão de que a página — que é essencialmente um elemento bidimensional — pareça ter profundidade, ou seja, pareça ser um elemento tridimensional. A Figura 1.4, a seguir, mostra um exemplo desta sensação de terceira dimensão que se pode provocar em uma página;
- **Animação** – Os elementos gráficos podem, de fato, ser usados para criar uma dimensão extra de informações, pois o elemento não é uma figura estática, e em seu espaço uma história inteira pode ser contada. É um recurso que deve ser usado sem exageros sob pena de desviar a atenção do usuário e gerar um efeito desagradável;

- **Som** – O som acompanhando as imagens caiu em desuso, e usar este artifício automaticamente na carga da página é um dos erros mais graves a se cometer;
- **Variabilidade** – Um elemento gráfico nem sempre precisa ser o mesmo, e a variação tem o poder de melhorar (ou, se mal planejada, confundir) a experiência do usuário. Há elementos gráficos em páginas da Web que permitem, por exemplo, expor várias imagens e elementos gráficos em sucessão, como se fossem *slides* sendo constantemente trocados. Outro exemplo é a variação nos títulos em função de ocasiões específicas. A Figura 1.5, a seguir, mostra um exemplo de variação de títulos que o Google usa para marcar datas especiais.

Assimile

Se um texto é "legível" não significa, automaticamente, que é um texto "agradável de se ler", mas apenas que é possível absorver seu conteúdo, ainda que por conta da fonte ou de outras características textuais, essa absorção seja desagradável.

Figura 1.3 | Exemplo de imagem dominante

Fonte: <http://www.apple.com>. Acesso em: 12 abr. 2016.

Figura 1.4 | Exemplo de elemento gráfico dando profundidade à página

Fonte: <http://www.basilgloo.com/>. Acesso em: 12 abr. 2016.

Figura 1.5 | Exemplos de variabilidade nos títulos

Fonte: <http://www.google.com>. Acesso em: 12 abr. 2016.

Por fim, a ausência de um elemento (ao qual alguns web designers ainda se apegam, infelizmente) nas imagens é fundamental, por ter caído em desuso há mais de uma década: as molduras de imagens. Não fazem mais sentido, ainda mais diante das tendências de design em vigor desde o início da década de 2000. E, principalmente, a partir do início da década de 2010, com os designs flat dominando tanto a Web quanto os aplicativos de dispositivos móveis, passou a ser garantia de que o *site* que as emprega (as molduras) será visto como tendo um trabalho amador e de mau-gosto (MILLER, 2011).

Responsividade

Segundo Sharkie e Fisher (2013), responsividade é a capacidade de um website (e, consequentemente de suas páginas) de se adaptar a qualquer dispositivo, em qualquer dimensão de tela e qualquer resolução, com qualquer tipo de interface de interação, apresentando as informações e elementos visuais e respondendo ao usuário sempre de forma coerente e agradável. Esta não é uma tarefa trivial, uma vez que hoje em dia temos no mercado uma gama enorme de dispositivos com tamanhos de tela, resoluções e tipos de interface diferentes. As variações de tamanho de tela e resolução são facilmente visíveis quando comparamos, por exemplo, computadores desktop, laptops, tablets e smartphones. Um notebook da Dell terá a dimensão de tela e resolução diferente de um computador de mesa da HP, que será diferente de um tablet da Samsung, que será diferente de um iPhone, por exemplo. No quesito interface de interação, um computador ou um laptop usam mouse e teclado, enquanto tablets podem usar teclado, mas têm o mouse substituído pelo toque direto na tela, enquanto os smartphones só têm a tela. Entendeu como toda essa variedade pode impor dificuldades a quem desenvolve o website? A Figura 1.6, a seguir, exemplifica as diferenças:

Figura 1.6 | Exemplos de dispositivos diferentes, exigindo responsividade

Fonte: Sharkie e Fisher (2013).

Na prática, para que um website tenha responsividade, ou seja, para que seu conteúdo seja adequadamente exposto em qualquer dispositivo, algumas características deverão estar presentes:

- **Identificação do dispositivo e do navegador** – o servidor que entrega as páginas deverá ser capaz de identificar todas as características relevantes de hardware e software do dispositivo que pede pelas páginas a serem apresentadas;
- **Formatos de página flexíveis** – adaptáveis a qualquer dimensão e a qualquer resolução de tela, dispondo os elementos da forma mais racional para cada dispositivo;
- **Tamanhos alternativos para elementos gráficos** – muito importante, porque as resoluções variam muito, e tamanhos diferentes de elementos gráficos podem afetar sensivelmente os tempos de carga da página;
- **Formulários em formatos e disposições flexíveis** – formulários estáticos feitos, por exemplo, para uma tela grande, gerarão dificuldades de leitura em telas pequenas;
- **Elementos textuais de tamanho e disposição flexível** – tamanhos de fonte devem variar de acordo com o tamanho da tela.

Existem várias técnicas, que serão vistas ao longo desta unidade curricular, para que um website seja responsivo e seu design atenda às necessidades de quaisquer tipos de dispositivos.

Reflita

Um computador de mesa ou um laptop tem uma única resolução de tela, obviamente, mas você já parou para pensar que um tablet ou um smartphone tem, não uma, mas sim duas resoluções de tela a serem consideradas? Como isso se explica?

Sem medo de errar

Após estudar a proposição inicial de Carla, você deverá fazer um levantamento mais detalhado dos requisitos do site, especialmente no que concerne às necessidades dos usuários, uma vez que o design centralizado no usuário é o principal mecanismo para atingir o sucesso do site.

Neste sentido, você deverá propor a Carla um processo iterativo de design, por meio do qual o design inicial proposto deverá ser mensurado junto aos usuários, e melhorado periodicamente com base nos resultados obtidos.

No que concerne à arquitetura e aos elementos do site, um bom caminho a se seguir para defini-los é seguir as etapas relacionadas abaixo:

- Em termos de arquitetura, uma página central do tipo "portal", isto é, uma página em que chamadas para as produções dos vários colaboradores são constante e dinamicamente dispostas, pode ser o ponto de partida;
- Em termos de cores, você deverá estudar os vários esquemas de cores possíveis, apontando a Carla os efeitos subliminares de cada um deles, para que ela decida qual melhor se encaixa à proposta do site;
- No que concerne à tipografia do site, você pode indicar que se a ênfase é em textos mais longos, uma fonte com serifa é a melhor indicação para o corpo dos textos, e uma fonte mais "alegre" (uma fonte cursiva, por exemplo) pode ser usada para os títulos e pequenos textos ao longo das páginas;
- Quanto aos elementos gráficos das páginas, como a ênfase inicial, deve ser o texto, as imagens devem ser utilizadas no sentido de dar apoio aos conteúdos, não sendo elementos principais nas páginas. Uma imagem de cabeçalho, por exemplo, para ilustrar subjetivamente o texto, pode ser usada no design inicial, com mensurações de atratividade para o usuário apontando que caminho seguir nas interações futuras;
- Com relação à responsividade, a proliferação de dispositivos com características de tamanho, resolução e interface com usuários diferentes, é fundamental que o site e todas as suas páginas sejam desenhados de forma responsiva, para que os visitantes do site de Carla se sintam bem servidos e tenham uma experiência positiva independentemente do dispositivo que escolham ou que tenham à mão.

Para exemplificar os elementos acima para Carla, busque sites que demonstrem a adequação (ou a inadequação) de cada um dos elementos acima, munindo sua cliente dos insumos necessários para tomar as decisões que vão fazer de seu site um sucesso de público e crítica.

Avançando na prática

Evitando que o cliente cometa erros graves

Descrição da situação-problema

Joel é um programador com mais de 15 anos de experiência na indústria de softwares. Com todo seu conhecimento, ele decidiu que é hora de empreender.

Como seu hobby, além da programação, são os jogos eletrônicos, ele decidiu criar um jogo, a que deu o título de "O Sabotador de Orion", um jogo de ação com tema espacial, em que o jogador deve se infiltrar nas forças invasoras que ameaçam a Federação Galáctica e destruí-las antes que cheguem até Turglax, o planeta-capital da Federação e, após dominá-lo, escravize todos os cidadãos interplanetários.

Em que pese Joel ser um excelente programador, e um criativo proponente de jogos, sua ideia do que deve ser o Website do jogo parece ser um tanto questionável: quando ele lhe propõe o trabalho, logo de cara informa que gosta muito do estilo do site de um outro jogo, intitulado "Galaxion". Você visita o site, que pode ser encontrado no endereço <http://www.galaxion.com> (acesso em: 11 abr. 2016. Caso esteja fora do ar, há um armazenamento histórico permanente em <http://webcache.googleusercontent.com/search?q=cache:www.galaxion.com>, acesso em: 11 abr. 2016).

Em sua opinião, o site do jogo "Galaxion" não deveria ser usado como modelo para o site de Joel, e você decide apontar-lhe os motivos, por meio de um relatório que aponta elementos de design que prejudicariam a imagem do jogo de Joel.

Resolução da situação-problema

Alguns dos pontos problemáticos mais evidentes do site são:

- Olhando para a página não é possível identificar quem é o público-alvo ao qual a mesma se destina, e isso porque o design não foi feito com um público-alvo específico em mente. Em outras palavras: quem fez o design deste site não estava pensando em seus usuários potenciais.
- Sob o ponto de vista de cores, não há um esquema definido, com uma miscelânea de cores sendo usada em textos e elementos gráficos que não contribuem para uma mensagem sobre o site. A cacofonia visual, ou seja, a falta de qualidade em seu design, só faz confundir e causar mal-estar em quem visita o site.

- Há uma coleção desconexa de fontes, sem a preocupação de harmonia: fontes em estilo antigo (letra negra) se misturam com fontes modernas, que se misturam com fontes sem serifa, que se misturam com fontes com serifa, e por aí vai. Mais cacofonia visual, mais confusão e desagrado ao visitante.
- O pano de fundo, imitando uma galáxia, é feito com a repetição de pequenos quadros, e a animação só distrai, sem adicionar à experiência do usuário.
- As pequenas imagens se movendo pela tela não fazem sentido: vemos um planeta, duas naves espaciais desconexas (uma delas com direitos de uso protegidos, sendo usada indevidamente) e um crucifixo (!), tudo isso se movimentando aleatoriamente pela tela.

As imagens estáticas não só têm molduras, como as molduras têm sombreamento, ambos os estilos em desuso há muito tempo.

Faça valer a pena

1. Observe as afirmações a seguir:

1. O foco do processo deve ser o usuário e as tarefas que este usuário precisa desempenhar, e este foco deve começar cedo e ser contínuo.
2. O comportamento do usuário deve ser mensurado empiricamente ao longo de todo o processo.
3. O design é um processo iterativo, isto é, deve seguir um fluxo de implementação, mensuração e correção, repetindo-se estes três estágios até que os resultados esperados sejam atingidos.
4. O design deve estar sempre a serviço da tecnologia, utilizando-se os elementos mais recentes da tecnologia no site para assim expressar a mensagem de forma moderna e atual.

São coerentes com o foco no usuário:

a) 1, 2 e 3, apenas.
b) 1, 2 e 4, apenas.
c) 1, 3 e 4, apenas.
d) 2, 3 e 4, apenas.
e) 1, 2, 3 e 4.

2. O conceito de comunicação descentralizada funcionava, e adotando muitos de seus princípios em suas redes particulares, os militares publicaram seus estudos e despertaram o interesse das universidades

norte-americanas, que rapidamente se interessaram por desenvolver os protocolos de comunicação nascentes para assim interligar seus computadores e centros de pesquisa..
Durante as décadas de 1970 e 1980, os principais mecanismos de comunicação e acesso à informação via Internet eram:

a) FTP, Telnet, HTTP.
b) Telnet, SMTP, HTTP.
c) FTP, SMTP, HTTP.
d) FTP, Telnet, SMTP.
e) Web, HTML, HTTP.

3. A autora Krystin Cullen (2012) define tipografia como sendo o processo que torna a linguagem visível aos olhos, usando como ferramenta a formatação do texto por meio de tipos (fontes) que atribuem lhe fluência e (ainda segundo a autora) vida.
Qual das seguintes alternativas contém uma regra da tipografia?

a) Não devem ser usadas fontes sem serifa em textos longos.
b) Não se deve misturar fontes sem serifa com fontes serifadas.
c) Fontes em letra negra (*blackletter*) só devem ser usadas em textos sacros.
d) A primeira regra da tipografia é que não se deve falar sobre tipografia.
e) Não existem regras em tipografia.

Seção 1.2

Percepção de informações digitais para tecnologias web

Diálogo aberto

Olá!

Estou certo de que a discussão e os exemplos sobre a experiência de uso e o design centralizado no usuário já começaram a iluminar o assunto, não é verdade? Pois, afinal de contas, faz sentido que um website seja tão bom quanto sua capacidade de atrair e manter um público, e que cada visitante saia satisfeito com os resultados que obteve por meio das informações e interações com o site. Para tanto, vários elementos do site devem ser planejados e executados de forma a colaborar com a experiência do usuário: cores, tipografia, elementos gráficos e responsividade. Tudo isso colabora com (e, se não tomarmos cuidado, prejudica) a experiência do usuário.

Mas será que o cuidado com esses elementos é tudo o que você precisa para ter um bom design dos websites que logo estará construindo? Não, ainda falta um elemento importante: agora que você foi apresentado às partes que formam um website, está na hora de olhar para o todo. E para aplicar este conceito no contexto prático que estamos trabalhando, vamos retomar a situação.

Até o momento, Carla está contente com o que você lhe apresentou. Ela entendeu e aceitou suas sugestões para melhorar a experiência do usuário por meio das escolhas mais adequadas de cores, fontes, imagens e responsividade para seu site, e quer saber mais sobre como criar o site ideal para seus propósitos de divulgação de cultura. Nesta etapa, você levará a ideia adiante, mostrando a Carla que para que seu website seja um sucesso, ele deverá respeitar as regras da Gestalt, que uma vez bem utilizadas, permitirão que o site faça sentido muito além da tecnologia: ele estará em congruência com as tendências psicológicas de seus visitantes. Para tanto, você deverá demonstrar a Carla o que é a Gestalt, quais são suas regras, e como elas se aplicam no desenvolvimento de websites.

É um desafio e tanto, mas com certeza será igualmente interessante e instigante.

Pronto para mais essa?

Então, vamos lá!

Não pode faltar

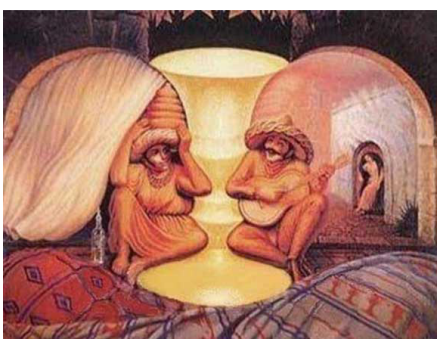
Você já olhou para o céu em um dia de muitas nuvens e pensou que uma nuvem qualquer se parecia com esse ou aquele objeto, ou mesmo com um animal ou pessoa? De onde vem essa nossa capacidade de associar objetos e formas com outros objetos e formas com os quais encontramos alguma semelhança? Quais são os princípios dessa característica, que é inerente ao ser humano? Como podemos usar essa característica presente em todos nós em nosso benefício na hora de planejar um website? Essas e outras perguntas são respondidas com o auxílio da Gestalt, um dos campos mais interessantes de estudos da Psicologia.

O que é Gestalt?

Segundo Bock, Furtado e Teixeira (2008), Gestalt é um termo de tradução difícil, mas que consensualmente se traduz como "forma" ou "configuração", mas nenhum desses termos é utilizado ("Psicologia da Forma" ou "Psicologia da Configuração") por não corresponderem ao seu significado dentro dos estudos de Psicologia. Na verdade, a Gestalt é o ramo da Psicologia criado no início do século 20, pelos alemães Max Wertheimer, Wolfgang Kohler e Kurt Koffka, para estudar a percepção e a sensação causada pelas formas nos seres humanos.

Esse ramo da Psicologia se alicerça sobre o fato de que a realidade e nossa percepção da realidade podem ser coisas bastante díspares. Não precisamos ir muito longe para constatar que esta afirmação é verdadeira, bastando para tanto olhar para a Figura 1.7 a seguir:

Figura 1.7 | Exemplo de disparidade entre a realidade e a percepção da realidade



Fonte: <http://www.wikiart.org/en/octavio-ocampo/forever-always>. Acesso em: 14 abr. 2016.

A maioria das pessoas, quando observa pela primeira vez a Figura 1.7 enxerga um casal de idosos se olhando, mas uma observação mais detalhada da figura mostra que na realidade temos um casal mais jovem, sentado, com uma terceira jovem mais ao fundo. O que parecem ser as cabeças dos idosos nada mais é que um par de arcos de uma construção.

O que a Gestalt postula é que nossa mente naturalmente faz associações que buscam transformar partes de uma figura em um todo coeso, mesmo que este todo coeso exista apenas em nossas mentes, e não na realidade (como a figura mostra com clareza). Nas palavras do próprio Kurt Koffka (1921): "o todo é maior que a soma das partes".

Princípios da Gestalt

A Gestalt tem alguns princípios básicos, que buscam identificar as diferentes maneiras de interpretarmos a realidade, gerando resultados diferentes com base em nossas percepções (BOCK; FURTADO; TEIXEIRA, 2008). O professor de design Felipe Fernandes (2012) aponta que são cinco os princípios básicos da Gestalt:

1. **Proximidade** – Tendemos a agrupar objetos com base em sua proximidade e enxergar o todo composto por esses objetos, mesmo que na realidade façam sentido individualmente;

2. **Semelhança** – Tendemos a agrupar objetos com características semelhantes (formato, tamanho, cor, material, etc.), enxergando o conjunto como um todo;

3. **Continuidade** – Tendemos a enxergar objetos alinhados como se fosse um todo contínuo, um caminho, mesmo que individualmente esse caminho não exista;

4. **Fechamento** – Tendemos a completar conjuntos de objetos cuja distribuição "quase" gera uma forma, efetivamente criando em nossas mentes essa forma;

5. **Pregnância** – Usamos nossas experiências passadas para compor imagens ou completar imagens incompletas e assim dar-lhes significado.

É bem provável que as definições acima precisem ser mais detalhadas para que façam sentido, não é verdade? É exatamente isso que vamos fazer nos itens que se seguem. E mais: vamos ver e exemplificar, além do que significam estes princípios sob o ponto de vista visual, como são aplicados em páginas Web. Um detalhe: alguns autores adicionam, ainda, dois princípios: a Segregação (derivada da proximidade) e a Unidade (derivada do Fechamento). Para fins de simplificação, ficaremos com os cinco princípios clássicos definidos acima.

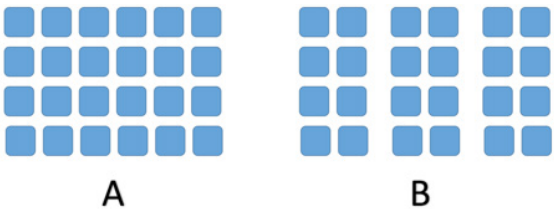
Assimile

A Gestalt postula que nossa mente busca fazer sentido do todo, atribuindo-lhe coesão, mesmo que seja formado por partes independentes.

Proximidade

Observe a Figura 1.8 A e B, a seguir:

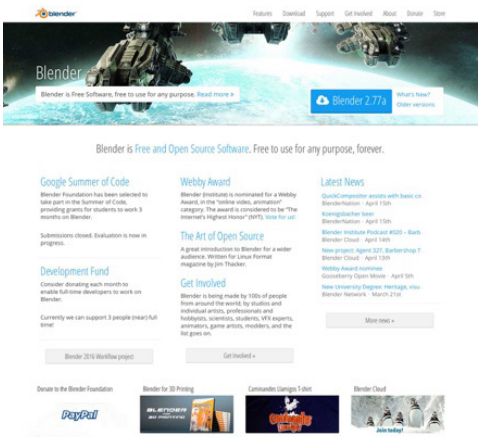
Figuras 1.8 | Exemplo do conceito de Proximidade em Gestalt



Fonte: elaborada pelo autor.

Nossa tendência é considerar o conjunto de objetos da Figura 1.8-A como sendo um objeto único e coeso. Já no caso da Figura 1.8-B, nossa tendência é considerar o conjunto de objetos (formado pela mesma quantidade de objetos, aliás) como sendo três blocos distintos, coesos entre si, mas distintos um do outro. Este efeito é resultado do princípio da proximidade (FERNANDES, 2012). A Figura 1.9, a seguir, exemplifica este princípio em um website:

Figura 1.9 | Exemplo de proximidade



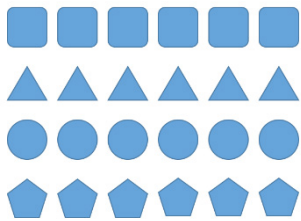
Fonte: <http://www.blender.org>. Acesso em: 3 maio 2016.

O agrupamento dos links (títulos em azul claro) em colunas torna natural seu agrupamento. É importante que itens agrupados tenham coesão semântica, isto é, possam ser colocados em uma mesma categoria de forma que sua colocação nesta categoria faça sentido.

Semelhança

Observe a Figura 1.10, a seguir:

Figura 1.10 | Exemplo de agrupamento por semelhança



Fonte: elaborada pelo autor.

Apesar de as distâncias entre as linhas serem as mesmas, tendemos a enxergar não um bloco único de itens, mas sim quatro linhas. Nossa tendência é enxergar elementos semelhantes por forma e/ou por cor como pertencentes a grupos diferentes (MILLER, 2011). A Figura 1.11, a seguir, exemplifica este princípio da Gestalt em um website:

Figura 1.11 | Exemplo de agrupamento por semelhança

Fonte: <http://www.globo.com>. Acesso em: 18 abr. 2016.

As cores dos blocos — vermelho para assuntos políticos, laranja para entretenimento e verde para esportes — são mais relevantes para nosso agrupamento mental que as colunas em que as matérias são agrupadas.

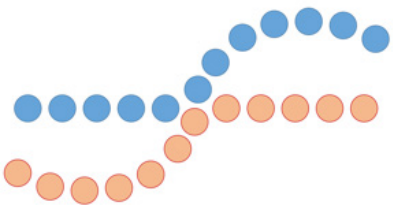
Reflita

Por que, no exemplo anterior, o site decidiu agrupar notícias de assuntos diferentes com cores diferentes?

Continuidade

Observe a Figura 1.12, a seguir:

Figura 1.12 | Exemplo de agrupamento por continuidade

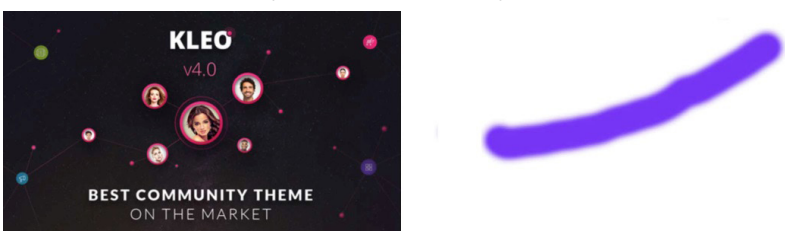


Fonte: elaborada pelo autor.

Apesar de as cores criarem dois agrupamentos distintos, também criamos um grupo formado por uma linha reta contínua e outro formado por uma curva contínua, apesar de não haver continuidade entre as formas (são formas distintas, discretas, separadas no espaço) ou mesmo nas cores. Miller (2011) chama a atenção para essa nossa capacidade de criar a continuidade.

A Figura 1.13, a seguir, mostra um website em que naturalmente criamos continuidade entre pontos discretos:

Figura 1.13 | Exemplo de agrupamento por continuidade (Traço na cor púrpura mostra uma linha contínua que formamos ao olhar para o todo)



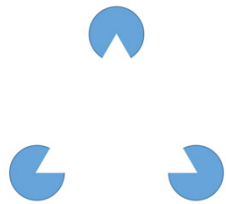
Fonte: adaptada de: <http://themeforest.net/item/kleo-next-level-wordpress-theme/full_screen_preview/6776630>. Acesso em: 3 maio 2016.

Os ícones com fotos se agrupam naturalmente em uma linha, dando a ideia de continuidade ao que na verdade são pontos discretos.

Fechamento

Observe a Figura 1.14, a seguir:

Figura 1.14 | Exemplo de agrupamento por fechamento

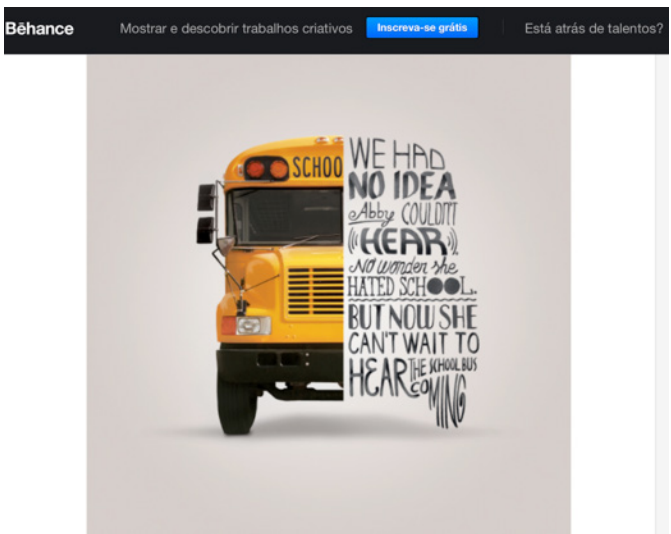


Fonte: elaborada pelo autor.

Na Figura 1.14, apesar de haver apenas três semicírculos, tendemos a "fechar" um triângulo completo no centro. O princípio do fechamento, segundo Miller (2011), aponta justamente esta capacidade de inferir um todo "fechado" a partir de elementos efetivamente desconexos.

Na Figura 1.15, a seguir, podemos enxergar um exemplo de fechamento usado em um website:

Figura 1.15 | Exemplo de agrupamento por fechamento



Fonte: <https://www.behance.net/gallery/21741033/Cochlears-headlines->. Acesso em: 3 maio 2016.

Na Figura 1.15, a imagem do ônibus é completada pelas palavras, e o vemos completo mesmo que somente metade esteja sendo apresentada.

Pregnância

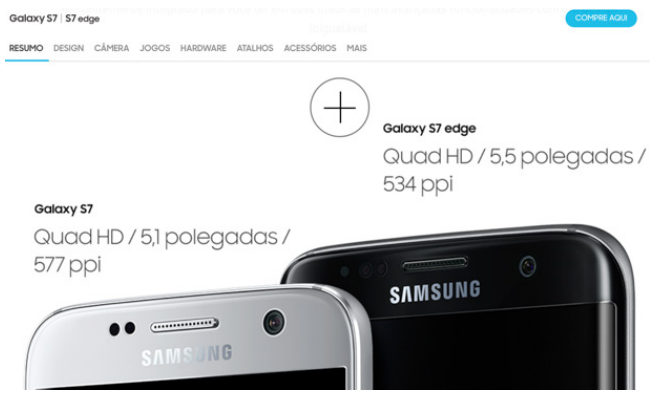
Observe a Figura 1.16, a seguir:

Figura 1.16 | Exemplo de pregnância

Fonte: adaptado de: <http://picshype.com/drawing-of-an-electric-guitar/electric-guitar-outline-stock/37855>. Acesso em: 3 maio 2016.

A pregnância, isto é, nossa capacidade de inferir o todo com base em uma parte e em nossas memórias e experiências (MILLER, 2011), permite que saibamos que a imagem da Figura 1.16 é, de fato, a representação de uma guitarra, mesmo que faltem informações na imagem em si. A Figura 1.17, a seguir, mostra um exemplo de pregnância no design de uma página da web:

Figura 1.17 | Exemplo de pregnância



Fonte: <http://www.samsung.com>. Acesso em: 3 maio 2016.

Na imagem, não há necessidade de mostrar os celulares inteiros, pois por inferência e pela memória que temos dos aparelhos, sabemos que se trata de smartphones.

O uso da Gestalt no design de websites

Segundo Miller (2011), um sistema efetivo de design tem precedência sobre os elementos individuais, pois, neste caso, o

observador percebe uma unidade coesa, e a utiliza para entender melhor as partes. A utilização desses mecanismos é livre e, não obstante serem todos bastante interessantes, é importante escolhe-los adequadamente, mantendo a harmonia e a ergonomia do produto final, evitando sempre o exagero.

Estes princípios devem ser levados em conta quando se planeja uma página ou um site inteiro, pois tendem a colaborar para com as mensagens sendo transmitidas quando bem utilizados.

Pesquise mais

O pesquisador Tiago Luís Ramos Silva Machado, da Universidade de Lisboa, publicou, em 2014, uma dissertação acerca da percepção de credibilidade com base no design visual do website. O capítulo 2, e em especial os itens 2.1 e 2.2, versa sobre design visual e dá ênfase à Gestalt. Vale a pena conferir.

MACHADO, T. **Design Visual e Credibilidade Percecionada na Web**. Lisboa: Universidade de Lisboa, 2014. Disponível em: <http://repositorio.ul.pt/bitstream/10451/18214/2/ULFBA_TES811.pdf>. Acesso em: 17 abr. 2016.

Exemplificando

O quadro "Família de Pássaros", do pintor mexicano Octavio Ocampo apresenta exemplos de Proximidade, Continuidade, Fechamento e Pregnância, ao mesmo tempo. Veja:

Figura 1.18 | Família de Pássaros, de Octavio Ocampo

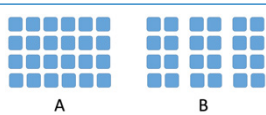
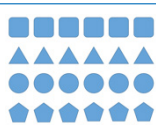
Fonte: <http://www.wikiart.org/en/octavio-ocampo>. Acesso em: 17 abr. 2016.

Sem medo de errar

Para mostrar a Carla as possibilidades e as vantagens da Gestalt, você pode começar apresentando a ela um breve documento explicando o que é a Gestalt e quais são seus princípios:

- O que é Gestalt?
  - Gestalt é o ramo da Psicologia criado para estudar a percepção e a sensação causada pelas formas nos seres humanos.
- Quais são os princípios da Gestalt?
  - **Proximidade** – agrupamento de objetos com base em sua proximidade, enxergando o todo composto por esses objetos, mesmo que na realidade façam sentido individualmente;
  - **Semelhança** – agrupamento de objetos com características semelhantes (formato, tamanho, cor, material, etc.);
  - **Continuidade** – agrupamento de objetos alinhados como se fossem um todo contínuo, um caminho;
  - **Fechamento** – compleição de conjuntos de objetos cuja distribuição "quase" gera uma forma, efetivamente criando em nossas mentes esta forma;
  - **Pregnância** – Uso de experiências passadas para compor imagens ou completar imagens incompletas e assim dar-lhes significado.

Em seguida, você poderá apresentar uma tabela, exemplificando simbolicamente e literalmente os princípios da Gestalt. Sugere-se, por exemplo, uma tabela como a que se segue:

Princípio	Definição	Exemplo Simbólico	Exemplo Literal na Web
Proximidade	...		http://www...
Semelhança	...		http://www...

Continuidade	...		http://www...
Fechamento	...		http://www...
Pregnância	...		http://www...

Atenção

É importante frisar que a adoção de princípios da Gestalt não demanda que todos sejam adotados, mas sim aqueles que fazem sentido de acordo com o design do site e a mensagem que se pretende passar ao usuário.

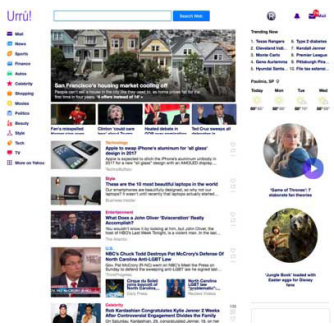
Avançando na prática

Identificando os princípios da Gestalt

Descrição da situação-problema

Lindolfo é um aluno de sua sala que, por motivos de saúde, perdeu a oportunidade de estudar o material sobre a Gestalt no design Web. Ele veio lhe pedir ajuda para encontrar os elementos da Gestalt presentes em um Website, o Urrú!, um portal de notícias que ele frequenta. Para tanto, ele trouxe até você uma imagem recente do site, representada na Figura 1.19, a seguir:

Figura 1.19 | Tela do site Urrú!



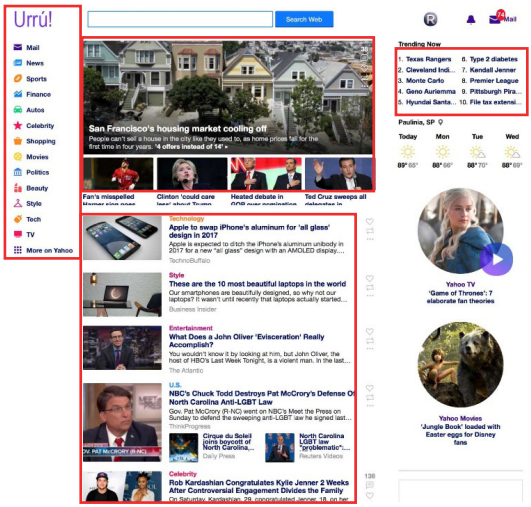
Fonte: adaptado pelo autor.

Quais elementos (princípios) da Gestalt estão presentes no site do Urrú?

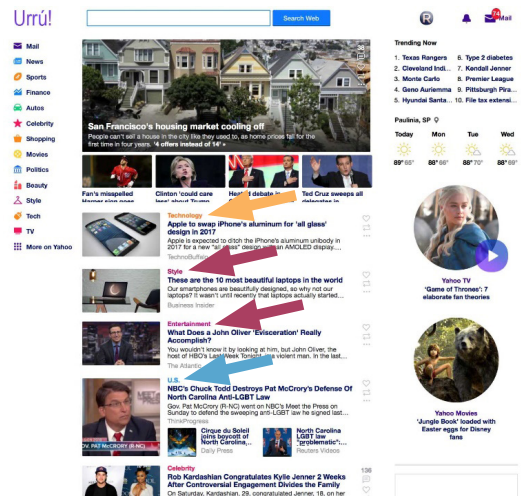
Dica: Os cincos princípios da Gestalt: Proximidade, Semelhança, Continuidade, Fechamento e Pregnância.

Resolução da situação-problema

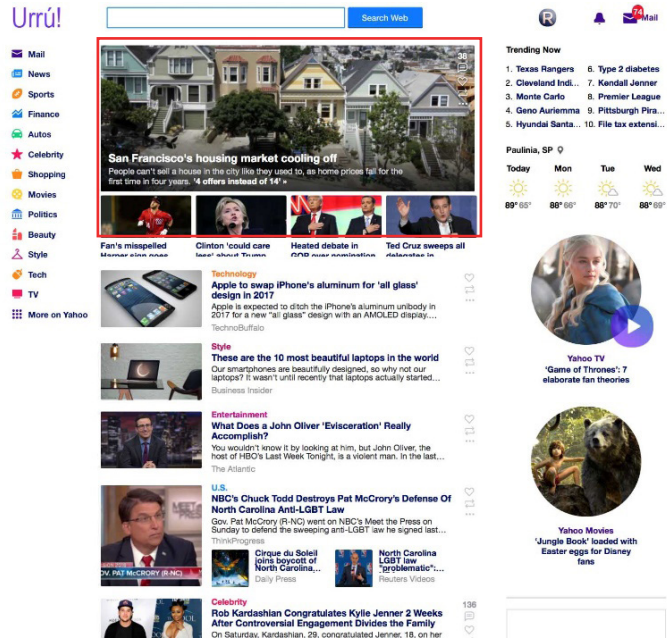
Podemos encontrar vários elementos da Gestalt no site apresentado por Lindolfo:



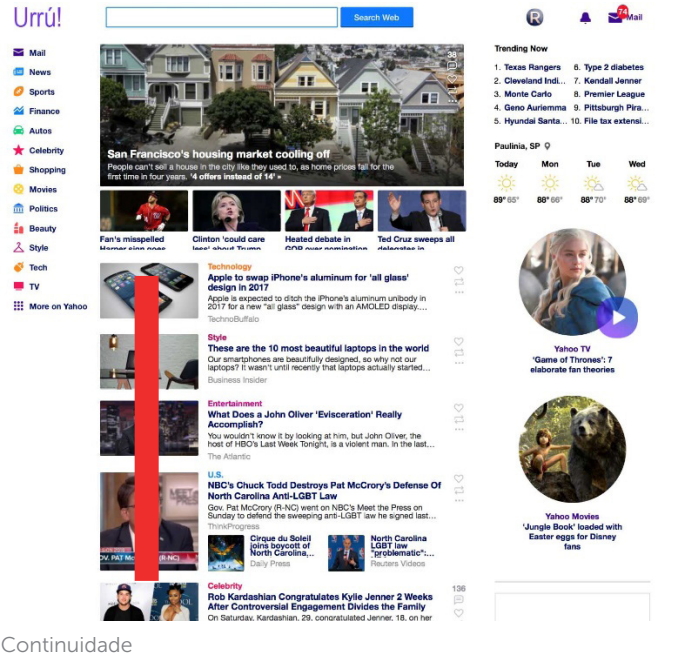
Proximidade



Semelhança (Cor do Assunto)



Fechamento



Continuidade

Faça valer a pena

1. O que a Gestalt postula é que nossa mente naturalmente faz associações que buscam transformar partes de uma figura em um todo coeso, mesmo que este todo coeso exista apenas em nossas mentes, e não na realidade. Nas palavras do próprio Kurt Koffka (1921): "o todo é maior que a soma das partes".

A respeito dos estudos da Gestalt, é verdadeiro afirmar que:

a) É um ramo de estudo criado para definir as melhores condutas de design para Websites.
b) É um estudo criado, muito antes da existência da Web, para melhorar o design em produtos gráficos impressos feitos a partir do início do século 20.
c) É um estudo criado para entender a percepção que temos das formas.
d) É o estudo da proximidade, da semelhança, da continuidade, do fechamento e da pregnância das ideias.
e) É o estudo da proximidade, da semelhança, da continuidade, do fechamento e da pregnância do design.

2. A Gestalt tem alguns princípios básicos, que buscam identificar as diferentes maneiras de interpretarmos a realidade, gerando resultados diferentes com base em nossas percepções (BOCK, FURTADO e TEIXEIRA, 2008).

Observe os itens a seguir:

I. Proximidade
II. Responsividade
III. Semelhança
IV. Continuidade
V. Pregnância

Considerando os itens apresentados acima, são princípios da Gestalt:

a) I, II, III e IV apenas
b) I, II, III e V, apenas
c) I, II, IV e V, apenas
d) I, III, IV e V, apenas
e) II, III, IV e V, apenas

3. Apesar de as cores criarem dois agrupamentos distintos, também criamos um grupo formado por uma linha reta contínua e outro formado por uma curva contínua, apesar de não haver continuidade entre as formas (são formas distintas, discretas, separadas no espaço) ou mesmo nas cores.

A tendência que temos de enxergar objetos alinhados como se fossem um todo contínuo, um caminho, mesmo que individualmente esse caminho não exista é chamada de:

a) Proximidade.
b) Semelhança.
c) Continuidade.
d) Fechamento.
e) Pregnância.

Seção 1.3

Projeto e arquitetura aplicadas as tecnologias Web

Diálogo aberto

Quando vemos a foto de um desses carros europeus esportivos de luxo, nossos olhos brilham, não é verdade? Enxergamos seu design arrojado e atrativo, suas cores, suas linhas harmônicas, e tudo parece dizer, bem baixinho, "velocidade", "precisão", "atenção total aos detalhes", "sensualidade", "luxo", etc. Muitos de nós nunca dirigimos um carro assim, mas de uma coisa podemos estar certos: o desempenho corresponde ao visual – do barulho do motor, passando pela resposta à troca de marchas e pela estabilidade nas curvas, até a velocidade estonteante a que conseguiríamos chegar. Em outras palavras: o que está embaixo do capô corresponde ao que vemos do lado de fora. O carro tem motor, suspensão, controle eletrônico, freios e tudo mais que corresponde ao seu design superior. Aliás, tudo isso faz parte de seu design superior.

Pois bem, no design Web temos que ter a mesma harmonia entre o que vemos "na lataria" e o que está "sob o capô". Até o momento vimos vários aspectos de design que dizem respeito ao que é visível em um website: foco no usuário, cores, fontes, imagens e Gestalt. Tudo isso é visível, é parte da experiência subjetiva do usuário, e vai contribuir (contra ou a favor) para sua experiência, para sua sensação ao navegar, para sua satisfação com o website por onde navega.

Agora é chegada a hora de darmos uma primeira olhada sob o capô do website, ou seja, olharmos para a arquitetura da infraestrutura que lhe dá sustentação. Vamos começar a olhar para as importantes peças que possibilitam que um website seja rico visualmente e útil funcionalmente.

No que concerne a nossa cliente, Carla sabe que quer criar um portal de cultura, e já está ciente de que os elementos de design visual deverão ser muito bem cuidados. Ela entendeu que você vai auxiliá-la a criar um ambiente agradável e funcional, e já sabe julgar por si mesma os vários aspectos que você lhe apresentou.

Nesta nova etapa do projeto, você deve apresentar a Carla os aspectos mais importantes da arquitetura de um Website, em dois níveis: em primeiro lugar, ela deve ter contato com a arquitetura "sob o capô", isto é, os aspectos de um website que não são visíveis, tais como a arquitetura cliente/servidor (navegador web/servidor web) entender quais e para que servem as linguagens de programação (de maneira genérica nesta etapa, claro), e os protocolos de comunicação. Em seguida, você deverá lhe apresentar os principais elementos da arquitetura visível de um website e suas páginas: títulos, menus, botões, corpos de texto e outros elementos, de forma que Carla compreenda o que compõe, de fato, uma página web e quais as opções de localização e tamanho destes itens.

Diante dessa tarefa, é natural que surjam alguns questionamentos: o que está envolvido em termos de processo de comunicação na obtenção de uma página web quando clicamos em um link? Quais são as linguagens e protocolos normalmente envolvidos? Como é a arquitetura dos componentes sobre os quais esta comunicação se dá? No que diz respeito a uma página, quais são seus elementos componentes? Há alguma distribuição mandatória ou pelo menos recomendável? O que não pode faltar na arquitetura de uma página na Web?

Essas e outras questões serão endereçadas a partir de agora.

Vamos em frente!

Não pode faltar

Vamos entender como é a arquitetura da Web, primeiramente em sua infraestrutura, e em seguida no que diz respeito aos sites e páginas.

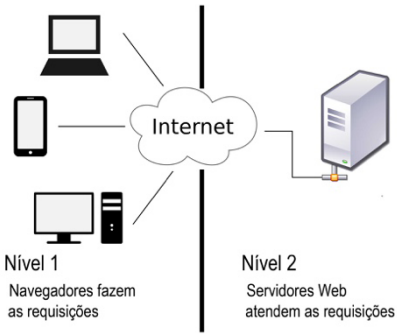
Arquitetura da Web

Você já parou para pensar o que acontece quando você clica em um link exposto em seu navegador Web e em poucos segundos o navegador expõe as informações que você requisitou? De onde vêm estas informações? Como chegam até seu navegador?

Bem, as respostas a essas questões passam pela arquitetura da Web em si. De forma simplificada, podemos afirmar que a Web é um serviço oferecido sobre a Internet, serviço este que se baseia inicialmente em uma arquitetura em dois níveis: clientes e servidores

(TITTEL e MINNICK, 2013). Podemos representar essa arquitetura conforme mostra a Figura 1.20, a seguir:

Figura 1.20 | Arquitetura cliente/servidor em dois níveis da Web



Fonte: elaborada pelo autor.

Essa arquitetura simples é poderosa o suficiente para que sobre ela criemos todo um leque de serviços e disponibilizemos tantas informações no período de dois dias do que as disponibilizadas desde o princípio da humanidade até o ano de 2003 (SIEGLER, 2010). Hoje em dia, o conjunto de todos os servidores da Web armazena, segundo Siegler (2010), dois petabytes de informações diariamente, sendo estas informações dos tipos: dados, textos, imagens, sons, vídeos, jogos, entre outros.

Os dois elementos básicos dessa arquitetura são (TITTEL; MINNICK, 2013):

- **Navegador (primeiro nível)** – Programa de computador que serve de interface entre o usuário e o conteúdo da Web. Responsável por: coletar entradas do usuário (cliques de mouse, textos, sons, vídeos), enviar requisições de conteúdo, enviar dados, receber conteúdo, exibir conteúdo (texto, som, vídeo, jogos e animações). São gratuitos e disponibilizados por vários fabricantes. Os principais do mercado são, segundo StatCounter (2016):
  - Google Chrome – 56,4%
  - Mozilla Firefox – 14,3%
  - Internet Edge (inclui antigo Explorer) – 12,5%
  - Apple Safari – 9.47%
  - Outros – 7,3%

- **Servidor (segundo nível)** – Programa de computador executado em algum dos servidores da Internet, que armazena os conteúdos e os distribui para os navegadores que os requisitam. Os principais servidores do mercado são (W3TECHS, 2016):
  - Apache (gratuito, código aberto) – 52,9%
  - Nginx (gratuito, código aberto) – 29,8%
  - Microsoft IIS – 11,9%
  - Outros – 5,4%

Linguagens, protocolos, elementos

Podemos citar os três elementos principais da comunicação Web:

- **HTML** – A linguagem, por meio da qual as informações da Web são trafegadas pela rede e depois interpretadas pelos navegadores para serem disponibilizadas ao usuário final, foi desenvolvida em 1989 e publicada em 1990 por Tim Berners-Lee. Leva o nome de *Hypertext Markup Language*, e é mundialmente conhecida por suas iniciais: HTML (LEVINE e YOUNG, 2010). Trata-se de uma linguagem de formatação de documentos, o que de certa forma a diferencia de uma linguagem de programação clássica: o resultado a que se propõe é apresentar documentos visualmente organizados e agradáveis ao usuário, e não criar programas que desempenham tarefas no computador.
- **HTTP** – O *Hypertext Transport Protocol* (sigla HTTP) é o protocolo por meio do qual as requisições de páginas Web são enviadas do navegador para o servidor e, em seguida, as páginas web são enviadas do servidor para o navegador (THOMAS, 2001). Trata-se de um protocolo *stateless*, ou seja, que não mantém o estado da comunicação. Uma vez atendido um pedido, o servidor não sabe em que estado está o navegador e vice-versa, diferente dos protocolos TCP e IP que mantêm o estado das comunicações (THOMAS, 2001).
- **URL** – *Universal Resource Location*, ou local universal de recursos, é o endereço dos recursos que buscamos na internet, via protocolo HTTP para apresentar em nosso navegador. Sua estrutura é mostrada na Figura 1.21, a seguir:

Figura 1.21 | Estrutura de uma URL (*Universal Resource Location*)

Fonte: elaborada pelo autor.

É importante observar na figura 1.21 que tanto o caminho de diretórios quanto o nome da página são opcionais. Em sua ausência, o servidor saberá em qual diretório buscar e qual a página padrão buscar, para enviar ao navegador.

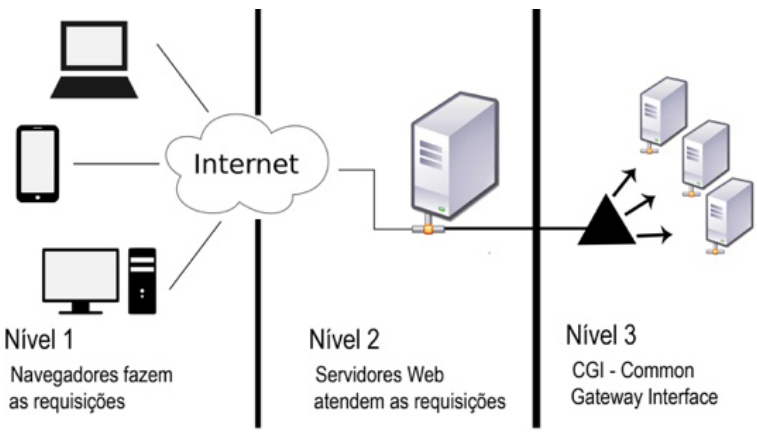
Reflita

Você sabia que o protocolo de comunicação em uma URL não precisa ser necessariamente o HTTP? Outros protocolos da Internet também podem ser usados aqui, como, por exemplo, o protocolo FTP para transmissão e recepção de arquivos.

Extensões da Arquitetura da Web - CGI

Lembra-se que aqui foi comentado que a Web, "inicialmente" tem uma arquitetura em dois níveis? Pois bem, esse era o caso nos primórdios da tecnologia, até pouco depois da metade da década de 1990. A partir daí, segundo o W3C (2011), mais precisamente em novembro de 1997, foi publicada a especificação do CGI ou *Common Gateway Interface* (em português: Interface Comum de Acesso (W3C, 2011)). Trata-se de uma especificação que propõe a comunicação do servidor Web (segundo nível) com outros servidores que ficam "por trás" deste. Esses demais servidores proveem aplicativos, linguagens e recursos não previstos na linguagem HTML. Na prática, transformam a arquitetura da Web em uma arquitetura de três níveis, como pode ser visto na Figura 1.22, a seguir:

Figura 1.22 | Arquitetura em três níveis da Web



Fonte: elaborada pelo autor.

Com essa nova arquitetura, a Web ganha a possibilidade de executar programas, receber dados de formulários, acessar bancos de dados e o principal: criar páginas dinâmicas, isto é, páginas que são montadas de acordo com a situação. As páginas dinâmicas são por nós utilizadas corriqueiramente, sem que pensemos muito a respeito. São páginas dinâmicas, por exemplo, que enxergamos todas as vezes que entramos em um portal de notícias, e vemos artigos que foram postados nos últimos minutos, com as notícias mais antigas removidas da página principal. Quem permite isso é a arquitetura em três níveis e a interface comum de acesso, a CGI.

Assimile

Inicialmente a arquitetura da infraestrutura da Web era em dois níveis, mas desde a introdução do CGI (*Common Gateway Interface*), em 1997, esta arquitetura passou a ser em três níveis.

Extensões da Arquitetura da Web - *Cookies*

Lembra-se que aqui foi dito que o protocolo de comunicação HTTP é *stateless*, (em português: que não mantém estado)? Isso significa que uma vez que uma requisição HTTP é atendida pelo servidor Web, este servidor não mantém nenhuma informação sobre o estado do cliente, ou seja, do navegador "na ponta de lá". Dessa forma, não há como o servidor saber, por exemplo, que partes do site o usuário já viu, que notícias mais lhe interessaram, que formulários ele já preencheu, etc.

Uma forma bem interessante por meio da qual os servidores Web guardam o estado da comunicação com os clientes (navegadores) é por meio dos *cookies* ("biscoitos", em português).

Um *cookie* é um pequeno arquivo de texto que o servidor grava no HD do usuário — mediante autorização, claro — com algumas informações acerca da comunicação com aquele site (LEVINE; YOUNG, 2010). Por meio dos *cookies*, o estado da comunicação daquele navegador com o servidor Web é mantido. Dessa forma, a navegação é mais agradável, pois o site sabe com mais precisão que conteúdo o usuário já viu e que conteúdo ele ainda não visitou.

Os *cookies* expiram após uma sessão de navegação, um marco que ocorre sempre que desligamos o navegador (encerramos o programa). Nesse momento, como parte das tarefas de encerramento do programa, os *cookies* são apagados (LEVINE; YOUNG, 2010).

Apesar de ainda muito utilizados, os *cookies* já não são a única maneira de os servidores Web guardarem informações acerca de seus usuários. O endereço IP da máquina do cliente, dados de registro (*login*), e outras características únicas a cada máquina do cliente, e mesmo a cada usuário de cada máquina do cliente, são mantidos em extensas bases de dados no servidor Web, o que permite que estes servidores saibam muitas coisas acerca dos usuários que os visitam. E se há aí uma discussão acerca de privacidade (afinal de contas, o servidor está guardando informações a respeito do usuário), a conveniência que é possível por conta deste mecanismo é aceita pela maioria dos usuários como "troca" pela privacidade (LEVINE; YOUNG, 2010).

Pesquise mais

Nem todos os *cookies* são inofensivos. Os *cookies* maliciosos são *cookies* que "abusam" do poder de manter informações sobre um usuário. Faça uma busca na web sobre o assunto e se inteire dos seguintes pontos:

- O que são *cookies* maliciosos?
- Como se diferenciam dos *cookies* não maliciosos (*cookies* normais)?
- Como identificar a presença de *cookies* maliciosos em um computador?
- Como eliminar *cookies* maliciosos de um computador?
- Como evitar que *cookies* maliciosos se instalem em um computador?

Elementos de uma página

Vamos agora entender outro aspecto da Arquitetura Web: a arquitetura das páginas que compõem um site. Os vários elementos de uma página Web são diferenciados por seus estilos, e quem controla os estilos destes elementos é um documento que compõe a página, chamado CSS, ou *Cascading Style Sheet*, em português: folha de estilos em cascata (TEAGUE, 2009). Esse documento estabelece como serão visualmente apresentados os vários elementos de uma página Web, e define fonte, tamanho da fonte, posição do elemento, distribuição dos elementos na página (colunas, largura das colunas, etc.) e várias outras características que serão vistas com mais detalhes em aula posterior, quando formos "entrar a fundo" em CSS.

Exemplificando

Um exemplo de formatação de estilo é a diferenciação entre o título de uma página e o texto dos parágrafos que se seguem. Os títulos são representados pela letra "h", seguida do número que corresponde ao nível do título, e o texto corrido é representado pela letra "p" (de parágrafo). Na folha de estilo ambos podem ser definidos como no exemplo a seguir: o título centralizado e na cor verde, e o texto corrido em fonte Arial com tamanho 14:

```
h1 { color: green;
     text-align: center; }
p { font-family: "Arial";
    font-size: 14px; }
```

A título de exemplo, vamos identificar os elementos de arquitetura de uma página Web em uma página qualquer. A Figura 1.23, a seguir, mostra uma página, com destaque para vários de seus elementos:

Figura 1.23 | Exemplo de elementos em websites

Fonte: adaptada de <http://www.webdesigner.com.br>. Acesso em: 27 abr. 2016.

Vamos observar em mais detalhes os elementos destacados na Figura 1.23 (TEAGUE, 2009):

1. **Logo do Site** – Geralmente ocupa uma posição de destaque, no canto superior esquerdo ou direito da página. É ativo e seu comportamento normal, quando clicado, é levar o usuário à página inicial do site (chamada em inglês de *main page* ou *home page*).

2. **Menu horizontal** – Geralmente posicionado no topo da página, oferece acesso às principais páginas do site. É um elemento ativo, sendo que cada elemento "avisa" quando passamos o mouse sobre ele (no caso, um destaque na cor cinza aparece em volta da palavra). Quando clicada uma opção, um submenu aparece, "descendo" a partir da opção selecionada (em inglês: *drop-down menu*).

3. **Opção Cadastre-se** – bastante necessária em sites nos quais o usuário tem acesso a conteúdo próprio e a customizações pessoais, tal como é o caso dos sites de comércio eletrônico (em inglês *e-commerce*).

4. **Opção Login** – Os usuários já cadastrados, quando querem acessar sua área pessoal no site, vão ao *login*, onde lhes são solicitados o nome e a senha sob os quais estão cadastrados.

5. **Slider (apresentador de slides)** – Elemento bastante comum hoje em dia, o *slider* é um elemento dinâmico que mostra figuras em sequência, mostrando alguns conteúdos para os quais se quer dar destaque na página. As figuras se sucedem em poucos segundos e são links ativos (quando clicadas, levam a páginas específicas no site).

6. **Título** – Elemento de estilo para o qual há definição na folha de estilo (CSS) da página. Destaca o nome do texto que se segue.

7. **Figuras e conteúdos em destaque** – Pequenos blocos com conteúdo para os quais se quer dar destaque na página. Geralmente acompanhados da parte introdutória dos textos para instigar o interesse do usuário.

Obviamente que ainda há vários elementos a serem discutidos sobre as páginas Web que estão ausentes do exemplo acima, tais como as divisões das páginas em colunas, os links de acesso (geralmente destacados em azul e sublinhados) e muitos outros que serão vistos ao longo das próximas aulas.

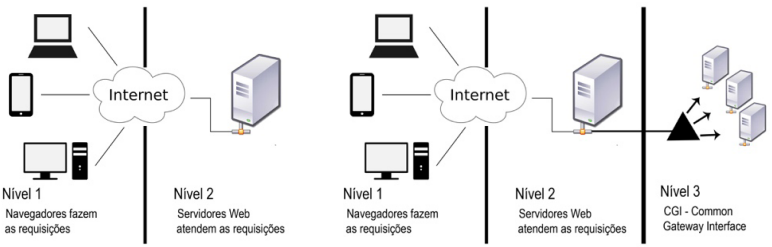
Pesquise mais

A Mozilla Foundation, criadora e mantenedora do navegador Firefox, disponibiliza um excelente texto para iniciantes em CSS, um "tutorial" que vai ajuda-lo a entender como funciona, para que serve e como utilizar o CSS de forma a obter os melhores resultados em suas páginas Web. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS/Getting_Started>. Acesso em: 17 abr. 2016.

Sem medo de errar

Para mostrar a Carla como funcionam os elementos da arquitetura da Web, podemos começar descrevendo como a arquitetura foi criada em dois níveis e depois evoluiu para três níveis, permitindo que, em páginas de conteúdo dinâmico, a interação com bancos de dados e servidores de aplicativos passasse a ser possível. A transição do modelo de dois níveis para o modelo atual em três níveis pode ser mostrada com duas figuras apresentadas lado a lado:

Figura 1.24 (a) e (b) | Arquitetura Web em dois níveis



A arquitetura em dois níveis... ...evoluiu para a arquitetura em três níveis.

Fonte: adaptada de <https://en.wikipedia.org/wiki/Client%E2%80%93server_model#/media/File:Client-server-model.svg>. Acesso em: 6 dez. 2017.

Em seguida, devemos pontuar para Carla os vários elementos que compõem a infraestrutura desta arquitetura, em especial os três mais importantes:

- **HTML** – a linguagem de formatação do conteúdo apresentado no navegador;
- **HTTP** – O protocolo de comunicação entre o navegador e o servidor Web;
- **URL** – *Universal Resource Location*, o endereço dos recursos que desejamos buscar na Web para exibição em nossos navegadores.

Para finalizar, você pode entrar em várias páginas da Web e conduzir Carla à identificação dos vários elementos que compõem uma página Web, destacando estes elementos da arquitetura:

- Logo;
- *Slider*;
- Título;
- Texto corrido;

- Menu (horizontal ou lateral);
- Chamadas para textos;
- Destaques;
- Links.

Dessa forma, Carla já poderá ir pensando em como vai querer ter as informações dispostas em seu site.

Avançando na prática

Identificando elementos de um portal de cultura

Descrição da situação-problema

Depois de entender sua explicação acerca dos elementos de uma página Web, Carla passou a prestar mais atenção às páginas que visita. Uma delas chamou a atenção da futura empreendedora, por sua organização e qualidade visual: o site do Instituto Moreira Salles, que pode ser encontrado no site <http://ims.com.br/ims> (acesso em: 27 abr. 2016). Ela trouxe seu laptop até você, com a página principal deste site carregada no navegador e pede ajuda para identificar os principais elementos do site. Você pode ajudá-la?

Lembre-se

As diferentes formatações da estrutura do texto e dos elementos do texto em si se dão por meio da folha de estilos da página, ou seja, do CSS.

Resolução da situação-problema

Quando olhamos para a página principal do site do Instituto Moreira Salles, representada na Figura 1.25 a seguir, o que vemos é algo assim:

Figura 1.25 | Página inicial do site do Instituto Moreira Salles

Fonte: <http://www.ims.com.br/ims/>. Acesso em: 27 abr. 2016.

Podemos identificar vários elementos já conhecidos nesta página principal, a saber:

- **Logo do site** – No canto superior esquerdo, com as letras estilizadas "IMS". É um link ativo que sempre conduz o usuário de volta à página inicial (*home page*);
- **Menu Horizontal** – Na parte superior da página, ao lado do logo. Cada uma das opções é um submenu do tipo *drop-down*, mostrando as várias opções e páginas do site.
- ***Slider*** – No caso, o *slider* está apresentando uma pequena chamada para o texto "Casa nova de Alice", com uma fotografia. As "bolinhas", logo abaixo do texto da chamada, mostram que são oito os "slides", e que estamos no sexto, neste momento.
- **Segunda Instância do Menu horizontal** – Logo abaixo do *Slider*, o site repete o menu horizontal, e não há nada de errado com isso. A parte superior do site poderia ser fixa, não se movendo com os movimentos do mouse arrastando a barra vertical lateral. Mas assim, perde-se uma faixa superior de tela para se exibir sempre o mesmo conteúdo. O site do IMS prefere repetir o menu horizontal, de forma que ele continua visível quando baixamos a tela. Observe ainda que esta segunda instância tem outras opções, além das opções do menu visto anteriormente.
- **Chamadas para conteúdos** – são as pequenas caixas com uma imagem e um pequeno trecho de texto, buscando despertar o interesse do leitor.

Faça valer a pena

1. O *Hypertext Transport Protocol* (sigla HTTP – em português significa: protocolo de Transferência de Hipertexto) é o protocolo por meio do qual as requisições de páginas Web são enviadas do navegador para o servidor e, em seguida, as páginas web são enviadas do servidor para o navegador (THOMAS, 2001).

O HTTP (*Hypertext Transfer Protocol*) é um protocolo dito *stateless*. O que significa isso?
a) O protocolo HTTP mantém uma tabela com o estado da comunicação com cada um dos navegadores com que o servidor se comunica.

b) O protocolo HTTP guarda o estado das comunicações, mas somente pela duração da sessão.
c) O protocolo HTTP depende de um alvará estadual de funcionamento para ser implantado sobre a rede de comunicações daquele estado.
d) O protocolo HTTP envia para o servidor o estado de memória e de armazenamento da máquina com quem o servidor se comunica
e) O protocolo HTTP não guarda o estado das comunicações entre servidores Web e navegadores.

2. O protocolo de comunicação HTTP é *stateless*, (em português: que não mantém estado). Isso significa que uma vez que uma requisição HTTP é atendida pelo servidor Web, este servidor não mantém nenhuma informação sobre o estado do cliente, ou seja, do navegador "na ponta de lá".

Observe os itens a seguir:
I. Proximidade
II. Responsividade
III. Semelhança
IV. Continuidade
V. Pregnância

Qual o mecanismo usado pelos sites Web para manter o estado da comunicação (manter informações sobre o usuário, por onde navega e outros dados de navegação naquele site), mesmo diante do fato de que o HTTP é um protocolo *stateless*?
a) HTML
b) Arquitetura em três níveis
c) Cookie
d) URL
e) CSS

3. Os programas de computadores que são executados no ambiente da internet, armazenam informações e distribuem o conteúdo para os navegadores que os requisitam. Lembrando que cada local de armazenamento tem suas estruturas específicas.

O Apache, o IIS e o Nginx são exemplos de que elemento da arquitetura da Web?
a) HTML
b) Servidor
c) Segundo Nível
d) URL
e) Navegador

Referências

ABNT - Associação Brasileira de Normas Técnicas. **ISO NBR 9241 11**: Requisitos Ergonômicos para Trabalho de Escritórios com Computadores Parte 11 - Orientações sobre Usabilidade. Rio de Janeiro: ABNT, 2000.

ARISTIMUNHA DA SILVA, D. Classificação dos Tipos. **Slideshare**, 2012. Disponível em: <http://pt.slideshare.net/denisealima/classificacao_tipografia>. Acesso em: 10 abr. 2016.

ARTY, D. Guia Sobre Cores - Teoria das Cores. **Chief of Design**, 2014. Disponível em: <http://chiefofdesign.com.br/teoria_das-cores/>. Acesso em: 10 abr. 2016.

_____. Saiba o que é e como usar a Gestalt em seus projetos. Disponível em: <https://www.chiefofdesign.com.br/gestalt/>. Acesso em: 5 dez. 2017.

BLEICHER, S. **Contemporary Color – Theory and Use.** 2. ed. Nova York: Cengage, 2012.

BOCK, A.; FURTADO, O.; TEIXEIRA, M. **Psicologia**. São Paulo: Saraiva, 2008.

CULLEN, K. **Design Elements:** Typography Fundamentals. Boston: Rockport Publishers, 2012.

FERNANDES, F. Princípios de Design e Introdução à Gestalt. **SlideShare**, 2012. Disponível em: <http://pt.slideshare.net/felipefernand3s/web-design-aula-3>. Acesso em: 14 abr. 2016.

HARTSON, R.; PYLA, P. **The UX Book**: Process and guidelines for ensuring a quality user experience. Nova York: Elsevier, 2012.

KOFFKA, K. **Perception**: An Introduction to the Gestalt Theories. Berlin: Osterwieck am Harz, 1921.

LEVINE, J.; YOUNG, M. **The Internet for Dummies**. Nova York: Wiley, 2010.

MILLER, Brian. **Above The Fold**: Understanding the principles of successful website design. Cincinati: How Books, 2011.

MOREIRA, Everton Lopes Paulo; BOTELHO, Roberto Debona; FERNANDES, Jocimar. Utilizando ergonomia, interface e design para melhorar a interação e a usabilidade do ambiente web. **Rev. Ambiente Acadêmico**, v. 2, n. 2, 2016. Disponível em <http://cachoeirodeitapemirim.multivix.edu.br/wp-content/uploads/2017/05/revista-ambiente-academico-edicao-4-artigo-7.pdf>. Acesso em: 5 dez. 2017.

Portal Educação. **Arquitetura básica de uma aplicação web**. Disponível em: <https://www.portaleducacao.com.br/conteudo/artigos/educacao/arquitetura-basica-de-uma-aplicacao-web/37326>. Acesso em: 6 dez. 2017.

ROCHA, Helder L. S. da. **Arquitetura da Web**. Disponível em: <http://www.argonavis.com.br/cursos/web/WebDesign_1.pdf>. Acesso em: 6 dez. 2017.

RYAN, J. **A history of the Internet and the digital future.** Londres: Reaktion Books, 2010.

SANTA MARIA, J. **On Web Typography.** Nova York: A Book Apart, 2014.

SHARKIE, C.; FISHER, A. **Jump Start Responsive Web Design.** Sydney: Sitepoint, 2013.

SIEGLER, M. G. Eric Schmidt: Every 2 Days We Create As Much Information As We Did Up To 2003. **TechCrunch**, 4 ago. 2010. Disponível em: <http://techcrunch.com/2010/08/04/schmidt-data/>. Acesso em: 23 abr. 2016.

SITE {404} PAGE Found. Disponível em:<http://www.404pagefound. com>. Acesso em: 9 abr. 2016.

SITE INTERNET LIVE STATS. Página Total Number of Websites. Disponível em: <http://www.internetlivestats.com/total-number-of-websites/>. Acesso em: 9 abr. 2016.

STATCOUNTER. StatCounter Global Statistics - Browser Usage. Disponível em: <http://gs.statcounter.com/#browser-ww-monthly-201503-201603>. Acesso em: 23 abr. 2016.

TEAGUE, J. **Speaking in Styles**: Fundamentals of CSS for Web Designers. Berkeley: New Riders, 2009.

THOMAS, S. **HTTP Essentials**: Protocol for Secure, Sacalble Websites. Nova York: Wiley, 2001.

TITTEL, E.; MINNICK, C. **Beginning HTML5 and CSS3 For Dummies**. Nova York: Wiley, 2013.

TRAVIS, D. User Experience: The ultimate guide to usability. **Udemy**, 2013. Disponível em: <https://www.udemy.com/ultimate-guide-to-ux/>. Acesso em: 9 abr. 2016.

W3C - World Wide Web Consortium. CGI: Common Gateway Interface. **Site W3C**, 5 dez. 2011. Disponível em: <http://www.w3.org/CGI/>. Acesso em: 24 abr. 2016.

W3TECHS. Usage of webservers for websites. Disponível em: <http://w3techs.com/technologies/overview/web_server/all>. Acesso em: 23 abr. 2016.

Ferramentas de desenvolvimento para tecnologias Web

Convite ao estudo

Preparados para iniciarmos nossos estudos nesta nova unidade? Perfeito, vamos lá! No assunto anterior, estudamos quais são os elementos necessários para podermos iniciar nosso desenvolvimento em Tecnologias web, como desfrutar da usabilidade web a nosso favor, como escolher a correta tipografia, cores, formas de navegabilidade, utilização de *sites* responsivos, como definir corretamente a arquitetura web a ser utilizada, sua teoria de aplicação e como desenvolver conteúdos para web e criar ferramentas de otimização para os *sites* de busca encontrarem seus conteúdos dentre milhares e milhares de *sites* disponíveis na web.

A partir desta unidade estudaremos como desenvolver os *sites* de forma simples e bem estruturada. Vamos estudar as ferramentas de desenvolvimento existentes no mercado.

Você conhecerá a linguagem de marcação, a base da programação para páginas web e poderá analisar as versões existentes no mercado, poderá elaborar diferentes folhas de estilos para as páginas web criando uma padronização e personalização do *site*, conhecerá e será capaz de utilizar ferramentas de uso comum no desenvolvimento de páginas e recursos Web, como PHP e Javascript, e, também, conhecerá e estará apto a utilizar ferramentas de uso comum no desenvolvimento de páginas e recursos Web, como JQuery e Python.

Com este estudo, temos certeza de que você terá plenas condições de começar a desenvolver grandes *sites* e ferramentas úteis para a web.

Para exemplificarmos essa situação, suponhamos que você foi contratado por uma empresa de desenvolvimento

de *sites* como estagiário e o Sr. Maurício, gerente de uma papelaria, entra em contato com a empresa para desenvolver um *site* para seu comércio. Como primeiro desafio no novo emprego, você ficará responsável pela elaboração e por todo desenvolvimento do projeto e, para iniciar, você precisará analisar as ferramentas disponíveis.

Analisar e compreender as ferramentas é um fator importantíssimo para se ter um comparativo de qual o melhor momento para utilizá-las, visto que nem todos os *sites* exigem os mesmos recursos, além da existência de inúmeras ferramentas disponíveis no mercado.

Nesta unidade vamos estudar as diferenças sobre as linguagens de marcação, o ponto inicial para começarmos a desenvolver páginas para a web. Então, vamos começar?

Seção 2.1

Ferramentas de desenvolvimento de tecnologias Web

Diálogo aberto

Nos dias de hoje, acredito que você já reparou que a navegação por *sites* na internet se tornou algo tão comum quanto comer, tomar banho, sair para passear e até mesmo dormir. E já reparou que às vezes essa navegação não é tão simples assim? Há *sites* que são simples, intuitivos, agradáveis, bonitos e, principalmente, fáceis de utilizar, aos quais sempre retornamos.

A criação de um *site* convidativo e agradável depende muito de qual linguagem você vai usar para desenvolver e isso é um ponto fundamental na qualidade final. Seja pela abertura de um *site* pelo computador ou pelo tablet ou até mesmo o celular. E vale lembrar que o celular, hoje, é um dos principais dispositivos de acesso à internet.

Já imaginou um *site* criado por você que ninguém visite? É comum acontecer, simplesmente porque quem o desenvolveu o fez sem uma base estruturada. Ainda bem que é só imaginação, não é verdade? A partir de agora você criará *sites* cada vez melhores e muito bem estruturados.

A estrutura do *site* é de fundamental importância pois é sobre ela que todo o *site* será desenvolvido. Observe a construção de um edifício e você verá que a estrutura dele deve ser muito reforçada e bem estruturada, caso contrário, o edifício não teria sustentação para ser construído e chegar ao objetivo final. E seu *site* será assim: com uma base reforçada, ele irá crescer a cada dia com as ferramentas de desenvolvimento que ensinaremos durante as aulas.

É baseando-se nessa estrutura que o *site* a ser criado para a papelaria do sr. Maurício deve ser bem observado e desenvolvido por você. É preciso conhecer as diferentes versões de linguagens de marcação e, a partir disso, elaborar esse projeto do *site* da papelaria.

Temos uma nova versão, que pode apresentar alguma incompatibilidade com os navegadores, ou a versão anterior que não apresenta esse problema, porém é desatualizada. Afinal, qual utilizar? E no futuro? Essa versão anterior funcionará ou precisará ser utilizada a nova versão e terá de ser reescrito o *site*?

Com base nesses questionamentos, estudaremos as principais diferenças existentes entre as versões mais utilizadas de linguagem de marcação.

Pronto para esse desafio?

Então, vamos lá!

Não pode faltar

Nesta unidade apresentaremos as ferramentas mínimas necessárias ao desenvolvimento de *sites*. E, para iniciarmos, a principal ferramenta é o HTML (*HyperText Markup Language*, ou no nosso português, Linguagem de Marcação de Hipertexto). Segundo Freeman (2008), o HTML é uma ótima linguagem de marcação para a descrição da estrutura das páginas web.

O HTML foi desenvolvido por Tim Berners-Lee, conforme afirma Silva (2011), e começou a ganhar popularidade quando o *browser Mosaic*, desenvolvido na década de 1990 por Marc Andreessen, começou a ser mais utilizado. A partir daí, os desenvolvedores de navegadores web começaram a utilizar o HTML como base, compartilhando as mesmas convenções, segundo Friedman (2014).

Assimile

Browser vem do verbo *to browse*, que significa folhear casualmente as páginas de um livro, e foi traduzido para o português como navegador, gerando a tão bem conhecida expressão "navegar na internet" (SILVA, 2011). A partir dessa informação, usaremos somente o termo em português: - navegador.

Browser Mosaic é considerado o primeiro navegador gráfico lançado.

Conforme o W3C (2010), hipertextos são elementos básicos de uma página web. Nestes elementos são criados links (ligações) para outras informações do mesmo *site* ou de outro *site*. Por exemplo, o *site* da Wikipédia é formado por hipertextos e podemos identificá-los com a cor em azul. Esses hipertextos, ao serem clicados, nos levam a ver informações na própria página ou em outra página dentro do próprio *site* e até mesmo para outros *sites* externos que não pertencem à Wikipédia.

A página é transferida de um computador remoto para o usuário, onde o *browser* faz o trabalho de interpretar os códigos naquele documento e mostra a página que o usuário visualiza no seu monitor. A Web está estruturada em dois princípios básicos: HTTP (*HyperText Transfer Protocol*) e HTML (*HyperText Markup Language*).

Segundo Silva (2011, p. 20), podemos conceituar hipertexto da seguinte forma:

> **Podemos resumir hipertexto como todo o conteúdo inserido em um documento para a web e que tem como principal característica a possibilidade de se interligar a outros documentos da web. O que torna possível a construção de hipertextos são os links, presentes nas páginas dos *sites* que estamos acostumados a visitar quando entramos na internet.**

Pesquise mais

Para conhecer mais sobre os padrões HTML, consulte também o livro: *Construindo Sites Adotando Padrões WEB*, de Marcelo da Silva Macedo, no qual o autor descreve técnicas para a construção de *sites* utilizando padrões Web (Web Standards) recomendados pelo W3C (*World Wide Web Consortium*) e como a sua utilização resulta em várias vantagens para os desenvolvedores e usuários (W3C, 2013).

De acordo com a entidade W3C (*World Wide Web Consortium*), liderada por Tim Berners-Lee, inventor da Web, são necessários três pilares para seu funcionamento, segundo Silva (2011):

- Um esquema de nomes para localização de fontes de informação na Web, chamado URL (*Uniform Resource Locator*, ou em português, Localizador Padrão de Recursos).
- Um protocolo de acesso para acessar estas fontes, hoje o HTTP.
- Uma linguagem de hipertexto, para a fácil navegação entre as fontes de informação: o HTML.

Assimile

Segundo Mendes (2015), o protocolo HTTP (*Hypertext Transfer Protocol*) é quem informa ao navegador como conversar com o servidor que possui a página com a relação dos cursos de redes. Sempre que você vir o protocolo HTTP, significará que você estará navegando pelas páginas na Internet.

Segundo o W3C (2010), a linguagem de marcação de hipertexto evoluiu para as versões HTML+, HTML2.0 e HTML3.0, de 1993 a 1995, quando várias mudanças foram propostas para melhoraria da linguagem e, somente em 1997, trabalhou na versão 3.2 da linguagem, fazendo com que ela fosse tratada como padrão de linguagem.

O HTML foi criado para ser uma linguagem independente, sem ter a obrigatoriedade de navegadores específicos, de plataformas próprias e outros meios de acesso. Dessa forma evitou-se que a Web fosse desenvolvida em uma base proprietária, com formatos incompatíveis e limitada para um determinado sistema operacional ou dispositivo que a utilizasse. Sendo assim, o HTML deve ser entendido universalmente, permitindo a reutilização dessa informação de acordo com as limitações de cada meio de acesso (W3C, 2013).

Os comandos em HTML são denominados de *Tags* (termo em inglês que significa "etiqueta"). Esses comandos têm uma sintaxe própria, e precisam seguir algumas regras:

- As *tags* devem aparecer entre os sinais de "menor que" (<) e "maior que" (>);
- A *tag* de finalização dos comandos possui uma barra (/) para indicar a finalização da mesma.

Exemplificando

Tag único: quando não precisamos finalizar o comando.
< br > - Este comando pula para próxima linha

Tag duplo: normalmente padrão dos comandos HTML
< font > Olá < /font > - Esta *tag* inicializa o comando com <font> para alterar os padrões da fonte e finaliza o comando com o < /font>

Todo o documento HTML fica contido entre os *tags*: <HTML> e </HTML>.

Uma página HTML apresenta três componentes básicos: estrutura principal, cabeçalho e corpo de página, sendo toda a página dividida em quatro *tags* básicas:

- <html> ... </html> - Indicam o início e o fim de um documento.
- <head> ... </head> - Indicam os parâmetros de configuração de cabeçalho do documento, como título e onde colocamos as configurações de *SEO*, que estudamos na unidade anterior.

- <title> ... </title> - Indica o título da página exibida no navegador. Esta *tag* deve estar sempre dentro das *tags* <head> </head>.
- <body> ... </body> - Envolvem a seção de corpo do documento. Aqui fica o conteúdo principal da página web.

Iniciada no ano de 2004 e incorporada pelo W3C em 2007, a especificação para HTML5 busca apresentar soluções relacionadas à implementação de uma versão da linguagem HTML moderna e compatível com as versões existentes, para tanto a linguagem deve ser única, podendo ser escrita tanto com sintaxe HTML como XML (*eXtensible Markup Language*, ou seja, Linguagem Extensível de Marcação, utilizada para gerar linguagens de marcação para necessidades especiais) e um aperfeiçoamento da marcação dos documentos.

Para as versões anteriores da linguagem HTML, a especificação final era aprovada por um comitê do W3C antes mesmo de ser totalmente implementada e a HTML5 não será concluída enquanto pelo menos duas implementações completas das funcionalidades da especificação não forem aplicadas. Essas especificações para a HTML5 definem uma sintaxe que é compatível tanto com a HTML quanto com a XHTML (*eXtensible Hypertext Markup Language,* isto é, Linguagem Extensível para Marcação de Hipertexto), linguagem com integração entre HTML e XML.

O surgimento do HTML5, segundo Silva (2011), tem como um dos principais objetivos facilitar a manipulação dos elementos, possibilitando, assim, ao desenvolvedor, modificar as características dos objetos, de forma que sejam transparentes para o usuário final, criando novas *tags* e permitindo modificar a função das demais.

Enquanto nas versões anteriores do HTML não existia um padrão na criação de seções de páginas e não havia padrão de nomenclaturas para os *IDs* (Identificadores), *tags* ou por meio de blocos de código entre *tags* chamados de classes, assim, não havia um método de captura automática para as informações do *site*, conforme W3C (2010).

Reflita

Com o surgimento do HTML5, muitas páginas na web têm se modernizado por meio dessa nova tecnologia, criando *sites* mais interativos. Porém, é uma tecnologia longe de ser totalmente implementada e com várias funcionalidades ainda inativas em alguns navegadores atuais e antigos. Tendo por base essa situação, compensa arriscar a migração para essa tecnologia hoje, sabendo-se que muitos usuários possuem navegadores antigos ou desatualizados? Reflita.

Conforme Silva (2011), alguns elementos e atributos tiveram funções e significados modificados para que pudessem ser reutilizados de forma mais eficaz. Elementos como <B>, utilizado para definir o texto em negrito e <I>, para definir o texto em itálico, são elementos descontinuados em versões desatualizadas de HTML e apresentam mais significado para usuários, assumindo funções diferenciadas em novas versões.

A Figura 2.1, abaixo, apresenta uma estrutura de linguagem de marcação utilizando HTML, enquanto a Figura 2.2 utiliza a linguagem de marcação HTML5, demonstrando a diferença na declaração de *tags* entre as duas versões.

Figura 2.1 | Exemplo de marcação HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html lang="pt-br">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>Título da página</title>
        <link rel="stylesheet" type="text/css" href="/estilos/main.css">
    </head>
    <body>
        <h1>Minha página HTML4</h1>
    </body>
</html>
```

Fonte: adaptada de Silva (2011).

Figura 2.2 | Exemplo de marcação HTML5 compatível com a sintaxe HTML da Figura 2.1

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title> Título da página </title>
        <link rel="stylesheet" href="/estilos/main.css">
    </head>
    <body>
        <h1>Minha página HTML5</h1>
    </body>
</html
```

Fonte: adaptada de Silva (2011).

O HTML5 apresenta uma organização de informações e escrita da página da web mais interativa e com mais lógica, não sendo necessária a instalação de *plug-ins*, o que facilita a reutilização de código para novos dispositivos.

Segundo Silva (2011), as principais diferenças entre a HTML e a HTML5 se originam no fato da HTML5 ter sido desenvolvida com a finalidade de substituir tanto a HTML, uma linguagem antiga, criada nos anos 90, quanto a XHTML, uma tentativa de aplicação do HTML com aplicação XML.

A semântica é um dos recursos mais interessantes do HTML5, com a ideia de que os padrões atuais de desenvolvimento possibilitam recursos de representação da informação. Com a denominação de Web Semântica, segundo Silva (2008), caso os computadores fossem capazes de realizar as tarefas de identificação dos significados, que hoje são de responsabilidade do usuário, o desempenho dos recursos seria melhor, podendo semanticamente entender as necessidades dos usuários e disponibilizando melhores resultados esperados.

Abaixo, a Figura 2.3 representa um comparativo da estrutura semântica entre as versões HTML e HTML5.

Figura 2.3 | Comparativo dos elementos estruturais entre HTML e HTML5

Fonte: adaptada de W3C (2013).

A Tabela 2.1 apresenta um descritivo das especificações dos elementos estruturais semânticos entre as versões HTML e HTML5.

Tabela 2.1| Comparativo dos elementos estruturais

Tag HTML	Tag HTML5	Especificação
	<header></header>	Especifica um cabeçalho de um documento ou seção.
	<nav></nav>	É uma seção do documento que agrupa links de navegação para outra parte do *site* ou aplicativo.
	<article></article>	Representa um conteúdo alto, relevante e independente.
	<aside></aside>	Define um bloco de conteúdo que faz referência ao conteúdo principal que o cerca.
	<footer></footer>	Normalmente conhecida como rodapé, área inferior.
	<section></section>	É um elemento genérico que pode abrigar outros elementos e sua principal função é definir seções em um documento.

Fonte: adaptada de W3C (2013).

Os navegadores atuais já suportam a grande parte das funcionalidades da versão HTML5, porém existem ferramentas desenvolvidas em linguagem *Javascript* que permitem detectar as funcionalidades não suportadas desta versão em cada navegador e, assim, o desenvolvedor pode criar uma alternativa para este recurso ainda não suportado nesses navegadores, de acordo com Silva (2011).

Segundo afirma Silva (2011), desenvolver um projeto web em HTML5 nos navegadores atuais é uma opção que pode ser considerada; caso você decida por ela, é preciso levar em conta algumas restrições e mecanismos existentes para contorná-las.

Pesquise mais

Para entender mais sobre os padrões regidos pelo W3C, para padronização de *sites* web, acesse: <http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf>. Acesso em: 4 dez. 2017.

Assimile

Plug-ins - São módulos para adicionar funções a outros programas maiores.

XHTML - É a sigla de *EXtensible HyperText Markup Language*, que significa Linguagem Extensível para Marcação de Hipertexto.

XML - É a sigla de *Extensible Markup Language*, que significa Linguagem Extensível de Marcação. É uma recomendação para gerar linguagens de marcação para necessidades especiais.

Sem medo de errar

É importante lembrar que as linguagens de marcação são fundamentais para a criação das páginas web, elas são as estruturas básicas da criação. A definição de qual versão iremos trabalhar vai definir no futuro como a página web vai funcionar e até evoluir. É importante, também, conhecer a diferença na semântica entre HTML e HTML5.

O Sr. Maurício entrou em contato com a empresa em que você trabalha para a criação de uma página web da papelaria. Você ficou encarregado de auxiliar o Sr. Maurício a criar essa página.

Sabemos agora, após conhecermos as versões das linguagens de marcação, que a linguagem de marcação HTML5 é a versão mais atualizada disponível e com ela podemos otimizar algumas declarações de *tags*.

A forma de declaração dessas *tags* é mais simplificada, permitindo uma melhor compreensão das linhas criadas e uma estrutura mais confiável. O fato de ser uma versão mais atual nos permite ter um tempo de uso mais longo dessa linguagem de marcação, e já estamos nos preparando para novas ferramentas futuras e *plug-ins* que serão criados para essa linguagem.

O avanço nas funcionalidades e na manipulação de elementos é outro ponto que nos leva a utilizar a linguagem na versão HTML5, e possibilita, ainda, a modificação das características dos objetos, de forma que seja transparente para o usuário final, criando novas *tags* e permitindo modificar a função das demais.

As páginas web tendem a ficar mais interativas com a utilização do HTML5 na sua criação, apesar de existir uma fatia do mercado utilizando navegadores antigos sem suporte a HTML5.

Atenção

Ao fazer o levantamento da linguagem de marcação a ser utilizada, é preciso levar em consideração o cuidado com os navegadores antigos ou sem atualização. Existem muitos usuários com navegadores nessa situação que podem não ter o acesso às funcionalidades corretas do *site* com HTML5.

Avançando na prática

Aparecendo para o mundo

Descrição da situação-problema

O escritório de contabilidade A_A Contabilidade está em pleno funcionamento na área de contabilidade há mais de 15 anos, porém o proprietário tem pouca divulgação dos seus serviços na cidade e gostaria de expandir mais seu escritório. Uma das soluções encontradas foi fazer a divulgação dos seus serviços na internet, onde o escritório poderá não só divulgar seus serviços para sua cidade, mas também para toda a região. Baseando-se nesta situação, é sua vez de levantar as informações para a criação do *site* e apresentar ao escritório de contabilidade como ele será. Preparado?

Lembre-se

Lembrando o estudo da unidade anterior, é importante levar em consideração a usabilidade que é a facilidade com que os visitantes de um *site* conseguem navegar e encontrar o que procuram nele.

Para levantar as informações para a criação do *site*, além de definir qual linguagem de marcação será utilizada, é necessário pesquisar e relembrar algumas lições aprendidas na unidade anterior, como ergonomia de recursos web, projeto e arquitetura aplicados à web e otimização de busca. Com o auxílio da tabela abaixo, defina quais são essas informações:

Linguagem de marcação	
Qual versão utilizar?	
Ergonomia de Recursos Web	
Usabilidade	
Navegabilidade	
Tipografia	
Cores	
Utilizará Responsividade?	
Arquitetura e Design de interface	
Itens que serão parte da interface	
Otimização de Máquina de Busca	
Utilizará SEO?	

Resolução da situação-problema

Após relembrarmos os itens da unidade anterior e levantarmos as informações por meio de pesquisas sobre qual linguagem de marcação usar, como podemos utilizar a usabilidade, como funcionará a navegabilidade da página, a tipografia e cores a serem adotadas, quais itens serão criados no design da página e se a página irá ter otimização de SEO e responsividade, podemos agora apresentar as informações ao escritório de contabilidade conforme abaixo:

Linguagem de marcação	
Qual versão utilizar?	- Linguagem na versão HTML5, devido às funcionalidades e manipulação de elementos. É a linguagem mais atualizada disponível.
Ergonomia de Recursos Web	
Usabilidade	- Não ter poluição visual no *site*, como imagens piscando ou conteúdos aglomerados;
Navegabilidade	- Utilização de links para as diversas áreas do *site*; - Exibir na parte de cima e no rodapé do *site* os telefones para contato;
Tipografia	- Fontes Arial ou Verdana, tamanho 12 ou 13;
Cores	- Utilizar as cores da logomarca da empresa;
Utilizará Responsividade?	- Sim, um dos motivos pelo qual utilizaremos o HTML5.
Arquitetura e Design de interface	
Itens que serão parte da interface	- Menus para navegação; - Criação de uma página inicial com um resumo dos serviços; - Ter textos condizentes com os serviços do escritório; - Utilizar imagens da empresa juntamente com os textos.
Otimização de Máquina de Busca	
Utilizará SEO?	- Aplicação de otimização de SEO.

Faça você mesmo

Com a ideia de usabilidade do *site* do escritório de contabilidade apresentada, chegou a hora de você desenhar um esboço de como seria essa página na internet. Lembrando que precisa estar conforme o que foi apresentado ao escritório.

Faça valer a pena

1. O HTML5 apresenta uma organização de informações e escrita da página da web mais interativa e com mais lógica, não sendo necessária a instalação de plug-ins, o que facilita a reutilização de código para novos dispositivos. As páginas web tendem a ficar mais interativas com a utilização do HTML5.

Sobre essa afirmação, indique a alternativa que melhor se adequa a ela:

a) As principais semelhanças entre a HTML5 e a HTML são originadas no fato da HTML5 ter sido desenvolvida com a finalidade de substituir tanto a HTML, uma linguagem antiga criada nos anos 90, quanto a XHTML.

b) Os padrões atuais de desenvolvimento do HTML5 ainda não possibilitam recursos de representação da informação.

c) XHTML (*EXtensible HyperText Markup Language*) é uma versão melhorada do HTML 5.

d) Os navegadores atuais já suportam a grande parte das funcionalidades da HTML5, porém existem ferramentas desenvolvidas em linguagem Javascript que permitem detectar as funcionalidades suportadas.

e) Desde as versões anteriores de HTML já existia um padrão na criação de seções de páginas e padrão de nomenclaturas para os IDs, *tags* ou classes.

2. Segundo Mendes (2015), o protocolo HTTP (*Hypertext Transfer Protocol*) é quem informa ao navegador como conversar com o servidor que possui a página com a relação dos cursos de redes. Sempre que você vir o protocolo HTTP, significa que você está navegando pelas páginas na Internet.

De acordo com a entidade W3C (*World Wide Web Consortium*), liderada por Tim Berners-Lee, inventor da Web, são necessários três pilares para o funcionamento do HTML, qual alternativa melhor apresenta esses pilares?

a) Esquema de nomes para localização de fontes de informação na Web, chamado URL; um protocolo para acesso das fontes HTTP e um catálogo de *sites* e endereços.

b) Esquema de nomes para localização de fontes de informação na Web, chamado URL; uma linguagem de hipertexto para facilitar a navegação entre as fontes de informação HTML e um protocolo de comunicação, chamado de HTTP.

c) Um catálogo de web *sites*; uma linguagem de hipertexto para facilitar a navegação entre as fontes de informação HTML e a ferramenta CCS.

d) URL (em português, localizador real de linguagens), HTTP, que é o protocolo de acesso para localização de fontes, e a ferramenta Javascript.

e) Um W3C, além de um protocolo para acesso das fontes HTTP, um catálogo de *sites* e HTML, que é uma linguagem de hipertexto facilitadora da navegação entre as fontes.

3. Sobre as afirmações abaixo:

Julgue os itens abaixo:

I) *Plug-ins* são módulos para adicionar funções a outros programas maiores.

II) XHTML é sigla de *EXtensible HyperText Markup Language*, em português: Linguagem Extensível para Marcação de Hipertexto.

III) O HTML5 possui a mesma semântica quando se refere a elementos <head>.

IV) XML é uma recomendação para gerar linguagens de programação para necessidades especiais.

Assinale a alternativa correta de acordo com as afirmações acima:

a) Apenas I e II

b) Apenas II e III

c) Apenas I, II e IV

d) Apenas II, III e IV

e) I, II, III, e IV.

Seção 2.2

Linguagem de marcação para Web

Diálogo aberto

Estudamos na seção anterior a importância de termos uma estrutura HTML para nossas páginas web e, assim, poderemos desenvolver páginas web com maior qualidade e dentro das normas estabelecidas pela W3C (*World Wide Web Consortium*).

Agora, você já sabe por onde iniciar a criação da página do Sr. Maurício, e em qual linguagem de marcação podemos desenvolver a estrutura básica da sua página.

O Sr. Maurício solicitou à empresa, onde você trabalha como estagiário, uma página web para a papelaria na qual ele é gerente. E você, como responsável pela criação desta página, precisa conhecer agora como criar as definições para personalização e padronização da página.

Andando pela rua da sua casa, pelo seu bairro, pela sua cidade e até mesmo durante uma viagem, você já deve ter observado as casas e prédios construídos, correto? Se não observou ainda, comece a observar. Com certeza, em algum desses lugares por onde passou, você já deve ter observado alguma casa ou prédio em construção.

Um dia você passou por uma dessas construções e ela estava apenas no aterramento e alguns dias depois, ao passar novamente por ali, você deve ter observado que as paredes já estavam sendo levantadas. Essa construção foi sendo preparada, até que, um dia, ela estava praticamente pronta e toda sua estrutura levantada, faltando apenas os acabamentos.

Passados alguns dias, você observou que essa construção já estava com portas, janelas, acabamento em gesso e pintura, ficando muito mais bonita e interessante do que o observado anteriormente.

Pois bem, caro aluno, é o acabamento para essa construção que iremos aprender nesta seção. A construção da qual falamos é sua página web somente com a linguagem de marcação, com sua estrutura básica e sem nenhum acabamento. Tenho certeza de que as páginas somente com sua estrutura não são tão atrativas

aos olhares dos visitantes, assim como as construções sem acabamento de janelas, portas e pinturas também não realizadas, pois não prendem a atenção dos que a visitam.

Quando adicionamos acabamento, ou seja, aplicamos folhas de estilo às nossas páginas, elas ficam mais atraentes e com visual mais elegante e bonito, inserindo cores, tipos de fonte e botões.

Estudaremos como é simples desenvolver as folhas de estilo e assim deixar as páginas da papelaria do Sr. Maurício com uma aparência mais elegante e com uma apresentação profissional.

Convido vocês a entrar neste universo que é a internet e a produzir páginas atrativas aos usuários e com um belo visual.

Não pode faltar

Olá, caros alunos. Vamos estudar nesta unidade como criar a estrutura da linguagem de marcação (HTML5) e como criar e aplicar as folhas de estilos nas páginas web criadas.

Para relembrar, HTML (*HyperText Markup Language*), ou seja, Linguagem de Marcação de Hipertexto, é a base para se criar uma página web, segundo Freeman (2008). Sem o HTML não seria possível o desenvolvimento dos websites e da própria internet.

Vimos as diferenças entre HTML e HTML5 e vimos também que, para iniciar um arquivo de página web, precisamos declarar as *tags* de comando. Com o surgimento do HTML5, alguns comandos perderam a compatibilidade e deixaram de funcionar como por exemplo a *tag* <font>, segundo Silva (2011).

Toda a estrutura das *tags* de comando HTML é criada baseada em uma série de elementos em forma de árvores, nos quais alguns comandos provêm de outros comandos, segundo a W3C (2013).

Para o início de um arquivo em HTML, devemos começar pela *tag* de comando <html> e finalizar com </html>, já no arquivo HTML5, começamos por uma instrução chamada *DOCTYPE* (Tipo de Documento), para que o navegador tenha informações sobre qual versão de código a marcação foi escrita, define W3C (2013).

Conforme Silva (2011), a declaração *DOCTYPE* deve ser feita na primeira linha da marcação HTML e nada deverá existir acima desta declaração de DOCTYPE, nem mesmo uma linha em branco.

Após a informação do tipo de documento, é inserida a tag de comando <html>, e diferentemente do HTML, em HTML5 definimos a linguagem do documento dentro dessa mesma *tag*, através do atributo LANG, por exemplo: *<htmllang="pt-br">*, vide quadro "Exemplificando" abaixo.

A *tag* HEAD é a parte da página onde ficam as informações que não serão visíveis aos usuários, que pode ser chamada de cabeçalho da página. No quadro "Exemplificando", podemos observar a declaração desta *tag* HEAD e algumas *tags* de metadados.

Assimile

Dentro da *tag* HEAD é onde declaramos as *tags* de scripts e de metadados, que são informações adicionais da página, como título da página, folhas de estilos, scripts, links e outras informações de configuração da página, assim como as informações de SEO, conforme estudamos na Unidade 1.

Segundo Silva (2011), uma das melhorias realizadas no HTML5 foi referente a *tag* responsável pela codificação de caracteres da página, chamada de *meta charset*. Em versões anteriores do HTML5, sua escrita é mais extensa, por exemplo *<meta http-equiv="Content-Type" content="text/html; charset=utf-8">*, já no HTML5, sua escrita pode ser definida como *<meta charset="utf-8">*. A forma de declaração das versões anteriores ainda é suportada no HTML5. Ficou mais simples, não é?

A necessidade de declaração do tipo de codificação de caracteres surgiu para permitir que pessoas de qualquer lugar do mundo possam acessar qualquer *site*, independentemente da sua localização global. Criou-se, então, a tabela chamada Unicode, uma tabela padrão, que possui milhões de caracteres de forma padrão, fornecendo um único número para cada caractere, não importa a plataforma, nem o programa, nem a língua. É utilizada pela maioria de sistemas e navegadores atualmente, permitindo, assim, que qualquer pessoa em qualquer lugar do mundo possa visualizar qualquer *site* de forma correta e sem erros de codificação, afirma W3C (2013).

A *tag* TITLE indica qual o título da página será apresentado na aba do navegador.

Outra declaração que pode ser utilizada como um elemento filho HEAD é a *tag* de comando LINK. Em HTML, existem dois tipos de *tags* de links: a *tag* <A LINK> que são links utilizados para levar o usuário a outras páginas ou arquivos e a *tag* LINK, que são links para arquivos de configuração que serão usadas na página, mas estão em fontes externas, segundo Silva (2011). Um exemplo da *tag* de comando LINK poderia ser: *<link rel="stylesheet" type="text/css" href="estilo.css">*, na qual o atributo *rel="stylesheet"* significa que aquele link é relativo à importação de um arquivo referente a folhas de estilo, e o atributo href="estilo.css" se refere ao nome do arquivo a ser utilizado, chamado de estilo.css.

Em HTML5 podem ser inseridos outros links relativos como o *rel="archives"* que significa a inserção de uma referência a uma coleção de material histórico da página. Por exemplo, o blog pode referenciar esta *tag* por usar um histórico.

Exemplificando

Vamos aproveitar este exemplo de uma página, com a estrutura básica de HEAD, no HTML5, copiar e colar no bloco de notas e salvar como index.html:

```
<!DOCTYPE html>
<html lang = "pt-br">
  <head>
    <meta charset="UTF-8">
    <title>Página de exemplo</title>
    <link rel="stylesheet" href="estilo.css">
  </head>
  <body>
    <h1> Aula de Tecnologia para Web </h1>
  </body>
</html>
```

Depois, carregue este arquivo salvo no navegador para ver o resultado.

Agora, vamos utilizar o exemplo da estrutura básica com os metadados de SEO, copiar e colar no bloco de notas e salvar como index.html:

```
<!DOCTYPE html>
<html lang = "pt-br">
  <head>
    <meta charset="UTF-8">
    <title>Página de exemplo</title>
      <meta name="description" content="Página de exemplo de HTML5">
    <meta name="keywords" content="lista de palavras-chave">
    <meta name="author" content="Kroton">
    <meta name="generator" content="HTML">
    <meta name="robots" content="all">
   <link rel="stylesheet" href="estilo.css">
  </head>
  <body>
     <h1> Aula de Tecnologia para Web </h1>
  </body>
</html>
```

E carregue este arquivo salvo no navegador.

A *tag* BODY, apresentada no exemplo, é a tag de comando responsável pela apresentação das informações aos usuários. Todo o conteúdo da página deve ser inserido dentro desta *tag*.

Reflita

O uso da linguagem de marcação para o desenvolvimento de páginas web vem evoluindo conforme a tecnologia evolui. Seria possível, nos dias de hoje, desenvolver um *site* sem a necessidade de utilização de recursos de scripts ou folhas de estilos utilizando somente o HTML ou HTML5? Fica o desafio lançado.

Folhas de estilos

As folhas de estilos, mais comumente conhecidas como CSS (*Cascading Style Sheets*), são utilizadas para definir a apresentação de documentos escritos em uma página com linguagem de marcação, permitindo assim aperfeiçoar as páginas HTML com definições exclusivas, conforme Carvalho (2004).

Para a versão do HTML, utiliza-se o CSS como padrão de folhas de estilo das páginas. Com a evolução para o HTML5, o padrão dessa nova versão passou a ser o CSS3, com aprimoramento e algumas modificações nas folhas de estilo.

Segundo Silva (2011), as folhas de estilo têm como funcionalidades:

- Economia de tempo na criação de páginas;
- Diminuição do código da página;
- Maior velocidade para carregar a página;
- Maior facilidade para manter ou modificar as páginas;
- Maior controle na criação do *layout* da página.

Segundo o W3C (2013), o uso das folhas de estilo, em versões anteriores, deveria ser declarado dentro do próprio arquivo de página web criada. Isso impediria que suas definições pudessem ser utilizadas em diversas páginas ao mesmo tempo. Para manter a mesma definição em todas as páginas do *site*, tínhamos que declarar o código em todas as páginas.

A evolução do HTML fez com que o CSS evoluísse também, e atualmente, os arquivos de folhas de estilo podem ser declarados em arquivos auxiliares com a extensão .css. A criação da linguagem CSS em arquivos auxiliares facilita a criação de forma modular, separado por classes, permitindo a padronização de todo o conjunto

de páginas, de forma que é possível reutilizar essas definições em outros projetos, conforme W3C (2013).

Podemos declarar as folhas de estilos de três formas, segundo Silva (2011), baseando-se na sua localização: o *inline*, onde o código CSS está junto com o código HTML; o *linkado*, quando utilizamos um arquivo externo; e o *incorporado*, quando é declarado no início da página HTML.

A sintaxe do CSS, assim como no HTML, não é *case sensitive*, ou seja, permite-se utilizar tanto as letras maiúsculas como as minúsculas e permite também que a utilização de múltiplos espaços seja tratada como espaço simples, conforme Silva (2007).

Os arquivos de folhas de estilo são criados baseados em regras CSS e essa regra de sintaxe deve seguir o padrão: *elemento {atributo1: valor; atributo2: valor ...}*, em que o elemento descreve o mesmo da tag HTML, mas sem os sinais de maior e menor. O atributo seria a propriedade do elemento, por exemplo, font-size. E o valor é a configuração válida deste atributo, por exemplo, 16pt (16 pontos) para *font-size*. Podem ser utilizadas múltiplas declarações em um único elemento. Para isso basta apenas separar os atributos e seus valores com ponto e vírgula (;). Para finalizar as declarações, após a última declaração, não é necessário o uso do ponto e vírgula.

Exemplificando

Exemplo de uma declaração de folha de estilo, inserindo cor de fundo da página:

```
Body {
        background: blue
}
```

Como no exemplo anterior, precisamos salvar este arquivo dentro da mesma pasta da página HTML como estilo.css

Segundo Mazza (2012), podemos utilizar alguns *sites* alternativos para utilização de fontes, sem a necessidade de ter a fonte ou conhecer suas características, como: para carregar a fonte Press Start 2P (<https://www.google.com/fonts/specimen/Press+Start+2P>. Acesso em: 1 jun. 2018) com um visual diferente, basta adicionarmos a seguinte ao HEAD da sua página:

<linkhref='http://fonts.googleapis.com/css?family=Press+Start+2P'rel='stylesheet'>

Uma das vantagens de se utilizar o CSS é a personalização de vários recursos, como links para outras páginas e arquivos, que, se utilizados da forma padrão, nos darão um texto simples como forma de botão.

Exemplificando

Vamos criar um botão simples com efeito CSS, mas para isso precisamos criar a classe .botao dentro do arquivo CSS, chamado estilo.css:

```
.botao {
font:bold 20px Tahoma, Geneva, sans-serif;
font-style:normal;
color:#ffffff;
background:#1568db;
border:0px solid #ffffff;
text-shadow:0px -1px 1px #222222;
box-shadow:2px 2px 5px #000000;
-moz-box-shadow:2px 2px 5px #000000;
-webkit-box-shadow:2px 2px 5px #000000;
border-radius:10px 10px 10px 10px;
-moz-border-radius:10px 10px 10px 10px;
-webkit-border-radius:10px 10px 10px 10px;
width:65px;
padding:10px 34px;
cursor:pointer;
margin:0 auto }
.botao:active {
cursor:pointer;
position:relative;
top:2px }
```

E assim criamos o arquivo index.html

```
<!DOCTYPE html>
<html lang = "pt-br">
  <head>
    <meta charset="UTF-8">
    <title>Página de exemplo</title>
    <link rel="stylesheet" href="estilo.css">
  </head>
  <body>
    <button>Botão Normal</button>
    <button class=”botao”> Botão Normal </button>
  </body>
</html>
```

Observe que nesse exemplo, utilizando a *tag* BUTTON, é apresentado um botão simples no *site*, padrão do HTML, já a segunda tag apresentará um botão com a classe .botao na tag e na tela o botão aparecerá com o efeito de descida e com o cursor do mouse mudando a forma quando estiver sobre o botão.

A utilização das folhas de estilo nos *sites* permite um dinamismo maior e uma apresentação visual mais elaborada da página criada.

Pesquise mais

Para conhecer mais sobre a utilização de folhas de estilo em HTML5, consulte também o livro: *HTML5 e CSS3, Domine a web do futuro*, de Lucas Mazza, no qual o autor descreve uma visão do futuro da web com o HTML5 e a aplicação do CSS3 como forma de personalização e otimização, utilizando os novos recursos e melhorando a semântica das páginas web.

Sem medo de errar

Para termos uma construção atraente, ela precisa ser bem construída, bem estruturada e com acabamento que permita ao visitante observá-la e ter uma visão harmoniosa entre pintura das paredes e portão, modelo de portas e janelas. Sem cores carregadas e sem combinações.

Assim também deve ser uma página web como a do Sr. Maurício. A página da papelaria estará visível na web para todos acessarem. Como responsável pela página do Sr. Maurício, você terá de criar uma página atraente, e é importante ter uma estrutura HTML bem definida e, posteriormente, criar a pintura harmoniosa da página, ou seja:

- Criar as definições de folha de estilo para personalizar a página com cores, botões e links necessários.
- Uma página web permite a utilização de várias folhas de estilo em um único arquivo HTML e é importante pensarmos em uma forma que seja fácil a atualização no futuro, e dessa forma é melhor a criação de um arquivo de folha de estilo externo ao arquivo da página web e utilizamos a forma linkada de declaração CSS.

- Pode-se criar um arquivo auxiliar de folha de estilo para padronização de cada item específico por exemplo, um arquivo auxiliar para padronização de fontes e cores da página e outro para padronização de botões e links.

Conhecer e ser capaz de elaborar diferentes folhas de estilo para páginas web é importante para que se desenvolva páginas com qualidade e com facilidade de atualização ou modificação se necessário em suas definições.

Atenção

Vimos que as folhas de estilos são criadas em arquivos externos à estrutura HTML, porém é possível criar dentro da própria estrutura, mas cuidado em fazer desta forma, pois você terá que inserir essa mesma estrutura em todas as páginas do *site* a ser criado.

Avançando na prática

Personalizando uma página web

Descrição da situação-problema

Sua amiga Joana está interessada em criar uma página para publicar os poemas que ela escreve, e solicitou a você para desenvolver uma página na internet onde serão expostos esses poemas.

Nesta página, ela deseja alguns botões de menu, por exemplo:

- Página Inicial
- Poemas
- Contato

Na Página Inicial, deverá constar o último poema escrito pela Joana, e em Poemas, uma listagem com todos os poemas escritos por ela. Na página de Contato, ela deseja apenas que seja exibido seu e-mail e telefone. Joana gostaria que sua página tivesse cores e fontes agradáveis.

Com base nessa situação, sua função é estruturar a página web e criar as folhas de estilo dessa página. Mas como você fará essa tarefa? Vamos lá? Mãos à obra!

Lembre-se

Não se esqueça de criar e definir as personalizações em folhas de estilo (CSS) e de criar as classes e os links dentro do arquivo HTML chamando essas personalizações.

Resolução da situação-problema

Para o desenvolvimento desta atividade, é necessário a criação de página com estrutura HTML5 e seu arquivo auxiliar de folha de estilo. Abaixo segue uma estrutura básica simples de criação desta página.

Página Inicial index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8"/>
    <title>Pagina de poemas da Joana</title>
    <link rel="stylesheet" href="estilo.css"/>
</head>
<body>
<!-- Botões de Menu -->
<ul class="menu">
   <li><a href="index.html">Inicial</a>
   <li><a href="poemas.html">Poemas</a>
   <li><a href="contato.html">Contato</a>
</ul>
<!-- Parágrafo de título do texto da página -->
<h1>Pagina de poemas da Joana</h1>
<p>Bem vindos a minha pagina de poemas </p>
</body>
</html>
```

Página de Poemas.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8"/>
  <title>Pagina de poemas da Joana</title>
  <link rel="stylesheet" href="estilo.css"/>
</head>
<body>
```

```
<!-- Botões de Menu -->
<ul class="menu">
  <li><a href="index.html">Inicial</a>
  <li><a href="poemas.html">Poemas</a>
  <li><a href="contato.html">Contato</a>
</ul>
<!-- Parágrafo de título do texto da página -->
<h1>Poemas</h1>
<p>Poema 1</p>
<p>Poema 2</p>
<p>Poema 3</p>
<p>Poema 4</p>
</body>
</html>
```

Página de contato.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
   <meta charset="UTF-8"/>
   <title>Pagina de poemas da Joana</title>
   <link rel="stylesheet" href="estilo.css"/>
</head>
<body>
<!-- Botões de Menu -->
<ul class="menu">
  <li><a href="index.html">Inicial</a>
  <li><a href="poemas.html">Poemas</a>
  <li><a href="contato.html">Contato</a>
</ul>
<!-- Parágrafo de título do texto da página -->
<h1>Contato</h1>
<p>Entre em contato pelo telefone (xx) xxxx-xxxx </p>
</body>
</html>
```

Arquivo auxiliar de folha de estilo estilo.css

```
/* Personalização do corpo da página */
body {
    padding-left: 11em;
        background-color: lightgrey;
    font-family: Georgia, "Times New Roman", Times, serif;
}
```

```
/* Personalização do parágrafo H1*/
h1 {
    color: blue;
    font-family: Helvetica, Geneva, Arial, SunSans-Regular, sans-serif
}
/* Personalização do menu */
ul.menu {
    list-style-type: none;
    padding: 0;
    margin: 0;
    position: absolute;
    top: 2em;
    left: 1em;
    width: 9em
}
ul.menu li {
    background: white;
    margin: 0.5em 0;
    padding: 0.3em;
    border-right: 1em solid black
}
ul.menu a {
    text-decoration: none
}
a:link {
    color: blue
}
a:visited {
    color: purple
}
/* Personalização do parágrafo de texto*/
p {
    color:green;
}
```

Agora que você já conheceu como funcionam as definições de folhas de estilo, é sua vez de criar uma página web própria. Seu professor solicitou a criação do seu currículo na web, portanto, é hora de você criar sua página e divulgar aos seus amigos seu currículo para futuros contatos. Em seguida, compartilhe com seus colegas de sala.

Faça valer a pena

1. Para criarmos definições de propriedades e valores para personalização em uma página web, podemos utilizar um arquivo auxiliar com essas definições independente do arquivo principal. Assim, em caso de alteração da página, não precisamos editar o arquivo principal, somente este arquivo auxiliar.

Esse arquivo auxiliar é definido por:

a) XML (*eXtensible Markup Language*).
b) XHTML (*eXtensible Hypertext Markup Language*).
c) JavaScript.
d) CSS (*Cascading Style Sheets*).
e) HTML.

2. Faça uma análise dos dados que contém trechos de código em linguagem CCS abaixo:

Trecho 1:

```
<p style="color: blue; margin: 20px 20px 10px 100px">
  Este é o início do texto
</p>
```

Trecho 2:

```
<head>
  <link rel="stylesheet" href="default.css">
</head>
```

Conhecendo os tipos de CSS quanto a maneiras de utilizar folhas de estilo, esses trechos referem-se a:

a) incorporado e linkado.

b) linkado e inline.

c) incorporado e inline.

d) linkado e incorporado.

e) inline e linkado.

3. Sobre CSS (*Cascading Style Sheets*), de acordo com Carvalho (2004), as folhas de estilo são utilizadas para definir a apresentação de documentos escritos em uma página com linguagem de marcação, aperfeiçoando as páginas HTML de forma exclusiva.

Qual alternativa apresenta corretamente as funcionalidades de CSS e CSS3?

a) Economia de tempo na criação de páginas, além da diminuição do código empregado, porém com menor velocidade no carregamento da página.

b) Permite uma maior facilidade para manter ou modificar páginas, com economia de tempo e menor controle na criação de *layout* da página.

c) Diminuição do código da página, maior controle da criação do *layout* da página e economia de tempo na criação de páginas.

d) Maior velocidade no carregamento da página, aumento do código da página além do maior controle na criação do *layout* dela.

e) Aumento no tempo para criação das páginas, porém com carregamento mais rápido e permite maior facilidade para manter ou modificar as páginas.

Seção 2.3

Linguagem de desenvolvimento de tecnologias Web

Diálogo aberto

Olá, queridos alunos, espero que estejam gostando dos conceitos apresentados até o momento na disciplina de tecnologias para web e para dispositivos móveis. Já vimos o conceito sobre as diferentes linguagens de marcação e sua estrutura e também vimos, os conceitos sobre as folhas de estilo, permitindo deixar as páginas web mais atrativas aos usuários que as visitam.

Já perceberam o quanto é interessante um parque de diversões? Ele tem uma área geográfica e, nesta área, o parque monta sua estrutura toda. Desde a estrutura básica de alicerces para os brinquedos e local reservado do próprio parque, como toda a estrutura de acesso ao parque.

Entenderam a ideia da estrutura? Pois é, assim também acontece com as páginas web: precisamos criar uma estrutura básica de alicerce e a estrutura reservada na qual podemos alterar as informações conforme as necessidades.

Você frequentou algum parque de diversões há 10 anos? Já retornou anos depois ou recentemente a este parque? Caso tenha retornado, notou algo diferente nele, algum brinquedo novo, alguma atração nova? E, se você não retornou, por qual motivo? Não gostou do parque ou não retornou por não ter nenhuma atração nova? Você deve estar se perguntando, qual a ligação entre frequentar um parque de diversões e esta seção de ensino, não é?

Em páginas web acontece a mesma situação, e precisamos estar sempre atualizando as páginas com novas funções. Os estudos em PHP e JavaScript são importantes para conhecermos essas ferramentas, adicionando recursos e funções na página.

Quando uma pessoa acessa uma página web e retorna a essa página algum tempo depois, provavelmente encontra algo que lhe interessa, alguma informação nova, alguma funcionalidade interessante ou, até mesmo, retorna pela sua utilidade. E se uma vez

acessada, a pessoa não retorna mais a essa página? Provavelmente, ela não tem algo que lhe chame a atenção ou até mesmo pelo fato de a página estar sempre da mesma forma.

E você, como responsável pela criação da página do Sr. Maurício, precisa agora conhecer as linguagens de desenvolvimento PHP e JavaScript que, diferentemente da linguagem de marcação, nos permitem elaborar uma página mais dinâmica e interativa. Uma página web sempre vai precisar de páginas adicionais para compor todo o *site*, portanto, a página do Sr. Maurício vai precisar de páginas adicionais também. Como manter todas estas páginas com o mesmo padrão e a mesma simetria? O Sr. Maurício gostaria de saber, também, como está o acesso à sua página, quantas pessoas estão visitando sua página. Então, por que não criar novas funcionalidades para a criação da página ao Sr. Maurício?

Você terá conhecimento destas linguagens e será capaz de utilizar as ferramentas de uso comum no desenvolvimento de páginas web, com recursos para deixá-las com maior qualidade e praticidade.

E, assim, os clientes do Sr. Maurício poderão usufruir da página web criada com recursos novos e interessantes. Este é só o começo do instigante mundo do desenvolvimento web!

Não pode faltar

Caros alunos, estudaremos nesta seção as linguagens de desenvolvimento e, com elas, deixar os nossos *sites* mais interativos e dinâmicos de uma forma prática e bem simples.

Podemos construir páginas web somente utilizando a linguagem HTML, mas você perceberá, em determinado momento, que a construção da sua página estará simples, sem nenhuma funcionalidade atrativa ou dinâmica, limitada aos recursos básicos do desenvolvimento somente em linguagem de marcação. A construção somente utilizando HTML e folhas de estilo deixará o *site* de forma estática sem a interação do usuário com a página, como citam Welling e Luke (2005).

PHP

A linguagem de desenvolvimento, ou como chamamos também linguagem de programação PHP, foi criada em 1994 por Rasmus Lerdorf e inicialmente era chamada de *Personal Home Page Tools* (Ferramentas de Página Pessoais) e era composta por um conjunto de scripts em linguagem C, utilizado para a criação de páginas dinâmicas de monitoração de currículos. Atualmente, a linguagem PHP significa *PHP Hypertext Preprocessor*, segundo Dall'Oglio (2015).

Com o passar do tempo, a linguagem desenvolvida por Rasmus passou a ser utilizada por mais pessoas e foram adicionados novos recursos até que, em 1995, seu código fonte foi liberado e mais desenvolvedores se uniram ao projeto. O PHP, desde o seu surgimento, nunca parou de evoluir e com o tempo foi sendo reescrito e novas versões foram lançadas, conforme Dall'Oglio (2015). Na atualidade, o PHP é a linguagem de desenvolvimento mais utilizada no mundo, conforme Figura 2.4. Por ser uma linguagem de uso público e sem custo, é possível utilizar para criação e venda de *sites* devido à licença de uso livre. A versão mais utilizada nos dias de hoje é a 5.3, estando já disponível a nova versão 7.0.

Figura 2.4 | Linguagem de desenvolvimento web mais utilizadas no mundo

© W3Techs.com	usage
1. PHP	82.2%
2. ASP.NET	15.8%
3. Java	2.7%
4. static files	1.5%
5. ColdFusion	0.7%

Fonte: adaptada de W3Techs.

Segundo Welling e Luke (2005), o PHP é uma linguagem para inserção de scripts dentro de uma página. É no servidor em que está hospedada a página web, que esses scripts são executados e interpretados. As páginas são criadas em HTML e, juntamente, pode ser introduzida a linguagem PHP. Todas as vezes que a página é visitada, o servidor interpreta a linguagem e a executa, gerando o HTML ou outra funcionalidade criada.

As vantagens da utilização do PHP em páginas web, conforme Welling e Luke (2005), são:

- Maior desempenho na execução da linguagem;
- Permite conexão com diferentes bancos de dados;
- Permite a criação de bibliotecas integradas para realizar tarefas comuns na web;
- Baixo custo;
- Aprendizagem fácil e simples;
- Permite a utilização em vários sistemas operacionais diferentes;
- Código-fonte disponível;
- Facilidade em se obter suporte.

Todo documento PHP tem como extensão o .php, fazendo com que o servidor entenda que o documento é composto pela a linguagem de desenvolvimento e execute as suas funcionalidades corretamente.

Diferentemente do CSS (*Cascading Style Sheets*), o PHP não está envolvido com o layout da página e sim com seu funcionamento.

Segundo Lobo (2007), a sintaxe do script em PHP é similar à do HTML em suas declarações, sendo iniciadas e finalizadas com *tags*, porém as *tags* em PHP são diferentes em seus nomes de comando. Essas *tags,* ou delimitadores, podem ser colocadas em qualquer parte da página HTML para que o interpretador (servidor) identifique

e execute o que está programado. Existem quatro formas para inserirmos as *tags* ou delimitadores em linguagem PHP dentro de uma página web:

```
<?php
Seus comandos;
?>
```

ou

```
<?
Seus comandos;
?>
```

ou

```
<script language="php">
Seus comandos;
</script>
```

ou

```
<%
Seus comandos;
%>
```

Das quatro formas apresentadas, a segunda é a forma mais utilizada de declaração de *tags*, com a abreviação da declaração PHP. Com exceção das linhas de comando de controle, as linhas de código devem ser finalizadas com ponto e vírgula (;), conforme Lobo (2007).

Exemplificando

Exemplo de uma página web com declaração php:

```
<html>
  <head>
    <title>PHP Teste</title>
  </head>
  <body>
  <?
    echo "<p>Olá Mundo</p>";
  ?>
  </body>
</html>
```

Segundo Lobo (2007, p. 11), assim como em qualquer outra linguagem de desenvolvimento, as variáveis são definidas como:

> **As variáveis de sistema são endereços de memória representados por um identificador, que podemos até mesmo chamar de nome. Estes nomes permitem que identifiquemos a variável para guardar e depois recuperar um valor qualquer.**

Pesquise mais

Variáveis são apenas alguns dos recursos disponíveis em PHP. Convido você a consultar o livro: LOBO, E. J. R. **Criação de sites em PHP**. São Paulo: Digerati Books, 2007. Nele são explicados todos os principais recursos do PHP, do mais básico ao avançado, com uma linguagem simples e fácil, possuindo também exemplos práticos e completos de utilização desses recursos que proporcionam o desenvolvimento de sites dinâmicos em PHP.

Em linguagem de desenvolvimento PHP, as variáveis têm seu nome iniciado com o caractere cifrão ($) seguido de um texto que, por regra, deve ser uma letra ou o caractere *underline* (_). Outra característica do PHP é o fato de as variáveis serem *Case Sensitive*, ou seja, as letras maiúsculas são diferentes das letras minúsculas nas variáveis. Devido ao PHP utilizar algumas variáveis já definidas dentro da sua programação com letras maiúsculas, é melhor evitar criar variáveis com nomes em maiúsculas, segundo Barreto (2000).

Exemplificando

Exemplo de variáveis em PHP:
A variável $nomefaculdade é diferente da variável $NomeFaculdade

Na Figura 2.5 é apresentado um script modelo de declaração da linguagem de desenvolvimento PHP juntamente com a linguagem de marcação HTML, com a utilização de uma varável em PHP.

Figura 2.5 | Modelo de declaração de linguagem de desenvolvimento inserido dentro do HTML

```
<html>
<head>
    <title><? echo "Hello World"; ?></title>
</head>

<body>
<?
  $teste = "Hello World";
  echo $teste;
?>
</body>
</html>
```

Fonte: adaptada de Barreto (2000).

Neste modelo, aparecerá somente o texto "Hello World" na página web, e o comando *echo* é responsável por apresentar na tela o texto armazenado na variável *$teste,* exibindo o código-fonte da página apresentada na Figura 2.5. Na Figura 2.6, você poderá observar que somente é apresentada a codificação em HTML, a linguagem em PHP fica oculta para o usuário, devido a sua interpretação feita pelo servidor, afirma Niederauer (2011).

Figura 2.6 | Resultado do código-fonte após a execução da página web no servidor

```
<html>
<head>
    <title>Hello World</title>
</head>

<body>
Hello World</body>
</html>
```

Fonte: adaptada de Barreto (2000).

O uso do PHP poderia ser aplicado em:

- Criação de páginas para web, simples ou com acesso a banco de dados, como *sites* de faculdades, lojas virtuais e sites simples com formulários;
- Criação de sistemas de gerenciamento de empresas para internet;
- Criação de aplicações desktop para empresas, juntamente com o uso de extensões;
- Aplicações para celulares;
- E várias outras aplicações para serem utilizadas tanto em Linux como Windows.

JavaScript

A linguagem de marcação HTML tem a finalidade de estruturar uma página web, enquanto a personalização da página fica por conta das folhas de estilos. Segundo Silva (2010), as funcionalidades de uma página web são criadas pelas linguagens de desenvolvimento. Por exemplo, não é possível utilizar a linguagem de marcação HTML para realizar o envio de dados através de um formulário, e esse envio seria feito pelo JavaScript.

Para realizar o envio dos dados desse formulário é necessário o uso de uma linguagem de desenvolvimento que consiga manipular e processar essas informações, de acordo com Silva (2010).

Pesquise mais

Consulte o livro *JavasScript – Guia do Programador* (SILVA, M. S. **JavaScript Guia do Programador**. São Paulo: Novatec Editora, 2010) e aprenda mais sobre os fundamentos da linguagem, partindo dos conceitos mais básicos até a utilização das funcionalidades em uma página web, criando recursos mais avançados e interativos com JavaScript.

JavaScript é uma linguagem de desenvolvimento interpretada criada em 1995 por Brendan Eich da Netscape como uma extensão da linguagem HTML para o navegador Navigator 2.0, a fim de proporcionar maior interatividade a uma página web, conforme Rocha (1999).

Segundo Silva (2010), diferentemente da linguagem de desenvolvimento PHP, que é interpretada no servidor, JavaScript é uma linguagem de desenvolvimento criada para ser interpretada do lado do cliente, ou seja, depende do navegador do usuário para realizar a interpretação e funcionamento da linguagem. Esse recurso é possível, pois existe um interpretador JavaScript nos navegadores atuais.

A linguagem JavaScript é uma linguagem de desenvolvimento baseada em objetos, isto é, define as estruturas, propriedades dos navegadores e os elementos da página HTML como objetos, e são manipuladas por eventos, operadores e expressões criadas pelo usuário. Oferecendo recursos de interatividade ausentes no HTML, permitindo, assim, a criação de páginas dinâmicas e interativas, de acordo com Rocha (1999).

Por exemplo, caso uma página web possua um formulário de contato, podemos utilizar o JavaScript para realizar a validação dos

campos digitados pelo usuário e envio dos dados, recursos ausentes no HTML, que é utilizado unicamente como linguagem de marcação.

Com o crescente uso do JavaScript, a linguagem passou a ter um Padrão Web proposto pela ECMA (*European Computer Manu facturers Association*), uma associação europeia para padrões de sistemas de informações, segundo Rocha (1999).

Assimile

O uso de JavaScript é de extrema importância para as páginas web, porque permite um controle maior de várias funcionalidades e validações, gerando um controle de funcionamento maior à página web. Além de permitir a integração com as demais linguagens existentes como PHP, jQuery e Python.

A conformidade com os Padrões Web segue dois princípios básicos. O primeiro é escrever JavaScript não obstrutivo, isto é, o conteúdo da página deve ser funcional e presente. Deve-se utilizar para aumentar a usabilidade da página e utilizar scripts externos à página HTML. O segundo princípio é o da melhoria progressiva, onde esses princípios são interdependentes e se completam, cita Silva (2010).

É possível com JavaScript, segundo Rocha (1999), realizar diversas funcionalidades integradas à linguagem HTML, como:

- Gerar documentos em sua visualização baseado nas informações do usuário, como um relatório em PDF, com os dados que o usuário deseja;
- Realizar operações matemáticas e computacionais, como criar um script para exibir na tela a idade de um usuário baseado na sua data de nascimento;
- Abrir novas abas ou janelas do navegador, realizar trocar de informações entre essas janelas, manipular propriedades do navegador como o histórico de páginas visitadas e abrir janelas *pop-ups,* por exemplo;
- Manipular o conteúdo do documento, alterando propriedades da página, dos elementos HTML e validando toda a página como objetos, por exemplo, informando se o usuário e senha de uma página são válidos;
- Por meio de eventos, pode-se interagir com o usuário na página, como criar mensagem de boas-vindas para um usuário ao entrar na página web.

Para inserirmos o JavaScript dentro de uma página web, existem três formas, segundo Rocha (1999):

- Dentro de *tags* HTML com <SCRIPT> ... </SCRIPT>;
- Em um arquivo externo, importado pela página web;
- Dentro de descritores HTML sensíveis a eventos.

Formas de declarar a *tag* SCRIPT para uso da linguagem JavaScript:

```
<script>
   Código JavaScript
</script>
```

ou

```
<script type="text/javascript">
   Código JavaScript
</script>
```

Exemplificando

Figura 2.7 | Trecho do código em JavaScript que exibirá uma mensagem "Alô, Mundo! Cheguei" na tela do usuário.

```
<head>
...
<script type="text/javascript">
    alert("Alô Mundo!\nCheguei.");
</script>
</head>
...
```

Fonte: adaptada de Silva (2010).

Na Figura 2.7, em Exemplificando, temos um exemplo de uma linguagem JavaScript, em que aparecerá para o usuário uma caixa de alerta com a mensagem "Alô, Mundo! Cheguei", conforme Silva (2010).

Segundo Silva (2010), a linguagem JavaScript tem a capacidade de alterar, definir e controlar dinamicamente uma página web, com relação à sua formatação de cores, textos, links e, inclusive, alterar a posição de elementos HTML de uma página, podendo utilizar as folhas de estilos ligadas à página e até mesmo alterar suas funcionalidades.

Tabela 2.2 | Tabela comparativa entre HTML, PHP e JavaScript

HTML	PHP	JavaScript
É interpretado e executado pelo navegador.	É executado no servidor.	É interpretado e executado pelo navegador.
Cada *tag* HTML é parte de sua estrutura.	Trata o HTML como texto.	Cada *tag* HTML é tratada como objeto que pode possuir métodos, propriedades (ou atributos) e eventos.
Responsável pela estrutura da página.	Realiza funções junto ao servidor.	Realiza funções junto ao navegador.

Fonte: elaborada pelo autor.

Na Tabela 2.2, temos um comparativo entre HTML, PHP e JavaScript, com algumas diferenças entre essas linguagens e suas funcionalidades.

Reflita

Uma das inovações da versão HTML5 é o aumento da interatividade somente com o uso da linguagem. É possível a substituição do JavaScript pela versão do HTML, sabendo-se que o JavaScript é utilizado para criar dinamismo e interatividade nas páginas web? JavaScript pode ser deixado de lado na criação de páginas?

Sem medo de errar

As linguagens de desenvolvimento web, como PHP e JavaScript, adicionam, às páginas web, interatividade e dinamismo, resultando em um *site* mais atrativo ao usuário.

Para o *site* da papelaria do Sr. Maurício, precisamos criar mais de uma página web, contendo outras informações, como uma página sobre a empresa, uma página sobre os produtos vendidos e uma página com um formulário de contato para o usuário enviar uma mensagem ao Sr. Maurício. E como criar mais de uma página e manter o padrão entre elas? Precisamos criar todas as páginas com o mesmo padrão, mesmo tamanho de cabeçalho e rodapé.

Nosso objetivo é criar essa padronização entre as páginas e, para resolver essa questão, por meio das linguagens PHP e JavaScript.

No levantamento de novas funcionalidades para a criação de página, deve-se começar pesquisando essas versões, tanto do PHP

que será executado no servidor quanto ao JavaScript que será executado nos navegadores.

A criação desta padronização deve-se começar pela criação de uma estrutura de cabeçalho que deseja, como padrão, um rodapé e como será o corpo da página.

Com essa definição em mãos, aplicamos em arquivos independentes e, por meio da linguagem PHP, incluímos todos os itens dentro de um único arquivo de página web. Podemos utilizar o JavaScript através de um arquivo externo para manter as funcionalidades nas páginas web e, adicionando ele mesmo através da declaração incorporada, mantendo um padrão único entre as páginas.

Atenção

É importante lembrar que as linguagens PHP e JavaScript possuem variáveis em sua sintaxe e sua forma de declaração é parecida, não havendo a necessidade de informar qual o tipo da variável. Essa questão é de extrema importância para um funcionamento correto da página.

Avançando na prática

Validando informações

Descrição da situação-problema

A empresa ContrataRH é uma agência de empregos e sabe que a tecnologia é uma tendência crescente e deseja migrar a forma de trabalho manual para a internet. Hoje a empresa recebe os currículos impressos e os armazena, gerando desperdício e grande acúmulo de papel na empresa.

A necessidade da empresa é desenvolver uma página na web na qual os candidatos, que queiram cadastrar seus currículos, acessem a página e, por intermédio de um formulário, com os mesmos campos existentes no currículo, cadastrem suas informações, porém essas informações precisam ser validadas para que o usuário não envie informações incompletas ou errôneas por meio do formulário.

Você deverá criar as validações neste formulário, a fim de resolver este problema. Para isto, você precisa pesquisar e apresentar as formas de interação do usuário dessa página, como validação e formatação dos campos. Sabendo-se que JavaScript permite essa

interação do usuário com a página, é possível realizar esta validação e formatação com HTML5 e/ou PHP?

Você deverá apresentar à empresa um comparativo entre as linguagens, que permitem este tipo de interação, por meio de um exemplo de cada linguagem possível.

Lembre-se

É importante lembrar que as linguagens HTML5 e JavaScript são interpretadas pelo navegador, enquanto a linguagem PHP é executada e interpretada pelo servidor para resolução desta situação.

Resolução da situação-problema

Pesquisando as linguagens envolvidas nesta questão, para a resolução da situação apresentada, podemos utilizar as linguagens HTML5 e JavaScript para a validação, tanto utilizando somente HTML5 ou somente JavaScript, como também utilizando as duas linguagens integradas.

As validações e formatações podem ser criadas por meio de scripts em JavaScript, e podem ser configurados os campos de nome, telefone, celular, e-mail, CPF, data de nascimento, por exemplo, para que os candidatos cadastrem as informações padronizadas de acordo com as validações.

A linguagem PHP é executada e interpretada pelo servidor, portanto as validações e a formatação dos campos seriam realizadas pelo servidor e não no navegador do usuário.

Faça valer a pena

1. Analise as afirmações abaixo sobre linguagem de programação web.

I) Existem limitações quando se utiliza apenas a linguagem HTML para o desenvolvimento de páginas web, dificultando a interação usuário-página, por isso outras linguagens podem ser empregadas, como o PHP.

II) A linguagem PHP significa PHP *Hypertext Preprocessor*, possui uso público, sem custo; quando desenvolvida, era composta por um conjunto de scripts em linguagem C.

III) PHP é uma linguagem para inserção de scripts dentro de uma página, porém é no servidor em que está hospedada a página web que esses scripts são executados e interpretados.

IV) PHP não está envolvido com o layout da página e sim com seu funcionamento, diferentemente do CSS (*Cascading Style Sheets*).

Assinale a alternativa correta:

a) As alternativas I, II, III e IV estão corretas.

b) Apenas as alternativas I, III e IV estão corretas.

c) Apenas as alternativas II e III estão corretas.

d) Apenas as alternativas I e IV estão corretas.

e) Apenas as alternativas I, II e IV estão corretas.

2. Segundo Lobo (2007), a sintaxe do script em _____ é similar com a do _____ em suas declarações, sendo iniciadas e finalizadas com *tags*, porém as *tags* em _____ são diferentes em seus nomes de comando. Essas *tags,* ou delimitadores, podem ser colocadas em qualquer parte da página HTML para que o interpretador (servidor) identifique o que é linguagem _____ e execute o que está programado.

Assinale a alternativa que completa corretamente a afirmação acima:

a) JavaScript, CSS, CSS, HTML.

b) HTML, PHP, HTML, JavaScript.

c) PHP, HTML, PHP, PHP.

d) PHP, CSS, PHP, PHP.

e) HTML, PHP, CSS, PHP.

3. Sobre as afirmações abaixo:

I) Em linguagem de desenvolvimento PHP, as variáveis têm seu nome iniciado com o caractere cifrão ($) seguido de um texto, que por regra, deve ser uma letra ou o caractere *underline* (_).

II) Outra característica do PHP é o fato de as variáveis serem *Case Sensitive*, ou seja, as letras maiúsculas são diferentes das letras minúsculas nas variáveis.

III) As variáveis de sistema são endereços em scripts representados por um identificador, que podemos até mesmo chamar de nome.

IV) JavaScript é uma linguagem de desenvolvimento criada para ser interpretada do lado do cliente, ou seja, depende do navegador do usuário para realizar a interpretação e o funcionamento da linguagem.

Assinale a alternativa que apresenta as afirmativas corretas:

a) Apenas I e II são verdadeiras.

b) Apenas I e III são verdadeiras.

c) Apenas I, II e III são verdadeiras.

d) Apenas I, II e IV são verdadeiras.

e) Apenas II, III e IV são verdadeiras.

Referências

BARRETO, M. V. de S. **Curso de Aplicações Web em PHP.** Aracajú, 2000.

CARLOS H. P. S. **Criar um site responsivo com HTML5 e CSS3 – Parte 1/3**, 2014. Disponível em: <http://www.carloshps.com.br/blog/criar-site-responsivo-com-html5-e-css3-parte-1-de-3>. Acesso em: 11 dez. 2017.

CARVALHO, A. **HTML 4.1 & CSS 2.1: Manual Completo**. 2. ed. São Paulo: Book Express, 2004.

COSTA. G. **Conhecendo novas tags do html**. 2014. Disponível em <http://www.tutorialwebdesign.com.br/html-basico-parte-2>. Acesso em: 11 dez. 2017.

FREEMAN, E. **Use a cabeça: HTML com CSS e XHTML**. Rio de Janeiro: Alta Books, 2008.

HERTEL R. **Diferença entre HTML e HTML5.** 2017. Disponível em: <https://www.hostinger.com.br/tutoriais/diferenca-entre-html-e-html5/>. Acesso em: 10 dez. 2017.

LOBO, E. J. R. **Criação de sites em PHP.** São Paulo: Digerati Books, 2007.

LUTZ, M.; ASCHER, D. **Aprendendo Python.** 2. ed. Porto Alegre: Bookman, 2007.

MACEDO, M. S. **Construindo Sites Adotando Padrões WEB**.Rio de Janeiro: Ciência Moderna, 2004.

MAZZA, L. **HTML5 e CSS3 – Domine a web do futuro**. Sáo Paulo: Casa do código, 2012.

MENDES, D. R. **Rede de Computadores – Teoria e Prática.** São Paulo: Novatec Editora, 2007.

ROCHA, H. da. **Desenvolvendo Web Sites Interativos com JavaScript.** 1. ed. São Paulo, 1999.

SILVA, M. S. **Construindo Sites com CSS e (X) HTML**. São Paulo: Novatec Editora, 2007.

SILVA, M. S. **Criando Sites com HTML**. São Paulo: Novatec Editora, 2008.

SILVA, M. S. **HTML5 – A linguagem de marcação que revolucionou a web**. São Paulo: Novatec Editora, 2011.

SILVA, M. S. **JavaScript Guia do Programador.** São Paulo: Novatec Editora, 2010.

UPINSIDE Treinamentos. **Curso de HTML5 - Comparando HTML Com o HTML5!** (Aula 04). Disponível em: <https://www.youtube.com/watch?v=U22G0Dc8VIM>. Acesso em: 10 dez. 2017.

W3C. **World Wide Web Consortium**. Disponível em: <http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf>. Acesso em: 6 mar. 2016.

WELLING, L.; THOMSON, L. **PHP e Mysql Desenvolvimento Web.** 3. ed. Rio de Janeiro: Elsevier, 2005.

Tecnologias de desenvolvimento para dispositivos móveis

Convite ao estudo

Caro(a) estudante, bem-vindo(a) à terceira unidade no estudo de tecnologias para web e para dispositivos móveis. Até o momento, você explorou várias tecnologias usadas para o desenvolvimento web, desde ferramentas que são utilizadas no lado do cliente até as ferramentas usadas no lado do servidor.

Com o avanço tecnológico, dispositivos móveis, como *smartphones* e *tablets,* são cada vez mais utilizados, pois passaram a acessar a rede mundial de computadores de qualquer lugar e a qualquer momento. O avanço do *hardware* só é efetivo se o *software* também evoluir, portanto, novas tecnologias foram e continuam sendo desenvolvidas para atender essa nova demanda.

A partir desse momento, você conhecerá várias tecnologias associadas à evolução e ao desenvolvimento para dispositivos móveis. Tal conhecimento lhe abrirá uma nova gama de possibilidades profissionais. Veja um exemplo dessas possibilidades com a ideia de dois alunos.

Dois jovens empreendedores estão montando em sua cidade natal um Pub, do inglês *public house,* que designa um tipo de bar muito popular no Reino Unido, República da Irlanda e outros países de influência britânica, onde são servidas bebidas e comida rápida. Os sócios sempre gostaram muito de tecnologia e, em uma recente viagem à Irlanda, estiveram em um Pub que chamou muito a atenção deles por ser um local muito tecnológico, com *tablets* em suas mesas. Os frequentadores

tinham acesso a um cardápio totalmente digital e interativo, possibilitando obter informações de tudo o que é oferecido no Pub, bem como realizar os próprios pedidos diretamente do *tablet*. Tal tecnologia agilizou muito seus pedidos e propiciou um ambiente de muita descontração com os amigos, visto que não precisavam ficar a todo o momento sendo interrompidos pelos garçons ou chamá-los para realizar um pedido. Os empreendedores querem essa tecnologia para abrilhantar ainda mais o seu Pub, e para possibilitar essa implantação, você foi contratado para auxiliá-los a avaliar o universo *mobile*, as principais tecnologias para desenvolvimento de aplicações móveis, sistemas operacionais para dispositivos móveis e gerenciamento das informações, viabilizando a melhor solução para disponibilizar este serviço aos clientes. Em situações assim, em que podemos combinar as tecnologias móveis com diferentes tipos de negócios, quais tipos de dispositivos móveis devem ser utilizados? Como garantir que temos a melhor aplicação da TI móvel em nosso estabelecimento? Essas e várias outras questões deverão ser respondidas para que uma boa solução seja disponibilizada para realização de sua tarefa.

Para solucionar esse problema, nesta unidade você conhecerá a evolução do *hardware* móvel, verá o que define uma aplicação móvel, quais os sistemas operacionais mais utilizados nesse universo e, por fim, conhecerá as opções de desenvolvimento de aplicativo para esse universo.

E aí, pronto para este desafio? Então, vamos lá!

Seção 3.1

Definições e aplicações de TI móvel

Diálogo aberto

Está na hora de iniciar o trabalho de avaliação do universo *mobile* e as possibilidades de uso no Pub dos jovens empresários. Você deve expandir seu conhecimento acerca dos tipos de dispositivos móveis disponíveis, realizando um levantamento histórico e evolutivo, suas definições e aplicações da TI móvel, demonstrando aos sócios a melhor forma de utilizar esta tecnologia no Pub. Nesta primeira seção, você será apresentado ao universo *mobile,* avaliando os tipos de dispositivos móveis disponíveis no mercado, realizando um levantamento histórico e evolutivo e as definições e aplicações da TI móvel.

Perguntas como "Quais tipos de dispositivos móveis devem ser utilizados?", "Como as tecnologias móveis evoluíram?" e "Como garantir que temos a melhor aplicação da TI móvel para um determinado negócio?", entre outras, deverão ser respondidas para que você possa apresentar a arquitetura mais indicada para o caso. O resultado dos trabalhos desta etapa será um relatório com tudo aquilo que for definido, a saber: evolução das tecnologias móveis e quais dispositivos móveis que devem ser utilizados. Quantas coisas novas, não? Pois é, assim você vai aprofundando seus conhecimentos acerca dos dispositivos móveis e computação, preparando-se cada vez mais para o mercado de trabalho, que precisa muito de profissionais com estes conhecimentos. Mãos à obra e boa sorte!

Não pode faltar

Está na hora de voltarmos nossas atenções para o universo *mobile,* avaliando os tipos de dispositivos móveis disponíveis no mercado, realizando um levantamento histórico e evolutivo e as definições e aplicações da TI móvel. É por aí, portanto, que vamos começar.

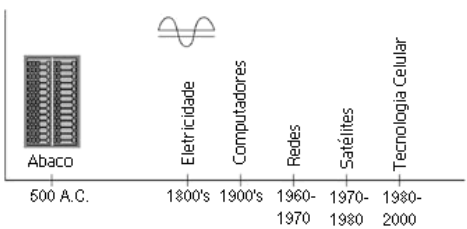
Evolução dos dispositivos e computação móvel

Segundo Yuan (2004), os primeiros adventos e sistemas de armazenamento móveis podem ser vistos a partir da era da eletrônica. Um sistema móvel computacional, como qualquer outro tipo de sistema, pode ser conectado a uma rede. A conectividade à rede, entretanto, não é um pré-requisito para ser um sistema móvel computacional.

Em 1960, os computadores puderam se comunicar a partir das redes e houve o início da comunicação *wireless* (tecnologia sem fio que permite a conexão entre dispositivos sem a necessidade de utilizar cabos de conexão entre eles) pelos militares. Em busca de qualidade de transmissão e recepção e da remoção de obstáculos entre as comunicações, em 1970 surgiram as comunicações via satélite e a sua comercialização da tecnologia. Através das comunicações via satélite houve um aumento significativo na qualidade e confiança dos serviços, porém o custo ainda era um grande obstáculo. Em 1990, a tecnologia celular tornou-se comercialmente viável, trazendo benefícios inigualáveis à transmissão de dados.

A Figura 3.1 ajuda a entender a evolução da Computação Móvel, tendo sua origem na máquina de calcular Ábaco, passando pela criação dos computadores, redes e satélites, até chegar à tecnologia celular, que permitiu o uso de Aplicações Móveis.

Figura 3.1 | Linha do tempo da computação móvel



Fonte: Muchow (2004, p. 105).

Assimile

De acordo com Mateus e Loureiro (2008), podemos destacar as seguintes evoluções nas tecnologias *mobile*:

- Em 1983: os americanos lançaram a primeira rede de celular.

- 1991: Teve início a tecnologia microcelular, e os EUA lançaram os padrões TDMA e CDMA.
- 1992: Início do GSM – Group Speciale Mobile, padrão Pan Europeu.
- 1994: Início dos sistemas CDPD (Celular Digital Packet Data), PCS (Personal Comunications Services), TDMA e CDMA.
- 1995: Projeto Iridium, modelo de satélite de baixa órbita.
- 1997: Aprovação do Wireless, padrão IEEE 802.11.
- 1999: Aprovação dos padrões IEEE 802.11b e 802.11a, em que utilizavam uma frequência de 2.4 e 5 GHz.
- 2000: Destaque para o Wi-fi
- 2003: Enfim, tecnologia 3G
- 2006: Utilização do padrão 802.11n.
- 2010: tecnologia 4G.

Personal Digital Assistant

De forma simplificada, Moll (2006) define o *PDA* como um computador de dimensões reduzidas, dotado de significativa capacidade computacional, o qual cumpre desde funções básicas até funções avançadas, com possibilidade de interconexão com um computador pessoal e uma rede sem fios, para prover acesso a correio eletrônico e Internet.

Os *PDA*s possuem quantidade de memória suficiente para prover uma quantidade significativa de funcionalidades e diversos *softwares* para várias áreas de interesse, porém, ainda encontram-se notavelmente distantes da eficácia de um computador de mesa ou portátil como o *notebook* ou *laptop*.

Os antecessores do *PDA* incluem o *Psion Organiser* e o *Sharp Wizard*. Estes primeiros aparelhos, feitos para serem computadores portáteis, foram lançados em meados dos anos 1980 e incluíam pequenos teclados para a entrada, uma tela pequena e recursos básicos, como um relógio com alarme, calendário, agenda telefônica e calculadora. O suporte para *softwares* especializados como jogos e planilhas também foi incluído. O *Psion Organiser* II, apresentado na Figura 3.2 - a, lançado em 1986, foi especialmente popular, e mais de meio milhão desses aparelhos chegou a ser vendido.

Em 1993, a empresa *Apple* apresentou o *Newton MessagePad*, visto na Figura 3.2 - b, com um preço de 700 dólares. Ele forneceu aos usuários um bloco de notas eletrônico, agenda de compromissos, calendário, agenda telefônica e aplicativos de arquivo de endereços. Algumas das inovações de *Newton* se tornaram recursos padrão dos *PDA's,* incluindo uma tela sensível com caneta especial, capacidade de reconhecimento de escrita, porta de infravermelho e uma baia de expansão. A *Apple* parou de fabricar o *Newton* em 1998.

O *PalmPilot* original, apresentado na Figura 3.2 - c, foi lançado em março de 1996 pela empresa *Palm Computing* (na época, de propriedade da empresa *U.S. Robotics*). Ele custava menos de 300 dólares, executava seu próprio sistema operacional *Palm OS*, cabia no bolso de uma camisa e se sincronizava com os computadores dos clientes. O *PalmPilot* funcionava com baterias, era fácil de usar e podia armazenar milhares de contatos, compromissos e anotações. Parte de seu tamanho reduzido se devia à ausência de um teclado. Os usuários usavam uma caneta especial e a linguagem *Graffiti* para introduzir os dados.

A *Microsoft*, por sua vez, desenvolveu diversos formatos de computação portátil, incluindo o *PenWindows* e os computadores *Tablet*. Em novembro de 1996, a *Microsoft* lançou o *Windows CE*, seu primeiro sistema operacional para aparelhos móveis. Diversos fabricantes, como a *HP, Compaq, Toshiba* e *Casio* adotaram-no, e passou a ser chamado de *Handheld PC,* Figura 3.2 - d, (literalmente, PC de mão): o primeiro competidor baseado em *Windows* para o *PalmPilot*. A partir desse ponto, temos grandes empresas fabricantes disputando o mercado e, para atrair o público para seu produto, passaram a oferecer cada vez mais aplicativos que poderiam ser úteis aos clientes, o que leva à necessidade de *hardwares* mais eficientes em termos de processamento e memória.

Figura 3.2 | Aparelhos PDA

a) b) c) d)

Fonte: adaptada de: <http://mycalcdb.free.fr/><https://msu.edu/~luckie/gallery/newton130.gif> <http://phonedb.net/img/palm_pilot_pro.jpg> <http://www.wowelectrons.com/pdas>. Acesso em: 13 dez. 2017.

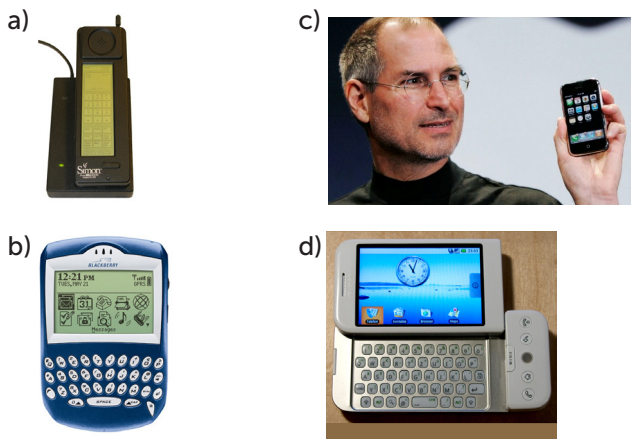
Smartphones e tablets

Segundo Morimoto (2009), os *smartphones* são considerados a união dos PDA e Palms. Claro, com possibilidades de conexões múltiplas, como 3G e Wi-Fi e diversos recursos de interatividade e comunicação.

O Blackberry 6210 (Figura 3.3 - b) foi o *smartphone* que entrou no gosto dos executivos em 2001, possuindo teclado QWERTY e entrada USB, sendo considerado um dos 100 principais *gadgets* (equipamento com propósito e função específica, chamados de *gadgets* dispositivos eletrônicos portáteis) de todos os tempos. O iPhone foi o primeiro *smartphone* da *Apple* (Figura 3.3 - c), lançado em 2007 por Steve Jobs, logo se tornando um grande sucesso. Ele foi apresentado como "um iPod, um celular e um comunicador móvel" em um único *gadget* e utiliza o mundialmente conhecido Sistema Operacional iOS. O HTC Dream (Figura 3.3 - d), lançado em 2007, foi o *smartphone* com o primeiro sistema operacional *Android*, então baseado no *Linux*. O *software* foi adquirido posteriormente pela *Google*, sendo considerado um dos principais sistemas operacionais para dispositivos móveis do mundo.

O Galaxy Nexus foi o primeiro celular da Samsung no mercado de *smartphones*, utilizando sistema Android 4.0, feito em parceria com a *Google*.

Figura 3.3 | Primeiros smartphones



Fonte: adaptada de: <http://www.mobilecollectors.net/%20phone/3320/IBM-Simon> <http://s.glbimg.com/po/tt/f/304x0/2013/01/28/blackberry-6210.jpg> <https://upload.wikimedia.org/wikipedia/commons/6/64/HTC_Dream_opened.jpg> <https://images.rapgenius.com/68d81b5c84469e8c5bdd00f43f576a69.1000x750x1.jpg>. Acesso em: 13 dez. 2017.

Reflita

Uma das frases mais marcantes na era tecnológica da década de 1990 foi dita por Steve Jobs no lançamento do iMac em 1998: "A lot of times, people don't know what they want until you show it to them." (WEEK, 1998). Traduzindo a famosa frase de Steve Jobs, "Na maioria das vezes, as pessoas não sabem o que querem, até mostrarmos a elas".

Uma década após essa frase histórica, como podemos relacioná-la ao mundo dos dispositivos móveis? Ela ainda pode ser considerada uma frase atual?

O próximo grande marco no mercado móvel foi realizado por Steve Jobs em 2010 com o lançamento do iPad. De acordo com Paulino e Oliveira (2013), o lançamento do iPad da Apple, em 2010, revolucionou a maneira de ver conteúdos na Internet. A introdução dos *tablets* tem como característica fundamental a interatividade, com interface sensível ao toque e altamente interativa. É um aparelho digital portátil, pessoal, em formato de prancheta, dotado de capacidades básicas de um computador. A portabilidade, acesso à Internet e suporte multimídia são alguns dos atributos que tornam esta máquina um genuíno representante da tecnologia móvel do século XXI, bem como um dos modelos de negócio que mais recebem apostas pelas multinacionais. Um dos primeiros dispositivos deste conceito é o GRIDPAD, lançado em 1989, sensível ao toque de uma caneta do tipo stylus11. A partir de então, dispositivos correlatos começaram a ser desenvolvidos e receberam o nome de *slate computers* e *pen computers*.

Após o lançamento do iPad, diversos fabricantes, tais como Sony, Motorola e Samsung, começaram a "corrida" pelos *tablets*. O que diferencia os demais *tablets* em relação ao iPad é o sistema operacional e os aplicativos desenvolvidos para cada tipo. Os iPads usam o sistema operacional iOS, criado pela Apple, sendo de fácil uso e estável. Os demais *tablets* citados fazem uso do sistema operacional Android e têm como ponto forte uma oferta maior de aplicativos, não estando limitados apenas àqueles permitidos pelo próprio desenvolvedor, como no caso da Apple.

Pesquise mais

O artigo "O uso dos tablets nas organizações: análise do impacto no ambiente de trabalho" é um convite a uma leitura acerca das mudanças impetradas pela tecnologia no ambiente das organizações, tendo como foco o uso dos *tablets* no meio empresarial. FONTES, Gustavo Pacheco; PASSANEZI, Paula Meyer Soares. O uso dos tablets nas organizações: análise do impacto no ambiente de trabalho. 2012. Disponível em: <http://www.revistaseletronicas.fmu.br/index.php/rms/article/view/23/pdf_1>. Acesso em: 13 dez. 2017.

Exemplificando

O uso de *tablets* favoreceram vários segmentos de diversos setores, entre eles podemos destacar os restaurantes, que fazem uso de tablets para realização de pedidos dos seus clientes, interagir com a cozinha, caixa, comissão dos garçons e até mesmo o estoque. Tudo muito simples e com alguns toques na tela *touch screen*.

Muito bem, estudamos a evolução e aplicações dos dispositivos de computação móvel, agora vamos em frente!

Sem medo de errar

Sabendo que os donos do empreendimento querem modernizar o seu Pub, permitindo que os frequentadores tenham acesso a um cardápio totalmente digital e interativo, realizando seus pedidos diretamente de um dispositivo móvel como, por exemplo, um *tablet*, você irá, neste momento, auxiliar os sócios na escolha do melhor dispositivo e tecnologia a ser aplicada.

Em recente reunião, os sócios comentaram que tinham uma limitação de orçamento, pois haviam destinado boa parte do capital para a locação do endereço de instalação do Pub, adequações na estrutura física e com parcerias e contratos. Está na hora de se reunir com os clientes, realizar o levantamento de requisitos dos equipamentos e responder às seguintes questões:

- Que tipo de formato de equipamentos você deseja utilizar no Pub (Tablets, Smartphones, PDAs etc.)?
- Há alguma restrição quanto ao uso do sistema operacional a ser utilizado (Android, iOS etc.)?
- Será necessário que os dispositivos se comuniquem por rede 3G/4G ou apenas pela rede Wi-Fi?
- Qual o investimento que deve ser aplicado em equipamentos?

Todas as informações levantadas e todas as decisões sobre os dispositivos e orçamento devem ser registradas em um relatório a ser entregue na próxima reunião. O resultado dos trabalhos desta etapa será um relatório com tudo aquilo que for definido a título de arquitetura do sistema a ser criado, a saber:

- Evolução das tecnologias móveis;
- Dispositivos móveis que devem ser utilizados;
- Melhor aplicação da TI móvel para o negócio, sendo apresentado em uma tabela com 3 possíveis fornecedores, o valor de cada item e o valor final.

Avançando na prática

Escolhendo um dispositivo móvel

Descrição da situação-problema

Um jovem estudante acaba de ingressar no curso de Sistemas de Informação, em uma renomada faculdade de sua cidade. Ele nunca foi da área de computação, mas acredita que fez uma boa escolha. E certamente fez. O novo aluno iniciou suas aulas de posse de um *notebook* que acabou de comprar para poder acompanhar as aulas e verificou que seus colegas de classe usavam nas aulas, além do *notebook*, *smartphones* e *tablets* das mais diversas marcas e modelos.

Como o aluno nunca foi da área de computação e também não era muito atento às novas tecnologias, ele o procurou para auxiliá-lo a verificar o que esses dispositivos agregariam em seu dia a dia. Dentre as necessidades, ele precisará de um aplicativo para calculadora científica nas aulas de cálculo, uma agenda para registrar suas provas e atividades, e acesso a e-mails e redes sociais para se comunicar com seus colegas de sala.

Assim, você deve ajudá-lo e mostrar as vantagens e aplicações desses dispositivos, verificando se é necessário ou não a aquisição desses dispositivos.

Resolução da situação-problema

Para auxiliá-lo a tomar sua decisão, você deve realizar as seguintes atividades:

- Levantar os principais *smartphones* e *tablets* disponíveis no mercado;
- Avaliar as necessidades para auxiliar na escolha do dispositivo;
- Verificar quais as principais funcionalidades dos dispositivos;
- Especificar a arquitetura de cada um dos dispositivos;
- Levantar a relação custo-benefício;
- De posse de todas as informações coletadas, gerar um relatório a ser apresentado ao colega.

Acesse aos principais sites de vendas e levante, para cada *tablet* escolhido, as seguintes informações:

- Funcionalidades;
- Custo-benefício;
- Vantagens e desvantagens.

Faça valer a pena

1. Na década de 1990, grandes inovações tecnológicas transformaram a sociedade em uma "sociedade da informação". Tal feito decorre de uma revolução tecnológica que surgiu no final da Segunda Guerra Mundial e, desde então, vários profissionais dedicaram e continuam se dedicando a essa evolução com objetivos a curto, médio e longo prazo.
Observe o texto a seguir e escolha a alternativa que completa mais adequadamente este texto:

Os _____ a partir das _____ puderam se _____.
Mais tarde houve o início da comunicação _____.

a) satélites – wireless – comunicar – redes
b) computadores – redes – comunicar – eletrônica
c) militares – redes – comunicar – wireless
d) satélites – redes – comunicar – eletrônica
e) computadores – redes – comunicar – wireless

2. A evolução da Computação Móvel teve sua origem na máquina de calcular Ábaco, passando pela criação dos computadores, redes e satélites até chegar à tecnologia para celulares, que permitiu o uso de Aplicações Móveis. Diante da nova demanda, os sistemas de transmissão também tiveram que evoluir.

Analise as afirmações:

I. O padrão IEEE 802.11b e 802.11a trabalha com 5 GHz e 2.4 GHz, respectivamente.
II. A WECA lançou o selo Wireless Facility (Wi-Fi).
III. A tecnologia 3G surge em 2003.
IV. A tecnologia 4GS está sendo implementada.
V. IEEE 802.11 é um dos padrões wireless.

Analisando as afirmações acima, são verdadeiras:

a) I e II.
b) II e III.
c) III e IV.
d) III e V.
e) Apenas V.

3. Um grande marco na tecnologia móvel aconteceu em 2010, quando Steve Jobs lançou o iPad, revolucionando a maneira de ver conteúdos na Internet. A introdução dos *tablets* tem como característica fundamental a interatividade, com interface sensível ao toque e altamente interativa.
Atualmente podemos classificar os *tablets* em duas categorias: os iPad e "demais *tablets*".

Considere as afirmações acerca dessas duas categorias:

I. Os *tablets* possuem menor oferta de aplicativos que os iPad.
II. O iPad possui maior capacidade de armazenamento que os demais *tablets*.
III. Uma das grandes diferenças entre ambos é o sistema que gerencia memória, acesso, *hardware* etc.

a) Apenas a alternativa I é verdadeira.
b) Apenas a alternativa II é verdadeira.
c) Apenas a alternativa III é verdadeira.
d) Apenas as alternativas I e II são verdadeiras.
e) Apenas as alternativas II e III são verdadeiras.

Seção 3.2

Sistemas operacionais e arquitetura de informações móveis

Diálogo aberto

> Sem software, um computador é basicamente um monte inútil de metal. Com software, um computador pode armazenar, processar e recuperar informações, tocar música e reproduzir vídeos, enviar e-mail, pesquisar a Internet e se envolver em muitas outras atividades valiosas (TANEMBAUM, 2008, p. 21).

Da mesma forma que um computador precisa de um sistema para ser útil, os dispositivos móveis também precisam, e esse é nosso assunto desta seção. Vamos entender como os dispositivos móveis são gerenciados por um *software* chamado Sistema Operacional, conhecendo a estrutura dos principais sistemas do mercado.

Dando continuidade ao seu projeto de implementação de dispositivos móveis no Pub, você demonstrará para os sócios as vantagens do desenvolvimento de um aplicativo para o sistema operacional Android, realizando um levantamento do histórico do Android e suas características, e também apresentar quais seriam as vantagens e desvantagens do sistema Android diante do seu concorrente iOS. Perguntas como "Quais *tablets* suportam Android?", "Quais requisitos para instalação e uso?", entre outras, deverão ser respondidas para que você possa apresentar a arquitetura mais indicada para o caso.

Percebeu que seu conhecimento acerca das tecnologias móveis está aumentando a cada atividade? Quantas coisas novas, não? Pois é, assim você aprofundará seus conhecimentos acerca dos dispositivos móveis e computação, preparando-se cada vez mais para o mercado de trabalho, que precisa muito de profissionais com estes conhecimentos.

Mãos à obra e boa sorte!

Não pode faltar

Definição de sistema operacional

Os *softwares* podem ser divididos em duas categorias: programa de sistema e programas aplicativos. Na primeira categoria está o *software* responsável por gerenciar os recursos do dispositivo, por exemplo, memórias, processador, dispositivos de entrada e saída, etc., e é exatamente por gerenciar todos os recursos que esse *software* recebe um nome especial: **sistema operacional (SO)**. Já na segunda categoria estão os aplicativos que trabalham sobre o sistema operacional, atendendo às necessidades do usuário. Portanto, é certo concluir que o aplicativo depende diretamente do sistema operacional, o que nos leva à dedução de que certos aplicativos podem funcionar em um determinado sistema operacional e não funcionar em outro. Vamos entender um pouco mais a arquitetura e a função de um sistema operacional.

De acordo com a Techtudo (2016), *Kernel* significa núcleo, funcionando como o "cérebro" de um sistema operacional. É parte fundamental nos sistemas operacionais, sendo a ligação entre processamentos dos dados e os programas. Ele é responsável por interligar o *hardware* e *software*, gerenciando e permitindo que aplicativos sejam executados e usem os recursos da máquina. O *Kernel* desempenha seu papel a partir do momento que a máquina é ligada, detectando o *hardware* e garantindo que todos os dispositivos e processos estejam funcionando. Quando o sistema operacional é carregado, o núcleo parte para o gerenciamento de processos, arquivos, memória, periféricos etc. O *Kernel* também escalona quais processos serão executados pelo processador.

Segundo VivaoLinux (2016), o *Kernel* é a parte interna do sistema operacional, fornecendo todos os serviços para outras partes do sistema, gerenciando *hardware* e distribuindo os recursos do sistema. Os *Kernels* podem ser dos seguintes tipos:

- Kernel Monolítico: aquele formado por um só bloco, contendo todos os módulos e subsistemas em um único executável binário, apresentando como característica permitir que funções essenciais sejam executadas através do *Kernel* Space. Possui melhor desempenho do que outras arquiteturas de *kernel*, mas apresenta manutenção de programação mais demorada, pois

deve ser recompilado e substituído por completo, se necessitar implementá-lo. Como exemplos desse tipo de *Kernel*: Linux, Unix, BSD e Ms-Dos;

- MicroKernel: é um tipo de *Kernel* que executa suas funcionalidades fora do Kernel Space, conversando com subsistemas que estão no User Space (parte do sistema de memória em que são executados os processos do usuário) através de mensagens. Ele provê serviços essenciais, como gerenciamento de memória, gerência de *threads* e comunicação entre processos e tarefas. São exemplos: OpenSolaris e MINIX;
- Kernel Híbrido: apresenta características dos dois tipos de *Kernel* vistos, possuindo apenas funções principais e se comunica com módulos chamados de "servidores", como o serviço de impressão. Caso um servidor falhe, o sistema operacional continua funcionando. São exemplos: Windows NT, XP, Vista, 7, 8 e Mac OS;
- Exokernel: usado para virtualização de sistemas, cria uma camada de *Kernel*, dando controle aos outros sistemas que são executados na camada superior. Tem conceito similar ao microkernel e é de difícil projeto, pois trata diferentes sistemas operacionais e deve permitir acesso a diferentes recursos de *hardware* de cada máquina. Exemplo: Xen.

De acordo com Gomes, Fernandes e Ferreira (2012), o sistema operacional acessa a memória e coordena a sua utilização pelos processos dos usuários, garantindo sua segurança de acesso. A maioria dos sistemas utiliza o conceito de memória virtual. O sistema deve garantir que cada processo tenha espaço na memória, fornecendo uma proteção a ele para não haver sobrescrição e utilização por outro, e possibilite que a aplicação não use mais memória que a existente fisicamente.

Memória virtual é um processo que usa a memória principal como cache para armazenamento secundário, permitindo assim o compartilhamento seguro e eficiente da memória entre os programas. A memória virtual consiste em recursos de *hardware* e *software* com três funções básicas:

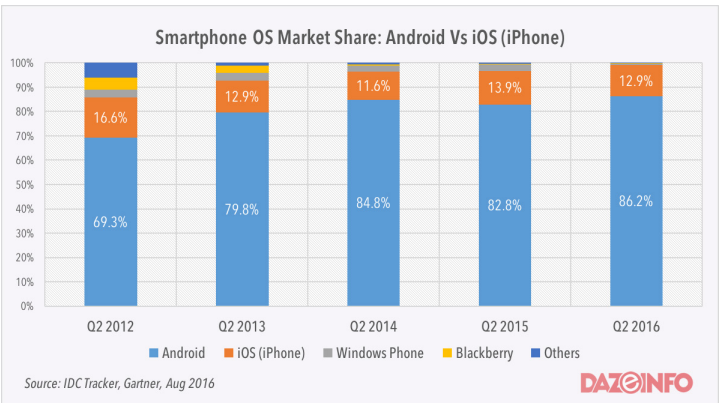
- Realocação: garante que cada processo tenha o seu próprio endereço, começando do zero;

- Proteção: não permite que um processo utilize um endereço de memória que não lhe pertença;
- Paginação: possibilita a uma aplicação utilizar mais memória do que fisicamente existe.

Sistemas operacionais para dispositivos móveis

Atualmente, dois sistemas operacionais se destacam no mercado para dispositivos móveis: o Android e iOS. Nossa observação pode ser confirmada a partir do estudo desenvolvido pela MOONTECHNOLABS (2017), cujo resultado está na Figura 3.4. Desde 2012, o sistema operacional Android é o mais amplamente utilizado e vem crescendo a cada ano, terminando o ano de 2016 com 86,2% do mercado. O segundo SO mais utilizado é o iOS da Apple, que terminou 2016 com 12,9% do mercado. Como você pode perceber, somando os dados dos dois grandes sistemas, estamos falando de 99,1% do mercado. Portanto, vamos agora aprender um pouco sobre os dois principais sistemas operacionais para dispositivos móveis, para que você possa escolher a melhor opção para o Pub.

Figura 3.4 | Comparativo entre Android e Apple



Fonte: <https://www.moontechnolabs.com/apple-vs-android-comparative-study-2017/>. Acesso em: 19 dez. 2017.

O sistema operacional Android

De acordo com Lecheta (2013), o Android aparece em um momento de necessidade de um ambiente poderoso e flexível para executar aplicativos em um celular. É um sistema operacional para dispositivos móveis, baseado no sistema operacional Linux, contando com aplicações já instaladas e um ambiente de desenvolvimento poderoso.

O Android é um sistema operacional desenvolvido pela Google e que está presente em vários aparelhos dos mais conhecidos e importantes fabricantes, como Sony, Motorola, Samsung e LG, sendo considerado uma das mais populares plataformas do mundo. O Android tem seu núcleo baseado no Linux, além de possuir um código aberto, com a possibilidade dos fabricantes de personalizar suas versões.

A data de lançamento do Android foi 2008, e desde então tem sido aperfeiçoado pela Google com muitas novidades, apresentando atualizações que ocorrem periodicamente, e ficou famoso por receber o nome de doces (exemplos: KitKat, Lollipop). A última versão lançada, 8.0, leva o nome de Oreo™, com a promessa de ser duas vezes mais rápido que seus antecessores (ANDROID, 2017). Um ponto interessante do Android é que cada fabricante tem uma versão personalizada, por esse motivo é diferente em cada celular. O fabricante cria então um Android personalizado, podendo alterar a interface e os variados aplicativos, chamado ROM. Todos os ROMs têm em comum o uso de aplicativos da Google, como Gmail, Google Now e Google Maps.

Arquitetura do sistema operacional Android

A possibilidade de variações no Android deve-se a sua arquitetura (Figura 3.5). Na camada inferior está seu *Kernel* (núcleo), no qual todos os *softwares* são comumente escritos em linguagem assembly e linguagem C. O gerenciamento de memória, em baixo nível no Android, é feito pelo Linux Kernel. A descrição da memória virtual é feita no Linux através de segmentação e paginação. A segmentação divide a memória em dois espaços, sendo eles o Kernel Space (Espaço Kernel) e o User Space (Espaço Usuário). A paginação evita que tenha blocos de memória contínuos para cada processo, o que sobrecarregaria a memória. A paginação divide a memória em pedaços de tamanho fixo (páginas), e segmentos de código são alocados e mapeados para essas páginas. Se a memória já preencheu todas as páginas possíveis, é realizada a substituição de páginas quando o processo em execução requisita que uma nova página seja alocada.

Acima do núcleo está o Android Runtime (Android em tempo de execução). Para entendermos essa camada, vamos fazer uma comparação com um computador desktop. Quando você abre um navegador, um processo é aberto no núcleo do sistema operacional,

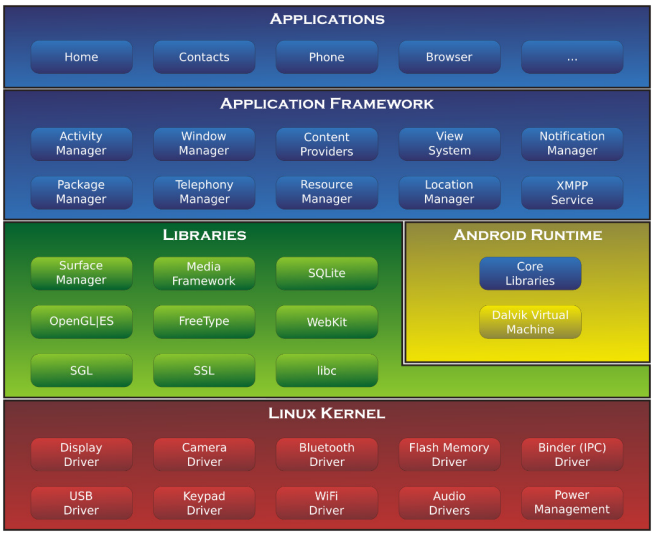
já no sistema Android isso é diferente: mesmo seu núcleo Linux sendo capaz de executar múltiplos processos simultâneos, quando um aplicativo é aberto um processo é iniciado em uma máquina virtual chamada *Dalvik*. Portanto, cada aplicativo executado em um dispositivo Android cria uma instância dentro da máquina virtual *Dalvik*. A execução de aplicativos em máquinas virtuais oferece uma série de vantagens. Em primeiro lugar, as aplicações estão essencialmente em caixas separadas (cada aplicação em uma instância), na medida em que não podem prejudicar o sistema operacional ou outras aplicações caso a aplicação dê algum problema. Além disso, aplicativos em execução em uma máquina virtual não podem acessar diretamente o *hardware* do dispositivo. Este nível de abstração torna a plataforma de aplicativos neutra, na medida em que nunca está ligada a nenhum *hardware* específico. A máquina virtual *Dalvik* foi desenvolvida pelo Google e depende do *kernel* Linux para funcionalidades de baixo nível (acesso ao *hardware*). Foi especificamente projetada para permitir que várias instâncias funcionem de forma eficiente dentro das restrições de recursos de um dispositivo móvel. Para que a máquina virtual possa executar os aplicativos, estes devem ser convertidos para o formato executável *Dalvik* (.dex), felizmente esse processo é feito internamente pelo sistema.

Além dos recursos já apresentados, o sistema Android possui uma vasta biblioteca de classes para que tudo funcione de forma integrada. Essa gama de possibilidades facilita o trabalho dos programadores no processo de desenvolvimento de aplicativos. As bibliotecas podem ser entendidas como um conjunto de instruções que orientam o dispositivo a tratar diferentes tipos de dados. Exemplo: a biblioteca do *framework* de mídia suporta reproduzir e gravar vários formatos de áudio, vídeo e imagem.

O *framework* de aplicação é a próxima camada e inclui todas as funções básicas do telefone, como alocações de recursos, aplicações do telefone etc. Os desenvolvedores possuem total acesso ao *framework* de aplicações do Android.

Ocupando o topo da pilha estão as aplicações em si, nas quais estão as funções básicas do dispositivo, como fazer chamadas telefônicas, acessar a Web ou a lista de contatos. É a camada que mais será utilizada pelo usuário.

Figura 3.5 | Arquitetura do SO Android



Fonte: <https://de.wikipedia.org/wiki/Android_(Betriebssystem)>. Acesso em 19 dez. 2017.

O Android não está presente apenas em celulares, ele também aparece em *tablets*, TVs, computadores, relógios, sendo um dos sistemas operacionais mais populares, ganhando público desde em aparelhos mais simples até os modelos top de linha.

Em 2007 foi criado a *Open Handset Alliance* (OHA), uma aliança de várias empresas com o objetivo de criar padrão aberto para telefonia móvel. Fazem parte da OHA empresas como a Google, HTC, Dell, Samsung, LG, Motorola, entre outras.

Assimile

Os aplicativos, em um dispositivo móvel que utiliza o sistema operacional Android, executam seus processos em uma máquina virtual chamada Dalvik. Essa tecnologia foi projetada especialmente para a plataforma Android a fim de otimizar a utilização de memória.

O sistema operacional iOS

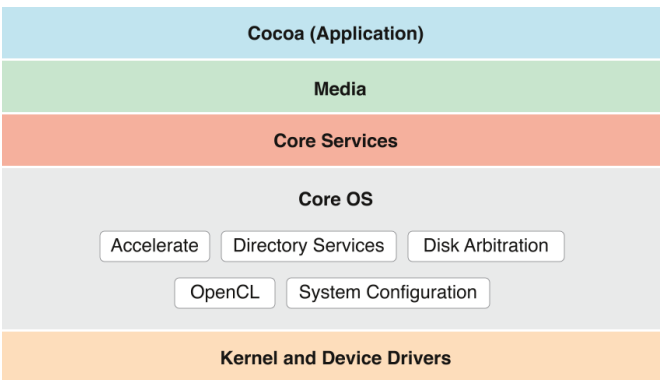
O primeiro smartphone da Apple foi lançado em 2007 e, como já sabemos, um *hardware* sem *software* não é nada útil, portanto, o sucesso que a Apple alcançou não é atribuído somente ao *hardware*, mas também ao seu sistema operacional, que primeiro recebeu o nome de iPhone OS, mas foi atualizado mais tarde para **iOS**.

O iOS foi originalmente desenvolvido para ser executado no iPhone, porém seu sucesso foi tamanho que, desde então, vem sendo expandido para suportar o iPod, o iPad, o Apple Watch e a Apple TV (COSTELLO, 2017).

O objetivo principal da Apple ao desenvolver o iOS foi criar um sistema que tirasse o máximo proveito do *hardware*, proporcionando ao usuário satisfação em termos de velocidade e operabilidade. O iOS foi pioneiro em vários recursos como, por exemplo, o Game Center e a assistente pessoal Siri. Segundo o site oficial da Apple, o iOS 11 (seu mais novo sistema operacional) traz novas possibilidades para realidade aumentada em jogos e aplicativos.

Ao contrário do Android, o iOS é um sistema fechado e nenhuma outra empresa pode utilizá-lo. Sua arquitetura é dividida em cinco camadas (Figura 3.6), sendo que na camada inferior estão os componentes de baixo nível (mais próximo da linguagem de máquina) como *drivers* de dispositivos, arquivos de segurança etc.; na camada Core OS estão algoritmos que aceleram operações complexas, controle de acesso a dados, controle do *hardware* permitindo o processamento paralelo nas centrais de processamento (GPU e CPU), dentre outras características. A camada Core Services possui a implementação necessária para serviços como integração de e-mails, agendas etc.; na camada Media são definidas as opções de mídia que o sistema suporta; a camada de aplicação Cocoa é a principal responsável pela aparência de aplicativos e sua capacidade de resposta às ações dos usuários.

Figura 3.6 | Arquitetura do sistema iOS



Fonte: <https://developer.apple.com/library/content/documentation/MacOSX/Conceptual/OSX_Technology_Overview/CoreOSLayer/CoreOSLayer.html>. Acesso em: 19 dez. 2017.

Pesquise mais

O site da Apple possui uma vasta documentação para que desenvolvedores de *software* possam consultar e compreender a arquitetura. Acesse <https://developer.apple.com/library/content/documentation/MacOSX/Conceptual/OSX_Technology_Overview/About/About.html>. Acesso em: 19 dez. 2017, e aprenda mais sobre a arquitetura do sistema iOS.

Comparação Android *x* iOS

Android é baseado em Linux, que é um sistema *open source*, então só esse fato já traz uma série de diferenças entre este e o iOS.

Exemplificando

Segundo a Tudocelular (2016), pesquisadores de uma empresa de segurança chamada Zimperium encontraram uma das piores vulnerabilidades da história do Android em 2015, que atingia os dispositivos quando recebiam uma mensagem MMS, não sendo necessário abrir a mensagem. Essa falha atingiu 95% dos *smartphones* da versão 2.2 a 5.1 e se deve ao processamento de uma biblioteca de arquivos de mídia chamada Stagefright. Os *hackers* invasores enviavam a mensagem, conseguiam acesso ao *smartphone*, podendo roubar arquivos, ler e-mails e obter credenciais.

Segundo NIELD (2016), a Google permite que seus usuários e desenvolvedores tenham mais liberdade na maneira de trabalhar com seu sistema. Veja no Quadro 1 algumas diferenças entre esses dois sistemas.

Quadro 3.1| Android x iOS

	Android	iOS
Desenvolvedor	Google, Open Handset Alliance	Apple Inc.
Customização	Permite alta customização	Customização restrita
Versão inicial	23 de setembro de 2008	29 de julho de 2007
Código fonte	Código aberto	Fechado
Sistema base	Linux	OS X, UNIX
Serviços em nuvem	Integração nativa com o armazenamento em nuvem do Google.	Native integration with iCloud
Segurança	Médio grau de confiança	Alto grau de confiança
Lojas de aplicativos alternativos	Várias lojas alternativas de aplicativos diferentes da Google Play Store oficial	Não permite outras lojas.

Fonte: Adaptado de: <https://www.diffen.com/difference/Android_vs_iOS>. Acesso em: 19 dez. 2017.

Reflita

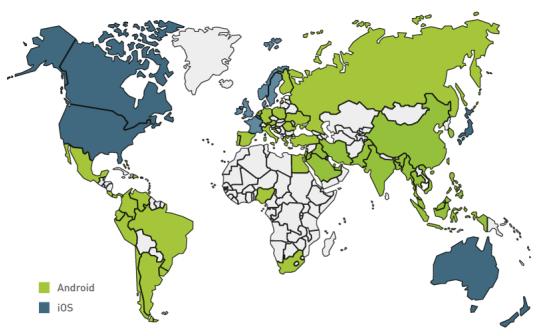
Na Figura 3.4 é apresentado o resultado de uma recente pesquisa comparativa da quantidade de usuários que utilizam cada sistema operacional disponível para dispositivos móveis. Desde seu lançamento, o Android é o mais utilizado. A partir desses dados, podemos concluir que criar aplicativos para Android é mais vantajoso que criar para iOS?

Sem medo de errar

Após entender o que é um sistema operacional e conhecer as duas principais opções para dispositivos móveis, chegou o momento de você apresentar para os sócios as vantagens do desenvolvimento de um aplicativo para o sistema operacional Android.

Comece sua apresentação mostrando a evolução histórica do Android, em seguida mostre as características desse sistema e quais seriam as vantagens e desvantagens do sistema Android diante do seu concorrente iOS. Insira em sua apresentação dados como na Figura 3.7, que mostra a quantidade de usuários de cada sistema dividido por regiões ao redor do mundo. Procure outras fontes para que suas informações sejam reforçadas. Para finalizar, prepare uma tabela com os possíveis dispositivos que poderiam ser usados com o sistema Android. Nessa tabela deve constar o nome do dispositivo, duas referências para compra com valor, versão do sistema disponível para o dispositivo e uma estimativa de investimento que os sócios deverão fazer.

Figura 3.7 | Uso de sistema operacional por região



Fonte: <https://www.moontechnolabs.com/apple-vs-android-comparative-study-2017/>. Acesso em: 19 dez. 2017.

Avançando na prática

Questão de segurança

Descrição da situação-problema

O responsável pela financeira de cartões de crédito está necessitando de um sistema para viabilizar a seus clientes a opção de pagamentos via crédito e débito através do *smartphone*. O sistema operacional que será utilizado é o Android, e uma as maiores preocupações é a questão de segurança. Você foi procurado para prestar uma consultoria e verificar como está a questão de segurança das distribuições Android e suas vulnerabilidades, buscando, assim, determinar a distribuição que deve ser utilizada.

Resolução da situação-problema

Para realizar seu trabalho de consultoria, prepare um relatório com dados em uma tabela com: (i) Levantamento das distribuições Android; (ii) Levantamento dos aspectos de segurança das distribuições; (iii) Principais vulnerabilidades. Essas informações devem ser obtidas através de fontes oficiais, como o endereço oficial da página do fabricante, bem como artigos sobre tecnologias.

Exemplo:

Quadro 3.2 | Distribuições Linux

Distribuição Android	Aspectos de Segurança	Vulnerabilidades
Android 5.0 - Lollipop	Versão não possibilita a atualização do sistema operacional. O aparelho desliga e reinicia sem ter instalado a atualização.	Reinicializações constantes de aplicativos,apresentando problema de gerenciamento de memória ou perda de memória.

Fonte: elaborado pelo autor.

De posse de todas as informações coletadas, gerar um relatório a ser apresentado ao seu cliente.

Faça valer a pena

1. O sistema operacional é o principal em qualquer sistema computacional, pois ele gerencia todos os recursos disponíveis. Cada SO possui um *kernel* (núcleo) que desempenha seu papel a partir do momento em que a máquina é ligada, detectando o *hardware* e garantindo que todos os dispositivos e processos estão funcionando.

Observe o texto a seguir e em seguida escolha a alternativa que completa mais adequadamente este texto:

Kernel significa _____, sendo o _____ de nosso sistema. É parte fundamental nos _____, sendo a ligação entre _____ dos dados e os _____.

a) núcleo – cérebro – sistemas operacionais – processamento – programas
b) cérebro – núcleo – sistemas – programas – processamentos
c) sistema operacional – cérebro – núcleo – processamento – programas
d) núcleo – sistema operacional – sistemas – manipulação dos dados – programas
e) cérebro – núcleo – sistemas – processamento – programas

2. Considere as seguintes afirmações sobre o sistema Android.

I. Para os fabricantes de celulares, a grande vantagem em utilizar tal sistema é o fato de ele ser baseado em Linux e possuir código aberto.
II. A finalidade do grupo Open Handset Alliance (OHA) é padronizar o sistema Android.
III. O sistema Android usa a máquina virtual JVM (Java Virtual Machine)

Assinale a alternativa correta.

a) Apenas a alternativa I está correta.
b) Apenas a alternativa II está correta.
c) Apenas as alternativas I e II estão corretas.
d) Apenas as alternativas II e III estão corretas.
e) Apenas a alternativa III está correta.

3. O sistema operacional iOS foi lançado em 2007 inicialmente somente para iPhone. Devido ao grande sucesso, o sistema passou a compor os demais dispositivos da família Apple e hoje é possível encontrá-lo em dispositivos como relógio e TV.

O sistema iOS é divido em 5 camadas. Escolha a opção que contém as funções que são atribuídas à camada Core OS.

a) *Softwares* que definem as opções de mídia que o dispositivo suporta.
b) *Softwares* que definem a aparência de aplicativos e a interação com o usuário.
c) *Softwares* que fazem a "ligação" entre o *hardware* e as demais camadas.
d) *Softwares* que controlam as centrais de processamento.
e) *Softwares* que controlam a integração entre serviços.

Seção 3.3

Desenvolvimento de aplicações móveis

Diálogo aberto

Caro(a) estudante, bem-vindo(a) à última seção da unidade destinada a familiarizá-lo com o mundo dos dispositivos móveis. Você já conheceu a história e a evolução desses dispositivos na primeira seção; na segunda você pôde conhecer um pouco sobre a arquitetura e o funcionamento, entendendo o que é um sistema operacional e quais as principais opções para esse universo. Nesta terceira seção você conhecerá quais são as principais tecnologias usadas para o desenvolvimento de aplicações para os dispositivos móveis.

Depois do estudo desta seção, você poderá completar mais uma etapa do seu trabalho para o Pub.

Você criará uma apresentação em *slides* para os sócios, mostrando as possibilidades de desenvolvimento dos aplicativos móveis. Realize um levantamento das principais tecnologias para desenvolvimento de aplicações para TI móvel. Perguntas como "O que são aplicações móveis?", "Como desenvolver aplicações móveis?", "Quais ferramentas e linguagens mais utilizadas", entre outras, deverão ser respondidas para que você possa apresentar a arquitetura mais indicada para o caso.

Para concluir sua tarefa, apresentaremos nesta seção uma introdução sobre aplicações para dispositivos móveis, discutiremos as diferenças entre as tecnologias para desenvolvimento nativa e híbrida, e ainda entenderemos o que significa o termo *cross*-plataforma dentro desse contexto.

Então, vamos dar início a essa nova aventura que lhe abrirá uma nova gama de possibilidades profissionais.

Bons estudos!

Não pode faltar

De acordo com a Infobase Interativa (2015), o elemento essencial e o que dá utilidade aos *smartphones* e *tablets* são os aplicativos. Os aplicativos (ou apps) são o motivo principal de tornar esses dispositivos tão úteis e adaptados às nossas rotinas. São os apps presentes no aparelho móvel que fazem com que ele seja uma ferramenta multifuncional, possibilitando desde a realização de tarefas simples, como ouvir uma música, até detectar raios cósmicos presentes em atividades de partículas subatômicas (app DECO, produzido pelo Centro de Astrofísica de Partículas IceCube, Universidade de Wisconsin).

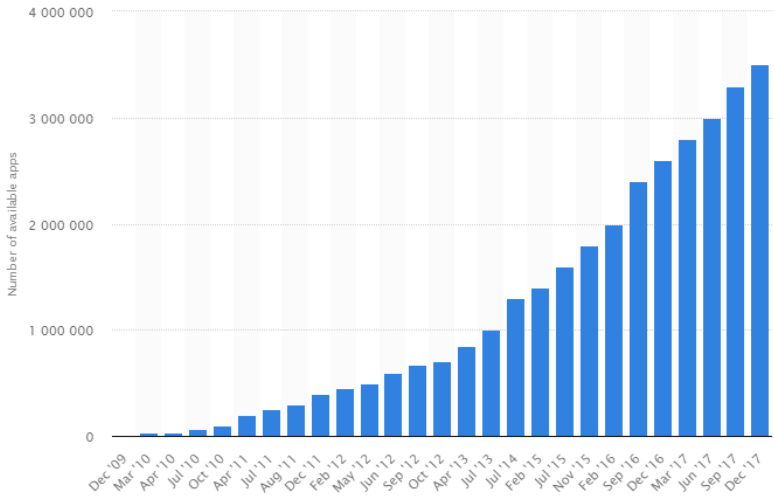
Exemplificando

Segundo a Infobase Interativa (2015), algumas aplicações móveis merecem destaque:

- App Google Maps: detalha cerca de 200 países, possuindo navegação GPS orientada por voz e rotas de transporte público em mais de 800 cidades. Disponível para Android, Windows Mobile e iOS;
- Instagram: aplicativo gratuito para gravação e postagem de vídeos, produção e divulgação fotográfica, compatível com inúmeras redes sociais;
- Easy Taxi: ferramenta de auxílio a passageiros e taxistas em todo o Brasil, fornecendo informações sobre a quantidade de táxis disponíveis nas proximidades e, para o taxista, a região exata em que se encontra o passageiro que solicitou o serviço;
- WhatsApp Messenger: aplicativo multiplataforma que permite envio e recebimento de mensagens através de um aparelho celular sem nenhum custo adicional por SMS.

Esses são apenas alguns exemplos de aplicativos, pois, segundo o site <www.statista.com>, em dezembro de 2017 a Google Play Store contava com um banco de aplicativos de cerca de 3,5 milhões (Figura 3.8).

Figura 3.8 | Quantidade de aplicativos na Google Play Store de 2009 a 2017



Fonte: Statista (2018, p. 1).

Os dados disponíveis da maior concorrente da Google não são tão atualizados. Segundo a mesma fonte, em janeiro de 2017 a App Store contava com um banco de cerca de 2,2 milhões de aplicativos (STATISTA, 2018, p. 1). Se compararmos os dados de ambas as empresas no mesmo período (janeiro de 2017), a quantidade de apps é semelhante, o que mostra o equilíbrio na disputa de mercado pelas duas maiores empresas do ramo.

Somando os aplicativos disponíveis em ambas as lojas, no mês de janeiro de 2017, chegamos à quantia de mais de 4 milhões de aplicativos. E agora fazemos a grande pergunta: como esses aplicativos são desenvolvidos?

Assimile

O desenvolvimento de aplicativos para dispositivos móveis são divididos em duas categorias: **desenvolvimento nativo** e **desenvolvimento híbrido**. Além dessas duas categorias, existe o termo cross-plataforma, utilizado para identificar um certo tipo de tecnologia de desenvolvimento. Vamos entender cada um desses termos.

Desenvolvimento nativo

Embora seja possível fazer maravilhas com um computador, ele é capaz de entender apenas dois dígitos: 0 e 1. Mas, então, como é possível usar aplicativos, navegar na Internet, dentre tantas outras atividades? Para que seja possível usufruir dos recursos computacionais, é necessário "traduzir" tudo que se faz para a linguagem que a máquina entende, e esse processo é o que chamamos de compilação. Segundo Delgado e Ribeiro (2017, p. 396), "a principal função de um compilador é converter uma sequência de caracteres, que representa instruções de um programa, de acordo com as regras de uma determinada linguagem, em código de máquina (instruções em binário que o hardware sabe executar diretamente)". Portanto, quando um *software* é criado em uma determinada linguagem de programação, para que esse programa possa funcionar em um dispositivo eletrônico, seja computador, celular, *tablet*, etc., ele passa pelo processo de compilação, convertendo todos os comandos em sequências de 0 e 1 para que os dispositivos possam executar.

Vale ressaltar que o processo de compilação muda de acordo com o paradigma de programação (procedural, orientado a objetos etc.) e com a linguagem de programação utilizada (C, Java, C++, C# etc.). Cada fornecedor cria seu próprio mecanismo de compilação a fim de garantir a melhor performance de seu produto. Segundo Delgado e Ribeiro (2017, p. 396):

> **O compilador é, assim, um elemento fundamental no desempenho de um computador. O mesmo programa compilado por dois compiladores diferentes e executado no mesmo computador pode apresentar tempos de execução significativamente distintos, dependendo das otimizações feitas e do grau de conhecimento que o compilador tem da arquitetura em que o programa vai ser executado.**

Toda a discussão apresentada sobre a compilação de um programa nos leva direto à essência do desenvolvimento nativo para dispositivos móveis. Portanto, desenvolvimento nativo é quando um aplicativo é escrito em uma determinada linguagem de programação e o código fonte do programa é compilado para uma determinada plataforma (*software* + *hardware*) (DUA, 2017).

Esse processo também é conhecido como *build* (construir), pois a partir de um código fonte é construído um aplicativo para uma determinada plataforma.

Vantagens de construir um aplicativo nativo:

Como o aplicativo é construído especificamente para uma determinada plataforma, a comunicação entre *hardware* e *software* apresenta um melhor desempenho e a probabilidade de falhas é menor.

Segundo DUA (2017), as vantagens de uma aplicação nativa são:

(i) Os aplicativos nativos são muito rápidos e receptivos porque são criados para essa plataforma específica. Nesse caso, o código de máquina gerado pelo *build* é específico para o *hardware* e para o sistema operacional do dispositivo, consequentemente a comunicação é direta, não sendo necessária nenhuma extensão (*plug-in*).

(ii) Apresentam melhor desempenho. Como *hardware* e *software* trabalham de maneira dedicada, a ocorrência de falhas é menor.

(iii) São distribuídos em lojas oficiais de aplicativos. Somente programadores cadastrados nas lojas oficiais conseguem disponibilizar seus aplicativos, e quando um usuário instala um aplicativo a partir de uma dessas lojas, ele não precisa se preocupar com a presença de vírus.

(iv) São mais interativos, intuitivos e correm muito mais suave em termos de entrada e saída do usuário.

(v) Permite que os desenvolvedores acessem o conjunto de recursos completo já otimizado para aquele *hardware*;

(vi) Em geral apresentam uma melhor experiência do usuário, pois o fluxo é mais natural devido à existência de padrões específicos de interface de usuário (user interface – UI) para cada plataforma.

Mesmo com tantas vantagens, o desenvolvimento nativo também possui desvantagens, sendo uma delas consequência da própria definição: se o código fonte é compilado para uma determinada plataforma, isso significa que não irá funcionar em outra plataforma, e essa é a grande questão do desenvolvimento nativo.

Cada plataforma possui um conjunto de características específicas, bem como ferramentas próprias para o desenvolvimento. Portanto, se uma empresa deseja criar uma aplicação nativa que funcione em duas plataformas diferentes, ela terá que contar com duas equipes de profissionais distintas, o que aumentará significativamente o custo do projeto (DUA, 2017).

Já sabemos que os dois grandes sistemas operacionais no mercado de dispositivos móveis são o Android, da Google, e o iOS, da Apple, portanto, vejamos as principais características e ferramentas disponíveis para a construção de aplicativos nativos para esses sistemas.

Android:

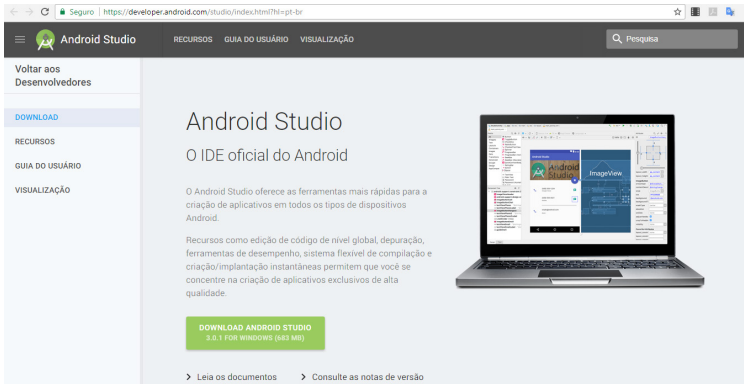
Segundo Sinicki (2017), o primeiro passo para um desenvolvedor que almeja criar aplicativos para o sistema da Google é aprender uma das linguagens de programação disponíveis para essa plataforma. Dentre as opções estão:

- Java – Atualmente é a linguagem de programação oficial para o desenvolvimento nativo para Android. É uma linguagem poderosa que pertence ao grupo do paradigma orientado a objetos e requer um tempo para seu aprendizado.
- Kotlin – Embora seja uma linguagem de programação nova, é considerada a segunda linguagem de programação oficial para Android (SINICKI, 2017).
- C/C++ – Embora seja uma opção, é necessário adicionar o Android Native Development Kit (NDK), que consiste em um conjunto de ferramentas que permite usar código C e C++ em aplicativos Android.
- C# – Linguagem de programação criada pela Microsoft que tem como diferencial funcionar em ferramentas que permitem a compilação em mais de uma plataforma. Voltaremos a falar dessa opção quando abordarmos o desenvolvimento cross-plataforma nesta seção.

Após escolher a linguagem de programação a ser utilizada, o próximo passo consiste em escolher uma ferramenta de desenvolvimento, mais conhecida por IDE (*Integrated Development Environment*). Uma IDE é um ambiente constituído por um conjunto de ferramentas que funcionam de maneira integrada para a construção de *softwares*.

Uma das IDE mais utilizadas para programação em Java é o Eclipse, porém o uso dessa ferramenta para o desenvolvimento de aplicativos Android foi descontinuada pela Google, que criou o Android Studio e o tornou o IDE oficial (Figura 3.9).

Figura 3.9 | Site oficial do Android Studio



Fonte: <https://developer.android.com/studio/index.html?hl=pt-br>. Acesso em: 16 jan. 2018.

Instalando o Android Studio, todas as ferramentas necessárias para o desenvolvimento Android ficam à disposição do programador. Embora seja a ferramenta oficial, não é a única. Ainda nesta seção falaremos de outra opção.

iOS:

O desenvolvedor que deseja criar aplicativos nativos para o sistema da Apple deve seguir as mesmas etapas que o desenvolver Android: escolha da linguagem e escolha da IDE. Entretanto, as opções são mais limitadas, pois existem apenas duas linguagens de programação e apenas uma IDE para o desenvolvimento nativo iOS.

A primeira linguagem usada para criar *softwares* para OS X e iOS foi criada no início da década de 1980 como uma extensão da linguagem C, trata-se da linguagem Objective-C (ZHOU, 2010). Com o advento das linguagens de programação, a Apple dedicou-se a criar uma linguagem de programação mais dinâmica e, em 2014, na Worldwide Developers Conference (WWDC), a Apple apresentou sua nova linguagem, à qual deu o nome de Swift (CLEVERISM, 2017).

Tratando-se da questão de ambientes de desenvolvimento para os dispositivos da Apple, a única IDE disponível é o Xcode, e para sua execução é necessário um computador da marca.

Desenvolvimento híbrido

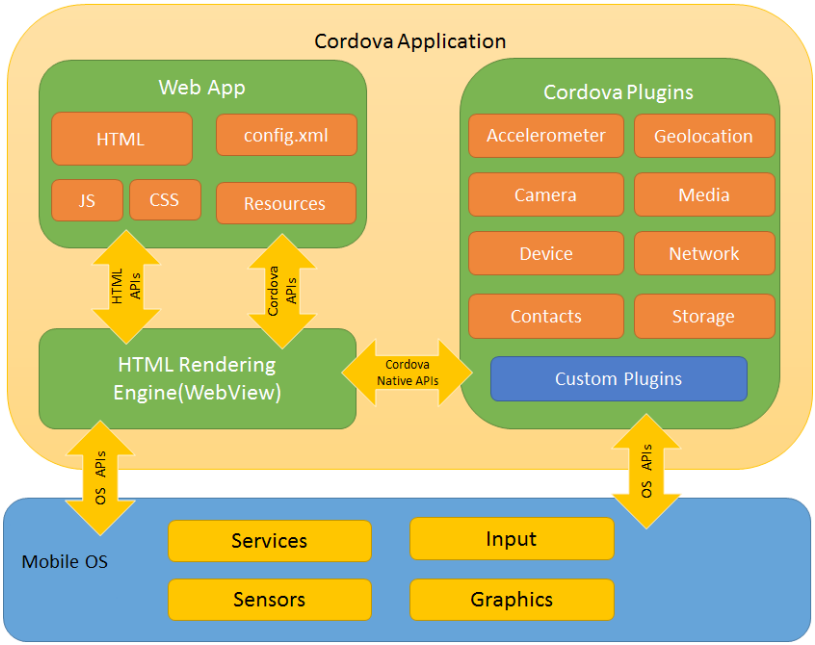
Com as informações vistas até o momento, fica evidente que para desenvolver um aplicativo que funcione nas duas principais plataformas do mercado são necessárias duas frentes de trabalho, cada uma atuando com uma linguagem de programação e em uma IDE diferente. Certamente essa situação não agrada um cliente que deseja "encomendar" um aplicativo. Uma das alternativas para contornar esse problema é recorrer ao desenvolvimento híbrido!

Um aplicativo híbrido é uma combinação de um aplicativo nativo com um aplicativo web (DUA, 2017). Desde o surgimento da Internet, muitas e muitas páginas de Internet, são criadas diariamente, então, não seria interessante (e extremamente produtivo) se existisse uma maneira de transformar a página pronta em um aplicativo para dispositivo móvel? Pois essa solução já existe e faz parte do desenvolvimento híbrido.

Um aplicativo híbrido é composto por duas partes: (i) uma parte contendo o código construído usando HTML, CSS e JavaScript. Como o desenvolvimento híbrido é caracterizado pela combinação de um aplicativo *web* com um aplicativo nativo, as tecnologias usadas para construir uma página *web* são utilizadas para construir a estrutura do aplicativo; (ii) a segunda parte consiste em um núcleo (*shell*) que irá trazer as funcionalidades nativas e fará com que o código do aplicativo execute em uma *Webview*.

A tecnologia envolvida na criação de uma página *web* você já teve a oportunidade de estudar neste livro, portanto, vamos explicar a segunda parte que compõe um aplicativo híbrido. A comunicação entre a página *web* e os recursos nativos de um dispositivo móvel é feita através de um outro *software* que fará essa ponte entre os recursos. Nesse processo, a página *web* é encapsulada e transformada em um aplicativo, sendo possível disponibilizá-la em qualquer loja de aplicativos. O projeto Apache Cordova é um *framework* capaz de realizar essa tarefa de encapsulamento da página *web*. Na Figura 3.11 está ilustrado o encapsulamento feito pelo Cordova: existe um aplicativo *web* (página *web*) feito com os recursos que já conhecemos e, através de *plug-ins*, é possível acessar recursos nativos dos dispositivos móveis, como câmera, mídia etc. Nesse processo, o Cordova encapsula esse conjunto de código e faz a interface entre o código e o sistema operacional do dispositivo móvel.

Figura 3.11 | Arquitetura de encapsulamento do Apache Cordova



Fonte: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>. Acesso em: 16 jan. 2018.

A partir da proposta do projeto Apache Cordova, diversos fabricantes criaram ambientes de produção para o desenvolvimento híbrido, alguns são gratuitos e outros necessitam de licença.

Pesquise mais

Na página oficial do projeto Apache Cordova existe uma lista com 13 opções de ambientes de desenvolvimento de aplicativos híbridos. Explore alguns desses ambientes acessando: <https://cordova.apache.org/>. Acesso em: 16 jan. 2018.

Vantagem x desvantagem do desenvolvimento híbrido

Dentre as vantagens em se optar pelo desenvolvimento híbrido, segundo DUA (2017), vale destacar: (i) utilizar as tecnologias *web* (HTML, CSS, JavaScript) é mais fácil do que uma linguagem de programação; (ii) o custo final do aplicativo é menor, pois um único código pode ser usado para diversas plataformas, usando uma tecnologia como o Cordova; (iii) o desenvolvimento é mais rápido.

Porém, também existem desvantagens em utilizar essa forma de desenvolvimento: (i) os aplicativos tendem a ser mais lentos que os nativos, pois existe uma camada de *software* intermediária entre o aplicativo e o *hardware*; (ii) existe uma dependência de *plug-ins* para que recursos como câmera e mídia possam ser acessados e, sendo extensões, a permanência desses no mercado não é tão segura quanto um recurso nativo; (iii) Como os recursos são utilizados por *plug-ins*, um mesmo aplicativo pode apresentar resultados diferentes para cada plataforma.

Portanto, embora o fato de um único aplicativo ser aceito em diversas plataformas através do desenvolvimento híbrido, optar por esse caminho deve ser uma escolha consciente, pois também existem desvantagens.

Desenvolvimento cross-plataforma (multiplataforma)

O desenvolvimento de aplicativos cross-plataforma está relacionado à ideia de um único código fonte poder ser executado em diferentes plataformas, portanto, um aplicativo híbrido é por essência cross-plataforma (DUA, 2017). Entretanto, existe uma outra forma de desenvolvimento cross-plataforma que está relacionada às tecnologias nativas, trata-se da plataforma Xamarin, criada pela Microsoft.

Reflita

O desenvolvimento híbrido tem como característica ser cross-plataforma, assim como o desenvolvimento usando a plataforma Xamarin. O que difere a plataforma da Microsoft dos demais aplicativos híbridos? Quais as vantagens e desvantagens entre ambas?

Após investir na plataforma .NET e na linguagem C#, a Microsoft aproveitou o nicho de mercado dos aplicativos para dispositivos móveis e todo o dilema da disputa entre os dois grandes sistemas operacionais e, em 2013, firmou uma parceria com a plataforma Xamarin, anunciando uma nova solução para o desenvolvimento de aplicativos (MANCHESTER, 2013). Com essa tecnologia é possível que os desenvolvedores criem um aplicativo, que é executado em muitas plataformas, usando a linguagem de programação C#, dentro do Visual Studio, um ambiente de desenvolvimento da própria Microsoft. Com uma base de código compartilhada em C#,

os desenvolvedores podem usar ferramentas Xamarin para escrever aplicativos nativos para Android, iOS e Windows com interfaces de usuário nativas e compartilhar código em várias plataformas.

Pesquise mais

Na página oficial do projeto Xamarin existe uma rica fonte de documentação sobre a plataforma e como desenvolver aplicativos usando essa tecnologia.

Explore um pouco dessas informações acessando: <https://www.xamarin.com/>. Acesso em: 18 jan. 2018.

Certamente a grande vantagem do projeto Xamarin é utilizar um único código fonte para os dois grandes sistemas operacionais. Entretanto, vale a pena ressaltar que é uma tecnologia relativamente nova, e seu futuro depende da resposta do mercado com o passar do tempo. Outro detalhe importante é que o código fonte deve ser compilado para a plataforma "alvo" (Android, iOS ou Windows), e a compilação para um dispositivo da Apple só é possível usando um dispositivo da marca.

Agora você já tem uma ideia das principais tecnologias disponíveis no mercado para o desenvolvimento de aplicativos para dispositivos móveis. Escolha uma opção e embarque nesse mundo.

Sem medo de errar

Depois de conhecer as principais tecnologias e ferramentas utilizadas para o desenvolvimento de aplicativos móveis, você tem condições de elaborar uma apresentação para os sócios do Pub, mostrando quais as opções existentes para que eles possam decidir.

- No primeiro *slide* apresente dados sobre os dois principais sistemas operacionais no mercado. Os dados disponíveis em: <https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/> e <https://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/>. Acesso em: 12 jan. 2018 são ótimas opções.

- Como os sócios querem implementar um cardápio digital e interativo, no segundo *slide* apresente um orçamento comparando o valor de três fornecedores de dispositivos Apple e três Google, pois a escolha do dispositivo impactará na escolha da tecnologia a ser usada para o desenvolvimento do aplicativo.
- No terceiro *slide* apresente as vantagens e desvantagens, caso eles escolham um dispositivo da Google.
- No quarto *slide* apresente as duas tecnologias disponíveis, caso ele escolha o dispositivo da Google (híbrido ou nativo). Nesse momento, deve ser enfatizado que a escolha impactará no custo final do projeto.
- No quinto *slide* apresente as vantagens e desvantagens, caso eles escolham um dispositivo da Apple.
- No sexto *slide* apresente as duas tecnologias disponíveis, caso ele escolha o dispositivo da Apple (híbrido ou nativo). Nesse momento, deve ser enfatizado que a escolha impactará no custo final do projeto.
- No sétimo *slide* apresente a sua opção para os sócios, revelando as vantagens e desvantagens.
- No último *slide* apresente um cronograma e um orçamento, caso os sócios aceitem a sua proposta.

Avançando na prática

Criando um aplicativo informativo

Descrição da situação-problema

Um fisioterapeuta lhe procurou para fazer um projeto em parceria. Ele gostaria de fazer um aplicativo móvel voltado para funcionários de empresas que fazem atividade repetitiva, e lhe explicou que gostaria que, nesse aplicativo, tivesse os tipos de atividade repetitiva mais comuns no dia a dia dos funcionários, por exemplo, postura ao sentar, manuseio do teclado e mouse etc. Sobre o funcionamento do aplicativo, ele adiantou que gostaria que tivesse texto e imagem. O cliente não tem nenhuma ideia sobre o custo de um aplicativo, portanto, seu trabalho é apresentar-lhe as tecnologias que podem ser empregadas, e assim ele terá uma noção de custo.

Resolução da situação-problema

O primeiro passo no desenvolvimento de qualquer aplicativo é o levantamento de requisitos, e para isso existem diversas técnicas (*briefing*, *brainstorming*, entrevista etc.). Nesse caso, a entrevista é a melhor opção, visto que o cliente é apenas uma pessoa. Durante a entrevista é importante anotar todas as especificações que o cliente deseja e ir discutindo as possibilidades, afinal o cliente não tem conhecimento técnico e isso pode levá-lo a pedir funcionalidades muito complexas que levariam a um aumento expressivo no valor do produto.

Feito o levantamento de requisitos, seu próximo passo é apresentar um rascunho da solução e, nesse caso, você optou por apresentar uma sequência de *slides* com as telas do aplicativo simulando como seria a navegação.

Terminada a etapa de levantamento de requisitos, chegou o momento de você discutir com o cliente qual tecnologia usar para implementar a solução. Após fazer a análise do projeto, você verificou que o aplicativo iria conter uma navegação simples, com conteúdo composto somente por texto e imagem, além de o orçamento do cliente ser limitado. Devido a essas características, você decidiu propor para o cliente implementar um aplicativo híbrido e preparou uma nova apresentação em *slides* expondo as vantagens e o motivo de você ter escolhido essa tecnologia.

Somente após o levantamento de requisitos e a escolha da tecnologia a ser usada é que tem subsídios para elaborar um orçamento. Apresente o orçamento juntamente com um cronograma.

Faça valer a pena

1. Os aplicativos são o motivo principal de tornar os dispositivos móveis tão úteis e adaptados às nossas rotinas. Aplicativos são desenvolvidos usando linguagens de programação, e qualquer programa precisa passar por um processo de "tradução", no qual os comandos são traduzidos para uma linguagem que a máquina entende, ou seja, para a linguagem de máquina. Esse processo de tradução possui o nome de compilação, e o programa responsável por fazer é chamado de compilador.

Analise as asserções a seguir e a relação proposta entre elas.

I. "O compilador é, assim, um elemento fundamental no desempenho de um computador. O mesmo programa compilado por dois compiladores diferentes e executado no mesmo computador pode apresentar tempos de execução significativamente distintos..." (DELGADO; RIBEIRO, 2017, p. 396).

PORQUE

II. Cada arquitetura de *hardware* exige um certo conjunto de instruções de máquinas, e o compilador, sendo um programa, depende das instruções que foram programadas e da maneira como o código foi projetado para otimizar o desempenho.

a) As asserções I e II são proposições verdadeiras e a II é uma justificativa correta da I.
b) As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa correta da I.
c) A asserção I é uma proposição verdadeira e a II é uma proposição falsa.
d) A asserção I é uma proposição falsa e a II é uma proposição verdadeira.
e) As asserções I e II são proposições falsas.

2. Uma das tecnologias usadas para desenvolver aplicativos para dispositivos móveis é conhecida como desenvolvimento nativo. Esse tipo de desenvolvimento é caracterizado quando um aplicativo é escrito em uma determinada linguagem de programação (como Java, por exemplo) e o código fonte do programa é compilado para uma determinada plataforma.

Considere as afirmações abaixo sobre desenvolvimento nativo e escolha a opção correta.

I. O Android Studio não é o único ambiente de desenvolvimento que possui disponíveis todas as funcionalidades nativas de um sistema Android.
II. O Xcode é o único ambiente de desenvolvimento que possui disponíveis todas as funcionalidades nativas de um sistema iOS.
III. A utilização da plataforma Xamarin juntamente com a linguagem C# não caracteriza o desenvolvimento nativo.

a) Apenas a alternativa I está correta.
b) Apenas a alternativa II está correta.
c) Apenas a alternativa III está correta.
d) As alternativas I e II estão corretas.
e) As alternativas II e III estão corretas.

3. Uma das tecnologias usadas para desenvolver aplicativos para dispositivos móveis é conhecida como desenvolvimento híbrido. Esse tipo de desenvolvimento é caracterizado pelo uso de tecnologias de desenvolvimento web (HTML, CSS e JavaScript), juntamente com um outro *software* (por exemplo, o Cordoval), que fará a interface com os recursos nativos dos dispositivos.

Considere as afirmações abaixo sobre desenvolvimento nativo e escolha a opção correta.

I. O projeto Cordova funciona através de recursos nativos, consequentemente o uso de *plug-ins* não é necessário.
II A existência de diversos ambientes para o desenvolvimento híbrido influencia no desempenho do desenvolvedor, pois para cada ambiente ele terá que usar uma determinada linguagem de programação.
III. Uma das desvantagens do desenvolvimento híbrido é que as aplicações podem se comportar de maneira diferente em cada ambiente, pois existe um *software* que faz a mediação entre o aplicativo e o *hardware*.

a) Apenas a alternativa I está correta.
b) Apenas a alternativa II está correta.
c) Apenas a alternativa III está correta.
d) As alternativas I e II estão corretas.
e) As alternativas II e III estão corretas

Referências

ANDROID. **Open Wonder:** Introducing Android 8.0 Oreo™. 2017. Disponível em: <https://www.android.com/versions/oreo-8-0/>. Acesso em: 19 dez. 2017.

CLEVERISM. **Definition of Swift**. 2017. Disponível em: <https://www.cleverism.com/skills-and-tools/swift/>. Acesso em: 16 jan. 2018.

COSTELLO, Sam. **The History of iOS, from Version 1.0 to 11.0.** 2017. Disponível em: <https://www.lifewire.com/ios-versions-4147730>. Acesso em: 19 dez. 2017.

DELGADO, José; RIBEIRO, Carlos. **Arquitetura de computadores**. Rio de Janeiro: LTC, 2017.

DIFFEN, LCC. **Android vs. iOS.** 2017. Disponível em: <https://www.diffen.com/difference/Android_vs_iOS>. Acesso em: 19 dez. 2017.

DUA, Kinjal. **A Guide to Mobile App Development**: Web vs. Native vs. Hybrid. 2017. Disponível em: <https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/>. Acesso em: 16 jan. 2018.

FONTES, G.; PASSANEZI, P. M. S. **O uso dos tablets nas organizações:** Análise do impacto no ambiente de trabalho. 2012. Disponível em: <http://www.revistaseletronicas.fmu.br/index.php/rms/article/view/23/pdf_1>. Acesso em: 13 dez. 2017.

GOMES, Rafael Caveari; FERNANDES, Jean Alves R.; FERREIRA, Vinicius Corrêa. **Sistema Operacional Android**. 2012. 30 f. TCC (Graduação) - Curso de Engenharia de Telecomunições, Universidade Federal Fluminense, Niterói, 2012.

INFOBASE INTERATIVA (São Paulo). **Como desenvolver aplicativos móveis em 5 passos**. 2015. Disponível em: <http://www.iinterativa.com.br/como-desenvolver-aplicativos-moveis-em-5-passos/>. Acesso em: 12 jan. 2018.

LECHETA, Ricardo R. **Google Android**: aprenda a criar aplicações para dispositivos móveis com o Android SDK. 3. ed. São Paulo: Novatec, 2013.

MANCHESTER, Mina. **Microsoft and Xamarin Partner Globally to Enable Microsoft Developers to Develop Native iOS and Android Apps With C# and Visual Studio**. 2013. Disponível em: <https://www.xamarin.com/pr/xamarin-microsoft-partner>. Acesso em: 18 jan. 2018.

MATEUS, G.R.; e LOUREIRO, A.F. **Introdução À Computação Móvel**. 2. ed. Minas Gerais: EDUFMG, 1998.

MOLL, Cameron. **Mobile WEB Design**. United Kingdom: Mobile Web Book, 2006.

MORIMOTO, Carlos E. **Smartphones**: Guia Prático. GDH Press, 2009.

MUCHOW, John W. **Core J2ME**: Tecnologia e MIDP. São Paulo: Pearson Makron Books, 2004.

NIELD, David. **Android vs. iOS: which is the best mobile OS?** 2016. Disponível em: <https://www.t3.com/news/android-vs-ios-showdown>. Acesso em: 19 dez. 2017.

PAULINO, Rita de Cássia Romeiro; OLIVEIRA, Vivian Rodrigues de. **A tecnologia como norteador das mudanças nos processos de produção jornalística do impresso aos tablets**. 2013. Disponível em: <http://www.ufrgs.br/alcar/encontros-nacionais-1/9o-encontro-2013/artigos/gt-historia-da-midia-digital/a-tecnologia-como-norteador-das-mudancas-nos-processos-de-producao-jornalistica-do-impresso-ao-tablets>. Acesso em: 13 dez. 2017.

SINICKI, Adam. **I want to develop Android Apps: What languages should I learn?** 2017. Disponível em: <https://www.androidauthority.com/develop-android-apps-languages-learn-391008/>. Acesso em: 16 jan. 2018.

STATISTA. **Number of available applications in the Google Play Store from December 2009 to December 2017**. 2018. Disponível em: <https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/>. Acesso em: 12 jan. 2018.

STATISTA. **Number of available apps in the Apple App Store from July 2008 to January 2017**. 2018. Disponível em: <https://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/ >. Acesso em: 12 jan. 2018.

TANENBAUM, Andrew S.; WOODHULL, Albert S. **Sistemas Operacionais**: projeto e implementação. 3. ed. Porto Alegre: Bookman, 2008.

TECHNOLABS, Moon. **Apple vs. Android:** A comparative study. 2017. Disponível em: <https://www.moontechnolabs.com/apple-vs-android-comparative-study-2017/>. Acesso em: 19 dez. 2017.

TECHTUDO. **O que é e como funciona o Kernel**. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2014/02/o-que-e-e-como-funciona-o-kernel-o-nucleo-do-seu-computador.html>. Acesso em: 12 dez. 2017.

VIVAOLINUX. **O Kernel Linux**. Disponível em: <https://www.vivaolinux.com.br/artigo/O-Kernel-Linux>. Acesso em: 7 mar. 2016.

WEEK, Business. **Steve Jobs on Apple's resurgence.** 1998. Disponível em: <http://allaboutstevejobs.com/sayings/stevejobsinterviews/bw98.php>. Acesso em: 13 dez. 2017.

YUAN, Michael Juntao. **Enterprise J2ME**: Developing Mobile Java Applications. New Jersey: Pearson Education, 2004.

ZHOU, Feifan. **Learn Objective-C**: A Brief History. 2010. Disponível em: <https://www.binpress.com/tutorial/objectivec-a-brief-history/37>. Acesso em: 16 jan. 2018.

Aplicação da tecnologia nativa para desenvolvimento Android

Convite ao estudo

Caro estudante, seja bem-vindo à última unidade do livro de tecnologias para web e dispositivos móveis. Desde a expansão da internet e do surgimento dos dispositivos móveis, a área de desenvolvimento para essas tecnologias ganhou grande visibilidade no mercado, sendo uma ótima oportunidade profissional. Você poderá conhecer as tecnologias usadas para a criação de páginas para internet, compreender a evolução dos dispositivos móveis e, ainda, reconhecer as ferramentas disponíveis para o desenvolvimento de aplicativos móveis. Nesta unidade, estudaremos as tecnologias envolvidas no desenvolvimento nativo para o sistema operacional Android.

Para conhecer e compreender os conceitos básicos envolvidos na programação de dispositivos móveis, você continuará o trabalho no Pub e, ao final da unidade, entregará um protótipo da tela de seleção de alguns itens do menu do cardápio. O primeiro passo para completar esse desafio é elaborar um diagrama com algumas classes que farão parte do sistema. O segundo passo consiste em documentar a IDE Android Studio e a arquitetura de um aplicativo para Android. Esse passo é importante, pois futuramente você poderá contratar outro profissional para orientá-lo e esse documento o auxiliará a entender o mundo móbile. E, por fim, você poderá apresentar o protótipo de tela para os sócios do Pub.

Para finalizar nosso estudo, na primeira seção conheceremos um pouco sobre o mundo da programação orientada a objetos e a linguagem Java. Na segunda seção, aprenderemos a instalar e a usar o ambiente de desenvolvimento Android Studio e, na terceira unidade, implementaremos o protótipo do menu.

Bons estudos!

Seção 4.1

O paradigma de programação orientado a objetos

Diálogo aberto

Caro aluno, iniciamos agora um estudo voltado à prática do desenvolvimento de aplicativos para ambientes móveis. Nosso primeiro passo é fazer uma introdução do paradigma da orientação a objetos e da linguagem Java. Segundo Cass (2017), Java é uma das três linguagens de programação mais utilizadas e procuradas no mercado. Portanto, seu aprendizado é, sem dúvida, um diferencial para o profissional de TI.

Continuando o projeto de automatização do atendimento do Pub, nesse momento você deverá fazer um esboço das classes que serão utilizadas no sistema e organizá-las em forma de diagrama. Esse documento não precisa ser apresentado aos sócios, pois é uma documentação mais técnica e poderá auxiliá-lo no desenvolvimento e na manutenção do sistema.

Para a realização dessa etapa de documentação, você conhecerá o paradigma da programação orientada a objetos e seus quatro pilares. Aprenderá, também, como escrever classes em Java e como elas se tornam objetos, além de trabalhar com herança na mesma linguagem.

Aproveite esta grande oportunidade! Bons estudos!

Não pode faltar

Introdução ao paradigma de orientação a objetos

Dizer que uma linguagem de programação segue o paradigma orientado a objetos significa que a forma de organizar os comandos e os recursos disponíveis fazem parte de um padrão com algumas regras bem definidas. No contexto da programação de

computadores, um paradigma é um jeito, uma maneira, um estilo de se programar. De acordo com Santos (2003), no paradigma de programação orientada a objetos, os dados a serem processados e os mecanismos desse processamento devem ser considerados em conjunto, com base em um modelo.

Quando representamos elementos reais de forma simplificada e padronizada, criamos um modelo para eles. "Modelos são representações simplificadas de objetos, pessoas, itens, tarefas, processos, conceitos, ideias etc., usados comumente por pessoas no seu dia a dia, independente do uso de computadores." (SANTOS, 2003, p. 2).

A ideia da programação orientada a objetos ganhou impulso na década de 1970 e, no começo da década de 1980, Bjarne Stroustrup integrou a orientação a objetos na linguagem C, o que resultou no C++, tida como a primeira linguagem OO usada em massa (THE UNIVERSITY OF TENNESSEE, [s.d.]). No início dos anos 1990, um grupo da Sun, liderado por James Gosling, desenvolveu uma versão mais simples do C++, que foi batizada de Java.

Características e princípios do paradigma de orientação a objetos

Machado (2015) entende que o paradigma da orientação a objetos é fundamentado por quatro características:

Abstração: a abstração está relacionada à definição precisa de um objeto, o que inclui sua identificação (nome), suas características (ou propriedades) e o conjunto de ações que ele desempenha. A abstração de um objeto será representada por uma estrutura conhecida como **classe**.

Assimile

Classes são estruturas das linguagens de programação orientadas a objetos e criadas para conter os dados que devem ser representados e as operações que devem ser efetuadas com estes dados para determinado modelo (SANTOS, 2003, p. 14).

Cada classe terá os seguintes componentes:

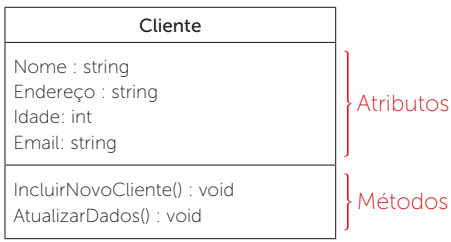
- Um nome que identificará o objeto.
- Atributos ou propriedades: são as variáveis (ou campos) que ajudarão a caracterizar o objeto. Por exemplo: nome, idade, endereço etc.
- Métodos (ações que o objeto poderá efetuar).

Exemplificando

Vários programas são desenvolvidos para atender às necessidades de clientes (aplicativos de banco, cinemas, lojas etc.), portanto, modelar o objeto "Cliente" é um dos primeiros passos. No projeto de um software OO, as classes são representadas graficamente e depois implementadas em uma linguagem de programação. Vamos modelar a classe Cliente (Figura 4.1, item a) e, depois, implementá-la na linguagem Java (Figura 4.1, item b).

Figura 4.1 | Classe cliente

a) Representação do objeto por uma classe



b) Implementação da classe na linguagem Java

```
1.  public class Client {
2.      String nome;
3.      String endereço;
4.      int idade;
5.      email: string
6.      public void IncluirNovoCliente (){
7.          //comandos
8.      }
9.      public void AtualizarNovoCliente(){
10.         //comandos
11.     }
12. }
```

Fonte: elaborada pelo autor.

Nesse primeiro contato, é importante esclarecer alguns pontos importantes sobre a estrutura da classe na linguagem Java.

- Na linha 1, temos a declaração da classe e o parâmetro *public* representa o tipo de acesso à classe, nesse caso, público (qualquer outra classe poderá acessar a classe Cliente). O segundo parâmetro *class* é padrão e o terceiro é o nome da classe.
- Para indicar o início e o fim de classes e métodos, são utilizadas chaves. A chave da linha 1 marca o início da classe e a chave na linha 12 marca o fim.
- Das linhas 2 a 5, os atributos da classe são declarados de acordo com a sintaxe Java: **tipo variável**.
- Na linha 6, há a declaração de um método com os seguintes parâmetros: (i) *public* representa o tipo de acesso ao método; (ii) *void* significa que o método fará alguma coisa e não devolverá nenhum valor para o sistema; (iii) *IncluirNovoCliente* representa o nome do método; (v) () indica que todo método terá um par de parênteses, pois caso o método receba informações do sistema, esses devem vir entre parênteses.
- A classe deve ser salva com o nome *Cliente.java*.

E assim construímos, de maneira simplificada, uma classe na linguagem Java.

Reflita

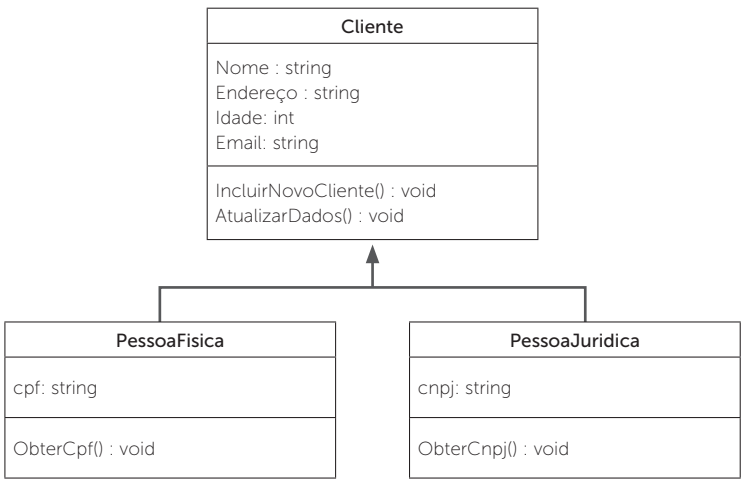
Mesmo com restrições impostas pela linguagem de programação, o desenvolvedor tem liberdade para escolher o nome das variáveis, dos métodos e das classes. Genericamente, esses nomes são conhecidos como "identificadores". No entanto, o bom senso deve sempre prevalecer nas escolhas. Não seria razoável criar o nome desses elementos conforme a função que executam no contexto do programa? Qual seria o sentido, por exemplo, de chamar de "multiplicação" uma classe que executa operações de adição?

Herança: por meio dessa característica do paradigma OO, um objeto filho herdará características e comportamentos do objeto pai.

No projeto de um software OO, as classes podem ser construídas reaproveitando partes de outras, recurso que caracteriza a herança. Vamos incrementar a modelagem iniciada na Figura 4.1 para ilustrar o uso da herança e sua implementação em Java.

Figura 4.2 | Exemplo de herança

a) Representação gráfica de herança



b) Implementação de herança em Java.

```
1.    public class PessoalFisica extends Client {
2.        String cpf;
3.        public void ObterCpf(){
4.            //comandos
5.        }
6.  }

1.    public class PessoalJuridica extends Client {
2.        String cnpj;
3.        public void ObterCnpj(){
4.            //comandos
5.        }
6.  }
```

Fonte: elaborada pelo autor.

Implementar a herança na linguagem Java é simples, pois, como podemos observar nas linhas 1 da Figura 4.2, item b, basta utilizar o comando *extends* para que uma classe receba como herança os atributos e métodos de outra. Pontos que merecem atenção:

- A classe Cliente é chamada de classe base, classe pai ou, ainda, super.
- Tanto as classes PessoaFisica como PessoaJuridica são chamadas de classes filhas.
- As classes filhas possuem TODOS os atributos e métodos da classe pai + os seus.
- Cada classe deve ser criada em um arquivo distinto e salvo como: PessoaFisica.java -– PessoaJuridica.java

Polimorfismo: já sabemos que um objeto filho herda características e ações de seu objeto pai, situado hierarquicamente acima do primeiro. Contudo, em certos casos, precisaremos definir ações do objeto de outra forma. Assim, polimorfismo consiste em dar outra forma a alguma ação herdada do objeto pai.

Para exemplificar o uso do polimorfismo na linguagem Java, vamos continuar trabalhando com as classes Cliente, PessoaFisica e PessoaJuridica (Figura 4.3).

Figura 4.3 | Polimorfismo implementado em Java

```
1.    public class PessoalFisica extends Client {
2.        String cpf;
3.        public void ObterCpf(){
4.            //comandos
5.        }
6.  Public Void AtualizarDados (){
7.            //comandos na classe filha
8.        }
9.  }

1.    public class PessoalJuridica extends Client {
2.        String cnpj;
3.        public void ObterCnpj(){
4.            //comandos
5.        }
6.  Public Void AtualizarDados (){
7.            //comandos na classe filha
8.        }
9.  }
```

Fonte: elaborada pelo autor.

Na Figura 4.3, temos a implementação de métodos polifórmicos. Nas linhas 6, o método AtualizarDados() está sendo novamente

declarado (lembrando que ele já existe na classe pai), porém com comandos que são diferentes daqueles implementados na classe pai.

Encapsulamento: o termo "encapsulamento" está relacionado à proteção ou ocultação dos dados do objeto.

Para exemplificar o último pilar da orientação a objetos, vamos encapsular o campo cpf na classe PessoaFisica:

Figura 4.4 | Exemplo de encapsulamento

```
1.    public class PessoalFisica extends Client {
2.        private String cpf;
3.        public String getCpf(){
4.            return cpf;
5.        }
6.  }
```

Fonte: elaborada pelo autor.

Na linha 2, o atributo cpf foi declarado como *private*. Isso significa que outras classes não têm acesso a esse dado. Para que ele possa ser usado, foi criado o método getCpf(), que retorna o valor guardado no cpf, ou seja, o atributo foi encapsulado por um método.

Para que as classes criadas possam ser utilizadas, é preciso criar uma instância dela. Esse processo é feito por meio do comando "new". Vejamos como ele funciona. Vamos criar uma classe por meio da qual acessaremos a classe Cliente e guardaremos um nome e uma idade de um cliente. Na linha 2, temos um método "especial" chamado "main". Em todo projeto Java deverá haver (obrigatoriamente) uma classe com esse método, que indica onde será o início da execução. Observe que, por meio dos comandos da linha 3, uma instância da classe Cliente será criada e a variável c1 dará acesso aos atributos e métodos da classe. Por isso, na linha 4, o nome João está sendo guardado no atributo *nome* da instância c1 e, na linha 5, o número 25 está sendo alocado no atributo *idade* da mesma instância.

```
1.    public class Principal {
2.        public static void main(string[] args){
3.            Cliente c1 = new Cliente();
4.            c1.nome = "João";
5.            c1.idade = 25;
6.        }
7.  }
```

Assimile

Não existe limite para a quantidade de instâncias que podem ser criadas. Isso depende da memória que o usuário tem em sua máquina. Portanto, poderíamos ter *n* instâncias da classe Cliente ou de qualquer outra.

Java é uma das linguagens mais utilizadas no mercado. Conhecer essa ferramenta, bem como a orientação a objetos, pode ser um diferencial para quem procura se destacar no mercado de trabalho.

Pesquise mais

No YouTube, há uma série completa de videoaulas de programação em Java. Disponível em: <www.youtube.com/watch?v=NZDzuve7kho&list=PLxQNfKs8YwvGhXHbHtxtoB-tRRv6r3Rlr&index=1>. Acesso em: 25 jan. 2018. Aproveite a oportunidade e tenha um diferencial em sua formação.

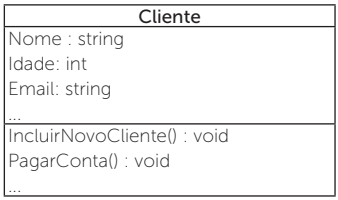
Demos o primeiro passo no desenvolvimento de aplicativos para dispositivos móveis. Aprender os fundamentos da orientação a objetos, bem como a linguagem Java, são vitais para o profissional que deseja atuar nessa área. Parabéns pelo caminho percorrido até o momento!

Sem medo de errar

Agora que conhece os principais pontos do paradigma orientado a objetos, você pode realizar mais uma etapa para o Pub. Foi proposto a você fazer um esboço de algumas classes que farão parte do sistema.

Para cumprir essa tarefa, você deve pensar nos principais objetos do mundo real que farão parte do sistema, como pessoas, alimentos e bebidas. Pense em outros objetos que possam ser modelados para o sistema e escreva as classes, em forma de diagrama, conforme exemplo na Figura 4.5.

Figura 4.5 | Exemplo para modelo de classes do Pub



Fonte: elaborada pelo autor.

Não se esqueça de estabelecer todos os atributos e métodos pertinentes à aplicação, bem como as relações de herança para os casos que forem adequados.

Prepare o documento e imprima uma versão. As classes o auxiliarão no processo de implementação.

Avançando na prática

Implementando uma classe na linguagem Java

Descrição da situação-problema

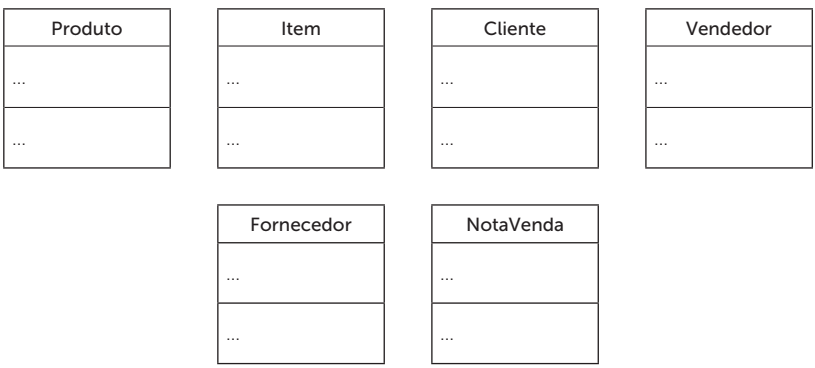
Muitas aplicações exigem que seja implementado um controle de estoque. O controle de estoque é uma área crucial para qualquer empresa que trabalhe com produtos, pois muitas ações podem ser tomadas utilizando-se resultados e dados. O responsável pela área de controle de estoque será capaz de prever quanto terá de comprar ou, ainda, fornecer dados para que as áreas de business inteligence (BI) possam criar estratégias de marketing.

Sabendo da importância dessa área, proponha um modelo simplificado de classes que possam fazer parte de um controle de estoque, apontando os atributos e métodos de cada uma delas.

Resolução da situação-problema

A complexidade das regras do controle de estoque, bem como a quantidade de classes, atributos e métodos poderão variar conforme a necessidade de cada sistema. Na Figura 4.6, há algumas classes que podem fazer parte de um controle de estoque.

Figura 4.6 | Algumas classes para controle de estoque



Fonte: elaborada pelo autor.

Termine de criar o diagrama propondo quais atributos e métodos poderiam fazer parte de cada modelo. Fique à vontade para propor novas classes.

Faça valer a pena

1. A programação orientada a objetos separa claramente a noção de **o que** é feito de **como** é feito:
"O que" é descrito como um conjunto de métodos – e às vezes com dados publicamente disponíveis – e suas semânticas associadas. Esta combinação de métodos, dados e semântica é muitas vezes descrita como um contrato entre o projetista da classe e o programador que a usa (ARNOLD et al., 2007).

Um método é um componente da classe que:

a) representa a estrutura da classe e dos seus atributos.
b) armazena o estado de uma classe e de seus atributos.
c) representa as variáveis de dados associadas à classe.
d) contém as ações (ou funcionalidades) de uma classe.
e) define os valores iniciais das variáveis da classe.

2. "Uma das boas referências em orientação a objetos ensina que classes são estruturas das linguagens de programação orientadas a objetos criados para conter os dados que devem ser representados e as operações que devem ser efetuadas com estes dados para determinado modelo" (SANTOS, 2003, p. 14).

Analise as afirmações que seguem e assinale a alternativa que contém apenas indicações de afirmações verdadeiras relacionadas a classes.

I) Embora não possua meio de representar dados e comportamentos juntos, uma classe, ainda assim, pode ser comparada a um modelo.
II) O rigor formal com que se escreve uma classe em Java restringe-se à correta colocação de letras maiúsculas e minúsculas em seu nome.
III) Classes são criadas por expressões contendo a palavra "new". Criar uma classe com base em uma definição de atributos é conhecido como "classificação".
IV) A unidade fundamental de programação da linguagem de programação Java é a classe. Classes fornecem a estrutura para os objetos.
a) Apenas a afirmação I é verdadeira.
b) Apenas a afirmação IV é verdadeira.
c) Apenas as afirmações III e IV são verdadeiras.

d) Apenas as afirmações II e III são verdadeiras.
e) Apenas as afirmações II e IV são verdadeiras.

3. Variáveis locais são declaradas dentro de um bloco de código, tal como o corpo de um método, em contraste com campos, que são declarados como membros de uma classe. Cada variável deve ter um tipo que precede seu nome quando a variável é declarada (ARNOLD et al., 2007).

Considerando o contexto apresentado, avalie as seguintes asserções e a relação proposta entre elas.

I) Campos declarados em uma classe são válidos por toda a extensão da classe, desde que declarados antes do método que utilizam esses campos.

PORQUE

II) A visibilidade dos elementos da classe varia em função do tipo de classe declarada.

A respeito dessas asserções, assinale a alternativa verdadeira.

a) As asserções I e II são proposições falsas.
b) A asserção I é uma proposição falsa e a II é uma proposição verdadeira.
c) A asserção I é uma proposição verdadeira e a II é uma proposição falsa.
d) As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa da I.
e) As asserções I e II são proposições verdadeiras,e a II é uma justificativa da I.

Seção 4.2

Ambiente de desenvolvimento Android Studio

Diálogo aberto

Caro estudante, bem-vindo a mais uma seção do universo de desenvolvimento de aplicativos para o sistema Android. Na seção anterior você deu um passo importante ao ter uma introdução do paradigma da programação orientada a objetos, bem como ter contato com a linguagem Java. O próximo passo nessa caminhada consiste em conhecer a principal (e oficial) ferramenta de programação para o sistema Android, além de conhecer a estrutura de um aplicativo.

Dando continuidade ao seu projeto de implementação de recursos digitais no Pub, nessa etapa seu trabalho não será uma entrega para os sócios, mas sim o desenvolvimento da documentação sobre a ferramenta de desenvolvimento e sobre alguns pontos da arquitetura de um aplicativo para Android. Como o universo do desenvolvimento para dispositivos móveis está em constante atualização, você deve documentar as características da IDE Android Studio bem como do arquivo *AndroidManifest.xml*. Todo software precisa ter uma documentação, pois novos membros podem ser contratos para a equipe e precisam ter acesso a tal documento para entender o mecanismo de trabalho.

Para cumprir seu trabalho você aprenderá nessa seção sobre ambiente de desenvolvimento Android Studio, os passos necessários para a instalação, os simuladores de Android e como é a arquitetura de um aplicativo para Android.

Em breve você poderá aplicar todo o conhecimento adquirido até o momento, continue dedicando-se a essa promissora área.

Bons estudos!

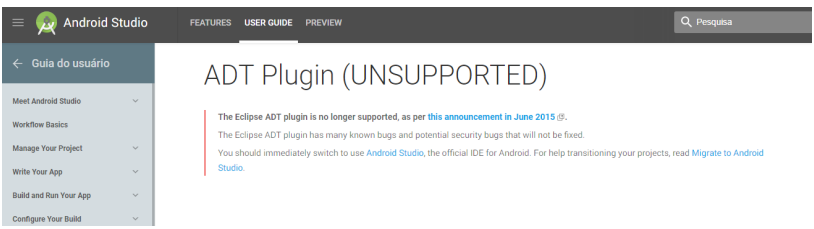
Não pode faltar

Apresentação do ambiente de desenvolvimento Android Studio

O desenvolvimento de aplicativos para dispositivos móveis possui práticas e recursos que são similares ao desenvolvimento de *softwares* para um computador convencional (PC). Em ambos os casos, o programador poderia abrir um simples editor de texto, como o bloco de notas, digitar seu código e fazer a compilação usando o compilador específico. Essa prática, embora muito utilizada nos primórdios da programação, não é mais utilizada no mercado de trabalho. Essa metodologia foi substituída por ambientes de desenvolvimento, mais especificamente, por ambientes de desenvolvimento integrado (IDE, do inglês, *Integrated Development Environments*). Segundo Silva (2016, p. 35), "Os IDEs são ferramentas que os desenvolvedores utilizam para diversos processos relativos ao código, como: criação, edição, compilação e depuração". Portanto, uma IDE possui uma série de ferramentas necessárias ao desenvolvimento de softwares já integrado ao ambiente, o que facilita e agiliza o desenvolvimento para o programador.

Uma forma de desenvolvimento de aplicativos para Android que foi amplamente utilizada consistia na combinação da IDE Eclipse em conjunto com o ADT (*Android Developer Tools*). Entretanto, caso você tente utilizar tal recurso, irá encontrar a mensagem da Figura 4.7 na documentação oficial do desenvolvimento para Android.

Figura 4.7 | Captura de tela da documentação oficial



Fonte: <https://developer.android.com/studio/tools/sdk/eclipse-adt.html>. Acesso em: 29 jan. 2018.

Isso acontece porque, em 2014, a Google anunciou a IDE Android Studio como a ferramenta oficial para o desenvolvimento de aplicações Android (MULLIS, 2017). Essa IDE trabalha em conjunto com o Android SDK (*Software Development Kit*) o qual dará acesso às funcionalidades do sistema operacional em questão. Ainda segundo Mullis (2017, p. 1), "*Java é necessário para escrever os programas, o Android SDK é necessário para fazer esses programas serem executados no Android e o Android Studio tem o trabalho de colocar tudo junto para você.*"

Instalação do ambiente de desenvolvimento Android Studio.

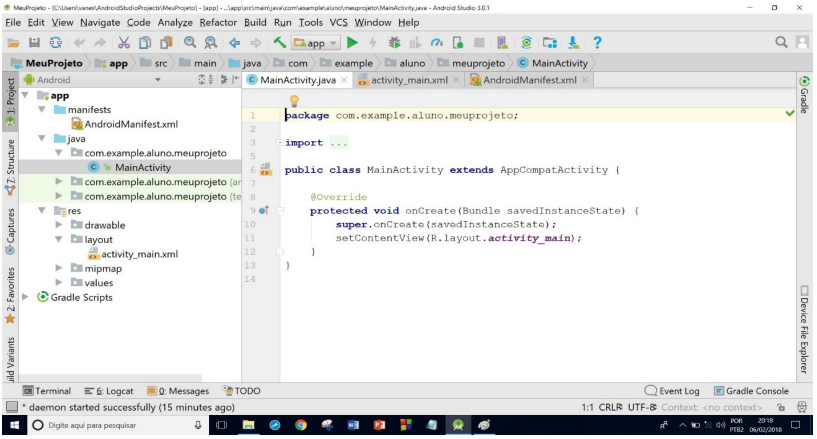
Para criar e compilar um aplicativo é necessário, em primeiro lugar, instalar o JRE (*Java Runtime Environment*), disponível para download no endereço <https://java.com/pt_BR/download/>. Acesso em: 05 fev. 2018. O segundo passo é instalar o JDK (*Java Development Kit*), disponível para download no endereço: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>. Acesso em: 29 jan. 2018. Certifique-se de instalar a versão de acordo o sistema operacional do seu computador. Por fim, é preciso instalar o ambiente de desenvolvimento Android Studio, disponível para download no endereço: <https://developer.android.com/studio/index.html>. Acesso em: 29 jan. 2018.

Reflita

A Google investiu dinheiro e tempo para construir e tornar o Android Studio a ferramenta oficial para o desenvolvimento de aplicativos para Android. Quais vantagens e desvantagens essa medida trouxe para os desenvolvedores?

Feito o download de todos os softwares, instale na seguinte sequência: (i) JRE; (ii) JDK; e (iii) o Android Studio. Durante a instalação do último, marque a opção para também instalar o Android SDK. Após preparar o ambiente, você pode criar um novo projeto, clicando em File > New > New Project. Ao criar um novo projeto, você verá algo semelhante à Figura 4.8 (semelhante porque depende de algumas escolhas).

Figura 4.8 | Ambiente do Android Studio



Fonte: elaborada pelo autor.

Ainda sobre a instalação, é possível que, durante a execução da IDE Android Studio, o ambiente ainda peça para que você instale ou atualize alguns recursos (lembre-se de que essas ferramentas estão em constante evolução). Fique atento às mensagens nas barras e siga as instruções quando for necessário.

Arquitetura de um aplicativo para Android

Existe um esforço dos desenvolvedores de IDE em tornar sua ferramenta prática. Diante dessa corrida, várias funcionalidades são incorporadas e vários métodos e bibliotecas são disponibilizados nesses ambientes. A consequência é que tais facilidades geram várias linhas de código antes mesmo de um programador escrever sua primeira linha. O Android Studio é uma das IDE que apresentam vários arquivos e várias linhas de código assim que um novo projeto é criado, conforme pode ser observado na Figura 4.8. Cada arquivo tem uma função específica dentro do projeto, e a junção deles é que criará o aplicativo Android. Vamos agora, começar a entender a arquitetura de um aplicativo para Android e explorar alguns dos principais arquivos.

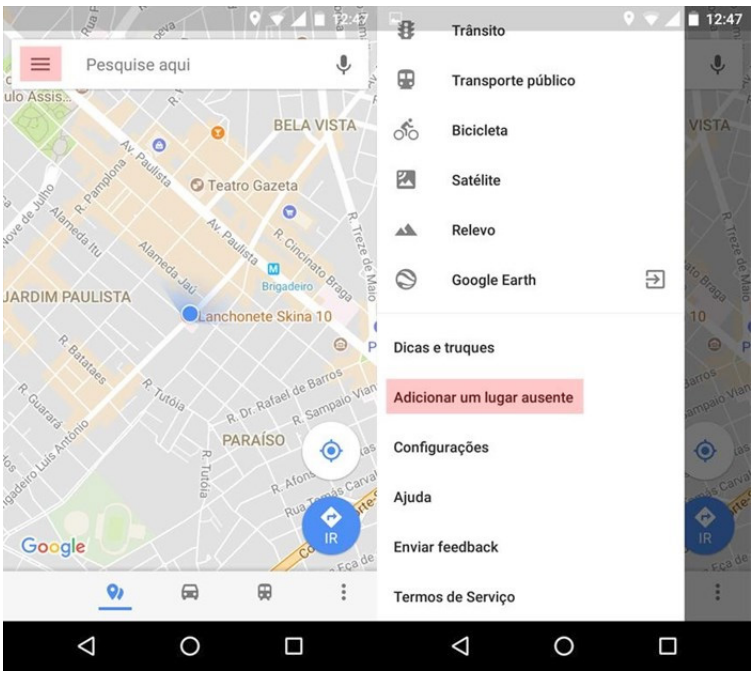
No mundo da programação para dispositivos Android um termo é bastante comum: "**Activity**".

> Activity é um componente de aplicativo que fornece uma tela com a qual os usuários podem interagir para fazer algo, como discar um número no telefone, tirar uma foto, enviar um e-mail ou ver um mapa. Cada atividade recebe uma janela que exibe a interface do usuário. Geralmente, a janela preenche a tela, mas pode ser menor que a tela e flutuar sobre outras janelas. (DEVELOPER, 2017, p. 1).

Exemplificando

Para entendermos o conceito de activity, vamos ver um exemplo usando um aplicativo do celular. Ao abrir o Google Maps, uma activity principal é iniciada, conforme outros recursos vão sendo abertos, novas activities podem ser iniciadas.

Figura 4.9 | Exemplo de activity no Android



Fonte: <https://www.tecmundo.com.br/internet/119792-aprenda-adicionar-local-google-maps-smartphone.htm> Acesso em: 29 jan. 2018.

Um aplicativo Android terá, no mínimo, uma activity, que é chamada de *Main Activity* (atividade principal). Essa activity representa o ponto inicial pelo qual o aplicativo executará e é dividida em dois arquivos: MainActivity.Java e activity_main.xml.

No arquivo activity_main.xml deverá ser codificado o *layout* dessa activity. Aqui serão inseridos textos, botões, caixas de seleção etc. É nesse arquivo que especificamos um *layout* responsivo, ou seja, um layout que se adapta automaticamente ao tamanho da tela do usuário. Podemos acessar essa funcionalidade através do gerenciador responsivo *Constraint Layout* e fazendo algumas configurações nos objetos.

No arquivo MainActivity.Java deverá ficar a parte de programação na linguagem Java, contendo questões relativas ao comportamento daquela determinada activity, como por exemplo, decisões e testes de validação. Se tratando de uma linguagem orientada a objetos, todos os objetos que forem usados no aplicativo serão "criados" (instanciados) através de alguma *activity*, que pode ser a principal, como também em outra.

O AndroidManifest.xml é outro arquivo essencial para um aplicativo Android, sem esse arquivo não é possível construir a aplicação. Ele fica no diretório raiz e possui informações que serão consultadas antes do aplicativo ser executado, como por exemplo, as bibliotecas usadas, a versão mínima do sistema Android necessária para executar a aplicação, dentre outras.

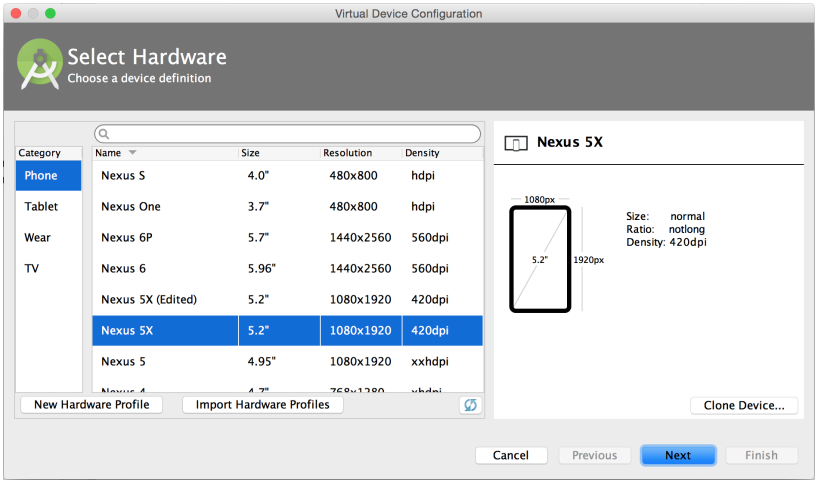
Assimile

Qualquer item adicionado no arquivo activity_main.xml, por exemplo, texto, botão, caixa de seleção, etc., deve ser referenciado dentro do arquivo MainActivity.Java, pois o correto funcionamento do aplicativo depende da ligação de todos os arquivos de construção.

Emuladores do sistema Android (Android Emulator)

Durante a criação do aplicativo, é necessário fazer vários testes, para isso são feitas inúmeras compilações (DEITEL; DEITEL; WALD, 2016). O resultado de cada compilação pode ser visualizado dentro do próprio ambiente de desenvolvimento, usando o *Android Virtual Device* (AVD). Após a instalação do Android Studio é provável que o ambiente não tenha nenhuma AVD instalada, portanto, você deverá instalar através do AVD *Manager*, que se encontra no menu Tools > Android > AVD Manager. A Figura 4.10 é uma captura de tela do ambiente para instalação do AVD.

Figura 4.10 | Instalando um AVD



Fonte: <https://developer.android.com/studio/run/managing-avds.html?hl=pt-br>. Acesso em: 6 fev. 2018.

Cada AVD define um perfil de hardware e software, ou seja, define as características de um celular, tablet, etc. A parte de hardware irá simular o funcionamento do hardware do dispositivo, já o software simulará a versão do Android, que está associada a uma API (*Application Programming Interface*). Confira na tabela 4.1, algumas versões do Android e sua respectiva API.

Tabela 4.1 | Versão Android x API

Nome	Versão	API
Oreo	8.1.0	API nível 27
Oreo	8.0.0	API nível 26
Nougat	7.1	API nível 26
Nougat	7.0	API nível 25

Fonte: adaptada de: <https://source.android.com/setup/build-numbers>. Acesso em: 19 fev. 2018.

É importante ressaltar que para cada projeto é escolhida uma versão da API para ser o destino do aplicativo (*target*), porém a escolha de uma API com nível mais alto não significa que o aplicativo não irá funcionar nas demais, significa apenas que algumas funcionalidades podem não estar disponíveis. Em sua máquina, você pode ter instalado quantas versões do sistema operacional Android desejar, para isso basta criar diversos emuladores (AVDs), um para cada versão do software.

Pesquise mais

Assista ao vídeo disponível em: <https://www.youtube.com/watch?v=3iacZE1wY9s>. Acesso em: 5 fev. 2018, e veja com detalhes como é feita a instalação dos recursos necessários para o funcionamento do Android Studio.

Agora que o ambiente está preparado, nosso próximo passo é aprender a criar aplicações para dispositivos móveis com o sistema operacional Android Studio. Na próxima seção, criaremos nosso primeiro projeto.

Sem medo de errar

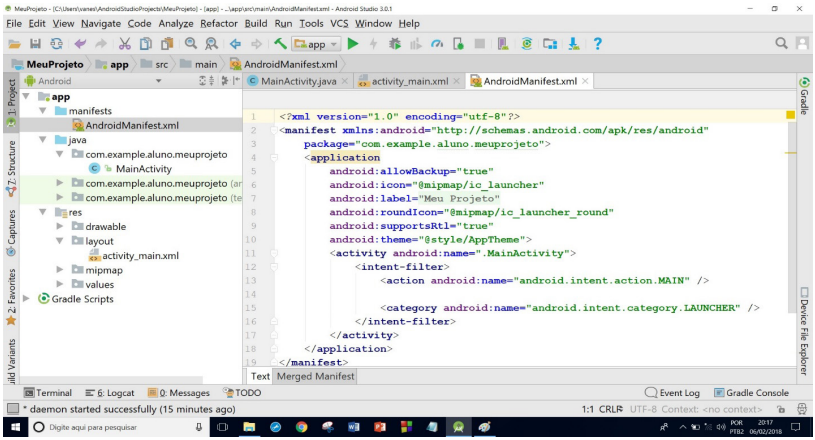
Você está em uma importante fase do projeto para o Pub, pois já acertou os detalhes com os sócios, inclusive já possui o esboço das classes que deverão ser implementadas no sistema. Porém, antes de efetivamente programar, você deve preparar a documentação do projeto conforme solicitado, a saber, a descrição de algumas características da IDE oficial e do arquivo AndroidManifest.xml.

Comece a documentação com informações sobre a versão da IDE que será utilizada no projeto. Em seguida deixe registrado como é feita a instalação, para isso os seguintes itens são necessários: (i) site oficial para download; (i) recomendações de espaço, memória RAM e processador para executar o ambiente de desenvolvimento; (iii) passo a passo para instalação, inclusive do emulador.

Terminada a primeira parte da documentação, passe a registrar o que é o arquivo de manifesto: o arquivo *AndroidManifest.xml* é considerado o coração do aplicativo Android, pois nele estarão todas as informações sobre a configuração do app.

Em seguida apresente uma figura de uma captura de tela de um projeto Android, destacando o arquivo em questão. Essa figura ajudará na apresentação de alguns recursos.

Figura 4.11 | O arquivo *AndroidManifest.xml*



Fonte: elaborada pelo autor.

Explique a localização do arquivo dentro da árvore de pastas que o Android Studio cria. Em seguida explique os comandos que aparecem na Figura 4.11.

A primeira linha indica a versão do xml que foi usada para gerar o código, além da codificação usada. Explique que o xml é uma linguagem de marcação (assim como o HTML) e também funciona através de *tags*.

Deixe claro que apenas duas *tags* são obrigatórias: <*manifest*> e <*application*> e que elas devem ocorrer somente uma vez, ao contrário das demais que podem aparecer 0 ou *N* vezes.

O próximo passo é explicar o que cada uma das *tags* obrigatórias faz. A primeira é o elemento raiz do arquivo *AndroidManifest.xml,* sua declaração é acompanhada de alguns atributos, sendo alguns deles:

(i) **xmlns:android.** Esse atributo define qual a namespace de trabalho do Android. Fazendo uma analogia, podemos comparar uma namespace com uma biblioteca de informações. O valor do atributo deve sempre ser "http://schemas.android.com/apk/res/android", caso uma configuração diferente seja feita, os comandos não serão reconhecidos.

(ii) **package.** Indica o pacote do seu projeto, esse nome serve como um identificador do seu projeto. Por exemplo, vários projetos podem implementar uma classe chamada "ConexaoBanco", o que diferencia uma classe da outra será o nome do pacote do projeto.

(iii) **android:versionCode.** Esse atributo identifica qual a versão do seu projeto. Vamos supor que o usuário está usando a versão 1.0 e você fez uma atualização para a versão 1.2. Assim que você altera o valor nesse parâmetro, o usuário do aplicativo recebe uma notificação que existe uma nova atualização.

(iv) **android:versionName.** Esse parâmetro é apenas uma identificação para o usuário.

Apenas explicados esses parâmetros, passe à apresentação da segunda tag obrigatória <*application*>. Primeiro explique que ela deve vir entre o início e o fim da *tag* <*manifest*>. Dentro dessa tag irão ficar todos os demais elementos do aplicativo, como, por exemplo, todas as *activities* que forem criadas. Observe que na figura 4.11 existe apenas uma *tag* <activity> </activity> (abrindo e fechando), portanto o aplicativo foi desenvolvido usando apenas uma activity.

Agora complete sua apresentação, consultando os endereços:

- <https://developer.android.com/guide/topics/manifest/manifest-intro.html?hl=pt-br>
- <https://developer.android.com/guide/topics/manifest/application-element.html?hl=pt-br>

E descreva mais alguns atributos das *tags* <*manifest*> e <*application*>.

Avançando na prática

Conhecendo uma activity

Descrição da situação-problema

Um gerente de uma empresa de desenvolvimento de software, percebendo que o desenvolvimento de aplicativos para dispositivos móveis se tornou um dos mercados mais rentáveis na área de software, decidiu treinar sua equipe para desenvolver aplicativos para o sistema Android usando a IDE oficial. Ao iniciar o projeto, ele percebeu que apenas saber programar não era suficiente, pois

os aplicativos dessa área possuem uma arquitetura diferente. Ao procurar conhecer esse novo universo, constatou que os aplicativos são desenvolvidos baseados em activities e lhe contratou para fazer uma apresentação aos programadores sobre as características desse elemento. Portanto, prepare uma documentação apontando alguns dos principais pontos.

Resolução da situação-problema

Comece sua apresentação explicando o que é uma activity:

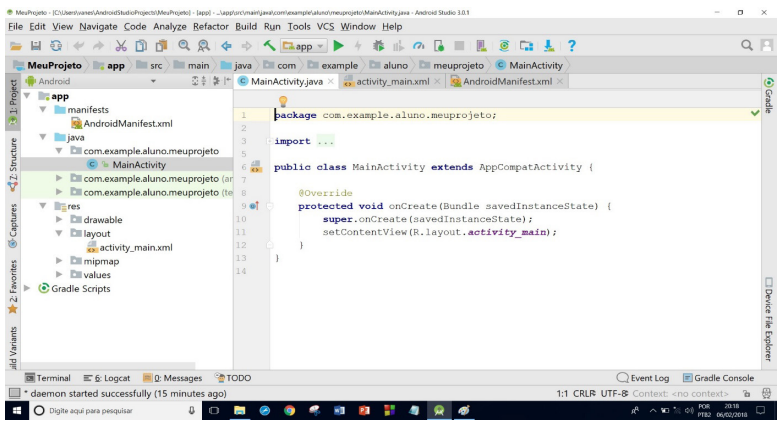
> **Activity é um componente de aplicativo que fornece uma tela com a qual os usuários podem interagir para fazer algo, como discar um número no telefone, tirar uma foto, enviar um e-mail ou ver um mapa. Cada atividade recebe uma janela que exibe a interface do usuário. Geralmente, a janela preenche a tela, mas pode ser menor que a tela e flutuar sobre outras janelas. (DEVELOPER, 2017, p. 1)**

Em seguida, explique que uma atividade depende de dois arquivos: MainActivity.Java e activity_main.xml e explique o que cada um deles representa no projeto.

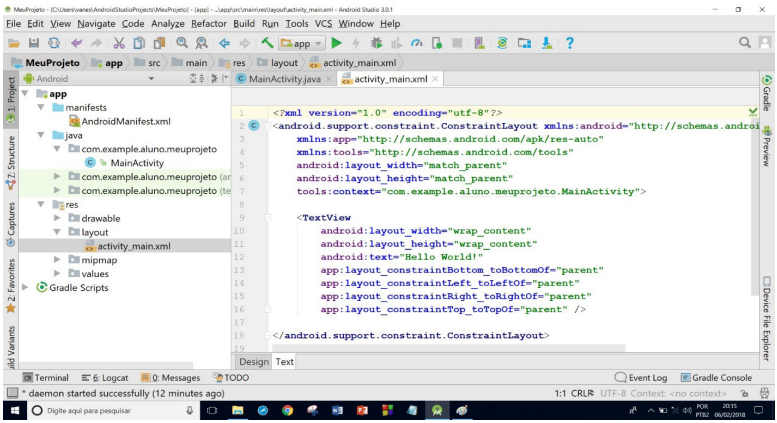
Em seguida, mostre uma captura de tela de cada arquivo, conforme Figura 4.12.

Figura 4.12 | Captura de tela – a) MainActivity.Java – b) activity_main.xml

a)



b)



Fonte: elaborada pelo autor.

Use as imagens para explicar a localização desses arquivos na árvore de pastas e explique que, dentro do arquivo *MainActivity.Java*, o método *setContentView(R.layout.activity_main)* é o responsável por fazer a ligação entre os dois arquivos.

Faça valer a pena

1. O ambiente de desenvolvimento de um aplicativo para o sistema Android depende do *Java Runtime Environment* (JRE). Esse software atua como uma camada entre o hardware e a aplicação, fazendo a tradução do código escrito em Java para a linguagem que o hardware entende.

Sobre a instalação do Android Studio, escolha a opção correta.

a) Devemos instalar primeiro o Android Studio para, só depois, instalar a JRE.
b) Devemos instalar primeiro o JDK, depois o Android Studio e, por fim, a JRE.
c) Devemos instalar primeiro a JRE, depois o SDK e, por fim, o Android Studio.
d) Devemos instalar primeiro o JDK, depois a JRE e, por fim, o Android Studio.
e) Devemos instalar primeiro a JRE, depois o Android Studio e, por fim, o SDK.

2. *Activity* é um componente de aplicativo que permite que as funcionalidades sejam implementadas. "Aplicativos geralmente têm várias *activities* pouco vinculadas entre si. Normalmente, uma *activity* em um aplicativo é especificada como 'principal', que é a apresentada ao usuário ao iniciar o aplicativo pela primeira vez. Cada *activity* pode, então, iniciar outra atividade para executar diferentes ações." (DEVELOPER, 2017, p. 1).

Considere as seguintes afirmações e a relação proposta entre elas.
I – Cada *activity* terá um arquivo com extensão Java (arquivo.Java) e um arquivo com extensão xml (arquivo.xml). Embora eles sejam dependentes, são implementados separadamente.

PORQUE

II – Dentro do arquivo com extensão Java programa-se o layout do aplicativo, enquanto no arquivo de extensão xml programa-se o comportamento da *activity*.

a) As afirmações I e II estão corretas e a segunda é uma justificativa válida da primeira.
b) A afirmação I está correta e a II está incorreta.
c) As afirmações I e II estão corretas, mas a segunda não é uma justificativa válida da primeira.
d) A afirmação I está incorreta e a II está correta.
e) As afirmações I e II estão incorretas.

3. Os aplicativos construídos no Android Studio podem ser testados tanto em um smartphone conectado, como no Android Emulator. Essa ferramenta pode ser instalada pelo próprio ambiente de desenvolvimento e simula um dispositivo físico em execução. Com o emulador é possível desenvolver e testar aplicativos para o sistema Android sem usar um dispositivo de hardware.

A respeito dos emuladores do sistema Android, escolha a alternativa correta.

a) Para cada versão do sistema operacional Android deve ser instalada uma imagem do sistema.
b) Para executar todas as versões do sistema Android é preciso instalar uma única imagem.
c) Para executar todas as versões do sistema Android é preciso instalar uma única imagem, porém a cada nova versão é preciso atualizar.
d) Ao escolher uma versão do sistema Android para simular no ambiente de desenvolvimento, as funcionalidades do aplicativo não irão funcionar nas versões anteriores.
e) Ao escolher uma versão do sistema Android para simular no ambiente de desenvolvimento não é mais possível alterá-la.

Seção 4.3

Desenvolvimento de um aplicativo para Android

Diálogo aberto

Caro aluno, chegamos à última seção do livro de tecnologias para web e para dispositivos móveis. Durante nosso percurso vários conceitos e tecnologias foram apresentadas trazendo a você a oportunidade de um diferencial no mercado.

Nessa última unidade já aprendemos um pouco sobre o mundo da programação orientada a objetos aplicados à linguagem Java. Já entendemos a função de três importantes arquivos que fazem parte da estrutura de aplicativo Android (MainActivity.java, activity_main.xml e Manifesto). Já vimos os passos necessários para a instalação do ambiente de desenvolvimento Android Studio, portanto, agora é hora de desenvolver seu primeiro aplicativo.

Dando continuidade ao seu projeto para o Pub, você deverá apresentar em uma *Activity* uma prévia do menu de comidas do pub. Levando em consideração a inclusão social, os sócios do Pub lhe solicitaram implementar um recurso que ao toque na tela a opção "clicada" seja falada pelo aplicativo, assim, deficientes visuais também poderão usufruir do menu.

Para que você possa desenvolver o menu, nesta seção iremos apresentar os recursos que permitem a inserção de textos e imagens no *layout*, além do recurso de voz para cliques na tela.

Bons estudos!

Não pode faltar

Toda vez que um programador inicia em uma nova linguagem de programação, faz parte da superstição criar o programa "hello world", portanto, não vamos contrariar a superstição! Faremos o desenvolvimento desta seção por meio de um aplicativo, no qual utilizamos caixa de texto, imagem e som. Portanto, iniciamos um novo projeto clicando em File > New > Project. Ao dar o comando na primeira tela, você poderá escolher (Figura 4.13):

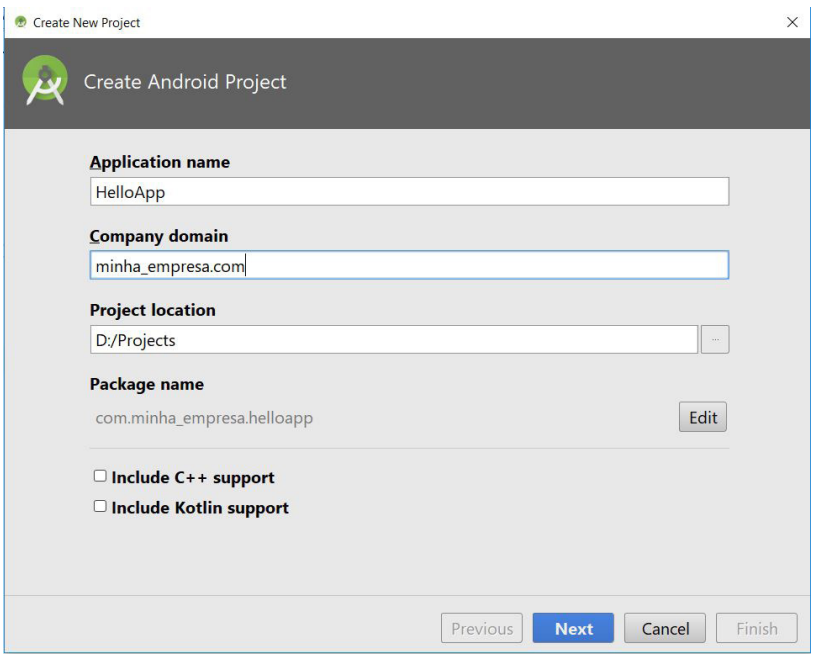
(I) Nome do projeto (*Application name*): Será o nome da sua aplicação, para nosso aplicativo, digite: *HelloApp*.

(II) Domínio da companhia (*Company domain*): Utilize o domínio de sua empresa, para nosso aplicativo, digite: minha_empresa.com.

(III) Localização do projeto (*Project location*): Escolha um local para que seu projeto seja salvo.

(IV) Nome do pacote (*Package name*): Esse campo é utilizado para identificar seu aplicativo dentro da *Google play*. O nome do pacote escolhido será o nome do pacote java onde estarão seus códigos-fontes. Esse valor não deve ser alterado em futuras versões do seu aplicativo, senão o *Google play* não conseguirá reconhecer a atualização. Existe um valor já criado para esse campo, composto por uma combinação dos campos (I) e (II). Primeiro vem o domínio da empresa seguido pelo nome do aplicativo.

Figura 4.13 | Captura de tela de novo projeto

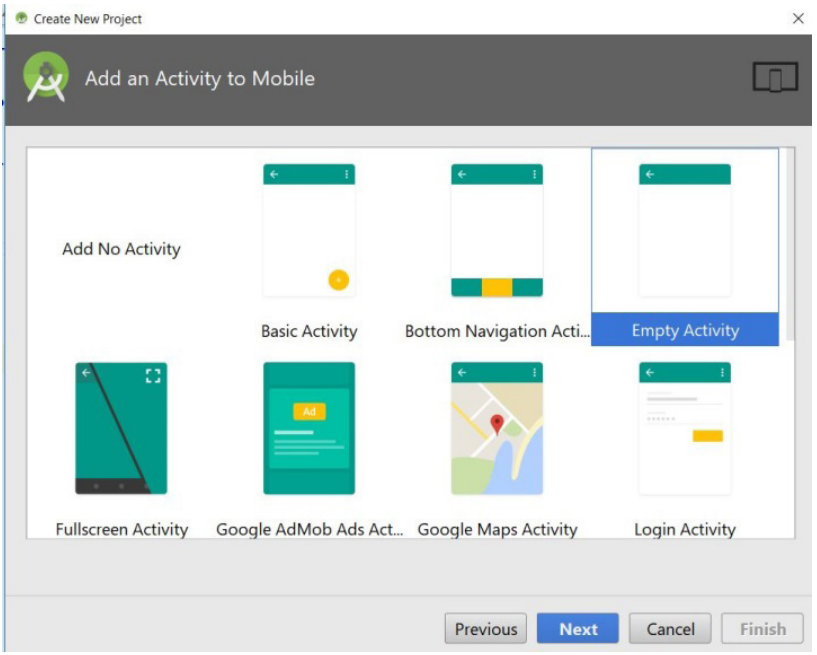


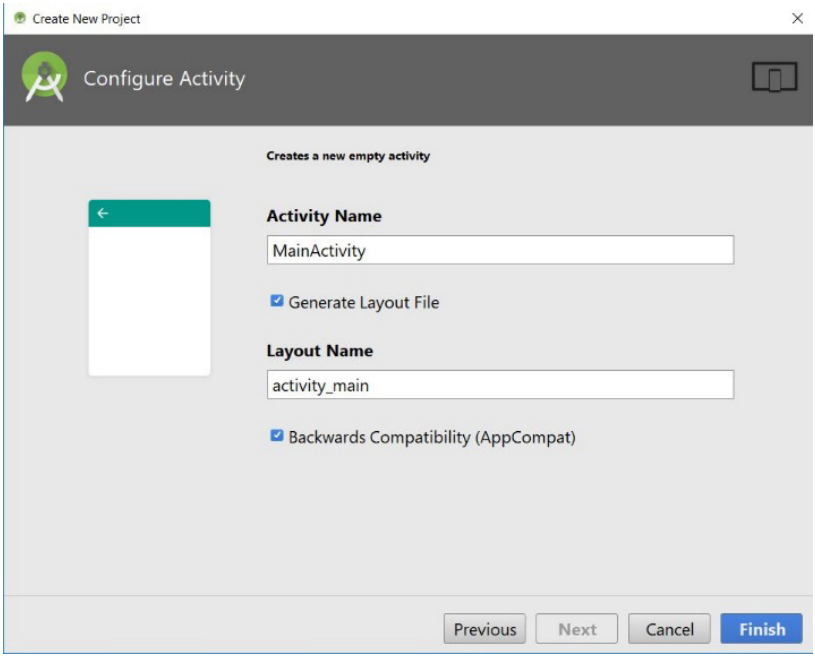
Fonte: elaborada pelo autor.

O passo seguinte consiste em escolher para qual dispositivo será o aplicativo, bem como a API mínima do Android. Para nosso app, escolha a API 23 (*Android 6.0 Marshmallow*). A escolha por APIs mais antigas permite que seu app funcione em um número maior de aparelhos, por outro lado, a cada nova versão são implementadas funcionalidades que podem não estar disponíveis nas anteriores. Então, você deve ter em mente: número de aparelhos x novas funcionalidades.

O próximo passo consiste na escolha do tipo de *activity*. Para nosso app, vamos escolher uma *activity* em branco e manter seus nomes originais, ou seja, *MainActivity* e *activity_main* (Figura 4.14).

Figura 4.14 | Adicionando uma nova *activity*



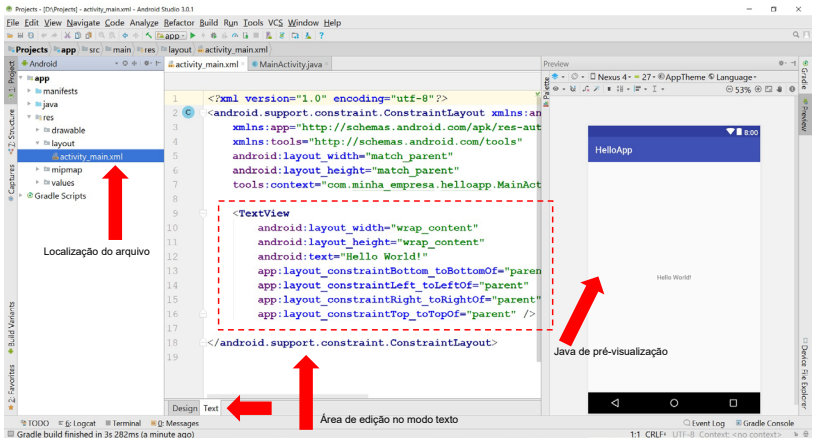


Fonte: elaborada pelo autor.

Ao criar o novo projeto, o ambiente abrirá com os arquivos da *activity* abertos, mas por padrão será exibido o *MainActivity.java*.

O ponto de partida do nosso aplicativo depende do *layout*, por isso vamos clicar no *activity_main.xml* e, com esse arquivo aberto, vamos acessar o menu *View* > *Tool Windows* > *Preview*. Nesse momento uma tela de pré-visualização é aberta com um texto no centro da tela, "*Hello World!*" (Figura 4.15). O texto é exibido através de um componente do *layout* chamado *TextView*. Da linha 9 a 17 está o código necessário para a inserção e configuração da posição do texto na tela. Como queremos escrever o nosso próprio app, apague esse conteúdo.

Figura 4.15 | Tela inicial de uma *activity* em branco



Fonte: elaborada pelo autor.

O comando *android.support.constraint.ConstraintLayout*, na linha 2, é utilizado para criar um *layout* em que a posição dos objetos seja feita de modo flexível (é bastante utilizado para criar *layouts* responsivos). Iremos alterar essa configuração no nosso projeto, de modo que o *layout* seja linear, com os elementos sendo alinhados verticalmente. Por isso vamos alterar o arquivo *activity_main.xml* para o código da Figura 4.16.

Figura 4.16 | *Layout* antes e depois da alteração

a) Usando *ConstraintLayout*

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context="com.minha_empresa.helloapp.MainActivity">
8
9
10
11 </android.support.constraint.ConstraintLayout>
```

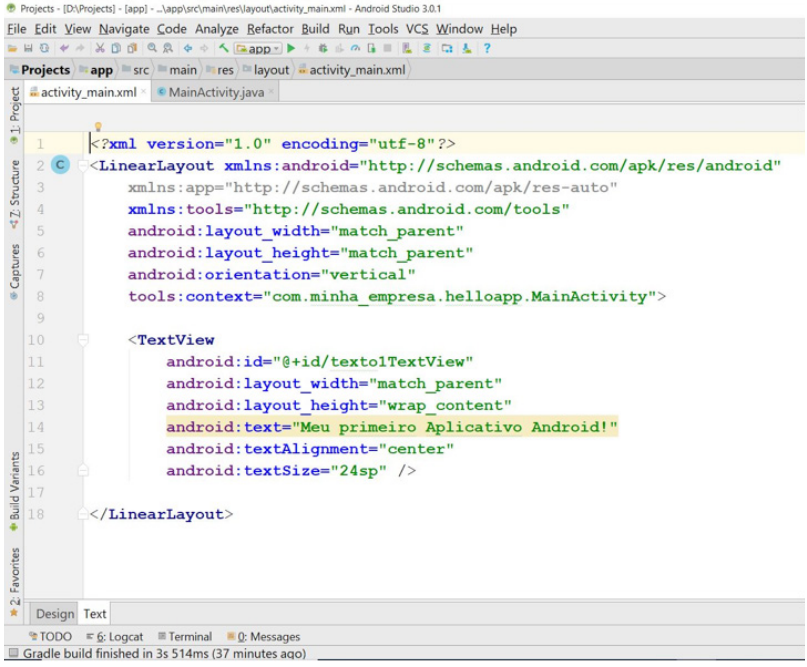
b) Usando *LinearLayout*

Fonte: elaborada pelo autor.

Mudamos na linha 2 para o *layout* linear, que pode resultar em um alinhamento vertical ou horizontal, por isso foi preciso especificar na linha 7 qual seria a orientação. Por fim, como a linguagem xml trabalha com *tags*, é preciso abrir e fechar, por isso também houve uma alteração na linha 11.

Agora iremos inserir nossa mensagem de boas-vindas, para isso vamos adicionar um componente *TextView*, digitando o código conforme Figura 4.17.

Figura 4.17 | Adicionando um *TextView* ao *layout*



Fonte: elaborada pelo autor.

Como é possível observar na Figura 4.17, usamos 7 parâmetros (linhas 10 a 16) para adicionar um único texto na tela do aplicativo, vamos entendê-los. Cada componente é inserido usando a tag correspondente. No caso de um texto, optamos por usar um *<TextView/>*. Dentro da tag devem vir especificados os parâmetros do componente. Dentre os parâmetros, dois são obrigatórios: *layout_width* e *layout_height*, o primeiro utilizado para especificar a largura e o segundo a altura do componente (DEVELOPER, 2017). Ao especificar a largura com o valor "match_parent", determinamos que seria do tamanho do componente que contém o texto, nesse caso, a própria tela. Ao especificar a altura com o valor "wrap_content", determinamos que a altura irá se adequar ao conteúdo.

Faça você mesmo

Altere os valores dos atributos *layout_width* e *layout_height* e veja as alterações que ocorrem no seu *layout*.

Os demais atributos são opções, entretanto, é de extrema importância que você sempre identifique os componentes com um nome (SILVA, 2016), para isso usa-se o comando da linha 11, no qual, após a barra, você pode especificar qualquer nome para seu componente.

Assimile

O nome de identificação de um componente deve ser único, não pode conter espaços nem acento. É uma boa prática de programação escolher nomes significativos e combiná-los com o próprio nome do componente. No nosso exemplo usamos texto1TextView, assim qualquer programador ao ver uma referência a esse componente saberá que se trata de um *TextView*.

No parâmetro *text* é digitado o texto que se pretende exibir na tela (linha 14). As linhas 15 e 16 são atributos apenas estéticos, formatamos o alinhamento do texto e o tamanho. Veja que usamos a unidade do tamanho em sp, isso significa "*pixel independente de escala*" e é a opção recomendada pela documentação oficial.

Exemplificando

Existem diversos controles que podem ser adicionados ao *layout*, vejamos alguns deles:

i. EditText: Caixa de texto utilizada para entrada de dados. Exemplos:

```
<EditText
  android:id="@+id/emailEditText"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:inputType="textEmailAddress" />

<EditText
  android:id="@+id/foneEditText"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:inputType="phone" />

<EditText
  android:id="@+id/senhaEditText"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:inputType="textPassword" />
```

ii. Button: Componente usado para criar botões com textos. Exemplo:

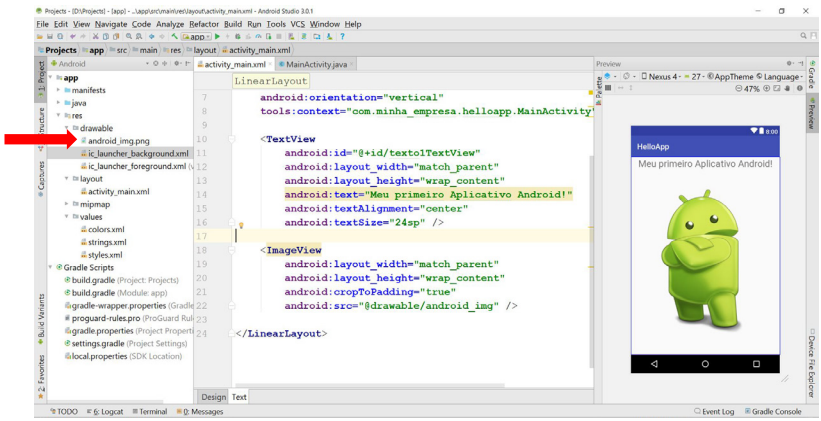
```
<Button
  android:id="@+id/simButton"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="Sim" />
```

iii. CheckBox: Componente usado para criar opções de seleção. Exemplo:

```
<CheckBox
  android:id="@+id/adicionalCheckBox"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
            android:text="Adicional" />
```

Agora vamos ver como inserir uma imagem na tela do aplicativo. O primeiro passo é copiar uma imagem do seu computador e colar na pasta *app* > *res* > *drawable*, isso pode ser feito tanto pelo ambiente do *Android Studio* quanto pelo *Explorer,* selecionando a pasta *drawable* no local que você escolheu para salvar seu projeto. Em seguida, no arquivo *activity_main.xml*, digite o código das linhas 18 até 22 conforme Figura 4.18.

Figura 4.18 | Inserindo uma imagem



Fonte: elaborada pelo autor.

O componente usado para adicionar imagens é o *<ImageView/>*, sendo necessário especificar seu tamanho com os atributos *layout_width* e *layout_height*, além de informar a fonte da imagem com o atributo *src* (abreviação de *source*). O atributo *cropToPadding* foi usado para que a figura fosse "cortada" para não ultrapassar os limites do seu preenchimento.

Reflita

Textos e imagens podem ser combinados para criar *layouts* sofisticados e atraentes para o usuário. Como as proporções entre esses dois componentes podem ser alteradas de modo que cada um ocupe metade da tela?

Existem outros atributos que podem ser usados para manipular as imagens, veja alguns deles:

- android:adjustViewBounds: Esse atributo é booleano, ou seja, aceita como valor true ou false. Quando configurado como true, ele ajuda a manter a razão de aspecto da imagem, em outras palavras, a imagem não é deformada conforme tamanho da tela.
- android:baselineAlignBottom: Esse atributo também é booleano e, quando ligado, alinha a imagem de acordo com a borda inferior.
- android:maxHeight: Atributo usado para especificar a altura máxima da imagem, aceitando como unidade px (pixel), dp (densidade da imagem), sp (pixel sem escala), in (polegadas) e mm (milímetros).
- android:maxWidth: Atributo usado para especificar a largura máxima da imagem e aceitar as mesmas unidades do atributo anterior.
- contentDescription: Esse atributo está relacionado à questão da acessibilidade do aplicativo. Ele é utilizado para atribuir um texto para ser lido quando o usuário tocar na imagem (DEITEL; DEITEL; WALD, 2016). Para isso, o usuário tem que estar com o *TalkBack* ativado no seu dispositivo (*Configurações > Acessibilidade > Talkback*).

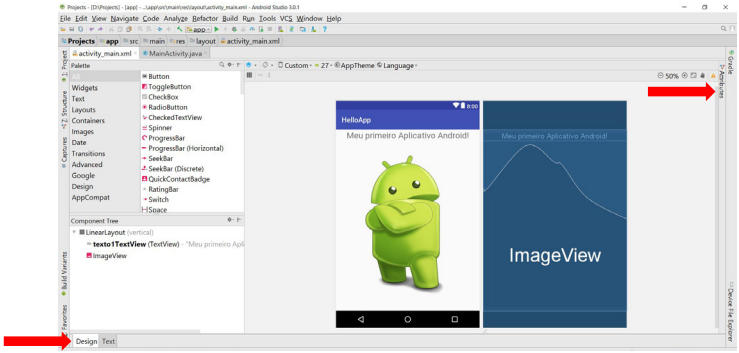
Para os componentes de texto, o *TalkBack* irá funcionar a partir do atributo *text*, por isso não é preciso especificar o *contentDescription*.

Pesquise mais

Caso ainda tenha dúvidas de como usar as imagens dentro do ambiente do Android Studio, assista ao vídeo e aproveite esse novo universo. Disponível em: <https://www.youtube.com/watch?v=3E8Ic4lZiE4>. Acesso em: 14 fev. 2018.

Para finalizar nosso livro, vamos ver uma maneira que facilita a criação de *layouts* no *Android Studio*. Até o momento, trabalhamos na criação do *layout* com comandos via linha de comando e, embora existam comandos que só podem ser adicionados dessa maneira, a inserção e disposição dos componentes pode ser feita usando o modo visual, ou como é mais comumente chamado "modo *design*". Para acessar a ferramenta basta clicar na aba Design, localizada na parte inferior da janela, conforme mostra a Figura 4.19.

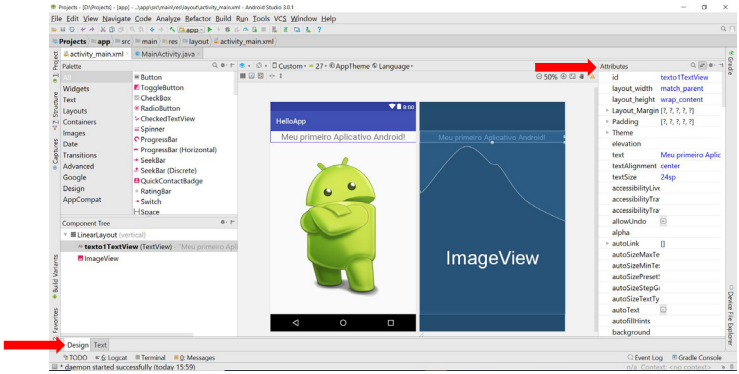
Figura 4.19 | Editor de layout no modo *Design*



Fonte: elaborada pelo autor.

Os modos texto e design podem ser alterados a qualquer momento, clicando em suas respectivas abas. Ao trabalhar no modo design também é possível configurar atributos via interface gráfica, acessando a janela de atributos conforme mostram as Figuras 4.19 e 4.20.

Figura 4.20 | Alterando atributos via interface gráfica



Fonte: elaborada pelo autor.

Aproveite os recursos e a facilidade do modo design e explore os componentes visuais do *Android Studio*. Com esta seção, demos alguns passos no caminho do desenvolvimento de aplicativos para dispositivos móveis, continue seus estudos nessa promissora área e tenha um grande diferencial no mercado de trabalho!

Sem medo de errar

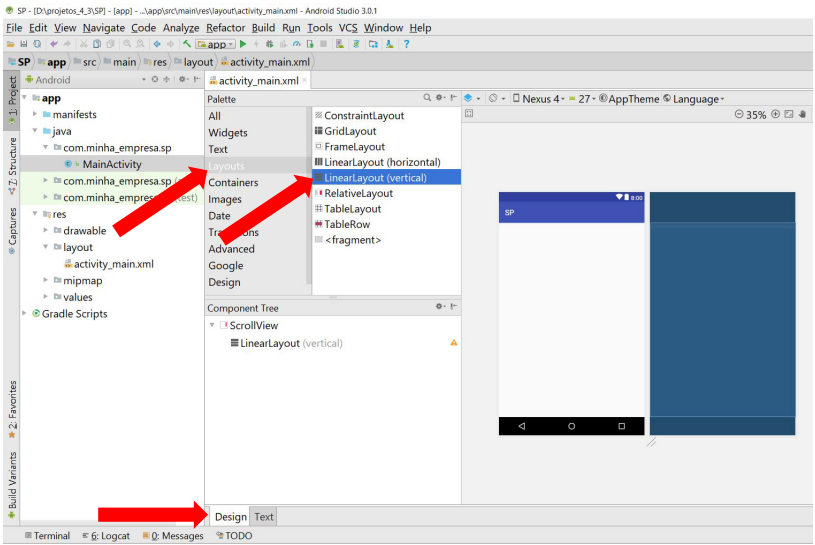
Chegou o momento de entregar um protótipo da tela do aplicativo para os sócios do Pub. Você viu que uma tela está associada a uma *activity*, que, por sua vez, está associada a um arquivo de *layout* com extensão xml.

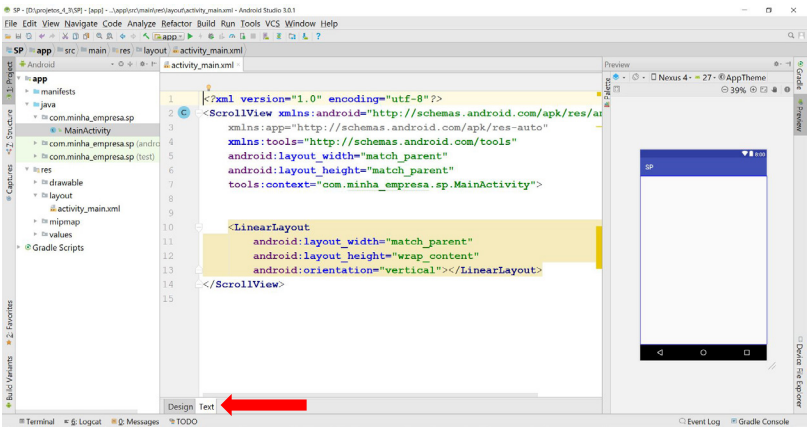
O primeiro passo para essa entrega é criar um projeto e não alterar o nome da *activity* padrão, portanto, criará a *activity_main.xml*.

O segundo passo é alterar a segunda linha do arquivo *activity_main.xml*, apagando o *android.support.constraint.ConstraintLayout* e digitando *ScrollView*. Esse componente permitirá que a tela deslize para cima e para baixo, conforme a quantidade de itens.

O terceiro passo é inserir um *LinearLayout* dentro do *ScrollView*. Você pode fazer esse passo usando o modo design. Dentro do menu *Layouts*, escolha a opção *LinearLayout(vertical)* e arraste para dentro da tela do aplicativo. Veja na figura 4.21 o modo design e o modo texto da tela até o momento.

Figura 4.21 | Formatação inicial da tela





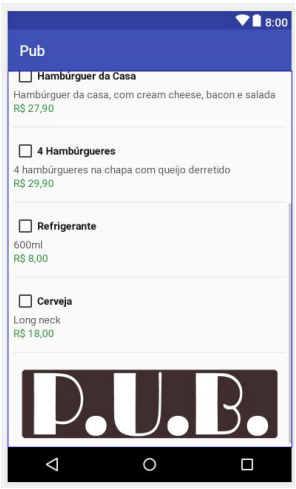
Fonte: elaborada pelo autor.

Use sua criatividade e explore os componentes usando o modo Design. Arraste os componentes para a tela e não se esqueça de atribuir um *id* para cada um dos componentes usados. Explore a formatação dos componentes (fonte, cor, tamanho etc.).

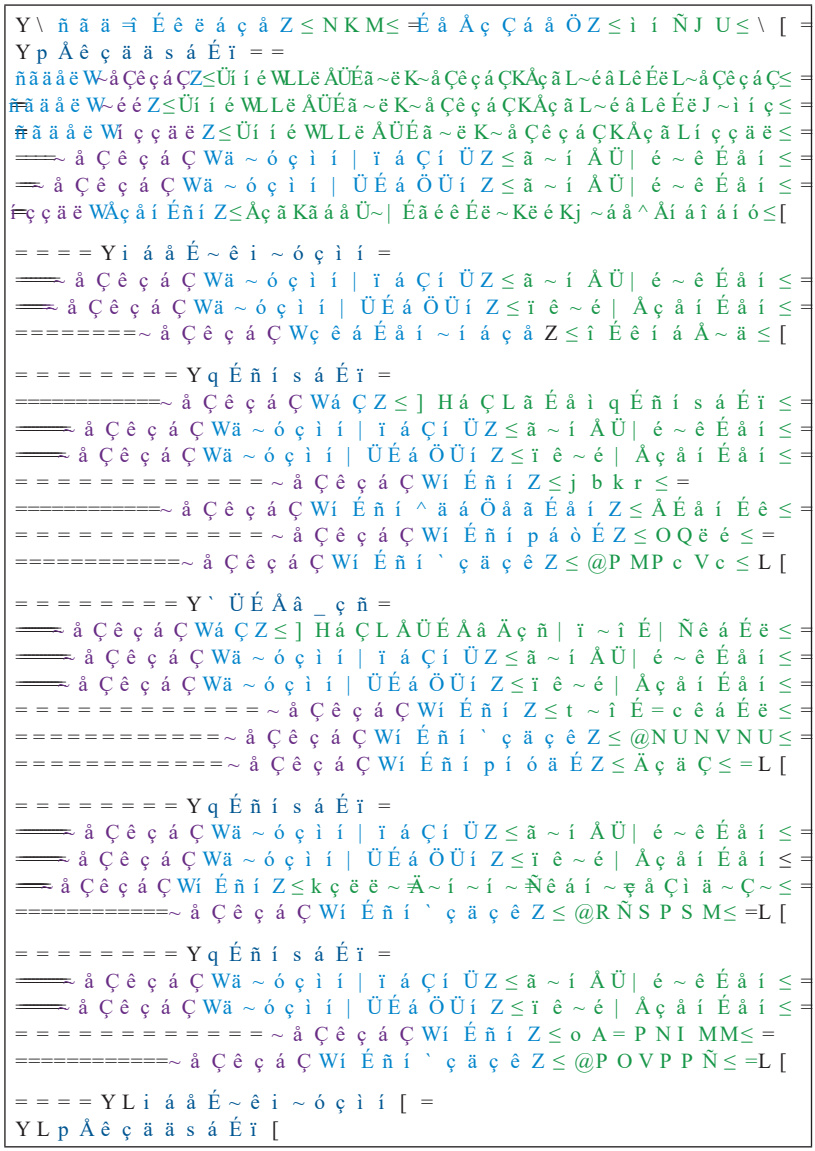
Para as imagens, você deve adicionar o atributo *contentDescription* a fim de que os deficientes visuais também usufruam dos recursos do aplicativo. Na Figura 4.22 – a está uma ideia para o menu; e na Figura 4.22 – b, parte do código.

Figura 4.22 | Ideia para o menu do Pub

a)



b)



Fonte: elaborada pelo autor.

Complete o código para o menu e aproveite o conhecimento adquirido nesta seção, aprofunde-o pesquisando sobre os recursos do *Android Studio*.

Avançando na prática

Trabalhando com eventos no Android Studio

Descrição da situação-problema

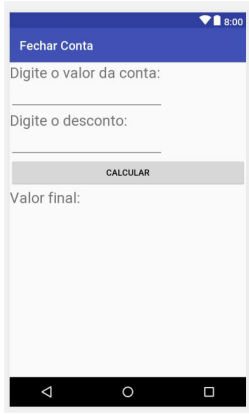
O dono de uma pizzaria lhe procurou para criar um aplicativo para Android para que seus funcionários pudessem fechar a conta dos clientes, aplicando um desconto ao valor total da conta. O desconto concedido varia conforme o cartão de fidelidade do cliente, sendo classificados em bronze (desconto de 7%), prata (desconto 10%) e ouro (desconto de 15%). Portanto, quando for até a mesa do cliente, o funcionário deve verificar qual é o cartão fidelidade e digitar no aplicativo o valor da conta e o respectivo desconto. Em seguida, o aplicativo exibirá o valor que o cliente deve pagar.

Resolução da situação-problema

Iremos trabalhar novamente com o *layout* linear, portanto no arquivo *activity_main.xml* é preciso alterar a segunda linha. Monte o *layout* conforme a Figura 4.23. Os componentes utilizados foram: um *EditText* para digitar o valor da conta e outro para digitar o desconto. Além deles, um botão para calcular e um *TextView* para exibir o valor a ser pago. Outros *TextView* foram usados apenas como rótulos de informações.

Figura 4.23 | Layout e código para o aplicativo

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.      xmlns:app="http://schemas.android.com/apk/res-auto"
4.      xmlns:tools="http://schemas.android.com/tools"
5.      android:layout_width="match_parent"
6.      android:layout_height="match_parent"
7.      android:orientation="vertical"
8.      tools:context="com.minha_empresa.calc_desc.MainActivity">
9.
10.     <EditText
11.         android:id="@+id/valorBrutoEditText"
12.         android:layout_width="500px"
13.         android:layout_height="wrap_content"
14.         android:inputType="numberDecimal" />
15.
16.     <EditText
17.         android:id="@+id/descontoEditText"
18.         android:layout_width="500px"
19.         android:layout_height="wrap_content"
20.         android:inputType="numberDecimal" />
21.
22.     <Button
23.         android:id="@+id/calcularButton"
24.         android:layout_width="match_parent"
25.         android:layout_height="wrap_content"
26.         android:text="Calcular"
27.         android:onClick="calcularDesconto" />
28.
29.     <TextView
30.         android:id="@+id/valorFinaltextView"
31.         android:layout_width="match_parent"
32.         android:layout_height="wrap_content"
33.         android:textSize="24sp"
34.         android:textColor="#228B22"/>
35.
36. </LinearLayout>
```



Fonte: elaborada pelo autor.

Na Figura 4.23 também estão trechos do código do arquivo *activity_main.xml* dos componentes que também serão utilizados no código-fonte java. Veja no código do botão, o parâmetro "*onClick*" foi usado no botão para ligá-lo a um evento que será trabalhado no arquivo com extensão java.

Na primeira linha do arquivo de extensão java estará a referência ao pacote java, o nome dependerá do que você informou no momento da criação do projeto. Você precisará importar bibliotecas para que os componentes funcionem (linhas 2 até 7). Também será preciso criar variáveis especiais para cada componente do *layout* que será utilizado (linhas 9 até 12). Precisará ligar as variáveis especiais aos seus respectivos componentes no *layout* (linhas 17 até 20). Por fim, criar o método que realiza o cálculo e exibe o resultado na tela (linhas 22 até 28).

```
N = = é ~ Åâ ~ ÖÉ ÅÇ ã K ã á å Ü ~ | É ã é ê É ë ~ K Å ~ ä Å | Ç É ë Å X

O = = á ã é ç ê í å Ç ê ç á Ç K ë ì é é ç ê í K î TK ~ é é K ^ é é ` ç ã é ~ í ^ Å í á î á í ó X
P = = á ã é ç ê í = ~ å Ç ê ç á Ç K ç ë K _ ì å Ç ä É X
Q = = á ã é ç ê í = ~ å Ç ê ç á Ç K î á É ï K s á É ï X
R = = á ã é ç ê í = ~ å Ç ê ç á Ç K ï á Ç Ö É í K b Ç á í q É ñ í X
S = = á ã é ç ê í = ~ å Ç ê ç á Ç K ï á Ç Ö É í K q É ñ í s á É ï X
T = = á ã é ç ê í = ~ å Ç ê ç á Ç K ï á Ç Ö É í K _ ì í í ç å X

U = = é ì Ä ä á Å Å ä ~ ë ë j ~ á å ^ Å í á î á í ó É ñ í É å Ç ë ^ é é ` ç ã é ~ í ^ Å í á î á í ó ê

V =     é ê á î ~ í É = b Ç á í q É ñ í = ã s ~ ä ç ê _ ê ì í ç X
N M =   é ê á î ~ í É = b Ç á í q É ñ í = ã a É ë Å ç å í ç X
N N =   é ê á î ~ í É = q É ñ í s á É ï = ã s ~ ä ç ê c á å ~ ä X
N O =   é ê á î ~ í É = _ ì í í ç å = ã ` ~ ä Å ì ä ~ ê X

N P =   ] l î É ê á Ç É
N Q =   é ê ç í É Å í É Ç í = ç á Ç ç å ` ê É ~ í É E _ ì å Ç ä É ë ~ î É Ç f å ë í ~ å Å É p í ~ í É F ê
N R =   =     ë ì é É ê K ç å ` ê É ~ í É E ë ~ î É Ç f å ë í ~ å Å É p í ~ í É F X
N S =   =     ë É í ` ç å í É å í s á É ï E o K ä ~ ó ç ì í K activity_main F X

N T =   =     ã s ~ ä ç ê _ ê ì í ç = Z =
=       =     = = Ñ á å Ç s á É ï _ ó f Ç E o K á Ç K valorBrutoEditText F X
N U =   =     ã a É ë Å ç å í ç = Z = =
= = Ñ á å Ç s á É ï _ ó f Ç E o K á Ç K descontoEditText F X
N V =   =     ã ` ~ ä Å ì ä ~ ê = Z = =
= = Ñ á å Ç s á É ï _ ó f Ç E o K á Ç K calcularButton F X
O M =   =     ã s ~ ä ç ê c á å ~ ä = Z = =
=       =     = = Ñ á å Ç s á É ï _ ó f Ç E o K á Ç K valorFinaltextView F X
O N =   õ

O O =   é ì Ä ä á Å ÷ ç á Ç Å ~ ä Å ì ä ~ ê a É ë Å ç å í ç E s á É ï ÷ á É ï F ô
O P =   =     Ñ ä ç ~ í = î ~ ä ç ê c á å ~ ä X
O Q =   =     Ñ ä ç ~ í = î ~ ä ç ê _ ê ì í ç = Z = =
=       =     ~ ä ç ~ í K parseFloat E ã s ~ ä ç ê _ ê ì í ç K Ö É í q É ñ í E F K
              í ç p í ê á å Ö E F F X
O R =   =     Ñ ä ç ~ í = Ç É ë Å ç å í ç = Z = =
=       =     ~ ä ç ~ í K é ~ ê ë É c ä ç ~ í E ã a É ë Å ç å í ç K Ö É í q É ñ í E F K
              í ç p í ê á å Ö E F F X
O S =   =     î ~ ä ç ê c á å ~ ä = Z = =
=       =     = = î ~ ä ç ê _ ê ì í ç = J = E î ~ ä ç ê _ ê ì í ç = G =
= = = E Ç É ë Å ç å í ç L N M M F F X
O T     ã s ~ ä ç ê c á å ~ ä K ë É í q É ñ í E p í ê á å Ö K î ~ ä ì É l Ñ Ê î ~ ä ç ê c á å ~ ä F F X
O U     õ
O V = = õ
```

Execute seu projeto em um emulador ou então conectando um celular com sistema Android ao computador.

Faça valer a pena

1. Um aplicativo Android é composto por atividades e para cada *activity* é gerado um arquivo xml e outro java. No arquivo xml é feita toda a configuração do *layout* e pode ser trabalhada tanto no modo texto como no gráfico. Ambos os modos possuem vantagens e desvantagens, cabe ao programador se apropriar de ambos os conhecimentos.

Considere as seguintes afirmações e a relação proposta entre elas.
I – Configurar um *layout* como LinearLayout é a melhor opção para os aplicativos.

PORQUE

II – Com essa opção os elementos são alinhados sequencialmente com uma orientação horizontal ou vertical, dependendo da escolha do programador.

a) As afirmações I e II estão corretas e a segunda é uma justificativa válida da primeira.
b) A afirmação I está correta e a II está incorreta.
c) As afirmações I e II estão corretas, mas a segunda não é uma justificativa válida da primeira.
d) A afirmação I está incorreta e a II está correta.
e) As afirmações I e II estão incorretas.

2. "Para quem está chegando de mundos um 'pouco' mais limitados em relação à interface gráfica para dispositivos móveis como, por exemplo, o cenário encontrado no desenvolvimento de aplicativos para aparelhos celulares mais antigos, a plataforma Android é um verdadeiro paraíso. Com dezenas de componentes visuais sofisticados, com efeitos e muitas características disponíveis para serem personalizadas, a plataforma Android está se tornando uma referência na quantidade e na qualidade dos componentes visuais existentes." (DEVMEDIA, 2017, p. 1)

A respeito dos componentes visuais de texto que podem ser usados no aplicativo, escolha a opção correta.

a) Um TextView só pode ser usado para exibir um texto.
b) Um TextView pode ser usado tanto para exibir um texto como para o usuário digitar valores.
c) Um EditText só pode ser usado para exibir um texto.
d) Um EditText pode ser usado tanto para exibir um texto como para o usuário digitar valores.
e) O atributo inputType do componente EditText é usado para especificar o tamanho do campo.

3. *Layouts* são compostos por diversos elementos gráficos, textos, imagens, caixas de seleção etc. Considere as afirmações a respeito de alguns componentes.
I – O atributo *contentDescription* é utilizado em imagens para adicionar um texto que será lido caso o usuário tenha o *talkback* ativado em seu celular.
II – O atributo *id*, utilizado para identificar os componentes, é um dos atributos obrigatórios.
III – Um componente *EditText* com o atributo *inputType*="numberDecimal" não aceita que o usuário digite valores diferentes do numérico.

Considerando as afirmações a respeito dos componentes visuais, escolha a opção correta.

a) Apenas a alternativa I está correta.
b) Apenas a alternativa III está correta.
c) Todas as alternativas estão corretas.
d) Apenas as alternativas I e II estão corretas.
e) Apenas as alternativas I e III estão corretas.

Referências

ARNOLD, K.; GOSLING, J.; HOLMES, D. **A linguagem de programação Java**. 4. ed. Porto Alegre: Bookman, 2007.

CASS, Stephen. **The 2017 Top Programming Languages**. Disponível em: <https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>. Acesso em: 25 jan. 2018.

DEITEL, Paul; DEITEL, Harvey; WALD, Alexander. **Android 6 para programadores:** uma abordagem baseada em aplicativos. Porto Alegre: Bookman, 2016.

DEVELOPER. **Atividades.** 2017. Disponível em: <https://developer.android.com/guide/components/activities.html?hl=pt-br>. Acesso em: 29 jan. 2018.

_____. **Introdução ao Android**. 2017. Disponível em: <https://developer.android.com/guide/index.html>. Acesso em: 15 fev. 2018.

DEVMEDIA. **Android Design**: Componentes visuais da plataforma Android. 2017. Disponível em: <https://www.devmedia.com.br/android-design-componentes-visuais-da-plataforma-android/31524>. Acesso em: 15 fev. 2018.

MACHADO, Henrique. **Os 4 pilares da Programação Orientada a Objetos**. Disponível em: <http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Acesso em: 30 ago. 2017.

MATTANO, G. **Entendendo a estrutura de um código Java**. Disponível em: <www.devmedia.com.br/entendendo-a-estrutura-de-um-codigo-java/24622#ixzz3GmnOTDLM>. Acesso em: 31 jan. 2018.

MULLIS, Alex. **Android Studio tutorial for beginners.** 2017. Disponível em: <https://www.androidauthority.com/android-studio-tutorial-beginners-637572>. Acesso em: 29 jan. 2018.

ORACLE. **Primitive Data Types**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>. Acesso em: 31 jan. 2018.

SANTOS, Rafael. **Introdução à programação orientada a objetos usando Java**. Rio de Janeiro: Campus, 2003.

SILVA, Diego (org.). **Desenvolvimento para dispositivos móveis.** São Paulo: Pearson, 2016.

THE UNIVERSITY OF TENNESSEE. **A Brief History of Object-Oriented Programming**. Disponível em: <http://web.eecs.utk.edu/~huangj/CS302S04/notes/oo-intro.html>. Acesso em: 31 de ago de 2017.

Anotações

Anotações

Anotações

ISBN 978-85-522-0732-0