FUNDAMENTOS PARA COMPUTAÇÃO

CAPÍTULO 2 - DE QUE MANEIRA OS PROGRAMAS SÃO EXECUTADOS EM UM SISTEMA COMPUTACIONAL?

Fábio Tavares Arruda

INICIAR

Introdução

Você já deve ter percebido que há computadores por todos os lugares, seja em casa, no trabalho, na escola, na faculdade, na padaria, nos supermercados, nos shoppings etc. Para que eles funcionem, há vários programas sendo executados. Eles, computadores e programas, têm o objetivo de resolver problemas ou diminuir o tempo de alguma atividade realizada pelo ser humano. Para que possamos utilizar o computador e seus programas precisamos de um programa específico para nos auxiliar, esse programa se chama sistema operacional. Então, neste capítulo, vamos estudar de que maneira os programas são executados em um sistema computacional.

Para iniciar este estudo, você fará algumas reflexões importantes para que possa integrar esses conceitos de como um sistema computacional funciona. Como podemos definir um sistema computacional? Quais as vantagens que o uso da

internet pode nos trazer? Quais as formas de acesso à internet? O que são redes de computadores? Como podemos criar um programa de computador? O que são linguagens de programação e quais suas características? Qual o papel dos sistemas operacionais?

A partir dessas reflexões, neste capítulo você estudará o conceito de sistemas operacionais: o que são e como podem ser aplicados. Estudará também os conceitos iniciais de linguagens de programação, suas ferramentas e técnicas. Além disso, será abordada a aplicação de sistemas de computação e suas aplicações práticas. Por fim, serão abordados os conceitos iniciais de redes de computadores e meios de acesso à internet.

2.1 Sistemas operacionais

A interação do ser humano com os computadores nem sempre foi como é hoje. O que mudou desde os primeiros computadores criados? Podemos perceber que, ao longo do tempo, os sistemas responsáveis por fazer a comunicação entre o usuário e o *hardware* do computador, os sistemas operacionais, estão sempre em rápida e constante evolução.

Um exemplo bem conhecido de sistema operacional é o Windows, que foi o responsável pela popularização do computador pessoal. Através dele, usuários não programadores começaram a utilizar o computador para resolver problemas do dia a dia. Com o avanço da tecnologia, os sistemas operacionais para *smartphones* também se popularizaram bastante e hoje são bem conhecidos, como o Android, criado pela empresa norte-americana Google, e o iOS, presente nos iPhones da também empresa norte-americana Apple.

VOCÊ SABIA?

Que o Windows 1.0 foi lançado em novembro de 1985? A primeira versão do Windows tinha apenas o jogo Reversi, um calendário, bloco de notas, calculadora, relógio, um prompt de comando, o Write, o Painel de Controle, o Paint e programas de comunicação. Ficou curioso para saber mais? Acesse: <https://canaltech.com.br/windows/do-windows-10-ao-windows-10-veja-como-o-sistema-mudou-nestes-30-anos-45911/ (https://canaltech.com.br/windows/do-windows-10-ao-windows-10-veja-como-o-sistema-mudou-nestes-30-anos-45911/)>.

Para usuários comuns de computador, pessoas que os utilizam para resolver seus problemas, as diferenças entre os vários sistemas operacionais são, em sua maioria, estéticas, ou seja, as diferenças consistem apenas na aparência. Para profissionais da computação, sistemas operacionais diferentes possuem características internas distintas e podem impactar diretamente no trabalho e nos programas utilizados.

2.1.1 Definições e conceitos básicos

Quando se pensa em sistemas operacionais é importante ter conhecimento do que é um programa de computador ou *software*, pois o sistema operacional é um tipo de *software*. De acordo com Fedeli, Polloni e Peres (2013), os *softwares* são conjuntos de instruções ou comandos dados a um computador para que ele execute tarefas como, por exemplo, impressão de relatórios, armazenamento de informações, cálculos matemáticos, transmissão de informação ou mostrando informações em um periférico de saída.

Como podemos definir um sistema operacional? De acordo com Brookshear (2013), é o sistema de *software* que controla a operação geral de um computador. Ele fornece os meios para que um usuário possa armazenar e obter arquivos, bem como requisitar a execução de programas.

Você já deve ter percebido que os sistemas operacionais trabalham para estabelecer a comunicação entre as aplicações (também chamados de *softwares* aplicativos) e o *hardware* (a parte física do computador). Além disso, eles são *softwares* complexos, pois trabalham tanto com aspectos de baixo nível (gerenciamento de memória e de dispositivos de entrada e/ou saída) como aspectos de alto nível (*browsers* para acesso à internet, editores de texto, entre outros).

Além da função citada anteriormente, os sistemas operacionais possuem outros aspectos e funções, como gerenciar recursos de *hardware* (processador, memória, entrada e saída), abstrair a camada de *hardware* para facilitar a programação e gerenciar arquivos. Mas como são feitos esses gerenciamentos de recursos?

Figura 1 - Funções do sistema operacional: utilizas para controlar o hardware do computador. Fonte: Elaborada pelo autor, 2018.

Antes de entendermos o gerenciamento do processador, precisamos saber o que é um processo. Todo *software* executável em um computador é organizado em diversos processos sequenciais ou apenas processos. Assim, um processo é simplesmente um programa em execução. (TANENBAUM; MACHADO FILHO, 2008)

Nesse contexto, para realizar o gerenciamento do processador, o sistema operacional possui recursos para dar ao usuário a ilusão de que os processos são executados de forma simultânea no computador. Cada processo é executado em uma parte do tempo e a alternância entre vários processos é tão rápida que o usuário pensa que a execução é realizada de forma simultânea. Caso o computador

possua mais de um processador, ou seja, tenha uma arquitetura multiprocessador, o conjunto de processos podem ser executado em diferentes processadores. Como o sistema operacional sabe qual processo executar e durante quanto tempo? O sistema operacional utiliza um tipo de algoritmo que vai alternar a execução dos processos. Eles são chamados de algoritmos de escalonamento e são utilizados para determinar qual processo deve ser executado em determinado momento e por quanto tempo.

O gerenciamento de memória é responsável por controlar o acesso e uso da memória principal do computador. Como os programas precisam dela para armazenar dados, há a necessidade de controlar esse armazenamento, pois o espaço necessário pode exceder o disponível, pode haver células de memória que não podem ser utilizadas porque já possuem dados armazenados, além de ser necessário identificar as células de memória vazias. (TANENBAUM; MACHADO FILHO, 2008)

Como é feito o gerenciamento de entrada e saída? Nesse caso, o sistema operacional irá controlar o acesso a dispositivos de entrada e saída. Para isso, ele possui uma coleção de controladores que são *softwares* capazes de se comunicar com os controladores de *hardware* dos dispositivos de entrada e saída. Um exemplo disso é quando precisamos utilizar uma impressora, o sistema operacional se comunica com ela através de *softwares* controladores e envia comandos para os controladores de *hardware* da impressora. (BROOKSHEAR, 2013)

Figura 2 - Os diretórios agrupam os arquivos semelhantes, assim como as pastas de documentos. Fonte: ESB Professional, Shutterstock, 2018.

No gerenciamento de arquivos, o sistema operacional deve coordenar o armazenamento de dados no computador. Com isso, ele guarda o registro de todos os arquivos armazenados, o local onde eles estão armazenados e o tipo do arquivo. Os arquivos também podem ser agrupados em pastas ou diretórios. (BROOKSHEAR, 2013)

2.1.2 Aplicações

Os sistemas operacionais podem ser classificados de acordo com vários aspectos, sendo um deles em relação à quantidade de usuários, como os monousuários que só permitem um usuário por vez para realizar as tarefas no sistema. Já os multiusuários permitem que mais de um usuário execute tarefas simultaneamente. Outra classificação para os sistemas operacionais é pelo número de tarefas, como os monotarefas, que permitem executar uma tarefa por fez; já os multitarefas permitem a execução de mais de uma tarefa “simultaneamente”. (BROOKSHEAR, 2013)

Temos também sistemas operacionais distribuídos, os quais rodam em mais de um computador, ou seja, vários computadores separados fisicamente rodando um único sistema operacional como se fosse em um único computador. Já os de tempo real são sistemas operacionais utilizados em situações críticas, em que há necessidade de confiabilidade e prazos compatíveis com a ocorrência de eventos externos. (FEDELI, POLLONI e PERES, 2013)

Atualmente existem diversos sistemas operacionais disponíveis no mercado, seja para computadores pessoais ou para *smartphones*. O mais popular para computadores pessoais é o Microsoft Windows, sendo que este é um sistema proprietário, ou seja, o seu código pertence e é mantido pela Microsoft. A primeira versão foi o Windows 1.0, lançada em 1985, e a mais atual é o Windows 10, lançada em 2014.

Já o Linux, criado por Linus Torvalds, é um sistema operacional livre e de código aberto, baseado no Unix, um sistema operacional multitarefa e multiusuário que tem a vantagem de rodar em uma grande variedade de computadores. Por ser de código aberto, qualquer desenvolvedor pode utilizar seu código e criar sua própria distribuição do Linux. Portanto, existem várias distribuições do Linux como o Ubuntu, Debian, Fedora etc.

Embora também seja baseado no Unix, o MacOS é um sistema operacional proprietário, desenvolvido e mantido pela empresa Apple. Uma característica particular do MacOS é que ele foi desenvolvido para ser utilizado exclusivamente nos computadores da Apple, os chamados computadores Macintosh. A primeira versão, chamada de MacOS Classic, foi lançada em 1984.

VOCÊ QUER VER?

O filme *Ela* conta a história de um escritor que acaba de comprar um novo sistema operacional para seu computador. Para a sua surpresa, ele acaba se apaixonando pela voz do programa, dando início a uma relação amorosa entre ambos. A história explora a relação entre o homem da atualidade e a tecnologia. Assista: <https://www.youtube.com/watch?v=SjZudKgH_OY (https://www.youtube.com/watch?v=SjZudKgH_OY)> (https://www.youtube.com/watch?v=SjZudKgH_OY).

Com o enorme crescimento de usuários de *smartphones*, o Android passou a ser o líder do mercado de sistemas operacionais móveis. Ele é baseado no núcleo do Linux, de código aberto, e desenvolvido pela Google. Assim, cada fabricante de aparelhos pode adaptá-lo para funcionar em *hardware* específico ou pode também utilizá-lo sem modificações.

2.2 Linguagens de programação

Após refletir sobre como podemos utilizar o computador através dos sistemas operacionais, pense em outros programas que podemos utilizar no computador e quais problemas eles resolvem. No dia a dia, nos deparamos com diversos programas de computador, seja no trabalho, na escola, na faculdade, nos supermercados ou farmácias. Em quais outros lugares podemos utilizá-los? Por que eles são tão importantes? Você já se perguntou como esses programas são criados?

Um algoritmo é um conjunto lógico de instruções definidas para resolver um problema específico (FEDELI; POLLONI; PERES, 2013). De forma intuitiva, já utilizamos algoritmos na vida diária para realizar nossas tarefas. Em uma linguagem mais simples, um algoritmo é um conjunto de passos para resolver um problema bem específico. Um exemplo utilizado no dia a dia é o algoritmo para tomar banho, quando realizamos os seguintes passos: 1º tirar a roupa, 2º entrar no *box*, 3º ligar o chuveiro, 4º passar o sabonete, 5º tirar o sabonete, 6º desligar o chuveiro, 7º se enxugar, 8º sair do *box*.

Para que um algoritmo seja executado em um computador, é necessário que ele seja escrito em uma linguagem de programação. O código é convertido (as instruções) em instruções binárias para que o computador entenda o que deve ser feito. Existem inúmeras linguagens de programação no mercado, como, por exemplo, C, Python, Pascal, PHP e Java, cada uma com suas características, propósitos, vantagens e desvantagens.

2.2.1 Conceitos iniciais

Um programa sequencial é um conjunto de rotinas programáveis executadas em sequência, ou seja, uma após a outra. De forma resumida, podemos organizar essas rotinas em três etapas: entrada, processamento e saída de dados (MANZANO; LOURENÇO; MATOS, 2016). Na entrada, o programa recebe os dados necessários para executar o algoritmo, eles podem ser enviados por um usuário, por um outro

sistema, ou por uma função do próprio programa. Após o recebimento dos dados, é possível realizar seu processamento, que nada mais é do que realizar uma transformação dos dados recebidos, seja através de operações lógicas ou aritméticas. Por fim, é realizada a etapa de saída, onde o computador disponibiliza o resultado do processamento dos dados.

Figura 3 - Etapas básicas que um programa deve seguir. Fonte: Elaborada pelo autor, 2018.

Ao realizar atividades no mundo real, nos deparamos com diversos tipos de dados. Quais os tipos que você consegue identificar nas suas atividades? Veja que temos os tipos primitivos de dados, ou seja, são aqueles definidos na especificação da linguagem de programação, os quais se dividem em numéricos, caracteres e booleanos. Os tipos numéricos representam os números e se dividem em real e inteiro, assim como na matemática. O caractere representa um único caractere que é escrito entre aspas, se houver mais de um caractere entre as aspas chamamos de cadeia de caracteres ou *string*. Os booleanos se caracterizam por poder ter dois valores, verdadeiro e falso.

No contexto das linguagens de programação, há também as palavras reservadas da linguagem, as quais não podem ser utilizadas pelo programador para outro fim a não ser aquele que foi definido. Algumas delas possuem a característica de poder armazenar dados, isso pode ser feito basicamente utilizando variáveis e constantes. Ao armazenar um valor em uma variável, esse valor pode ser alterado inúmeras vezes ao longo da execução do programa, entretanto, ao armazenar um valor em uma constante, esse valor não pode ser alterado durante a execução do programa. (MANZANO, LOURENÇO; MATOS, 2016)

Ao especificar o algoritmo de um programa, podem ser realizadas operações aritméticas, relacionais e lógicas. As operações aritméticas que podem ser realizadas são as operações de soma, subtração, multiplicação, divisão, resto da divisão, entre outras. As relacionais são realizadas utilizando os comandos de igual, diferente, maior que, menor que, maior ou igual, menor ou igual. Por fim, as lógicas são realizadas utilizando os operadores lógicos disjunção, conjunção e negação.

Em algumas situações, é necessário que o programa verifique qual bloco de código deve ser executado através de desvios condicionais, os quais podem ser utilizados de três formas: através do comando “se então”, através do comando “se então...senão então” e através do comando “caso”.

O comando “se então” executa o bloco de código dentro dele caso a condição do “se” seja verdadeira, se for falsa, o bloco de código não é executado. O comando “se então...senão então” verifica se a condição após o “se” é verdadeira, se for, ele executa o bloco de código após o “então”, se for falsa, ele executa o bloco de código após o “senão então”. Os comandos anteriores podem ser utilizados de forma encadeada quando é necessário verificar várias condições. Já o comando “caso” é utilizado quando uma condição precisa ser avaliada a partir de múltiplos valores e o uso de estruturas de decisão encadeadas pode ser trabalhoso.

Figura 4 - Exemplo de como um programa é escrito através de linhas de código. Fonte: isak55, Shutterstock, 2018.

Em outras situações, precisamos que determinado bloco de código seja executado mais de uma vez. Para isso, utilizamos os comandos de repetição, que podem ser de três formas: “enquanto...faça” (modo pré-teste), “repita...até” (modo pós-teste) e a instrução “para...de...até...passo...faça”. O comando de repetição pré-teste faz um

teste lógico no início da execução do comando para verificar se deve ou não executar o respectivo bloco de código. Enquanto a condição for verdadeira, o respectivo bloco de código será executado. Assim que a condição se tornar falsa, o programa deixará de executar o respectivo bloco de código e seguirá o caminho de execução sequencial do programa. O mesmo acontece com o pós-teste, a diferença é que, neste comando, o teste lógico acontece somente após a execução do respectivo bloco de código.

Por fim, temos o comando de repetição “para...de...até...passo...faça”, que não depende da análise de uma condição lógica, mas sim de uma contagem de valores que, ao atingirem determinado valor, interrompem o comando de repetição. Note que nesse caso o comando de repetição funciona com base em uma variável que controla a quantidade de iterações.

VOCÊ SABIA?

Podemos salvar o código dos nossos projetos na **nuvem**? E, além disso, é possível ter um histórico do que foi alterado no projeto, compartilhar o código com outros programadores e ter um controle de versão do projeto? Essas ferramentas permitem que tenhamos acesso ao projeto em qualquer computador ligado à internet. Uma das ferramentas que possuem essas características é o Git, que é um *software* livre e gratuito. Acesse: <https://git-scm.com/ (https://github.com/)>.

Além das variáveis e constantes que vimos anteriormente, existem duas outras estruturas para armazenar dados durante a execução do programa, que são os vetores e as matrizes. Essas estruturas permitem armazenar um conjunto de valores durante a execução do programa, além de que esses valores precisam ser do mesmo tipo de dados, por exemplo, um conjunto de números inteiros representados por 02, 10, 20 e 100. Assim, cada valor armazenado ocupa uma posição específica representada por um índice. Os vetores possuem uma única dimensão, como se fosse uma lista de valores. E as matrizes possuem duas dimensões, como se fossem uma tabela. (BROOKSHEAR, 2013)

2.2.2 Ferramentas e técnicas

Python é uma linguagem de programação de alto nível que possui características do paradigma orientado a objetos, paradigma imperativo e paradigma funcional. Uma das vantagens dessa linguagem é que ela permite ler o código facilmente, além de exigir poucas linhas de código em comparação a outras linguagens de programação. Por isso, vamos começar a programar e a entender códigos utilizando o Python.

VOCÊ QUER VER?

O filme *Os Estagiários* (2013) conta a história de dois grandes amigos que trabalham juntos como vendedores de relógios. Eles são pegos de surpresa quando seu chefe fecha a empresa. Precisando de um novo emprego, eles fazem a inscrição em uma seleção de estágio no Google. Mesmo sem terem a garantia de que serão contratados, eles viajam para a sede da empresa e lá precisam realizar atividades relacionadas ao desenvolvimento e manutenção de programas de computador, além de lidar com a diferença de idade com os demais estagiários.

Se você estiver usando versões recentes dos sistemas operacionais Linux ou MacOS X, o interpretador da linguagem Python já vem instalado. Se você estiver usando o sistema operacional Windows, acesse <https://www.python.org> para fazer o *download*. Assumindo que você está utilizando o Windows, depois de fazer o *download* e instalar o interpretador da linguagem, basta executar o IDLE (interpretador baseado em janelas). A partir disso, você já pode começar a escrever seus códigos em Python.

Para começar a se familiarizar com a linguagem, você pode executar comandos aritméticos simples como:

- 6 + 23 + 56
- (87 – 9) * 4
- (9 - (1 + 2)) / 3.0

Para armazenar dados em uma variável, precisamos atribuir um identificador a ela, ou seja, um nome. Podemos criar as seguintes variáveis:

- soma = 0
- media = 0

- numero = 7
- aluno1 = “Lucas da Silva”

Se você quiser imprimir algo no prompt de comando do IDLE basta escrever o comando *print* e digitar entre aspas o texto que deseja imprimir. Se precisa imprimir o valor que está armazenado em uma variável, basta colocar o nome da variável entre parênteses (como no exemplo anterior), veja:

- print “Aprendendo a programar”
- print “Programando em Python”
- print (num)

Para receber dados de entrada do usuário, podemos utilizar o comando “input”. Neste comando, definimos uma variável que vai receber o valor digitado pelo usuário, seguido do sinal de atribuição (=), seguido da palavra *input* e de um texto explicativo (entre parênteses e entre aspas) para indicar o que o usuário deve digitar. Veja como utilizar esse comando:

- num = input("Digite um número:")
- nota = input("Digite uma nota:")
- valor = input("Digite o valor do dólar:")

Em Python, podemos utilizar os comandos “se então” e “se então...senão então” através dos comandos “if condição:” e “if condição: ... else:”. Mas não tem o comando “caso”, então quando for necessário analisar múltiplas condições, é necessário utilizar os comandos anteriores de forma aninhada através do comando “if condição: ... elif condição: ... else:”.

Também temos os comandos de repetição “enquanto...faça” e “para...de...até...passo...faça”. O comando “para...de...até...passo...faça” pode ser utilizado usando a palavra “for”. E o comando “enquanto...faça” pode ser utilizado usando a palavra “while”.

Agora, vamos analisar alguns programas em Python para entender o que cada linha de código faz. Vamos começar com um programa que recebe notas de um aluno e calcula a média aritmética dessas notas.

```
nota1 = input ("Digite a primeira nota do aluno:")

nota2 = input ("Digite a segunda nota do aluno:")

media = (float(nota1) + float(nota2))/ 2
```

```
print (media)
```

Agora, vamos adicionar comandos de desvio condicional a esse código. Como podemos ver o programa verifica (na linha 4) se a nota é maior ou igual a 7, se for verdadeiro ele imprime na tela que o aluno foi aprovado, se for falso ele imprime na tela que o aluno foi reprovado.

```
nota1 = input ("Digite a primeira nota do aluno:")
nota2 = input ("Digite a segunda nota do aluno:")
media = (float(nota1) + float(nota2))/ 2
if media >= 7.0:
        print("Aluno aprovado. Parabéns!")
else:
        print("Aluno reprovado. Precisa estudar mais!")
print (media)
```

Como podemos utilizar as estruturas de repetição em Python? Vamos ver: temos a definição de um vetor de anos, na segunda linha temos uma estrutura de repetição “para...de...até...passo...faça” que em Python é representada pelo comando for. E, na última linha, temos o código que será executado enquanto tivermos valores no vetor.

```
alunos = ['João', 'Maria', 'Gabriel', 'Carolina']
for n in alunos:
print(n)
```

Para utilizar o comando “while”, atribuímos uma variável chamada contador e definimos uma estrutura de repetição que vai mostrar na tela o valor da variável “contador” enquanto o valor dessa variável for menor que 10, além disso, sempre que a condição for verdadeira incrementamos o valor da variável “contador”.

```
contador = 0
```

```
while contador < 10:
    print(contador)
    contador = contador + 1
```

2.3 Aplicação de sistemas de computação

Após entender o que é um programa de computador e como podemos criá-los, você acha que existem tipos de *software*? E como podemos pensar na estrutura de um programa ao nos deparar com a solicitação de um cliente? O processo de desenvolvimento de *software* não é semelhante à criação de qualquer outro produto, como um carro, uma peça de roupa uma televisão. Apesar de possuir métodos bem definidos, o processo não acontece da mesma forma para programas diferentes.

Nesse contexto, a engenharia de *software* é o ramo da ciência da computação que busca princípios, métodos e ferramentas para guiar o desenvolvimento de sistemas de *software* grandes e complexos. (BROOKSHEAR, 2013)

Um sistema de computação é um conjunto de componentes eletrônicos que processam informações através da utilização de *hardware* e *software*. Eles possuem três componentes: *hardwar*e, *software* e *peopleware*. O primeiro representa toda a parte física do computador, ou seja, as peças, o monitor, o teclado, o *mouse* etc. O segundo representa a parte lógica, ou seja, todos os programas que são executados sob o *hardware*. Por fim, o *peopleware* são as pessoas envolvidas no funcionamento do sistema, seja de forma direta ou indireta.

2.3.1 Conceitos iniciais

Como já vimos antes, o computador utiliza o sistema operacional para estabelecer a comunicação entre as aplicações e o *hardwar*e. Então, você deve ter percebido que existem diferentes tipos de programas. Vamos começar dividindo os programas de um computador em duas categorias: *software* de aplicação e *software* de sistema. (BROOKSHEAR, 2013)

Os de aplicação são os programas que utilizamos para executar uma tarefa específica como, por exemplo, programas de planilhas eletrônicas, editores de texto, gerenciadores de *e-mail*, editores de imagem, entre outros. Já os de sistema

realizam tarefas para promover a infraestrutura necessária para que os *softwares* aplicativos possam realizar suas funções. (FEDELI; POLLONI; PERES, 2013)

Os *softwares* de sistema podem ser de dois tipos: operacionais e utilitários. Como já vimos, os operacionais oferecem uma *interface* para que possamos solicitar que o *hardware* execute alguma função, além de gerenciar os recursos do computador. Por outro lado, os utilitários permitem a realização de tarefas fundamentais do computador, mas que não estão incluídas no sistema operacional, ou seja, eles estendem as funções dos sistemas operacionais. Exemplos de *softwares* utilitários são os para compactação de dados, limpeza de discos rígidos, acesso à internet, melhoria do desempenho do computador etc. (FEDELI; POLLONI; PERES, 2013)

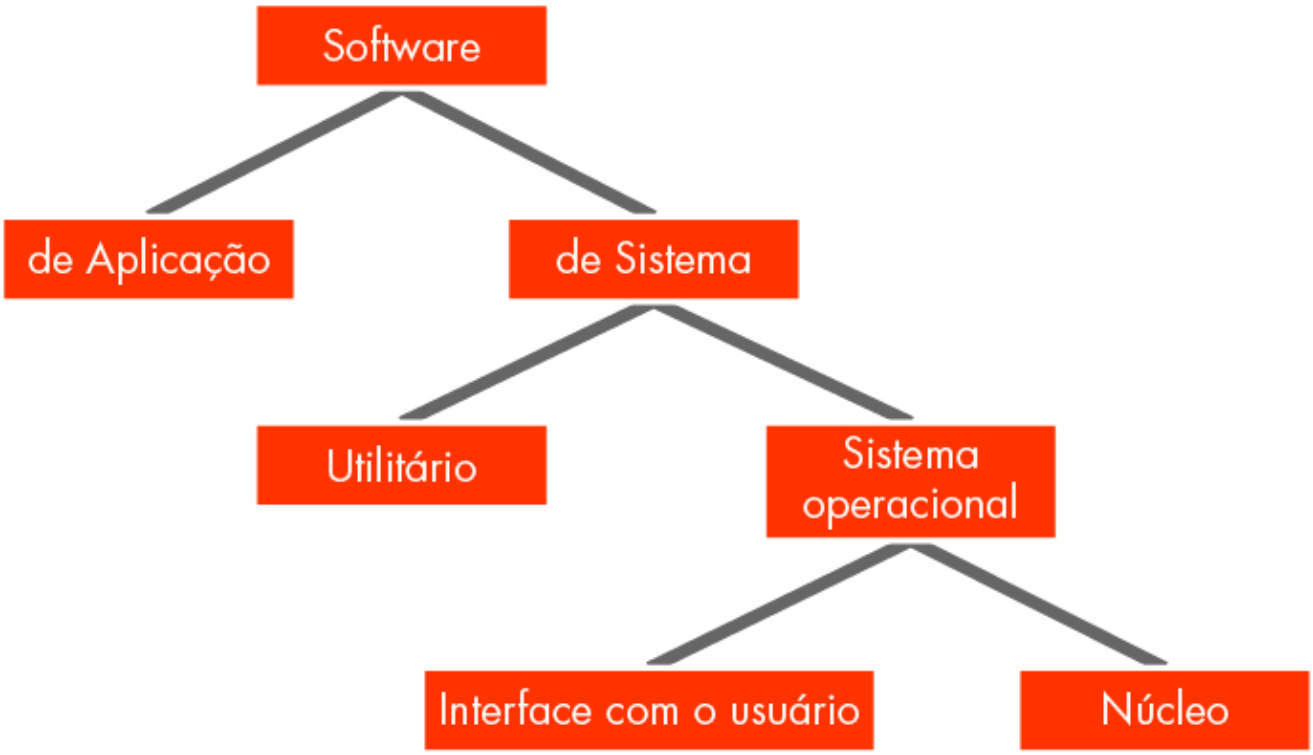


Figura 5 - Como podemos classificar os programas de computador. Fonte: BROOKSHEAR, 2013.

Você pode pensar que a diferença entre *software* de aplicação e utilitário pode ser um pouco confusa. Para tentar esclarecer melhor a diferença, vamos pensar da seguinte forma: se o pacote é parte da “infraestrutura de *software*” do computador, então ele é um *software* utilitário. Logo, um de aplicação pode evoluir para o estado de utilitário se sua funcionalidade se tornar uma ferramenta fundamental. (BROOKSHEAR, 2013)

2.3.2 Aplicações práticas

Você já se perguntou quais métodos e processos devem ser seguidos para desenvolver um programa? Bem, esse desenvolvimento não é apenas escrever código, então precisamos da engenharia de *software* para nos auxiliar durante esse processo. Assim, quando o cliente nos requisita um novo programa, todas as solicitações feitas por ele podem ser chamadas de requisitos do *software*, que são frequentemente classificados como requisitos funcionais e não funcionais.

Os requisitos funcionais representam serviços ou funcionalidades que o sistema deve fornecer, por exemplo, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações. Já os não funcionais são restrições que o sistema deve possuir, como as restrições no processo de desenvolvimento e as impostas por normas que regulamentam o domínio da aplicação. Ao contrário dos requisitos funcionais, os não funcionais aplicam-se, muitas vezes, ao sistema como um todo. (SOMMERVILE, 2011)

Vamos refletir um pouco, se especificarmos os requisitos de um *software* de forma imprecisa ou errada, o que pode acontecer? Imprecisão e erros na especificação dos requisitos de um sistema podem causar muitos problemas, como definir novos requisitos, fazer alterações nos artefatos já criados no sistema e até mesmo reescrever o código daquela funcionalidade. Assim, esses problemas podem gerar atrasos na entrega e aumentar os custos do projeto.

Então, quais são as etapas para especificar os requisitos de um sistema? Primeiro precisamos realizar um estudo inicial de viabilidade, para verificar se o sistema é viável de ser desenvolvido. Depois, realizamos a etapa de elicitação e análise de requisitos. Nessa atividade, é necessário trabalhar diretamente com os clientes e usuários finais do sistema para obter informações e detalhes sobre o domínio da aplicação, as funcionalidades que o sistema deve oferecer, o ambiente que o sistema vai funcionar, restrições de *hardware* etc.. De acordo com Sommerville (2011), as atividades do processo são:

- **descoberta de requisitos**: nessa atividade deve-se trabalhar diretamente com os clientes e usuários do sistema para identificar os requisitos desejados;
- **classificação e organização de requisitos**: essa atividade utiliza a coleção de requisitos que foram identificados na etapa anterior, agrupa requisitos relacionados e os organiza em grupos. A forma mais comum de agrupá-los é utilizando uma arquitetura do sistema para identificar os subsistemas e associá-los a cada subsistema;

- **priorização e negociação de requisitos**: como geralmente existem várias pessoas (clientes e usuários) envolvidas, os requisitos podem entrar em conflito. Portanto, é necessário priorizar os requisitos e resolver os conflitos por meio da negociação de requisitos. Normalmente, os clientes, usuários e desenvolvedores precisam se encontrar para resolver as diferenças e chegar a um acordo sobre os requisitos;
- **especificação de requisitos**: os requisitos são documentados e inseridos no próximo ciclo do desenvolvimento de *software*. Com isso, documentos formais ou informais de requisitos podem ser produzidos.

Para modelar um sistema e construir uma proposta de como funcionará o sistema solicitado pelo cliente, podemos utilizar diagramas de caso de uso, que é uma técnica de descoberta de requisitos, além de ser de fácil entendimento tanto para o cliente quanto para os desenvolvedores que trabalham no projeto. Eles fazem parte da linguagem UML (linguagem de modelagem unificada, do inglês *unified modeling language*) e identificam os atores envolvidos em uma interação do sistema. Eles devem ter o suporte de informações adicionais para descrever as interações do sistema, com o intuito de deixar claro o objetivo de cada caso de uso.

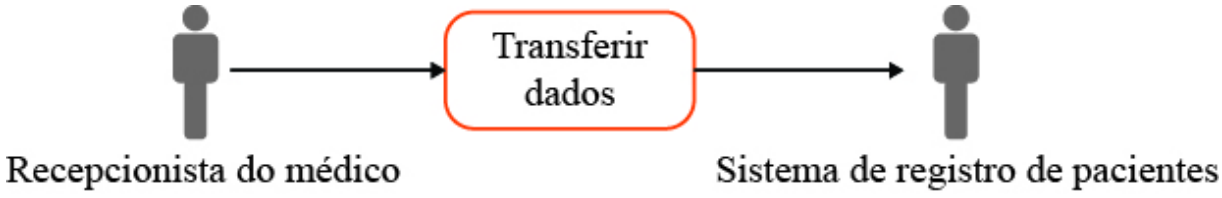


Figura 6 - Forma simples de caso de uso, onde o caso de uso é a elipse e os atores envolvidos são representados por figuras-palito. Fonte: SOMMERVILLE, 2011.

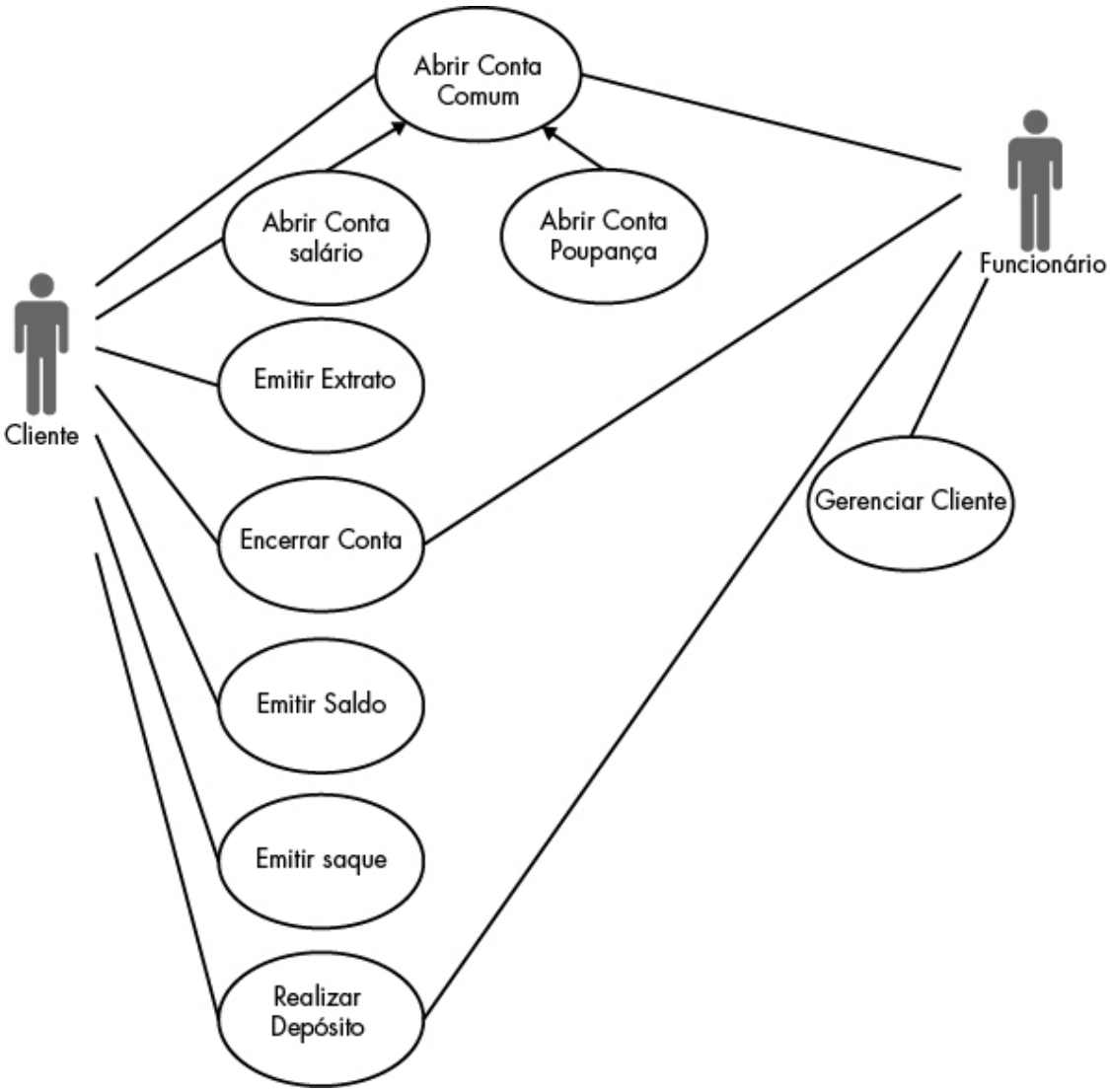
A modelagem de caso de uso é utilizada principalmente para modelar interações entre um sistema e os atores externos, os quais podem ser usuários ou outros sistemas. Nesse contexto, os diagramas de casos de uso podem nos dar uma visão simples e ampla de uma interação. Então, para complementar essa interação com detalhes, é necessário fornecer mais informações para entender o que está envolvido. Esses detalhes podem ser uma descrição textual, uma descrição estruturada em uma tabela ou um diagrama de sequência. (SOMMERVILLE, 2011)

CASO

Em um banco, há as funcionalidades de abrir conta, emitir extrato, realizar depósito, realizar saque, encerrar conta, registrar movimentação e emitir saldo. Além disso, uma conta pode ser uma conta comum, uma conta salário ou uma conta poupança. Nesse contexto, temos o cliente que vai interagir com sua própria conta e os funcionários que vão realizar atividades como gerenciar clientes, abrir conta, encerrar conta e realizar depósito.

Esse cenário é bem comum no nosso dia a dia e podemos perceber que o banco precisa de um sistema que auxilie seus funcionários e clientes a realizar todas essas atividades. Como poderíamos modelar um sistema com essas funcionalidades?

A seguir, veja um diagrama de caso de uso que modela todas as funcionalidades descritas e os atores envolvidos em cada interação, elaborado pelo autor deste material:



Os diagramas de casos de uso, assim como todos os outros diagramas especificados pela linguagem UML, podem ser criados utilizando *softwares* feitos com as especificações da UML. Existem muitos *softwares* que realizam essa atividade, um dos mais conhecidos é o Astah (anteriormente chamado de JUDE – Ambiente para desenvolvedores UML e Java). Ele possui versões pagas e uma versão gratuita, chamada de *Community*.

2.4 Redes e meios de acesso à internet

Até agora pensamos no computador e em programas de forma isolada, sem estarem conectados a outros computadores. Com a necessidade de compartilhar informações, dados e recursos, surgiram as redes de computadores, o que nada mais é do que dois ou mais computadores conectados de forma que dados possam ser enviados e recebidos entre eles. Através das redes de computadores, os usuários podem trocar mensagens, compartilhar recursos (como uma impressora, por exemplo), usar *softwares* e recursos de armazenamento de dados.

Nesse contexto, a internet é uma rede de redes global onde a comunicação é controlada por uma coleção aberta de padrões conhecida como a suíte de protocolos TCP/IP. Na criação da internet, o objetivo era desenvolver a habilidade de ligar uma variedade de redes de computadores de forma que ela funcionasse como um sistema conectado sem ser interrompida por desastres locais.

2.4.1 Conceitos iniciais

Muito do trabalho feito para a criação e manutenção da Internet foi patrocinado pelo governo dos Estados Unidos, por meio da Agência de Projetos de Pesquisa Avançados de Defesa (DARPA). Ao longo do tempo, o desenvolvimento da internet deixou de ser um projeto patrocinado pelo governo para ser um projeto de pesquisa acadêmico e, atualmente, é, em grande parte, um empreendimento comercial que liga uma combinação de escala global de LANs, MANs e WANs, envolvendo milhões de computadores. (BROOKSHEAR, 2013)

VOCÊ O CONHECE?

O criador da World Wide Web, ou internet, é Timothy John Berners-Lee (mais conhecido como Tim Berners-Lee), físico, cientista da computação e professor no MIT (Instituto de Tecnologia de Massachusetts), é o responsável pelo conceito de internet que temos atualmente e permite que possamos nos conectar com pessoas ao redor do mundo, além de ter acesso a informações e dados de qualquer lugar.

Você já ouviu falar nos conceitos de LAN, MAN e WAN? LAN representa uma rede de computadores local, ou seja, cobrem uma área limitada. Uma MAN representa uma rede de computadores maior que as LANs, podendo cobrir cidades próximas ou regiões metropolitanas. Já as WANs representam uma rede de computadores de longa distância, maiores que as MANs.

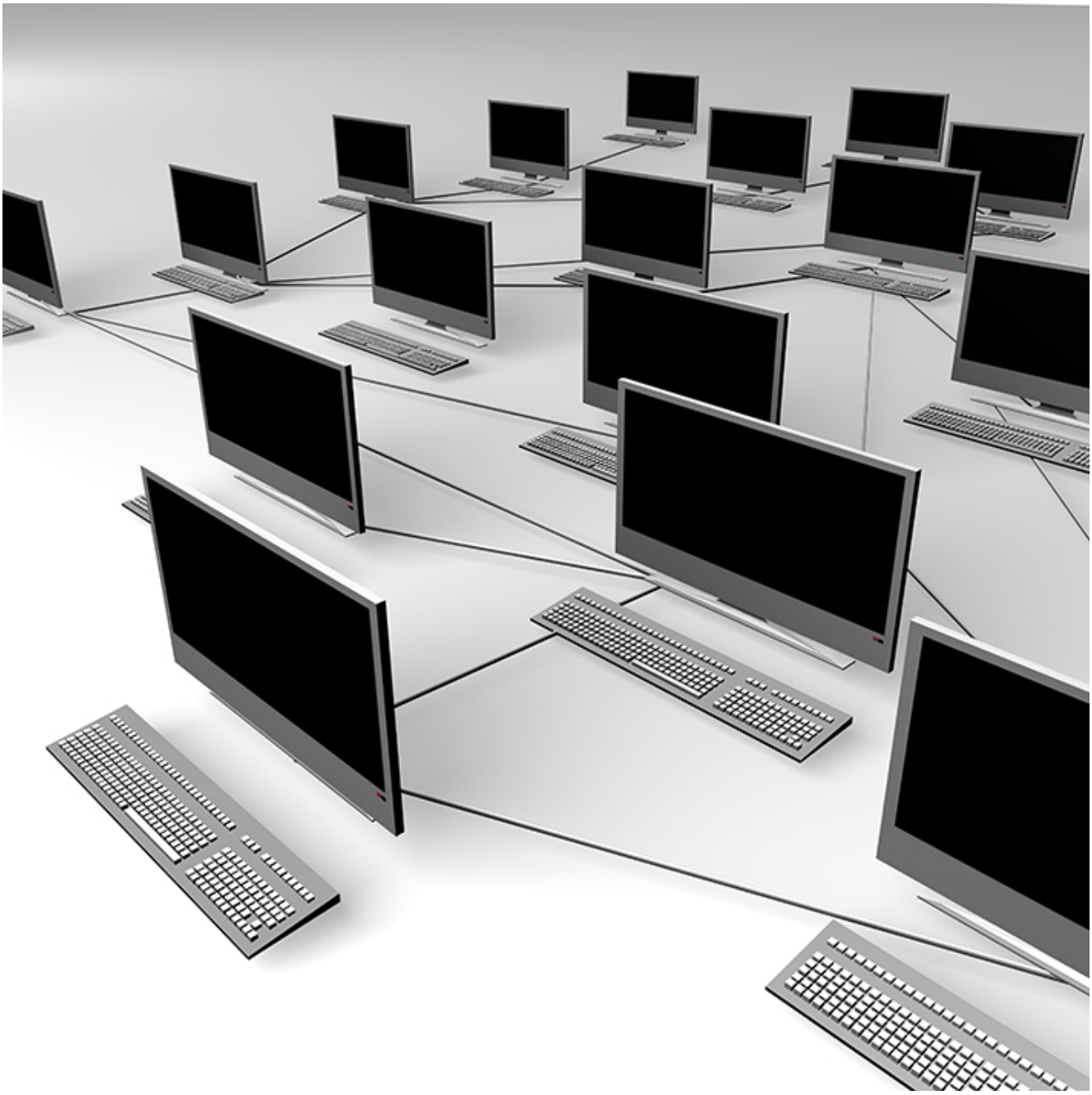


Figura 7 - Computadores conectados entre si, caracterizando uma rede de computadores. Fonte: mamanamsai, Shutterstock, 2018.

Segundo Carvalho e Lorena (2016), as redes de computadores podem ser caracterizadas por meio de diferentes critérios, como, por exemplo:

- meio de transmissão;
- topologia;
- extensão;
- protocolos de comunicação.

Com relação aos meios de transmissão, as redes de computadores podem utilizar diferentes meios, como cabo de pares trançados, cabo coaxial e cabo de fibras ópticas. Os cabos de pares trançados são formados por quatro pares de fios de cobre e não permitem interferência eletromagnética. Os coaxiais são utilizados geralmente por redes de televisão a cabo e possuem uma taxa de transmissão maior que que os pares trançados. Já os de fibras ópticas são os mais indicados para transmissão de dados e possui uma taxa de transmissão maior que a dos cabos coaxiais. As redes de computadores também podem utilizar meios de transmissão sem fio como *bluetooth*, RFID e *wireless*.

VOCÊ QUER LER?

Um texto sobre como a educação à distância evoluiu e se popularizou com ajuda da internet? Atualmente, existem cursos técnicos, bacharelados, licenciaturas e tecnólogos à distância. Esses cursos possuem vídeo-aulas, *web* conferências e disponibilizam ambientes virtuais para que o aluno tenha acesso a diversos tipos de materiais. Acesse: <https://www.infoescola.com/educacao/educacao-a-distancia/ (https://www.infoescola.com/educacao/educacao-a-distancia/)>.

Com relação à topologia, as redes de computadores podem ter uma topologia em barra, configuração em estrela, configuração em anel e redes em árvore. A topologia está relacionada ao formato de como os computadores estão conectados na rede. Na em barra, todos os computadores estão conectados através do mesmo meio, a configuração em estrela utiliza um *hub* ou *switch* para organizar seus componentes. Na em anel, cada computador está conectado a outros dois computadores e, nas em árvore, um computador está conectado diretamente a um ou mais computadores utilizando uma estrutura de árvore.

Com relação à extensão, vimos anteriormente o conceito de LAN, MAN e WAN. E com relação aos protocolos de comunicação, os computadores da rede podem se comunicar de forma diferente, através de arquiteturas, essas arquiteturas podem ser *peer-to-peer* ou cliente-servidor. Na arquitetura *peer-to-peer,* cada computador conectado à rede possui as mesmas atribuições e responsabilidades, já na arquitetura cliente-servidor, há um computador servidor que é responsável por centralizar e prover os recursos para os outros computadores da rede.

2.4.2 Técnicas e ferramentas

Segundo Carvalho e Lorena (2016), existem várias maneiras de se conectar um computador à internet. Para uso doméstico e em pequenas empresas, as cinco tecnologias mais utilizadas são:

- *modem* discado;
- ISDN;
- DSL;
- cabo;
- rede sem fio (*wireless*);
- rede via rádio.

Cabos e ligações de satélite são inerentemente mais compatíveis com transferências de dados de alta velocidade do que linhas telefônicas tradicionais, que foram instaladas com o objetivo de realizar comunicação por voz. Entretanto, diversos esquemas são desenvolvidos para estender essas ligações de voz e acomodar a transmissão de dados digitais (BROOKSHEAR, 2013). Vamos ver agora como as diferentes tecnologias existentes realizam a transmissão dos dados.

As tecnologias de *modem* discado e ISDN são consideradas de banda baixa, e as tecnologias DSL, cabo e rede sem fio são consideradas de banda larga. O *modem* discado se caracteriza por sua baixa velocidade de transmissão de dados (no máximo 56 kbps). Já a conexão ISDN transmite dados de forma digital com velocidade de no máximo 128 kbps.

A conexão DSL permite uma alta taxa de transmissão de dados, podendo ser de até 24 Mbps para *download*. Por outro lado, a conexão a cabo é uma das mais utilizadas e permite uma velocidade de, no máximo, 250 Mbps para residências e de até 400 Mbps para empresas. Redes sem fio (*wireless*) são cada vez mais utilizadas justamente por não precisar de cabos e estão presentes em todos os tipos de lugares, como cafés, escolas, bancos, lanchonetes, supermercados etc. Por fim, as redes via rádio utilizam uma conexão ponto-a-ponto.

Síntese

Você concluiu os estudos sobre como os programas são executados em um sistema computacional. Com essa discussão, esperamos que você se sinta apto para definir um sistema operacional, linguagens de programação, aplicações de sistemas de computação e redes e meios de acesso à internet.

Neste capítulo, você teve a oportunidade de:

- compreender a importância e as funcionalidades dos sistemas operacionais;
- entender como um programa de computador é feito e quais as características de uma linguagem de programação;
- definir o conceito de sistema de computação e como podemos propor um sistema a partir de requisitos do usuário;
- apresentar o conceito de redes de computadores e como essas redes podem ser utilizadas para acessar a internet.

Bibliografia

BROOKSHEAR, J. G. **Ciência da computação**: uma visão abrangente. 11. ed. Porto Alegre: Bookman, 2013.

CARVALHO, A. C. P. L. de; LORENA, A. C. **Introdução à computação:** hardware, software e dados. São Paulo: LTC, 2016.

PERES, M. Do Windows 1.0 ao Windows 10: veja como o sistema mudou nestes 30 anos. **Canaltech**. 20/11/2015. Disponível em: <https://canaltech.com.br/windows/do-windows-10-ao-windows-10-veja-como-o-sistema-mudou-nestes-30-anos-45911/ (https://canaltech.com.br/windows/do-windows-10-ao-windows-10-veja-como-o-sistema-mudou-nestes-30-anos-45911/)>. Acesso em: 14/03/2018.

FEDELI, R. D.; POLLONI, E. G. F.; PERES, F. E. **Introdução a ciência da computação**. 2. ed. São Paulo: Cengage Learning, 2013.

FERRAZ, T. Educação à Distância. **Info Escola**, [s/d]. Disponível em: <https://www.infoescola.com/educacao/educacao-a-distancia/ (https://www.infoescola.com/educacao/educacao-a-distancia/)>. Acesso em: 27/02/2018.

GIT. Disponível em: <https://git-scm.com/ (https://git-scm.com/)>. Acesso em: 19/03/2018.

JONZE, Spike. **Ela** [filme]. Direção (https://www.google.com.br/search?q=her+dire%C3%A7%C3%A3o&stick=H4sIAAAAAAAAAOPgE-LSz9U3yCk0S0sy1RLLTrbST8vMyQUTVimZRanJJflFAEpThtsmAAAA&sa=X&ved=0ahUKEwicheahlezZAhVGWpAKHZ7UAgsQ6BMI6QEoADAX) e produção Spike Jonze. Sony Pictures. 2h06min. Estados Unidos, 2013. Disponível em: < (https://www.youtube.com/watch?v=SjZudKgH_OY)https://www.youtube.com/watch?v=SjZudKgH_OY (https://www.youtube.com/watch?v=SjZudKgH_OY)>. Acesso em: 14/03/2018.

LEVY, Shawn. **Estagiários, os**. [filme]. Direção Shawn Levy. Produção Shawn Levy e Vince Vaughn. Fox Filme. 2h. Estados Unidos, 2013.

MANZANO, J. A. N. G.; LOURENÇO, A. E.; MATOS, E. **Algoritmos -** técnicas de programação. 2. ed. São Paulo: Érica, 2016.

PYTHON. Disponível em: <https://www.python.org (https://www.python.org)>. Acesso em: 14/03/2018.

SOMMERVILLE, I. **Engenharia de software**. 9. edição. São Paulo: Person Education, 2011.

SOUSÓLIVER, A. Engenharia de requisitos – o alicerce da engenharia de software. **Associação Brasileira de Melhoria em TI**, [s/d]. Disponível em: <http://www.abramti.org.br/node/115 (http://www.abramti.org.br/node/115)>. Acesso em: 28/02/2018.

TANENBAUM, A. S.; MACHADO FILHO, N. **Sistemas operacionais modernos**. 3. ed. São Paulo: Prentice-Hall, 2008.