UNIP
EDUCAÇÃO, POR SUA ESCOLHA

SISTEMAS DA INFORMAÇÃO

MATERIAL INSTRUCIONAL ESPECÍFICO

TOMO 12

CQA/UNIP – Comissão de Qualificação e Avaliação da UNIP

SISTEMAS DA INFORMAÇÃO

MATERIAL INSTRUCIONAL ESPECÍFICO

TOMO 12

Christiane Mazur Doi

Doutora em Engenharia Metalúrgica e de Materiais, Mestra em Ciências - Tecnologia Nuclear, Especialista em Cálculo e Matemática Aplicada, Especialista em Estatística Aplicada e Ciência de Dados, Especialista em Língua Portuguesa e Literatura, Especialista em Recursos Gramaticais para Revisão Textual, Engenheira Química e Licenciada em Matemática, com Aperfeiçoamento em Estatística e MBA em Engenharia Econômica. Professora titular da Universidade Paulista.

Tarcísio de Souza Peres

Mestre em Ciências, Especialista em Gestão de Projetos e Engenheiro da Computação, com MBA em Gestão de TI. Cursou Gestão Estratégica e Competitividade. Professor adjunto da Universidade Paulista.

Questão 1

Questão 1.[1]

Leia o texto a seguir.

*Uma relação **R** em um conjunto **S** é uma relação de equivalência se ela satisfizer todas as propriedades a seguir.*
*1. Reflexividade. Para todo a ∈ **S**, a **R** a;*

*2. Simetria. Para todos a, b ∈ **S**, a **R** b ⇔b **R** a;*

*3. Transitividade. Para todos a, b, c ∈ **S**, se a **R** b e b **R** c, então, a **R** c.*
*Uma partição de um conjunto **S** é uma coleção de subconjuntos disjuntos não vazios, cuja união é igual a **S**. Se **R** é uma relação de equivalência em um conjunto **S** e se x ∈ **S**, denota-se por [x] o conjunto de todos os elementos relacionados a x em **S** e chama-se esse conjunto de classe de equivalência de x.*

GERSTING, J. L. *Fundamentos matemáticos para a ciência da computação: um tratamento moderno de matemática discreta*. 5. ed. Rio de Janeiro: LTC, 2008 (com adaptações).

A partir dessas informações, considere que, em um cluster computacional, haja 10 computadores com configurações de hardware diferentes. Os administradores desse cluster pretendem desenvolver um algoritmo de escalonamento de tarefas que mantenha processos de uma mesma aplicação sendo executados em máquinas semelhantes, mesmo com estruturas arquiteturais distintas. Assim, os administradores fizeram uma tabela relacionando os computadores. Essa tabela foi montada considerando pares de computadores. Dessa forma, dois computadores do cluster fazem parte de uma linha na tabela se possuem alguma característica semelhante, ou seja, se apresentam algum tipo de relacionamento (por exemplo, quantidade de memória ou de núcleos de processamento semelhantes). Obviamente, apesar de não estar evidente na tabela, um computador também tem relação consigo mesmo. Os computadores estão numerados de 1 até 10 e a tabela resultante pode ser vista a seguir

Computadores com características semelhantes	
1	5
2	8
5	7
7	10
2	9
3	4
10	1
5	10
7	1
9	8

[1]Questão 12 – Enade 2023.

Com base nesse cenário e no conceito de relações de equivalência, assinale a opção correta.

A. A relação descrita pela tabela é uma relação de equivalência.
B. A relação descrita pela tabela apresenta a propriedade de simetria, mas não a de transitividade.
C. A relação descrita pela tabela apresenta a propriedade de transitividade, mas não a de simetria.
D. O subconjunto de computadores representados pelos números 1, 3, 5, 7 e 10 forma uma classe de equivalência.
E. O subconjunto de computadores representados pelos números 2, 3, 4, 8 e 9 forma uma classe de equivalência.

1. Introdução teórica

Relação de equivalência

Uma relação binária R em um conjunto S é um subconjunto de $S \times S$. Quando $(a, b) \in R$, escreve-se $a\,R\,b$. Uma relação R em S é uma relação de equivalência quando satisfaz, simultaneamente, as propriedades a seguir.

- Reflexividade: para todo $a \in S$, $a\,R\,a$.
- Simetria: para todos $a, b \in S$, $a\,R\,b \Rightarrow b\,R\,a$.
- Transitividade: para todos $a, b, c \in S$, $(a\,R\,b \text{ e } b\,R\,c) \Rightarrow a\,R\,c$.

A relação R é de equivalência: ela separa o conjunto S em grupos de elementos indistinguíveis segundo o critério definido por R. Esses grupos são as classes de equivalência. Para $x \in S$, a classe de equivalência de x é

$$[x] = \{y \in S : y\,R\,x\}$$

Há uma ligação fundamental entre os dois conceitos, como explicamos a seguir.

- Cada relação de equivalência em S determina uma partição de S em classes de equivalência (subconjuntos não vazios, dois a dois disjuntos, cuja união é S).
- Cada partição de S determina uma relação de equivalência: $a\,R\,b$ se, e somente se, a e b pertencem ao mesmo bloco da partição.

2. Análise das alternativas

A - Alternativa correta.

JUSTIFICATIVA. Uma relação de equivalência deve satisfazer, ao mesmo tempo, a reflexividade, a simetria e a transitividade. No enunciado, a reflexividade é assumida ("um computador também tem relação consigo mesmo"). A simetria decorre do sentido de "par de computadores semelhantes": se o computador "a" é semelhante ao computador "b", então "b" é semelhante ao "a". A transitividade é verificada porque os conjuntos sugeridos pela tabela formam grupos fechados: {1,5,7,10}, {2,8,9}, {3,4}, e {6}. Dentro de cada grupo, sempre que a R b e b R c, vale a R c (por exemplo, 1 R 5 e 5 R 7 implicam 1 R 7, o que também aparece na tabela como o par (7,1)).

B - Alternativa incorreta.

JUSTIFICATIVA. A alternativa afirma que existe simetria, mas não existe transitividade. Entretanto, a transitividade é satisfeita nos agrupamentos exibidos. Exemplos diretos: 1 R 5 e 5 R 10 implicam 1 R 10, que aparece como o par (10,1). Além disso, 2 R 8 e 8 R 9 implicam 2 R 9, que aparece como o par (2,9). Esse comportamento é o esperado quando os dados descrevem classes de equivalência, pois as classes correspondem a blocos de uma partição.

C - Alternativa incorreta.

JUSTIFICATIVA. A alternativa afirma que existe transitividade, mas não existe simetria. No contexto de "pares de computadores semelhantes", o par (a,b) representa semelhança mútua. Logo, se a R b, então b R a. Assim, a simetria está presente no critério usado para montar a tabela. Formalmente, isso equivale a interpretar cada linha como um par não ordenado que gera os dois pares ordenados (a,b) e (b,a), conforme a ideia de simetria em relações de equivalência.

D - Alternativa incorreta.

JUSTIFICATIVA. O conjunto {1,3,5,7,10} não pode ser uma classe de equivalência porque o computador 3 aparece relacionado somente com o computador 4 (par (3,4)), formando a classe {3,4}. Como 3 não está relacionado com 1, 5, 7 e 10, ele não pertence à mesma classe desses elementos. Em uma classe de equivalência, todos os elementos devem ser equivalentes entre si.

E - Alternativa incorreta.

JUSTIFICATIVA. O conjunto {2,3,4,8,9} reúne elementos de duas classes distintas: {2,8,9} e {3,4}. Não há, na tabela, relação entre 2 e 3 (nem entre 2 e 4), então não existe uma única classe que contenha simultaneamente esses cinco elementos. Uma classe de equivalência não pode "fundir" dois blocos diferentes de uma partição.

Alternativa correta: A.

3. Indicações bibliográficas

- EPP, S. S. *Discrete Mathematics with Applications*. 5. ed. Boston: Cengage, 2020.
- GERSTING, J. L. *Fundamentos matemáticos para a ciência da computação: um tratamento moderno de matemática discreta*. 5. ed. Rio de Janeiro: LTC, 2008.
- GRIMALDI, R. P. *Discrete and Combinatorial Mathematics: An Applied Introduction*. 5. ed. Boston: Pearson Addison Wesley, 2004.
- ROSEN, K. H. *Discrete Mathematics and Its Applications*. 8. ed. New York: McGraw-Hill Education, 2019.

Questão 2

Questão 2.[2]

Uma lista pode ser dividida em duas partes: o primeiro elemento (a cabeça da lista) e os demais elementos (sua cauda). Por exemplo, em uma lista de inteiros `[1, 2, 3, 4]`, a cabeça dessa lista é o valor inteiro `1`, enquanto sua cauda é a lista de inteiros `[2, 3, 4]`. Uma lista vazia é representada por `[]`.

O código a seguir define duas funções descritas em uma linguagem de programação funcional que manipulam listas de inteiros. A função `enade` recebe uma lista de inteiros e produz uma nova lista de inteiros. A função `auxiliar` é chamada pela função `enade` e possui dois parâmetros: um número inteiro e uma lista de inteiros. Essa função produz uma lista de inteiros.

```
enade :: [Int] -> [Int]
enade [] = []
enade (cabeca:cauda) = auxiliar cabeca (enade cauda)
auxiliar :: Int -> [Int] -> [Int]
auxiliar x [] = [x]
auxiliar x (cabeca:cauda)
  | (x `mod` 2 == 0) = x:cabeca:cauda
  | otherwise = cabeca:auxiliar x cauda
```

Considerando o código apresentado, é correto afirmar que se a função `enade` for executada recebendo como parâmetro de entrada a lista `[1, 2, 3, 4, 5, 6, 7, 8]`, o resultado será

A. `[].`

B. `[2, 4, 6, 8].`

C. `[1, 2, 3, 4, 5, 6, 7, 8].`

D. `[2, 4, 6, 8, 1, 3, 5, 7].`

E. `[2, 4, 6, 8, 7, 5, 3, 1].`

1. Introdução teórica

Linguagens funcionais costumam favorecer uma forma de programação em que a estrutura dos dados orienta a estrutura das funções. As listas, por sua definição em termos de cabeça e de cauda, oferecem um exemplo clássico dessa ideia. A recursão fornece o

[2]Questão 14 – Enade 2023.

mecanismo de processamento, e o pattern matching oferece uma forma expressiva e rigorosa de descrever os casos possíveis. Juntos, esses elementos compõem uma base conceitual importante para o estudo e a prática da programação funcional.

1.1. Programação funcional

Nas linguagens de programação funcionais, o foco do raciocínio do programador costuma recair sobre as funções e sobre a transformação de valores, em vez da sequência de comandos que modificam um estado de memória ao longo do tempo. Em termos conceituais, um programa é descrito como um conjunto de definições que relacionam as entradas e as saídas.

Em muitas linguagens desse paradigma (como Haskell, OCaml, F# e, em parte, Elixir), aparecem com frequência ideias como a imutabilidade, a composição de funções e as definições guiadas pela estrutura dos dados.

1.2. Listas

Uma lista é, em geral, uma estrutura de dados sequencial definida de forma indutiva. Isso significa que ela pode ser descrita por dois casos fundamentais: (1) a lista vazia e (2) uma lista formada por um elemento inicial, chamado de cabeça (head), seguido do restante da lista, chamado de cauda (tail).

Em notação típica de linguagens funcionais, a lista [1,2,3] pode ser vista como 1 : (2 : (3 : [])).Essa leitura é importante porque ela revela a "forma interna" da lista: há sempre um caso-base (a lista vazia) e um caso recursivo (cabeça + cauda).

1.3. Recursão

A recursão é a técnica em que uma definição faz referência a si mesma, de maneira controlada, para resolver um problema. Em programação funcional, a recursão costuma substituir os laços imperativos (for, while) em muitos contextos. Uma definição recursiva correta precisa de dois componentes: um caso-base, que interrompe a chamada recursiva, e um passo recursivo, que reduz o problema a uma versão menor dele mesmo. Em listas, isso se encaixa de forma natural: a lista vazia encerra o processo, e a lista com cabeça e com cauda permite continuar o processamento sobre a cauda.

Os algoritmos recursivos em listas exploram exatamente essa estrutura. Para calcular a soma dos elementos de uma lista, por exemplo, define-se que a soma da lista vazia é 0 (caso-base) e que a soma de uma lista não vazia é a cabeça somada à soma da cauda (passo recursivo). O mesmo esquema aparece em funções como o comprimento da lista, a busca de um elemento, a transformação de elementos (map), a filtragem (filter) e a redução (fold). O ponto importante é que a forma do algoritmo acompanha a forma do dado: se a lista é "vazia" ou "cabeça + cauda", a função também terá casos que refletem essa decomposição.

1.4. Pattern Matching

É nesse contexto que o pattern matching (casamento de padrões) se torna especialmente útil. Trata-se de um mecanismo de definição e de seleção de casos em que a própria forma do dado é usada para decidir qual regra aplicar. Em vez de escrever testes manuais para verificar se a lista está vazia ou para acessar seus elementos por índices, o programador declara padrões como [] (lista vazia) e x:xs (uma lista com cabeça x e cauda xs). Quando o valor recebido corresponde a um desses padrões, a linguagem associa automaticamente nomes às partes relevantes e executa a definição correspondente. Isso torna o código mais legível e mais próximo da estrutura conceitual do problema.

2. Análise da questão

O enunciado não nomeia explicitamente a linguagem, mas, pela notação usada, trata-se de uma escrita típica de Haskell (ou de um pseudocódigo fortemente inspirado em Haskell), que é uma linguagem funcional.

A ideia central da questão é perceber que a função foi construída com pattern matching e recursão sobre listas. Primeiro, ela distingue dois casos estruturais: o caso da lista vazia e o caso da lista não vazia, que é decomposta em cabeça (primeiro elemento) e cauda (restante da lista). Esse reconhecimento da forma da lista é o pattern matching. A partir dessa decomposição, a função principal trata a lista de modo recursivo: ela processa, em primeiro lugar, a cauda e só depois reinsere a cabeça no resultado já obtido. É isso que permite dizer, de forma intuitiva, que a lista é tratada de trás para frente, pois os últimos elementos ficam prontos antes dos primeiros.

A função auxiliar também usa pattern matching para separar o caso em que a lista está vazia do caso em que ela tem cabeça e cauda. Depois que o padrão da lista é reconhecido, entra a

regra condicional (as guardas): se o número a ser inserido for par, ele é colocado no início da lista; se for ímpar, ele é deslocado para mais adiante, até o final. Assim, ao longo do processo, os números pares vão sendo colocados na frente, enquanto os ímpares acabam ficando no fim.

Vamos aplicar essa lógica à entrada [1, 2, 3, 4, 5, 6, 7, 8]. O último elemento é o 8, que é par. Como ele é o primeiro a ser processado, a nova lista começa com [8]. Em seguida, entra o 7, que é ímpar. Então, ele vai para o final da lista já formada, ficando [8, 7]. Depois, entra o 6, que é par. Então, ela vai para o começo, produzindo [6, 8, 7]. Depois, entra o 5, que é ímpar. Então, ele vai para o final, resultando em [6, 8, 7, 5].

Continuando o mesmo raciocínio, entra o 4 (par), que vai para o começo, formando [4, 6, 8, 7, 5]. Depois entra o 3 (ímpar), que vai para o final, ficando [4, 6, 8, 7, 5, 3]. Em seguida, entra o 2 (par), que vai para o começo, gerando [2, 4, 6, 8, 7, 5, 3]. Por fim, entra o 1 (ímpar), que vai para o final, e o resultado final passa a ser [2, 4, 6, 8, 7, 5, 3, 1].

Portanto, o resultado da execução da função com a lista [1, 2, 3, 4, 5, 6, 7, 8] é [2, 4, 6, 8, 7, 5, 3, 1].

Alternativa correta: E.

3. Indicações bibliográficas

- BIRD, R. *Introduction to Functional Programming using Haskell*. 2. ed. Prentice Hall, 1998.
- CORMEN, T. H. et al. *Algoritmos: teoria e prática*. 4. ed. Rio de Janeiro: GEN LTC, 2024.
- DASGUPTA, S.; PAPADIMITRIOU, C.; VAZIRANI, U. *Algoritmos*. AMGH, 2009.
- HUTTON, G. *Programming in Haskell*. 2. ed. Cambridge University Press, 2016.
- O'SULLIVAN, B.; GOERZEN, J.; STEWART, D. B. *Real World Haskell*. O'Reilly Media, 2008.
- THOMPSON, S. *Haskell: The Craft of Functional Programming*. 3. ed. Pearson Education, 2015.

Questão 3

Questão 3.[3]

Computação em nuvem representa um conceito relativo ao compartilhamento de recursos, tais como capacidade de processamento, armazenamento, comunicação de dados e pessoal qualificado para manter sistemas computacionais disponíveis na internet. Quando esse compartilhamento constitui um serviço disponível para qualquer pessoa, o serviço é conhecido como nuvem pública. Quando as mesmas tecnologias são empregadas para uma única empresa, não permitindo que terceiros utilizem parte dos recursos, temos uma nuvem privada. Considerando as vantagens que o gerente de uma empresa espera obter na contratação de um serviço de nuvem pública, avalie as afirmativas.

I. Um serviço de nuvem pública proporciona a redução de custos operacionais, pois é possível dimensionar a necessidade desses recursos com base na demanda, visto que há momentos em que a empresa precisa de mais recursos e há momentos em que ela precisa de menos recursos.

II. Um serviço de nuvem pública gera aumento da velocidade de execução do software e de acesso ao software a partir de qualquer localidade, pois a nuvem pública garante acesso rápido em qualquer parte do mundo, o que contrasta com um servidor localizado na cidade da empresa.

III. Um serviço de nuvem pública viabiliza a redução do investimento inicial com equipamentos, com infraestrutura e com pessoal para iniciar a operação, visto que torna possível adiar a instalação desses recursos na empresa até que a operação se demonstre economicamente viável.

IV. Um serviço de nuvem pública propicia aumento de segurança da informação, pois as rotinas de segurança são responsabilidade do administrador da nuvem pública, não do contratante, o que contrasta com um servidor localizado dentro da empresa.

É correto apenas o que se afirma em

A. I e II.
B. I e III.
C. II e IV.
D. I, III e IV.
E. II, III e IV.

[3]Questão 20 – Enade 2023.

1. Introdução teórica

Computação em nuvem

A computação em nuvem organiza os recursos de TI (a saber, o processamento, o armazenamento, as redes e os softwares) como serviços acessados por rede, com o provisionamento sob demanda e o crescimento ou a redução de capacidade conforme a necessidade. Uma formulação recorrente na literatura descreve a nuvem por atributos como a elasticidade, a cobrança pelo uso (pay as you go) e o autoatendimento para provisionar os recursos.

Na nuvem pública, a infraestrutura pertence ao provedor e é compartilhada entre múltiplos clientes (com isolamento lógico). O valor gerencial costuma aparecer em três frentes abaixo descritas.

- **Economia e investimento**: substitui-se parte do investimento inicial (CAPEX) por despesa operacional (OPEX), com o pagamento conforme o consumo e sem a exigência da compra imediata de servidores para iniciar o serviço.
- **Escalabilidade e elasticidade**: a capacidade pode aumentar ou diminuir conforme o regime de picos e de vales de demanda, evitando o superdimensionamento permanente.
- **Acesso e desempenho percebido**: a nuvem amplia o acesso remoto, mas o desempenho não é necessariamente mais rápido porque depende de rede, de latência e de outros fatores. Aplicações web, por exemplo, podem ser mais lentas do que alternativas locais em certos cenários.

Em segurança, um ponto técnico central é que, em nuvem, a proteção não migra inteira para o provedor: há divisão de responsabilidades entre o provedor e o cliente, que varia de acordo com o modelo adotado - IaaS, PaaS ou SaaS - e as configurações do contratante).

2. Análise das afirmativas

I – Afirmativa correta.

JUSTIFICATIVA. A afirmativa descreve a elasticidade da computação em nuvem pública: a empresa pode aumentar ou reduzir recursos conforme a demanda. Isso tende a reduzir os desperdícios e os custos operacionais em comparação com a manutenção de infraestrutura superdimensionada o tempo todo.

II – Afirmativa incorreta.
JUSTIFICATIVA. A nuvem pública amplia o acesso remoto ao software, porém não garante, por si só, o aumento da velocidade de execução nem o acesso rápido em qualquer localidade. O desempenho depende de fatores como a latência de rede, a qualidade da conexão, a distância até a região do provedor e a arquitetura da aplicação.

III – Afirmativa correta.
JUSTIFICATIVA. A afirmativa está de acordo com uma vantagem clássica da nuvem pública: a redução do investimento inicial em equipamentos e na infraestrutura própria. Em vez de comprar os servidores e instalar toda a estrutura antes de operar, a empresa pode contratar os recursos sob demanda e expandi-los conforme sua necessidade.

IV – Afirmativa incorreta.
JUSTIFICATIVA. A segurança em nuvem pública não é de responsabilidade exclusiva do provedor. Há responsabilidade compartilhada entre o provedor e o contratante (cliente), que varia conforme o modelo de serviço contratado (IaaS, PaaS ou SaaS). Portanto, a justificativa apresentada na afirmativa está incorreta.

Alternativa correta: B.

3. Indicações bibliográficas

- BUYYA, R.; BROBERG, J.; GOSCINSKI, A. M. *Cloud Computing: principles and Paradigms*. Hoboken: Wiley, 2011.
- DOTSON, C. *Practical Cloud Security: a Guide for Secure Design and Deployment*. 2. ed. Sebastopol: O'Reilly Media, 2023.
- ERL, T.; MONROY, E. B. *Computação em nuvem: conceitos, tecnologia, segurança e arquitetura*. 2. ed. Porto Alegre: Bookman, 2024.
- MARINESCU, D. C. *Cloud Computing: Theory and Practice*. 3. ed. Amsterdam: Morgan Kaufmann, 2022.
- VELTE, A. T.; VELTE, T. J.; ELSENPETER, R. C. *Cloud computing, computação em nuvem: uma abordagem prática*. Rio de Janeiro: Alta Books, 2012.

Questão 4

Questão 4.[4]

A fim de melhorar a qualidade de vida de seus cidadãos e de criar eficiência nos serviços e nas operações urbanas, um grupo de vereadores de uma pequena cidade decidiu fazer algumas propostas de lei que, se aprovadas e devidamente implementadas, tendem a aproximar a cidade do conceito de "cidade inteligente" por meio da implementação de novos sistemas de software. O grupo de vereadores, preocupado com acessibilidade, consultou especialistas da área de interação humano-computador (IHC) e levantou informações a respeito de fundamentos de acessibilidade em IHC. Entre eles, estão os mencionados a seguir.

- **Fundamento 1.** Um produto ou serviço é equitativo quando é projetado de modo que possa atender a todos os usuários, independentemente de suas habilidades físicas, sensoriais ou cognitivas.
- **Fundamento 2**. Um produto ou serviço deve ter informações perceptíveis, o que pode envolver o uso de recursos como, por exemplo, texto alternativo para imagens ou contrastes de cores suficientes para usuários com deficiências visuais ou com daltonismo.

Considerando esse contexto, avalie as afirmativas.

I. Uma lei que prevê que semáforos devem exibir ícones universais associados a cada uma de suas cores está relacionada ao Fundamento 1.

II. Ainda que esteja relacionado à acessibilidade em IHC, o Fundamento 1 deixa de tratar da inclusão de pessoas cegas.

III. Dado que a cidade possui uma página na web para a obtenção de informações, uma lei que preveja a existência de telas digitais públicas que permitam o acesso às informações disponibilizadas é suficiente para caracterizar a aplicação do Fundamento 2.

É correto o que se afirma em

A. I, apenas.
B. II, apenas.
C. I e III, apenas.
D. II e III, apenas.
E. I, II e III.

[4]Questão 21 – Enade 2023.

1. Introdução teórica

Interação humano-computador

Na interação humano-computador (IHC), a acessibilidade é um tema central no projeto de sistemas digitais, de interfaces públicas e de serviços mediados por tecnologia. Em termos conceituais, ela diz respeito às condições que permitem que pessoas com diferentes características sensoriais, físicas e cognitivas percebam, compreendam e utilizem um artefato digital com autonomia e com segurança. Esse campo não se limita a adaptações para grupos específicos; ele envolve uma lógica de projeto que busca reduzir as barreiras desde a concepção da interface, aproximando-se das ideias de desenho universal e de design inclusivo. Nessa perspectiva, a qualidade de um sistema é avaliada também por sua capacidade de atender à diversidade humana real, e não por um usuário médio abstrato.

Em IHC, é importante distinguir a existência de um recurso tecnológico da efetiva acessibilidade desse recurso. A presença de uma tela, de um site, de um painel eletrônico ou de um aplicativo pode ampliar o alcance de informações, mas isso não significa, por si, que o conteúdo esteja acessível. A análise depende de como a informação é apresentada, de quais canais perceptivos são exigidos e de quais obstáculos foram removidos ou mantidos. Uma interface pode estar disponível e, ainda assim, excluir as pessoas por depender somente de uma certa cor, por ter o contraste insuficiente, por utilizar uma tipografia pouco legível, por não oferecer as alternativas textuais ou as sonoras, ou por exigir algumas interações incompatíveis com as tecnologias assistivas.

Dois conhecimentos são especialmente relevantes para esse tipo de avaliação. O primeiro é o princípio da equidade, ligado ao desenho de produtos e de serviços que possam ser usados por pessoas com perfis diversos, sem segregação e sem perda de funcionalidade para determinados grupos. A equidade, nesse contexto, implica projetar para a pluralidade de usuários, incluindo as pessoas cegas, as pessoas com baixa visão, as pessoas com daltonismo, as pessoas com limitações motoras e as pessoas com diferentes perfis cognitivos. O foco está em evitar que uma decisão de projeto transforme uma característica humana em barreira de acesso.

O segundo conhecimento é o da perceptibilidade da informação. Uma mensagem deve ser apresentada de modo perceptível para diferentes usuários, o que costuma exigir a redundância de sinais e o cuidado com a forma de representação. Em vez de depender de um único código perceptivo, como a cor, o projeto acessível combina os recursos complementares,

como o texto, os ícones, os símbolos, o contraste adequado, os padrões visuais, o feedback sonoro e outros mecanismos compatíveis com o contexto de uso. Essa redundância tem uma função cognitiva e sensorial: ela melhora a compreensão geral e reduz a exclusão de quem não percebe um dos canais empregados.

Também é necessário compreender que a acessibilidade e a usabilidade são conceitos relacionados, mas distintos. Um sistema pode ser relativamente fácil de usar para quem consegue acessá-lo e, ainda assim, ser inacessível para outros usuários. A avaliação consistente exige observar se a interface permite o acesso inicial à informação, se a informação pode ser percebida com clareza e se as ações exigidas podem ser realizadas por diferentes pessoas. Em ambientes urbanos e em serviços públicos digitais, essa análise ganha relevância adicional, porque o uso é heterogêneo, porque ocorre em condições variadas (a presença de ruído, a iluminação diversa, a pressa na interação, a circulação intensa) e envolve direitos de acesso à informação e aos serviços.

2. Análise das afirmativas

I – Afirmativa correta.
JUSTIFICATIVA. A associação de ícones universais às cores dos semáforos amplia o uso por pessoas com diferentes condições sensoriais, especialmente por pessoas com daltonismo, e se relaciona ao fundamento da equidade, pois favorece o atendimento de um conjunto mais amplo de usuários.

II – Afirmativa incorreta.
JUSTIFICATIVA. O fundamento da equidade abrange usuários com diferentes habilidades sejam físicas, sensoriais e até cognitivas. Portanto, ele inclui, sim, as pessoas cegas; a afirmativa exclui indevidamente esse grupo.

III – Afirmativa incorreta.
JUSTIFICATIVA. A mera existência de telas digitais públicas para acesso às informações não é suficiente para caracterizar a aplicação do fundamento das informações perceptíveis. Para isso, é necessário que a apresentação da informação seja acessível (por exemplo, com o contraste adequado, a legibilidade, a redundância de sinais e as alternativas para pessoas com deficiência visual).

Alternativa correta: A.

3. Indicações bibliográficas

- BARBOSA, S. D. J.; SILVA, B. S. *Interação humano-computador*. Rio de Janeiro: Elsevier, 2010.
- NIELSEN, J. *Usability Engineering*. Morgan Kaufmann, 1994.
- PREISER, W. F. E.; SMITH, K. H. *Universal Design Handbook*. 2. ed. McGraw-Hill, 2010.
- ROGERS, Y.; SHARP, H.; PREECE, J. *Design de interação: além da interação humano-computador*. 3. ed. Porto Alegre: Bookman, 2013.

Questão 5

Questão 5.[5]

Considere um cenário em que um computador seja organizado com múltiplos processadores, os quais compartilham a mesma memória RAM. Cada processador possui múltiplos núcleos. Nesse arranjo, o sistema operacional permite múltiplas threads, as quais podem ser dinamicamente alocadas para execução em diferentes núcleos e processadores. A partir das informações apresentadas nessa situação, assinale a opção correta.

A. Sistemas com múltiplos processadores devem alocar a mesma quantidade de memória RAM para cada processador do arranjo.
B. Como há múltiplos processadores, são desnecessários os semáforos, uma vez que não há acessos concorrentes a recursos compartilhados.
C. Como a exclusão mútua não é possível em arquitetura de múltiplos processadores, apenas uma aplicação pode ser executada de cada vez, mas com múltiplas threads.
D. Os processos que possuem múltiplas threads em execução são mantidos por meio de funções da biblioteca no código da aplicação e dispensam serviços do sistema operacional.
E. Dados trocados durante a comunicação entre processos podem ser armazenados nas áreas de memória compartilhada, mas o acesso a essas áreas é intermediado pelo sistema operacional.

1. Introdução teórica

Desempenho em sistemas computacionais

Em sistemas computacionais contemporâneos, o desempenho não depende apenas da frequência de um processador, mas da capacidade de executar múltiplas atividades de forma concorrente em arquiteturas com vários núcleos e, em certos arranjos, com múltiplos processadores físicos. Nesses ambientes, o sistema operacional organiza o uso da CPU, da memória e dos dispositivos de entrada e saída, distribuindo o trabalho entre unidades de execução e controlando a competição por recursos compartilhados.

Para compreender esse funcionamento, é indispensável distinguir os conceitos de processo e de thread, entender os mecanismos de escalonamento, conhecer os problemas de sincronização e reconhecer o papel do sistema operacional na comunicação entre processos.

[5]Questão 26 – Enade 2023.

1.1. Threads

Um processo é a instância de um programa em execução, com espaço de endereçamento próprio, com recursos associados e com contexto de execução. As threads são fluxos de execução dentro de um processo; elas compartilham o mesmo espaço de memória do processo e vários de seus recursos, mas têm um contexto de execução próprio, como o contador de programa e seus registradores. Essa distinção é central para a análise de concorrência, porque os processos e as threads podem executar simultaneamente em núcleos diferentes, produzindo ganhos de desempenho em tarefas paralelizáveis, ao mesmo tempo em que introduzem os riscos de inconsistência quando acessam os dados compartilhados sem a coordenação adequada.

O escalonamento dessas unidades de execução é uma função clássica do sistema operacional. Em arquiteturas com múltiplos núcleos, o kernel decide quais threads ou quais processos serão executados em cada núcleo, levando em conta as políticas de justiça, de prioridade, de responsividade e do uso eficiente da CPU. A mobilidade de threads entre os núcleos, comum em sistemas modernos, exige que a análise de concorrência não seja feita como se houvesse uma correspondência fixa entre uma thread e um processador. A execução é dinâmica, e a simultaneidade efetiva de acessos à memória compartilhada é uma consequência direta dessa organização.

Quando duas ou mais threads acessam a mesma estrutura de dados, e ao menos uma delas realiza a escrita, surge a necessidade de sincronização. Sem a sincronização, ocorre a condição de corrida, em que o resultado final passa a depender da ordem temporal das operações, produzindo comportamentos erráticos e difíceis de reproduzir. A exclusão mútua é o princípio que permite garantir que apenas uma thread por vez execute uma região crítica, preservando a consistência dos dados. Esse princípio é implementado por mecanismos como os mutexes, os semáforos, os monitores, as variáveis de condição e as operações atômicas. Em arquiteturas multiprocessadas, esses mecanismos permanecem essenciais, porque a execução paralela real amplia a chance de concorrência sobre os mesmos recursos.

Os semáforos podem ser usados tanto para a exclusão mútua quanto para a sincronização entre atividades, coordenando as ordens de execução e a disponibilidade de recursos. Seu uso correto exige disciplina de programação, pois falhas na aquisição e na liberação podem causar os deadlocks, a starvation ou a perda de desempenho por contenção excessiva. A presença de múltiplos processadores não elimina essas necessidades; ao contrário, torna mais evidente a importância de protocolos de sincronização bem definidos.

1.2. Comunicação entre processos

Embora processos tenham, em regra, espaços de endereçamento separados, o sistema operacional fornece os mecanismos para troca de dados, como os pipes, as filas de mensagens, os sockets, os sinais e a memória compartilhada.

A memória compartilhada é um dos mecanismos mais eficientes para a IPC (Interprocess Communication) local porque evita cópias repetidas de dados entre os espaços de usuário e de kernel após o estabelecimento da região compartilhada. O papel do sistema operacional é de criar, de mapear e de proteger essa região, além de controlar as permissões. Depois de mapeada, os processos acessam a área compartilhada diretamente, e a sincronização entre eles continua necessária para impedir as leituras e as escritas inconsistentes. A eficiência e a segurança caminham juntas: o ganho de desempenho da memória compartilhada exige mais cuidado com a coordenação de acessos.

Bibliotecas de threads fornecem as APIs (Interfaces de Programação de Aplicações) para a criação, a sincronização e o gerenciamento de threads, simplificando a programação concorrente. Mesmo assim, elas não tornam dispensáveis os serviços do sistema operacional. O kernel continua responsável por aspectos fundamentais, como o escalonamento, o gerenciamento de memória, a proteção, o tratamento de interrupções e o suporte às primitivas de sincronização e da IPC.

Em muitos sistemas, as threads utilizadas pelas aplicações são threads em nível de kernel, diretamente visíveis ao escalonador do sistema operacional. Em outros modelos, podem existir threads em nível de usuário, mas, ainda assim, a execução concreta depende da infraestrutura oferecida pelo sistema.

2. Análise das alternativas

A - Alternativa incorreta.

JUSTIFICATIVA. Em arquiteturas com os múltiplos processadores e com a memória compartilhada, não existe a exigência de que cada processador tenha "a mesma quantidade de memória RAM alocada". A memória principal pode ser compartilhada por todos os processadores (como em arranjos SMP/UMA, no tratamento didático clássico), e a distribuição lógica de uso da RAM não é feita por uma regra fixa de igualdade por processador. A afirmação descreve uma obrigação inexistente na teoria de sistemas operacionais.

B – Alternativa incorreta.

JUSTIFICATIVA. A existência de múltiplos processadores ou de múltiplos núcleos não elimina os acessos concorrentes a recursos compartilhados; ela aumenta a possibilidade de acessos simultâneos. Sempre que os processos ou as threads compartilham dados, pode ocorrer uma condição de corrida se não houver a coordenação adequada. Os semáforos e outros mecanismos de sincronização continuam necessários para proteger as regiões críticas e para ordenar o acesso aos recursos compartilhados.

C – Alternativa incorreta.

JUSTIFICATIVA. A exclusão mútua é possível em arquiteturas multiprocessadas e constitui um dos fundamentos da programação concorrente. Ela é implementada por mecanismos de sincronização, como os mutexes, os semáforos e as operações atômicas, com suporte do sistema operacional e do hardware. Também é incorreta a afirmação de que apenas uma aplicação pode ser executada por vez porque os sistemas multiprocessados são projetados justamente para permitir execução concorrente de múltiplos processos e threads.

D – Alternativa incorreta.

JUSTIFICATIVA. As bibliotecas de threads fornecem funções para a criação e o controle de threads no código da aplicação, mas isso não dispensa os serviços do sistema operacional. O sistema operacional permanece responsável pelo escalonamento, pela gerência de memória, pela proteção entre processos, pelas chamadas de sistema e pelo suporte aos mecanismos de sincronização e de comunicação entre processos. A alternativa generaliza de modo incorreto o papel das bibliotecas e exclui indevidamente a atuação do kernel.

E – Alternativa correta.

JUSTIFICATIVA. Na comunicação entre processos por memória compartilhada, os dados podem ser colocados em áreas de memória compartilhada entre os processos. O sistema operacional intermedeia a criação, o mapeamento e a proteção dessas áreas, controlando as permissões e o acesso autorizado. Após o mapeamento, os processos passam a acessar a região compartilhada para leitura e escrita, mantendo a necessidade de sincronização para garantir a consistência dos dados.

3. Indicações bibliográficas

- MACHADO, F. B.; MAIA, L. P. *Fundamentos de sistemas operacionais*. Rio de Janeiro: LTC, 2011.
- MACHADO, F. B.; MAIA, L. P. *Arquitetura de sistemas operacionais.* 5. ed. Rio de Janeiro: LTC, 2013.
- SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Fundamentos de sistemas operacionais*. 9. ed. Rio de Janeiro: LTC, 2015.
- STALLINGS, W. *Arquitetura e organização de computadores*. 8. ed. São Paulo: Pearson Prentice Hall, 2010.
- TANENBAUM, A. S.; BOS, H. *Sistemas operacionais modernos*. 4. ed. São Paulo: Pearson, 2016.

Questão 6

Questão 6.[6]

A técnica de virtualização de hardware consiste em emular um computador no qual a camada de software é executada sem que detalhes do computador físico e de seus componentes sejam expostos. Em um ambiente de computação distribuída, a técnica pode ser útil para que o sistema operacional e os softwares do usuário sejam executados em uma máquina virtual com características permanentes, em conformidade com o que foi projetado, verificado e validado, mesmo que um computador físico diferente seja empregado.

Com base nesse contexto, é correto

A. adquirir licenças de software do usuário no volume de uma licença para cada computador físico, o que favorece a economia de licenças, pois softwares em máquinas virtuais não correspondem a cópias extras.

B. utilizar um computador com capacidade extra de comunicação de dados para favorecer o seu desempenho, pois um sistema distribuído em máquinas virtuais consome mais recursos de rede do que o mesmo sistema distribuído sendo executado em um computador físico.

C. utilizar um computador com capacidade extra de memória principal para favorecer o seu desempenho, pois um sistema distribuído em máquinas virtuais requer mais espaço de memória física do que o mesmo sistema distribuído sendo executado em um computador físico.

D. adquirir computadores físicos com processadores similares para favorecer a compatibilidade, pois sistemas distribuídos fortemente acoplados compartilham recursos intensamente, funcionando de forma mais eficiente em máquinas virtuais quando os computadores físicos são compatíveis.

E. adquirir dispositivos de armazenamento físico com, pelo menos, o dobro da capacidade do dispositivo virtual a ser usado a fim de favorecer a disponibilidade, pois os sistemas operacionais da máquina virtual e da máquina física devem ter espaço equivalente de dados para que exista o mapeamento direto entre o dispositivo virtual e o dispositivo físico.

[6]Questão 27 – Enade 2023.

1. Introdução teórica

1.1. Virtualização de hardware

A virtualização de hardware é uma técnica que permite executar sistemas operacionais e aplicações em ambientes lógicos chamados máquinas virtuais, separados dos detalhes específicos do equipamento físico subjacente. Com isso, o software passa a operar sobre uma camada de abstração que preserva uma visão estável do computador, mesmo quando o equipamento real é diferente. Essa ideia é especialmente útil em ambientes distribuídos, nos quais a portabilidade, o isolamento e a previsibilidade da execução têm grande relevância.

Na virtualização de hardware, o computador físico hospeda uma ou mais máquinas virtuais. Cada uma delas possui processador virtual, memória virtual, dispositivos de armazenamento virtuais e interfaces de rede virtuais. Para o sistema operacional convidado, esse conjunto aparece como se fosse um computador próprio. O objetivo é desacoplar a execução do software em relação aos detalhes concretos da máquina física, o que facilita a administração, a mobilidade das cargas de trabalho e a padronização dos ambientes de execução.

1.2. Hipervisor e máquinas virtuais

A camada responsável por gerenciar essa abstração é o hipervisor. Ele controla o acesso das máquinas virtuais aos recursos físicos e distribui esses recursos entre elas. Em consequência, a máquina virtual não usa diretamente a memória, o processador ou os dispositivos físicos; ela usa representações controladas pelo hipervisor.

Isso produz isolamento entre ambientes e permite que diferentes sistemas operacionais coexistam sobre o mesmo equipamento físico. Ao mesmo tempo, essa intermediação introduz certo custo de gerenciamento, pois parte dos recursos físicos também precisa sustentar a própria virtualização.

1.3. Virtualização em sistemas distribuídos

Em computação distribuída, a virtualização ajuda a manter ambientes padronizados para a execução de serviços e aplicações em diferentes nós. Um mesmo sistema pode ser

projetado, testado e validado sobre um determinado conjunto de características virtuais e depois executado em outro hardware físico sem alteração da interface vista pelo software.

Essa propriedade favorece a migração, a replicação, a escalabilidade e a administração dos serviços distribuídos. Ainda assim, a virtualização não elimina totalmente o custo dos recursos físicos; ela reorganiza seu uso por meio de uma camada adicional.

1.4. Sobrecarga e uso de recursos

Uma consequência importante da virtualização é a sobrecarga de recursos. Uma máquina virtual precisa de memória para o sistema operacional convidado, para suas aplicações e para as estruturas de controle mantidas pelo hipervisor. Em muitos cenários, isso significa que a execução virtualizada demanda mais memória física do que a execução direta sobre hardware real equivalente. Essa diferença não decorre de uma mudança lógica no programa distribuído, mas do fato de que a virtualização acrescenta uma camada de gerenciamento entre o software e o equipamento físico.

2. Análise das alternativas

A - Alternativa incorreta.
JUSTIFICATIVA. O licenciamento de software não pode ser tratado, em regra, como simples economia automática por máquina física. Em muitos modelos de licenciamento, a máquina virtual conta como instância própria de execução, e as condições dependem do contrato do fabricante. Portanto, a afirmação generaliza de modo indevido.

B – Alternativa incorreta.
JUSTIFICATIVA. A virtualização pode introduzir alguma sobrecarga de rede em certos contextos, mas não é correto afirmar, de modo geral, que um sistema distribuído em máquinas virtuais consome mais recursos de comunicação de dados do que o mesmo sistema executado diretamente em máquinas físicas. Essa não é uma consequência necessária da técnica.

C – Alternativa correta.
JUSTIFICATIVA. A execução de um sistema distribuído em máquinas virtuais tende a exigir mais memória física do que a execução correspondente em máquinas físicas sem virtualização, porque cada máquina virtual precisa de memória para seu sistema operacional convidado e

para suas aplicações, além da memória consumida pela própria infraestrutura de virtualização. Assim, usar um computador com capacidade extra de memória principal favorece o desempenho nesse contexto.

D – Alternativa incorreta.
JUSTIFICATIVA. A semelhança entre processadores físicos pode ser relevante em cenários específicos, como migração entre hospedeiros, mas a justificativa apresentada é excessiva e não expressa uma exigência geral para o bom funcionamento de sistemas distribuídos em máquinas virtuais. O ponto central da virtualização é justamente reduzir a dependência direta do software em relação às particularidades do hardware físico.

E – Alternativa incorreta.
JUSTIFICATIVA. Não existe a exigência de que o armazenamento físico tenha pelo menos o dobro da capacidade do dispositivo virtual nem a necessidade de equivalência de espaço entre sistema operacional hospedeiro e sistema operacional convidado para que haja "mapeamento direto" entre dispositivo virtual e físico. O armazenamento virtual é abstraído e gerenciado pelo sistema de virtualização, não por correspondência rígida desse tipo.

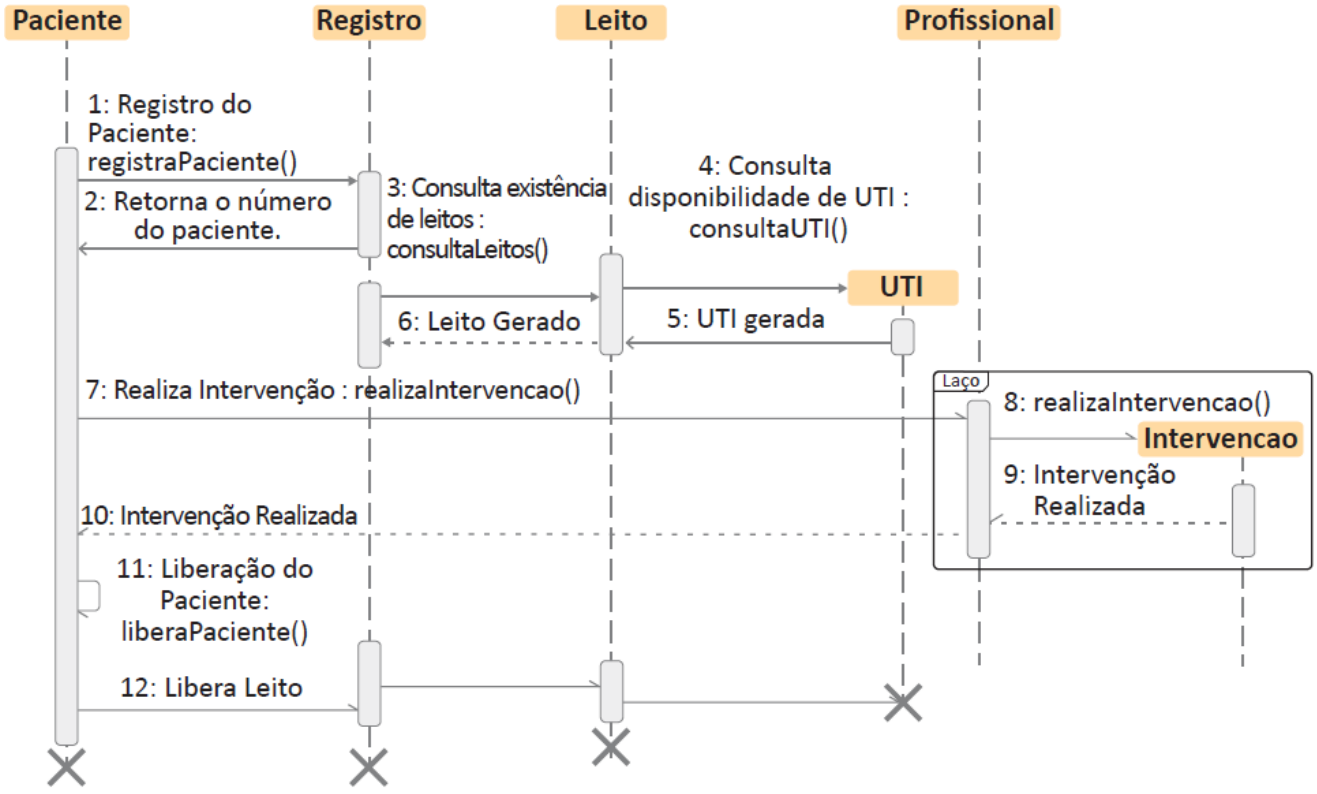
3. Indicações bibliográficas

- SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Operating System Concepts*. 10. ed. Hoboken: Wiley, 2018.
- SMITH, J. E.; NAIR, R. *Virtual Machines: versatile platforms for systems and processes*. San Francisco: Morgan Kaufmann, 2005.
- TANENBAUM, A. S.; BOS, H. *Modern Operating Systems*. 4. ed. Boston: Pearson, 2015.

Questão 7

Questão 7.[7]

Uma ONG decidiu construir um hospital de campanha para tratamento de pacientes diagnosticados com covid-19. Para auxiliar na gerência hospitalar, a ONG contratou alguns programadores voluntários para desenvolver um Sistema de Suporte à Decisão para Gestão Hospitalar. Esse sistema irá auxiliar no registro de todos os procedimentos diários realizados no paciente que dá entrada no hospital, desde sua internação até a saída, seja essa saída por recebimento de alta, por transferência ou por óbito. O sistema foi todo desenvolvido em Java de acordo com o Paradigma Orientado a Objetos. Durante o curto processo de análise, devido à urgência, foram construídos diversos diagramas em UML (Unified Modeling Language). Um desses diagramas relaciona a interação entre os objetos do sistema, o Diagrama de Sequência. Esse diagrama é apresentado a seguir.



Considerando o Diagrama de Sequência apresentado, assinale a opção correta.

A. Leito é uma subclasse de UTI.

B. O método registraPaciente() é implementado por Paciente.

C. Todos os objetos foram criados no instante de execução do Caso de Uso representado.

[7]Questão 16 – Enade 2021 (com adaptações).

D. O diagrama apresenta um erro ao não representar as mensagens de retorno depois da destruição dos objetos.
E. A mensagem 6 pode ser substituída pelo estereótipo <<create>> sem causar prejuízo à interpretação correta do Diagrama de Sequência representado.

1. Análise da questão

A questão cobra conhecimentos a respeito de diagramas de sequência, um dos tipos de diagramas UML (Unified Modeling Language). A seguir, faremos uma breve introdução ao tema e, em sequência, analisaremos o diagrama do enunciado.

1.1. Diagrama de sequência

No contexto dos diagramas UML, os **diagramas de sequência** são aqueles capazes de modelar a sequência de mensagens trocadas entre os objetos de um programa computacional, ao longo de uma interação.

Os principais elementos dispostos em um diagrama de sequência são elencados a seguir, com suas respectivas definições.

- **Linhas de vida**: representam objetos que participam de uma interação. Desse modo, cada instância é representada por uma linha de vida. Graficamente, são desenhadas como linhas verticais.
- **Mensagens**: são elementos que definem um tipo distinto de comunicação entre instâncias, em uma interação. Cada mensagem carrega informações de uma instância a outra. No diagrama, as mensagens são textos numerados que ocorrem entre as linhas de vida, sendo que as setas que conectam as linhas de vida indicam a ordem das mensagens transmitidas.
- **Barras de ativação**: indicam o período em que uma instância está ativa e processando uma mensagem. Essas barras são vistas no diagrama sobre as linhas de vida.
- **Fragmentos combinados**: são agrupamentos lógicos que contêm estruturas condicionais. No diagrama, são destacados por retângulos.

1.2. Análise do diagrama da questão

Com base no diagrama de sequência fornecido no enunciado da questão, a interação entre as instâncias do sistema pode ser resumida conforme os itens a seguir.

1. A instância Paciente inicia a interação invocando o método registraPaciente(). Isso representa um paciente sendo registrado no sistema, quando dá entrada no hospital.
2. A instância Registro retorna o número do paciente, que pode ser usado para identificar o paciente no sistema.
3. A instância Registro consulta a existência de leitos disponíveis no hospital por meio do método consultaLeitos().
4. A instância Registro também consulta a disponibilidade de UTI pelo método consultaUTI(). Isso pode ser necessário, caso o paciente precise de cuidados intensivos.
5. Como resultado das consultas, as instâncias Leito e UTI são geradas.
6. A instância Profissional realiza uma intervenção no paciente por meio do método realizaIntervencao(). Isso caracteriza algum tipo de procedimento médico que é realizado no paciente.
7. Depois de sair do laço de intervenção, o paciente é liberado pelo método liberaPaciente(). Isso pode representar o paciente recebendo alta, por exemplo.
8. Por fim, o leito que havia sido alocado ao paciente é liberado. Isso indica que o leito ficou disponível para outro paciente.

2. Análise das alternativas

A – Afirmativa incorreta.

JUSTIFICATIVA. A hierarquia entre classes não é exposta no diagrama de sequências. Desse modo, não é possível afirmar se uma classe do sistema é subclasse de outra.

B – Afirmativa incorreta.

JUSTIFICATIVA. Pelo diagrama, sabemos que a instância Paciente invoca o método registraPaciente(). No entanto, isso não significa, necessariamente, que o método é implementado por Paciente. Pode ser que esse método esteja implementado em outra classe, e é apenas chamado por Paciente.

C – Afirmativa incorreta.

JUSTIFICATIVA. O diagrama nos mostra que os objetos Leito e UTI são gerados durante a execução do caso de uso, conforme comentamos na análise do diagrama de sequência. No entanto, não há indicação de que as instâncias Paciente, Registro e Profissional são criadas durante a execução do caso de uso.

D – Afirmativa incorreta.

JUSTIFICATIVA. A destruição dos objetos, representada no diagrama por símbolos do tipo ✕, indica o momento em que a memória alocada para a criação das instâncias é liberada. Em um diagrama de sequência, não é obrigatório mostrar mensagens de retorno após a destruição de um objeto. Portanto, isso não é considerado um erro.

E – Afirmativa correta.

JUSTIFICATIVA. Alguns diagramas de sequência utilizam estereótipos. O estereótipo <<create>> indica a criação de um objeto. No diagrama de sequência do enunciado, a mensagem 6 diz "Leito Gerado", o que indica a criação do objeto Leito. Portanto, ela poderia ser substituída pelo estereótipo <<create>> sem causar prejuízo à interpretação correta do diagrama.

3. Indicações bibliográficas

- FOWLER, M. *UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos*. Porto Alegre: Bookman, 2005.
- IBM. *Diagramas de sequência*. Disponível em <https://www.ibm.com/docs/pt-br/rsm/7.5.0?topic=uml-sequence-diagrams>. Acesso em 12 dez. 2025.
- LEE, R. C.; TEPFENHART, W. M. *UML e C++: guia prático de desenvolvimento orientado a objeto*. São Paulo: Makron Books, 2001.

ÍNDICE REMISSIVO