

## **Aula 00**

*Engenharia de Software p/ SEE-DF-  
Temporários (Professor - Informática) -  
2020 - Pré-Edital*

Autor:

**Diego Carvalho, Equipe  
Informática e TI, Fernando  
Pedrosa Lopes**

08 de Julho de 2020

## Sumário

|                                                   |    |
|---------------------------------------------------|----|
| 1 – Arquitetura de Software .....                 | 2  |
| 2 – Coesão e Acoplamento .....                    | 4  |
| Exercícios Comentados .....                       | 6  |
| Lista de Exercícios.....                          | 11 |
| Gabarito .....                                    | 14 |
| 3 – Arquitetura em Camadas .....                  | 15 |
| Exercícios Comentados .....                       | 21 |
| Lista de Exercícios.....                          | 35 |
| Gabarito .....                                    | 42 |
| 4 – Arquitetura MVC (Model View Controller) ..... | 43 |
| Exercícios Comentados – CESPE.....                | 48 |
| Lista de Exercícios – CESPE.....                  | 70 |
| Gabarito - CESPE.....                             | 79 |
| 5 – Arquitetura Distribuída .....                 | 80 |
| Exercícios Comentados .....                       | 84 |
| Lista de Exercícios.....                          | 87 |
| Gabarito .....                                    | 88 |



# 1 – ARQUITETURA DE SOFTWARE

A Arquitetura de Software é a **organização ou a estrutura dos componentes significativos do sistema de software que interagem por meio de interfaces**. Uma arquitetura bem projetada deve ser capaz de atender aos requisitos funcionais e não-funcionais do sistema de software e ser suficientemente flexível para suportar requisitos voláteis.

A arquitetura é importante, na medida em que ela permite uma comunicação efetiva entre as partes interessadas, abrangendo a compreensão, negociação e consenso. Ademais, permite decisões tempestivas, i.e., **possibilita correção e validação do sistema antes da implementação**. Por fim, permite uma abstração reutilizável do sistema em situações diferentes com características similares.

**Galera, uma forma de organizar a arquitetura de um sistema complexo em partes menores é por meio da utilização de camadas de software**. Seguindo essa abordagem, cada camada corresponderá a um conjunto de funcionalidades de um sistema de software – sendo que as funcionalidades de alto nível dependerão das funcionalidades de baixo nível.

A separação em camadas fornece um nível de abstração através do agrupamento lógico de subsistemas relacionados entre si. **Parte-se do princípio de que as camadas de abstração mais altas devem depender das camadas de abstração mais baixas**. Portanto, isso permite que o sistema de software seja mais portátil e modificável.

**É importante salientar que mudanças em uma camada mais baixa, que não afetem a sua interface, não implicarão mudanças nas camadas mais superiores**. Do mesmo modo, as mudanças em uma camada mais alta, que não impliquem a criação de um novo serviço em uma camada mais baixa, não afetarão as camadas mais inferiores.

A arquitetura em camadas permite melhor separação de responsabilidades; decomposição de complexidade; encapsulamento de implementação; maior reuso e extensibilidade. No entanto, não são apenas benefícios! **Elas podem penalizar o desempenho do sistema e aumentar o esforço ou complexidade de desenvolvimento do software**.

**As camadas encapsulam bem algumas coisas, mas não todas. Isso, às vezes, resulta em alterações em cascata**. O exemplo clássico disto em uma aplicação corporativa em camadas é o acréscimo de um campo que precise ser mostrado na interface com o usuário e deva estar no banco de dados e assim deva também ser acrescentado a cada camada entre elas.

**Camadas extras podem prejudicar o desempenho**. Em cada camada, os dados precisam, tipicamente, ser transformados de uma representação para outra. O encapsulamento de uma função subjacente, no entanto, muitas vezes lhe dá ganhos de eficiência que mais do que



compensam esse problema. Uma camada que controla transações pode ser otimizada e então tornará tudo mais rápido.

Galera, camadas extras não necessariamente prejudicam o desempenho! Em geral, é isso que acontece. Porém, como foi dito, o encapsulamento de funções pode muitas vezes gerar ganhos de desempenho. Esse assunto é polêmico e já caiu em prova, portanto vocês devem ter atenção! **Bem, acredito que isso seja suficiente para posteriormente entender melhor cada arquitetura.**

**Funções do Arquiteto de Software:** priorizar casos de uso; análise arquitetural; construir prova de conceito arquitetural; estruturar o modelo de implementação; incorporar elementos de design; descrever a distribuição; identificar mecanismos de design; desenvolver guia de programação; identificar elementos de design; descrever a arquitetura em tempo de execução; desenvolver guia de design.

**(MPOG/ATI – 2015)** Embora normalmente os sistemas desenvolvidos se baseiem em padrões de arquitetura, cada um deles tem arquitetura totalmente específica, em consequência dos seus requisitos.

**Comentários:** De fato, eu posso usar padrões de arquitetura, tais como uma arquitetura em camadas, uma arquitetura distribuída, uma arquitetura mainframe ou uma arquitetura orientada a serviços. Embora cada sistema tenha uma arquitetura baseada em seus requisitos, elas não são TOTALMENTE específicas. (Errado).



## 2 – COESÃO E ACOPLAMENTO

Bem, falemos brevemente sobre esses dois importantes princípios de engenharia de software: coesão e acoplamento! Eles são fundamentais no conceito de arquitetura de software. Logo, preciso que vocês decorem a seguinte frase como se fosse um mantra: ***Uma boa arquitetura de software deve ter componentes de projeto com baixo acoplamento e alta coesão.***

*Como é, professor?* Acoplamento trata do nível de dependência entre módulos ou componentes de um software. Já a Coesão trata do nível de responsabilidade de um módulo em relação a outros. *Professor, por que é bom ter baixo acoplamento?* Porque se os **módulos pouco dependem um do outro, modificações de um não afetam os outros, além de não prejudicar o reúso.**

**Se esse princípio não for observado durante a construção da arquitetura de um sistema de software, pode haver problemas sérios de manutenção futura!** *Professor, por que é bom ter alta coesão?* Porque se os módulos têm responsabilidades claramente definidas, eles serão altamente reusáveis, independentes e simples de entender.



A Coesão está ligada ao princípio da responsabilidade única, que diz que uma classe deve ter uma e apenas uma responsabilidade e realizá-la de maneira satisfatória, i.e., a força funcional relativa de um módulo ou componente de software. **O acoplamento está ligado ao grau de conexão**



**entre módulos em uma estrutura de software.** Vejamos os tipos de coesão da pior para a melhor:

Vejamos os tipos de acoplamento:

| TIPO DE ACOPLAMENTO                    | DESCRIÇÃO                                                                                                                                                                        |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACOPLAMENTO POR CONTEÚDO               | Ocorre quando um módulo faz uso de estruturas de dados ou de controle mantidas no escopo de outro módulo.                                                                        |
| ACOPLAMENTO COMUM                      | Ocorre quando um conjunto de módulos acessa uma área global de dados.                                                                                                            |
| ACOPLAMENTO POR CONTROLE               | Ocorre quando módulos passam decisões de controle a outros módulos.                                                                                                              |
| ACOPLAMENTO POR DADOS                  | Ocorre quando apenas uma lista de dados simples é passada como parâmetro de um módulo para o outro, com uma correspondência um-para-um de itens.                                 |
| ACOPLAMENTO POR CHAMADAS DE ROTINAS    | Ocorre quando uma operação chama outra. Nesse nível de acoplamento, é comum e, quase sempre, necessário. Entretanto, realmente aumenta a conectividade de um sistema.            |
| ACOPLAMENTO POR USO DE TIPOS           | Ocorre quando o Componente A usa um tipo de dados definido em um Componente B. Se a definição de tipo mudar, todo componente que usa a definição também terá de ser alterado.    |
| ACOPLAMENTO POR INCLUSÃO OU IMPORTAÇÃO | Ocorre quando um Componente A importa ou inclui um pacote ou o conteúdo do Componente B.                                                                                         |
| ACOPLAMENTO EXTERNO                    | Ocorre quando um componente se comunica ou colabora com componentes de infraestrutura. Embora seja necessário, deve-se limitar a um pequeno número de componentes em um sistema. |

No contexto do projeto de componentes para sistemas orientados a objetos, coesão implica um componente ou classe encapsular apenas atributos e operações que estejam intimamente relacionados entre si e com a classe ou o componente em si. **A comunicação e a colaboração são elementos essenciais de qualquer sistema orientado a objetos.**

Há, entretanto, o lado sinistro desse importante (e necessária) característica. Como o volume de comunicação e colaboração aumenta (i.e., à medida que o grau de conexão entre as classes aumenta), a complexidade do sistema também cresce. **E à medida que a complexidade aumenta, a dificuldade de implementação, testes e manutenção do software também aumentam.**

**O acoplamento é uma medida qualitativa do grau com que as classes estão ligadas entre si.** Conforme as classes (e os componentes) se tornam mais interdependentes, o acoplamento aumenta. Um objetivo importante no projeto de componentes é manter o acoplamento o mais baixo possível. *Entenderam direito?* Agora vamos ver alguns exercícios!



## EXERCÍCIOS COMENTADOS

1. (CESPE - 2010 - ABIN - Oficial Técnico De Inteligência - Área de Desenvolvimento e Manutenção de Sistemas) Em sistemas de grande porte, um único requisito pode ser implementado por diversos componentes; cada componente, por sua vez, pode incluir elementos de vários requisitos, o que facilita o seu reúso, pois os componentes implementam, normalmente, uma única abstração do sistema.

### Comentários:

Pessoal, lembrem-se da coesão e acoplamento? Um componente que inclui elementos de vários requisitos tem baixa coesão, reduzindo sua reusabilidade.

**Gabarito:** Errado

2. (CESPE - 2013 - INPI - Analista de Planejamento - Desenvolvimento e Manutenção de Sistemas) De acordo com os princípios da engenharia de software relacionados à independência funcional, os algoritmos devem ser construídos por módulos visando unicamente ao alto acoplamento e à baixa coesão, caso a interface entre os módulos dê-se pela passagem de dados.

### Comentários:

Não, está invertido! Visa-se ao baixo acoplamento e à alta coesão!

**Gabarito:** Errado

3. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Análise de Sistemas) De acordo com o princípio da coesão de classes, cada classe deve representar uma única entidade bem definida no domínio do problema. O grau de coesão diminui com o aumento contínuo de código de manutenção nas classes.

### Comentários:

Uma classe deve ter uma única responsabilidade e executá-la de maneira satisfatória. Além disso, o grau de coesão tende a diminuir com o aumento contínuo de código de manutenção nas classes, mas isso não é regra.

**Gabarito:** Certo



4. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Análise de Sistemas) O acoplamento de métodos expressa o fato de que qualquer método deve ser responsável somente por uma tarefa bem definida.

**Comentários:**

Acoplamento? Não! Na verdade, é a coesão!

**Gabarito:** Errado

5. (CESPE - 2011 - MEC - Gerente de Projetos) A independência dos componentes é um dos atributos que reflete a qualidade do projeto. O grau de independência pode ser medido a partir dos conceitos de acoplamento e coesão, os quais, idealmente, devem ser alto e baixo, respectivamente.

**Comentários:**

Na verdade, é o contrário! Devem ser baixo e alto, respectivamente.

**Gabarito:** Errado

6. (CESPE - 2010 – INMETRO - Pesquisador) A coesão e o acoplamento são formas de se avaliar se a segmentação de um sistema em módulos ou em componentes foi eficiente. Acerca da aplicação desses princípios, assinale a opção correta.

- a) O baixo acoplamento pode melhorar a manutibilidade dos sistemas, pois ele está associado à criação de módulos como se fossem caixas-pretas.
- b) Os componentes ou os módulos devem apresentar baixa coesão e um alto grau de acoplamento.
- c) Os componentes ou os módulos devem ser fortemente coesos e fracamente acoplados.
- d) Um benefício da alta coesão é permitir realizar a manutenção em um módulo sem se preocupar com os detalhes internos dos demais módulos.
- e) A modularização do programa em partes especializadas pode aumentar a qualidade desses componentes, mas pode prejudicar o seu reaproveitamento em outros programas.

**Comentários:**

(a) Errado. Na verdade, essa é uma função da coesão e, não, do acoplamento; (b) Errado. Os componentes ou os módulos devem apresentar alta coesão e baixo grau de acoplamento; (c)





Correto. Essa é a regra geral: alta coesão e baixo acoplamento; (d) Errado. Esse é um benefício do baixo acoplamento; (e) Errado. Pelo contrário, quanto mais especializado, maior a coesão e maior a probabilidade de reaproveitamento em outros programas.

**Gabarito:** Certo

**7. (FCC - 2010 - TRT - 20ª REGIÃO (SE) - Analista Judiciário - Tecnologia da Informação)** No desenvolvimento de sistemas, no âmbito das relações intermodulares entre as classes, diz-se que o programa está bem estruturado quando há:

- a) maior coesão e maior acoplamento.
- b) menor coesão e maior acoplamento.
- c) menor coesão e menor acoplamento.
- d) maior coesão e menor acoplamento.
- e) apenas coesão ou apenas acoplamento.

**Comentários:**

Nosso mantra: alta coesão e baixo acoplamento!

**Gabarito:** D

**8. (FCC - 2010 - TCM-PA - Técnico em Informática)** Extensão natural do conceito de ocultação de informações, que diz: "um módulo deve executar uma única tarefa dentro do procedimento de software, exigindo pouca interação com procedimentos que são executados em outras partes de um programa", é o conceito de:

- a) coesão.
- b) enfileiramento.
- c) acoplamento.
- d) visibilidade.
- e) recursividade.

**Comentários:**

Módulo deve executar uma única tarefa? Divisão de responsabilidades, i.e., coesão.

**Gabarito:** A

**9. (FCC - 2008 - TRT - 18ª Região (GO) - Técnico Judiciário - Tecnologia da Informação)** Visando obter maior independência funcional, é adequado que o esforço seja direcionado ao projeto de módulos:



- a) que não usem estruturas de seleção.
- b) cujas tarefas tenham elevada coesão.
- c) cujas tarefas tenham coesão procedimental.
- d) que não usem estruturas de repetição.
- e) cujas tarefas tenham coesão lógica.

#### Comentários:

Maior independência funcional ocorre com alta coesão!

**Gabarito:** B

**10. (FCC - 2009 – TRT/03 – Analista de Sistemas)** Considerando o conjunto de tarefas que se relacionam em um módulo e o espectro de medidas da força funcional relativa dos módulos (coesão), a respectiva sequência, da pior para a melhor, é:

- a) sequencial, temporal e lógica.
- b) procedimental, coincidental e funcional.
- c) temporal, lógica e sequencial.
- d) temporal, comunicacional e sequencial.
- e) procedimental, funcional e lógica.

#### Comentários:

**Indesejáveis:** Coincidental, Lógica e Temporal; **Intermediárias:** Procedural, Comunicacional e Sequencial; **Desejável:** Funcional. Então, do pior para a melhor, temos: Temporal, Comunicacional e Sequencial.

**Gabarito:** D

**11. (UFG - 2017 – SANEAGO – Analista de Sistemas)** O emprego de boas práticas de projeto (design) de software visa resultar em um código:

- a) altamente acoplado e altamente coeso.
- b) altamente acoplado e fracamente coeso.
- c) fracamente acoplado e altamente coeso.
- d) fracamente acoplado e fracamente coeso.

#### Comentários:





Regra de Ouro de uma Arquitetura de Software: alta/forte coesão e baixo/fraco acoplamento!

**Gabarito: C**



## LISTA DE EXERCÍCIOS

1. **(CESPE - 2010 - ABIN - Oficial Técnico De Inteligência - Área de Desenvolvimento e Manutenção de Sistemas)** Em sistemas de grande porte, um único requisito pode ser implementado por diversos componentes; cada componente, por sua vez, pode incluir elementos de vários requisitos, o que facilita o seu reúso, pois os componentes implementam, normalmente, uma única abstração do sistema.
2. **(CESPE - 2013 - INPI - Analista de Planejamento - Desenvolvimento e Manutenção de Sistemas)** De acordo com os princípios da engenharia de software relacionados à independência funcional, os algoritmos devem ser construídos por módulos visando unicamente ao alto acoplamento e à baixa coesão, caso a interface entre os módulos dê-se pela passagem de dados.
3. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Análise de Sistemas)** De acordo com o princípio da coesão de classes, cada classe deve representar uma única entidade bem definida no domínio do problema. O grau de coesão diminui com o aumento contínuo de código de manutenção nas classes.
4. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Análise de Sistemas)** O acoplamento de métodos expressa o fato de que qualquer método deve ser responsável somente por uma tarefa bem definida.
5. **(CESPE - 2011 - MEC - Gerente de Projetos)** A independência dos componentes é um dos atributos que reflete a qualidade do projeto. O grau de independência pode ser medido a partir dos conceitos de acoplamento e coesão, os quais, idealmente, devem ser alto e baixo, respectivamente.
6. **(CESPE - 2010 – INMETRO - Pesquisador)** A coesão e o acoplamento são formas de se avaliar se a segmentação de um sistema em módulos ou em componentes foi eficiente. Acerca da aplicação desses princípios, assinale a opção correta.
  - a) O baixo acoplamento pode melhorar a manutibilidade dos sistemas, pois ele está associado à criação de módulos como se fossem caixas-pretas.
  - b) Os componentes ou os módulos devem apresentar baixa coesão e um alto grau de acoplamento.
  - c) Os componentes ou os módulos devem ser fortemente coesos e fracamente acoplados.
  - d) Um benefício da alta coesão é permitir realizar a manutenção em um módulo sem se preocupar com os detalhes internos dos demais módulos.



e) A modularização do programa em partes especializadas pode aumentar a qualidade desses componentes, mas pode prejudicar o seu reaproveitamento em outros programas.

**7. (FCC - 2010 - TRT - 20ª REGIÃO (SE) - Analista Judiciário - Tecnologia da Informação)** No desenvolvimento de sistemas, no âmbito das relações intermodulares entre as classes, diz-se que o programa está bem estruturado quando há:

- a) maior coesão e maior acoplamento.
- b) menor coesão e maior acoplamento.
- c) menor coesão e menor acoplamento.
- d) maior coesão e menor acoplamento.
- e) apenas coesão ou apenas acoplamento.

**8. (FCC - 2010 - TCM-PA - Técnico em Informática)** Extensão natural do conceito de ocultação de informações, que diz: "um módulo deve executar uma única tarefa dentro do procedimento de software, exigindo pouca interação com procedimentos que são executados em outras partes de um programa", é o conceito de:

- a) coesão.
- b) enfileiramento.
- c) acoplamento.
- d) visibilidade.
- e) recursividade.

**9. (FCC - 2008 - TRT - 18ª Região (GO) - Técnico Judiciário - Tecnologia da Informação)** Visando obter maior independência funcional, é adequado que o esforço seja direcionado ao projeto de módulos:

- a) que não usem estruturas de seleção.
- b) cujas tarefas tenham elevada coesão.
- c) cujas tarefas tenham coesão procedimental.
- d) que não usem estruturas de repetição.
- e) cujas tarefas tenham coesão lógica.

**10. (FCC - 2009 – TRT/03 – Analista de Sistemas)** Considerando o conjunto de tarefas que se relacionam em um módulo e o espectro de medidas da força funcional relativa dos módulos (coesão), a respectiva sequência, da pior para a melhor, é:

- a) sequencial, temporal e lógica.
- b) procedimental, coincidental e funcional.
- c) temporal, lógica e sequencial.
- d) temporal, comunicacional e sequencial.
- e) procedimental, funcional e lógica.



**11. (UFG - 2017 – SANEAGO – Analista de Sistemas)** O emprego de boas práticas de projeto (design) de software visa resultar em um código:

- a) altamente acoplado e altamente coeso.
- b) altamente acoplado e fracamente coeso.
- c) fracamente acoplado e altamente coeso.
- d) fracamente acoplado e fracamente coeso.



## GABARITO

1. ERRADO
2. ERRADO
3. CORRETO
4. ERRADO
5. ERRADO
6. CORRETO
7. LETRA D
8. LETRA A
9. LETRA B
10. LETRA D
11. LETRA C



### 3 – ARQUITETURA EM CAMADAS

**Inicialmente, os aplicativos eram combinados em uma única camada, incluindo Banco de Dados, Lógica do Aplicativo e Interface de Usuário.** O aplicativo, em geral, era executado em um computador de grande porte, e os usuários o acessavam por meio de *terminais burros* que podiam apenas realizar a entrada e exibição de dados. Essa abordagem tem o benefício de ser mantida por um administrador central.

As arquiteturas de uma camada têm um grave empecilho: os usuários esperam interfaces gráficas que exigem muito mais poder computacional do que o dos simples terminais burros. A computação centralizada de tais interfaces exige muito mais poder computacional do que um único servidor tem disponível, **e assim as arquiteturas de uma camada não consegue suportar milhares de usuários.**

**Temos, então, a Arquitetura Cliente-Servidor de duas camadas.** Para explicá-la, precisamos passar alguns conceitos básicos. Bem, ela é organizada como um conjunto de serviços, além de servidores e clientes associados que os acessam e os usam. Compõem esse modelo: servidores, que oferecem serviços; clientes, que solicitam os serviços; e uma rede que permite aos clientes acessarem esses serviços.

O processamento da informação é dividido em módulo ou processos distintos, sendo uma abordagem de computação que separa os processos em plataformas independentes que interagem, **permitindo que os recursos sejam compartilhados enquanto se obtém o máximo de benefício de cada dispositivo diferente.** Os principais componentes desse modelo são:

- **Servidores:** que oferecem serviços para outros subsistemas (Ex: Servidores de Impressão, Servidores de Arquivos, Servidor de Compilação, etc).
- **Clientes:** que solicita os serviços oferecidos pelos servidores. Geralmente são independentes, podendo ser executados simultaneamente.
- **Rede:** que permite aos clientes acessarem esses serviços. Quando clientes e servidores podem ser executados em uma única máquina, não são necessários.

**Clientes iniciam/terminam a comunicação com servidores, solicitando/terminando serviços distribuídos;** eles não se comunicam com outros clientes diretamente; eles são responsáveis pela entrada e saída de dados e comunicação com o usuário; eles tornam a rede transparente ao usuário; em geral, são computadores pessoais conectados a uma rede.

**Servidores realizam uma execução contínua; eles recebem e respondem solicitações dos clientes;** não se comunicam com outros servidores diretamente; prestam serviços distribuídos;

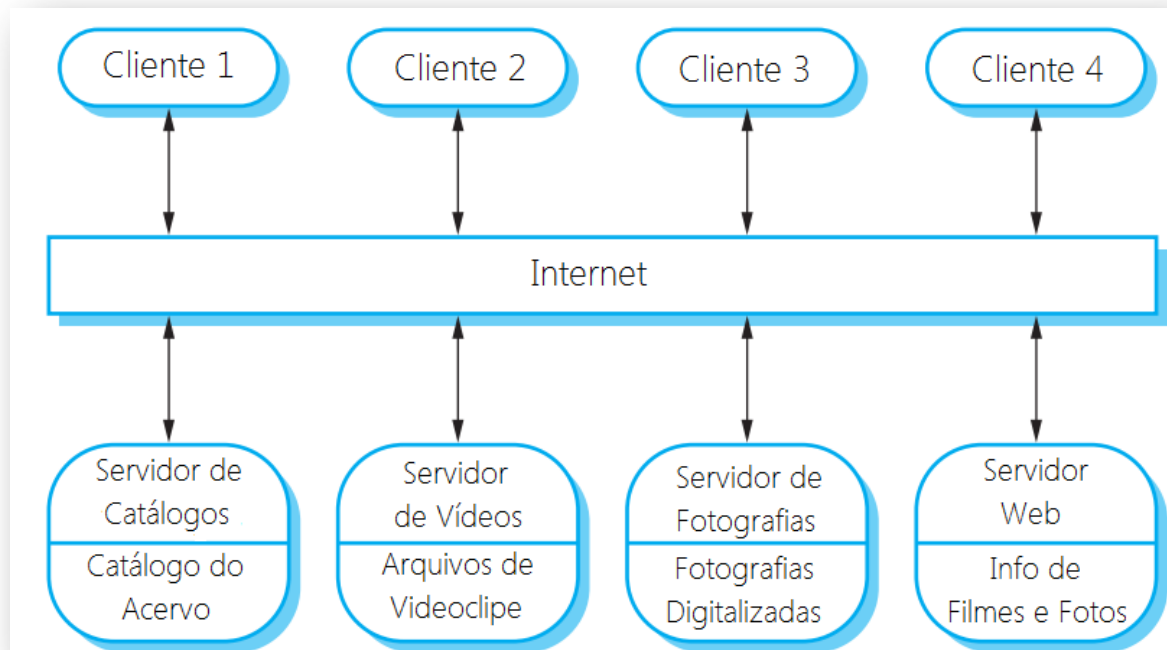




atende a diversos clientes simultaneamente; em geral, possuem um poder de processamento e armazenamento mais alto que de um cliente.

Os clientes talvez precisem saber os nomes dos servidores e os serviços que eles fornecem, mas os servidores não precisam saber sobre os clientes. Eles acessam os serviços pelo servidor por meio de chamadas remotas de procedimento usando um protocolo Request-Reply (Ex: HTTP).

**Essencialmente, um cliente faz um pedido a um servidor e espera até receber uma resposta.**



**Essencialmente, um cliente faz um pedido a um servidor e espera até receber uma resposta.**

A imagem acima mostra um exemplo de sistema baseado no modelo cliente-servidor. Ele é multiusuário e baseado na Web, para fornecer um acervo de filmes e fotografias. Nesse sistema, vários servidores gerenciam e apresentam tipos diferentes de mídia.

Os frames do vídeo precisam ser transmitidos rapidamente em sincronia, mas com uma resolução relativamente baixa. Podem ser comprimidos em um repositório e, assim, o servidor pode cuidar da compressão/descompressão do vídeo. **Fotografias devem estar em alta resolução, portanto devem ser mantidas em um servidor dedicado.**

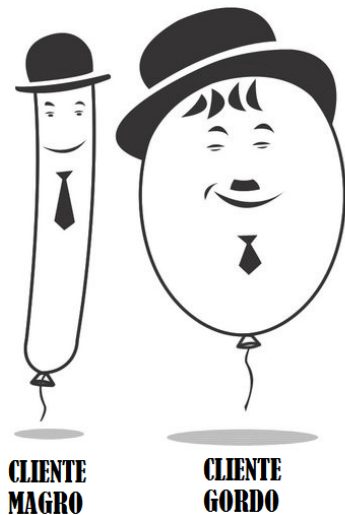
O catálogo deve ser capaz de lidar com várias consultas e de fornecer links para Sistemas Web de informações com dados do filme e videoclipe, e um sistema de e-commerce que apoie a venda desses. **O programa cliente é simplesmente uma interface integrada com o usuário, construída com o uso de um navegador Web para esses serviços.**

**A vantagem mais importante de um modelo cliente-servidor é que ele é uma arquitetura distribuída.** O uso efetivo de sistemas em rede pode ser feito com muitos processadores



distribuídos. É fácil adicionar um novo servidor e integrá-lo ao restante do sistema ou atualizar servidores de maneira transparente sem afetar outras partes do sistema.

As arquiteturas cliente-servidor de duas camadas podem ter duas formas: Cliente-Magro ou Cliente-Gordo. *Como é isso, professor?*



**No Modelo Cliente-Magro**, todo processamento da aplicação e o gerenciamento de dados é realizado no servidor. O cliente é responsável somente por executar o software de apresentação, portanto é magro!

**No Modelo Cliente-Gordo**, o servidor somente é responsável pelo gerenciamento de dados e o software do cliente implementa a lógica da aplicação e as interações com os usuários, portanto é gordo!

**As principais vantagens dos clientes magros são:** baixo custo de administração; facilidade de proteção; baixo custo de hardware; menor custo para licenciamento de softwares; baixo consumo de energia; resistência a ambientes hosts; menor dissipação de calor para o ambiente; mais silencioso que um PC convencional; mais ágil para rodar planilhas complexas; entre outros.

**As principais desvantagens dos clientes magros são:** se o servidor der problema e não houver redundância, todos os clientes-magros ficarão inoperantes; necessita de maior largura de banda na rede em que é empregado; em geral, possui um pior tempo de resposta, uma vez que usam o servidor para qualquer transação; apresenta um apoio transacional menos robusto; entre outros.

**As principais vantagens dos clientes gordos são:** necessitam de requisitos mínimos para servidores; apresenta uma performance multimídia melhor; possui maior flexibilidade; algumas situações se beneficiam bastante, tais como jogos eletrônicos, em que se beneficiam por conta de sua alta capacidade de processamento e de hardware específico.

**As principais desvantagens dos clientes gordos são:** não há um local central para atualizar e manter a lógica do negócio, uma vez que o código do aplicativo é executado em vários locais de cliente; é exigida uma grande confiança entre o servidor e os clientes por conta do banco de dados; não suporta bem o crescimento do número de clientes.

**Comparada à arquitetura de uma camada, as arquiteturas de duas camadas separam fisicamente a interface do usuário da camada de gerenciamento de dados.** Para implementar arquiteturas de duas camadas, não se pode mais ter terminais burros no lado do cliente; precisa-se de computadores que executem código de apresentação sofisticado (e, possivelmente, a lógica do aplicativo).

Sistemas Cliente-Servidor em duas camadas foram dominantes durante aproximadamente toda a década de noventa e são utilizados até hoje. Todavia, **para minimizar o impacto de mudanças nas aplicações, decidiu-se separar a camada de negócio da camada de interface gráfica**, gerando três camadas<sup>1</sup>: Camada de Apresentação, Camada Lógica do Negócio e Camada de Acesso a Dados.

- **Camada de Apresentação:** também chamada Camada de Interface, possui classes que contêm funcionalidades para visualização dos dados pelos usuários. Ela tem o objetivo de exibir informações ao usuário e traduzir ações do usuário em requisições às demais partes dos sistemas. O amplo uso da internet tornou as interfaces com base na web crescentemente populares.
- **Camada de Negócio:** também chamada Camada Lógica ou de Aplicação, possui classes que implementam as regras de negócio no qual o sistema será implantado. Ela realiza cálculos com base nos dados armazenados ou nos dados de entrada, decidindo que parte da camada de acesso de ser ativada com base em requisições provenientes da camada de apresentação.
- **Camada de Dados:** possui classes que se comunicam com outros sistemas para realizar tarefas ou adquirir informações para o sistema. Tipicamente, essa camada é implementada utilizando algum mecanismo de armazenamento persistente. Pode haver uma subcamada dentro desta camada chamada Camada de Persistência ou Camada de Acesso.

Nessa arquitetura, a apresentação, o processamento de aplicações e o gerenciamento de dados são processos logicamente separados executados por processadores diferentes. Seu uso permite a otimização da transferência de informações entre o servidor web e o de banco de dados. **As comunicações entre esses sistemas podem usar protocolos rápidos de comunicações de baixo nível.**

**Um middleware eficiente que apoia consultas de banco de dados em SQL (Structured Query Language) é usado para cuidar da recuperação de informações do banco de dados.** Em alguns casos, é adequado estender o modelo cliente-servidor de três camadas para uma variante com várias camadas, na qual servidores adicionais são incorporados ao sistema.

Esses sistemas podem ser usados quando as aplicações necessitam acessar e usar dados de bancos de dados diferentes. Nesse caso, um servidor de integração é colocado entre o servidor de aplicações e os servidores de banco de dados. **O servidor de integração coleta os dados distribuídos e apresenta-os como se fossem provenientes de um único banco de dados.**

**Um sistema de operações bancárias online é um exemplo de arquitetura cliente-servidor de três camadas.** O banco de dados de clientes fornece serviços de gerenciamento de dados; um

---

<sup>1</sup> Galera, diz-se Arquitetura em Três Camadas, 3-Layers Architecture ou 3-tiers Architecture. Apesar de muitas pessoas usarem os dois termos indiferentemente, eles não são iguais: Layers são camadas lógicas, i.e., pode haver três layers em uma única máquina e os Tiers são camadas físicas, i.e., pode haver apenas um tier por máquina. Entenderam? ;)



servidor web fornece os serviços de aplicação, como recursos para transferir dinheiro, gerar extratos, pagar contas, etc; e uma interface gráfica amigável fornece a apresentação dos dados ao cliente.

O próprio computador com um browser é o cliente. **Esse sistema é escalonável, pois é relativamente fácil adicionar novos servidores web quando o número de clientes aumenta.** O uso de uma arquitetura de três camadas, nesse caso, permite a otimização da transferência de informações entre o servidor web e o servidor de banco de dados.

As comunicações entre esses sistemas podem usar protocolos de comunicações de baixo nível. Um middleware eficiente que apoia consultas de banco de dados é usado para cuidar da recuperação de informações do banco de dados. **A arquitetura de três camadas que distribuem o processamento de aplicações entre os clientes são mais escalonáveis.**

**O tráfego de rede é reduzido em comparação com arquiteturas cliente-magro de duas camadas.** O processamento de aplicações é mais volátil e pode ser facilmente atualizado, pois está no centro. O processamento pode ser distribuído entre os servidores de lógica de aplicações e de gerenciamento de dados, conseguindo respostas mais rápidas.

*Vamos resumir?* Bem, havia a Arquitetura de uma Camada ou Monolítica! Nessa ocasião, um aplicativo era desenvolvido para ser usado em uma única máquina. **Esse aplicativo abarcava toda a funcionalidade em um único módulo**, i.e., a interface com usuário, lógica de aplicação e acesso a bancos de dados estavam presentes em um mesmo lugar.

Portanto a necessidade de compartilhar a lógica de acesso a dados entre vários usuários simultâneos fez surgir o modelo de arquitetura cliente-servidor em duas camadas. Dessa forma, **a base de dados foi colocada em uma máquina específica**, separada das máquinas que executavam, de fato, as aplicações ou em uma mesma máquina, porém em processadores diferentes.

Nessa arquitetura, **os aplicativos eram instalados em estações clientes contendo toda a apresentação e lógica de aplicação.** No entanto, quando havia alguma nova versão, tinha-se que reinstalar o software ou instalar a atualização em todas as (talvez milhares de) máquinas. Esse problema fez surgir a Arquitetura Cliente-Servidor em Três Camadas.

**Essa abordagem retirava a lógica de negócio da máquina do cliente e a centralizava em um servidor chamado Servidor de Aplicação.** Assim, o acesso ao Banco de Dados era feito seguindo regras de negócio contidas no Servidor de Aplicação, facilitando a atualização dos aplicativos. Contudo, atualizações na Camada de Apresentação precisavam ser distribuídas em todas as máquinas da rede.

Logo surgiu uma nova abordagem: Arquitetura Cliente-Servidor em Quatro Camadas! **Ela surgiu com a ideia de retirar a Apresentação do cliente e centralizá-la em um Servidor Web.** Assim, **não havia necessidade de instalar o aplicativo na máquina do cliente**, o acesso era feito por



meio de um navegador. As camadas eram: Dados, Aplicação, Apresentação e Cliente (Navegador Web).

**Grosso modo:** um usuário faz uma requisição por meio de um Navegador (Camada do Cliente), essa requisição é passada para um Servidor Web (Camada de Apresentação), que a processa e procura a regra de negócio correspondente no Servidor de Aplicação (Camada de Aplicação), que procura os dados no banco de dados (Camada de Dados).



## EXERCÍCIOS COMENTADOS

1. (CESPE - 2013 – STF - Analista de Sistemas) Quanto maior for o número de camadas, menor será o desempenho do software como um todo.

### Comentários:

Esse é um assunto polêmico! Em geral, é isso que acontece, i.e., quanto mais camadas, mais overhead. No entanto, há casos em que isso não é válido! Portanto, essa questão caberia recurso.

**Gabarito:** Certo

2. (CESPE - 2013 – STF - Analista de Sistemas) Cada camada tem comunicação (interface) com todas as demais camadas, tanto inferiores quanto superiores.

### Comentários:

Na verdade, as camadas só possuem comunicação/interface com suas camadas adjacentes.

**Gabarito:** Errado

3. (CESPE - 2013 – STF - Analista de Sistemas) Em uma arquitetura em camadas, a camada de persistência é responsável por armazenar dados gerados pelas camadas superiores e pode utilizar um sistema gerenciador de banco de dados para evitar, entre outros aspectos, anomalias de acesso concorrente dos dados e problemas de integridade de dados.

### Comentários:

Questão ótima! Persistência guarda os dados e, sim, pode utilizar um SGBD. *Por que?* Porque ele é capaz de tratar concorrência de dados e problemas de integridade! Tudo certinho...

**Gabarito:** Certo

4. (CESPE – 2009 – UNIPAMPA - Analista de Sistemas) Normalmente, a arquitetura em três camadas conta com as camadas de apresentação, de aplicação e de dados.

### Comentários:

Exato! Arquitetura em Três Camadas: Apresentação, Aplicação e Dados.

**Gabarito:** Certo



5. **(CESPE – 2009 – UNIPAMPA - Analista de Sistemas)** Em uma arquitetura em três camadas, na camada de aplicação, usualmente está um servidor de banco de dados que gerencia o conjunto de requisições.

**Comentários:**

Não, o Servidor de Banco de Dados usualmente se encontra na Camada de Dados.

**Gabarito:** Errado

6. **(CESPE – 2009 – UNIPAMPA - Analista de Sistemas)** O uso de middlewares é comum em aplicações de n camadas.

**Comentários:**

Exato! Utilizam-se middlewares para realizar uma comunicação mais eficiente.

**Gabarito:** Certo

7. **(CESPE – 2009 – UNIPAMPA - Analista de Sistemas)** Na camada de persistência dos dados em aplicações n camadas, podem ser utilizados o banco de dados orientado a objetos e o banco de dados relacionais.

**Comentários:**

É isso aí! É independente desse tipo de tecnologia.

**Gabarito:** Certo

8. **(CESPE – 2010 – BASA - Analista de Sistemas)** Nessa arquitetura (arquitetura multicamadas), quando são consideradas três camadas, a primeira camada deve ser implementada por meio do servidor de aplicação.

**Comentários:**

Por lógica, a primeira camada seria a Camada de Usuário ou a Camada de Dados. A Camada de Aplicação jamais seria a primeira camada.

**Gabarito:** Errado

9. **(CESPE – 2008 – IPEA - Analista de Sistemas)** Na arquitetura cliente-servidor com três camadas (three tier), a camada de apresentação, a camada de aplicação e o gerenciamento de dados ocorrem em diferentes máquinas. A camada de apresentação provê a interface do





usuário e interage com o usuário, sendo máquinas clientes responsáveis pela sua execução. A camada de aplicação é responsável pela lógica da aplicação, sendo executada em servidores de aplicação. Essa camada pode interagir com um ou mais bancos de dados ou fontes de dados. Finalmente, o gerenciamento de dados ocorre em servidores de banco de dados, que processam as consultas da camada de aplicação e enviam os resultados.

#### Comentários:

Questão perfeita! Refere-se a Camadas Físicas (Tiers), portando cada camada está localizada em uma máquina.

**Gabarito:** Certo

---

**10. (CESPE – 2010 – BASA - Analista de Sistemas)** Em arquitetura multicamadas, o servidor de aplicação nada mais é do que um programa que fica entre o aplicativo cliente e o sistema de gerenciamento de banco de dados.

#### Comentários:

É exatamente isso! Ele está entre a interface e os dados.

**Gabarito:** Certo

---

**11. (CESPE – 2010 – BASA- Analista de Sistemas)** Uma desvantagem dessa arquitetura (arquitetura multicamadas) é o aumento na manutenção da aplicação, pois alterações na camada de dados, por exemplo, acarretam mudanças em todas as demais camadas.

#### Comentários:

Não, a divisão em camadas reduz a manutenção da aplicação.

**Gabarito:** Errado

---

**12. (CESPE – 2011 – MEC – Analista de Sistemas)** O termo cliente é usado para designar uma parte distinta de um sistema de computador que gerencia um conjunto de recursos relacionados e apresenta sua funcionalidade para usuários e aplicativos.

#### Comentários:

Na verdade, a questão trata de um serviço!

**Gabarito:** Errado

---





**13. (CESPE – 2013 – FUB – Analista de Sistemas)** Aplicações cliente-servidor multicamadas são usualmente organizadas em três camadas principais: apresentação, lógica e periférico.

**Comentários:**

Na verdade, é apresentação, lógica e dados.

**Gabarito:** Errado

**14. (CESPE - 2008 – STJ - Analista de Sistemas)** A arquitetura de um sistema de software pode se basear em determinado estilo de arquitetura. Um estilo de arquitetura é um padrão de organização. No estilo cliente-servidor, o sistema é organizado como um conjunto de serviços, servidores e clientes associados que acessam e usam os serviços. Os principais componentes desse estilo são servidores que oferecem serviços e clientes que solicitam os serviços.

**Comentários:**

Exato! Trata-se de um conjunto de clientes e servidores que acessam e fornecem diversos serviços.

**Gabarito:** Certo

**15. (CESPE – 2009 – UNIPAMPA - Analista de Sistemas)** Nas aplicações cliente-servidor, em duas camadas, é simples acessar fontes de dados heterogêneas porque o legado de base de dados não precisa de drivers de conexões diferentes.

**Comentários:**

Na verdade, é necessário diversos drivers de conexões diferentes para acessar às bases de dados de fontes heterogêneas e não é simples! Em uma arquitetura cliente-servidor de três camadas, continua sendo necessário os drivers de conexões diferentes, porém é mais simples por haver uma camada responsável por gerenciar essas conexões.

**Gabarito:** Errado

**16. (CESPE – 2003 – TRE/RS - Analista de Sistemas)** Aplicações com arquitetura cliente-servidor são assimétricas, no sentido de que o cliente e o servidor possuem papéis diferentes na arquitetura de comunicações.

**Comentários:**

Perfeito, eles possuem papéis distintos.



**Gabarito:** Certo

---

**17. (CESPE – 2003 – TRE/RS - Analista de Sistemas)** O servidor, por possuir normalmente um hardware mais robusto, sempre deve executar a parte mais pesada do processamento.

**Comentários:**

Não! Por exemplo, quando se utiliza um cliente-gordo, a maior parte do processamento ocorre no cliente e, não, no servidor.

**Gabarito:** Errado

---

**18. (CESPE – 2003 – TRE/RS - Analista de Sistemas)** Do ponto de vista das funcionalidades de usuários, o servidor não precisa necessariamente de uma interface de usuário.

**Comentários:**

Perfeito, percebam que foi dito "*do ponto de vista das funcionalidades de usuários*". Uma vez que sejam executadas as funcionalidades esperadas, não há necessidade de uma interface de usuário.

**Gabarito:** Certo

---

**19. (CESPE – 2004 – SEAD/PA - Analista de Sistemas)** Em um modelo cliente-servidor em que o processamento é concentrado nos clientes e o armazenamento concentrado no servidor, observa-se uma baixa carga de tráfego na rede.

**Comentários:**

É um caso de Cliente-Gordo, em que o tráfego na rede é menor. No entanto nada garante que o tráfego na rede será baixo, ele pode ser altíssimo, porém é menor comparado ao Cliente-Magro.

**Gabarito:** Certo

---

**20. (CESPE – 2004 – SERPRO - Analista de Sistemas)** Uma das vantagens da arquitetura cliente-servidor é que parte da carga de processamento é retirada do servidor e colocada nos vários clientes.

**Comentários:**

Essa é uma das grandes vantagens da arquitetura cliente-servidor. Ela retira uma parte do processamento dos serviços (desonerando o servidor) e transfere para os clientes.



**Gabarito:** Certo

---

- 21. (CESPE – 2004 – STJ - Analista de Sistemas)** As camadas da arquitetura cliente-servidor de três camadas são: camada de interface de usuário, camada de regras de negócio e camada de acesso ao banco de dados.

**Comentários:**

Esses nomes mudam bastante, mas a questão está certa!

**Gabarito:** Certo

---

- 22. (CESPE – 2004 – STJ - Analista de Sistemas)** Na arquitetura cliente-servidor multicamadas, uma alteração na camada de acesso aos dados não afeta a camada de interface de usuário, desde que essas camadas estejam na mesma máquina.

**Comentários:**

Na verdade, mesmo que essas camadas estejam na mesma máquina, uma alteração na camada de acesso aos dados não afeta a camada de interface de usuário.

**Gabarito:** Errado

---

- 23. (CESPE – 2004 – STJ - Analista de Sistemas)** A arquitetura cliente-servidor multicamadas possui a vantagem de que a camada de interface de usuário pode se comunicar diretamente com qualquer outra camada, ou seja, não existe hierarquia entre camadas.

**Comentários:**

Não, a camada de interface se comunica diretamente apenas com a camada de negócio.

**Gabarito:** Errado

---

- 24. (CESPE – 2006 – DATAPREV - Analista de Sistemas)** Uma arquitetura cliente/servidor caracteriza-se pela separação do cliente, o usuário que acessa ou demanda informações, do servidor. Um exemplo típico é um navegador que acessa páginas na Internet. É uma arquitetura que permite o acesso a serviços remotos através de rede de computadores, e que tem como principal deficiência a falta de escalabilidade.

**Comentários:**

Na verdade, um dos benefícios da arquitetura cliente/servidor é a escalabilidade.



**Gabarito:** Errado

---

**25. (CESPE – 2006 – DATAPREV - Analista de Sistemas)** Arquiteturas cliente/servidor podem ser decompostas em mais de duas camadas. Uma configuração muito utilizada é aquela em que os clientes acessam informações por meio de servidores de aplicação, que por sua vez acessam servidores de banco de dados. Este tipo de arquitetura é conhecida como arquitetura em 3 camadas, ou three-tier.

**Comentários:**

Exato! O usuário utiliza um servidor de aplicação que acessa um servidor de banco de dados.

**Gabarito:** Certo

---

**26. (CESPE – 2007– TCU - Analista de Sistemas)** A arquitetura cliente-servidor tem por motivação sincronizar a execução de dois processos que devem cooperar um com outro. Assim, dadas duas entidades que queiram comunicar-se, uma deve iniciar a comunicação enquanto a outra aguarda pela requisição da entidade que inicia a comunicação.

**Comentários:**

Parece complicado, mas é simples! Uma entidade começa a comunicação e a outra aguarda a requisição da entidade iniciadora.

**Gabarito:** Certo

---

**27. (CESPE – 2009 – ANTAQ - Analista de Sistemas)** Os principais componentes da arquitetura cliente-servidor, que é um modelo de arquitetura para sistemas distribuídos, são o conjunto de servidores que oferecem serviços para outros subsistemas, como servidores de impressão e servidores de arquivos, o conjunto de clientes que solicitam os serviços oferecidos por servidores, e a rede que permite aos clientes acessarem esses serviços.

**Comentários:**

Perfeito, são clientes, servidores e uma rede que permite a comunicação.

**Gabarito:** Certo

---

**28. (CESPE – 2010 – BASA – Analista de Sistemas)** Em arquiteturas cliente-servidor multicamadas, na maior parte das aplicações, o browser é adotado como cliente universal.

**Comentários:**



Sim, pessoal! Nessa arquitetura, o browser (navegador) geralmente é o cliente.

**Gabarito:** Certo

---

**29. (CESPE – 2010 – BASA- Analista de Sistemas)** Em uma arquitetura cliente-servidor, os clientes compartilham dos recursos gerenciados pelos servidores, os quais também podem, por sua vez, ser clientes de outros servidores.

**Comentários:**

Exato! Os servidores gerenciam seus recursos que são compartilhados pelas dezenas, milhares, milhões de clientes. Além disso, um servidor pode acabar sendo um cliente de outro servidor.

**Gabarito:** Certo

---

**30. (CESPE – 2010 – BASA- Analista de Sistemas)** A Internet baseia-se na arquitetura cliente-servidor, na qual a parte cliente, executada no host local, solicita serviços de um programa aplicativo denominado servidor, que é executado em um host remoto.

**Comentários:**

Exato! A internet é isso: uma arquitetura cliente-servidor distribuída em que a parte cliente é executada no host local e solicita serviços de diversos servidores, que são executados em um host remoto.

**Gabarito:** Certo

---

**31. (CESPE – 2010 – BASA- Analista de Sistemas)** A arquitetura cliente-servidor viabiliza o uso simultâneo de diferentes dispositivos computacionais, do seguinte modo: cada um deles realiza a tarefa para a qual é mais capacitado, havendo a possibilidade de uma máquina ser cliente em uma tarefa e servidor em outra.

**Comentários:**

Apesar de a redação da questão estar confusa ("*tarefa para a qual é mais capacitado*"), a questão está certa em afirmar que uma máquina pode ser cliente em uma aplicação e servidora em outra.

**Gabarito:** Certo

---

**32. (CESPE – 2011 – MEC – Analista de Sistemas)** A arquitetura cliente/servidor proporciona a sincronização entre duas aplicações: uma aplicação permanece em estado de espera até que outra aplicação efetue uma solicitação de serviço.



### Comentários:

Exato! A redação é meio complicada, mas o que essa questão quis dizer é: uma aplicação (servidor) fica em estado de espera aguardando solicitações e outra aplicação (cliente) efetua a solicitação de serviços.

**Gabarito:** Certo

**33. (CESPE – 2011 – MEC – Analista de Sistemas)** A arquitetura cliente/servidor enseja o desenvolvimento de um sistema com, no máximo, duas camadas, quais sejam, cliente e servidor.

### Comentários:

Na verdade, pode ter quantas camadas forem necessárias.

**Gabarito:** Errado

**34. (CESPE – 2013 – FUB – Analista de Sistemas)** Entre as desvantagens de se executar todas as camadas de uma aplicação cliente-servidor no lado do servidor se destaca a dificuldade de atualização e correção da aplicação.

### Comentários:

Primeiro, se todas as camadas de uma aplicação cliente-servidor são executados no servidor, então não é uma arquitetura cliente-servidor. Executar a maioria das camadas no lado do cliente é uma dificuldade de atualização e correção de aplicação, na medida em que é necessária fazer essa manutenção em todas as máquinas clientes.

**Gabarito:** Errado

**35. (CESPE – 2017 – TRE/BA - Analista de Sistemas)** Com referência às arquiteturas multicamadas de aplicações para o ambiente web, assinale a opção correta.

a) Se, na camada de dados, for realizada uma alteração no banco de dados, o restante das camadas também será afetado.

b) O modelo de três camadas recebe essa denominação caso um sistema cliente-servidor seja desenvolvido mantendo-se a camada de negócio do lado do cliente e as camadas de apresentação e dados no lado do servidor.



- c) Cada camada é normalmente mantida em um servidor específico para tornar-se o mais escalonável e independente possível em relação a outras camadas, estando entre as suas principais características o eficiente armazenamento e a reutilização de recursos.
- d) O objetivo das arquiteturas multicamadas consiste na junção de responsabilidades entre os componentes das aplicações web, de modo a atender aos requisitos funcionais e não funcionais esperados pela aplicação.
- e) Na arquitetura de duas camadas — apresentação e armazenamento —, o computador que contiver a base de dados terá de ficar junto com os computadores que executarem as aplicações.

### Comentários:

(a) Errado, uma alteração no banco de dados alteraria apenas as classes da camada de dados, mas o restante da arquitetura não seria afetado por essa alteração; (b) Errado, ela recebe essa denominação quando um sistema cliente-servidor é desenvolvido retirando-se a camada de negócio do lado do cliente; (c) Correto. Cada camada desta arquitetura é normalmente mantida em um servidor específico para tornar-se mais escalonável e independente das demais. Cada camada é auto-contida o suficiente de forma que a aplicação pode ser dividida em vários computadores em uma rede distribuída; (d) Errado, o objetivo consiste na separação de responsabilidades entre os componentes das aplicações web, de modo que tenham alta coesão; (e) Errado. Nesta estrutura, a base de dados é colocada em uma máquina específica, separada das máquinas que executavam as aplicações.

**Gabarito:** Certo

### 36. (FCC - 2012 – TST - Analista de Sistemas) Uma arquitetura em camadas:

- a) possui apenas 3 camadas, cada uma realizando operações que se tornam progressivamente mais próximas do conjunto de instruções da máquina.
- b) tem, na camada mais interior, os componentes que implementam as operações de interface com o usuário.
- c) pode ser combinada com uma arquitetura centrada nos dados em muitas aplicações de bancos de dados.
- d) tem, como camada intermediária, o depósito de dados, também chamado de repositório ou quadro-negro.
- e) tem, na camada mais externa, os componentes que realizam a interface com o sistema operacional.



### Comentários:

(a) Errado. Não necessariamente possui três camadas – pode possuir 2, 3, 4, 5, etc; (b) Errado. Interface com o usuário é, em geral, na camada mais externa; (c) Correto. Não há nada que impeça isso; (d) Errado. O depósito de dados, em geral, fica na camada mais interna; (e) Errado. Interface com o sistema operacional, em geral, fica na camada mais interna.

**Gabarito:** C

**37. (FCC - 2015 – TCM/GO - Analista de Sistemas - Adaptada)** Quanto à Arquitetura em 3 Camadas, é necessário um arranjo que possibilite a reutilização do código e facilite sua manutenção e seu aperfeiçoamento. Deve-se separar Apresentação, Regra de Negócio e Acesso a Dados. Busca-se a decomposição de funcionalidades de forma a permitir aos desenvolvedores concentrarem-se em diferentes partes da aplicação durante a implementação.

### Comentários:

Perfeito! Essa é uma situação bastante comum em uma arquitetura em três camadas.

**Gabarito:** C

**38. (FCC - 2014 – TJ/AP - Analista de Sistemas)** Uma arquitetura muito comum em aplicações web é o Modelo Arquitetural 3 Camadas:

- I. Camada de Persistência.
- II. Camada de Lógica de Negócio.
- III. Camada de Apresentação.

Neste modelo, a correta associação dos componentes com as camadas é:

- a) I-Servidor de Banco de Dados - II-Servidor de Aplicação - III-Máquina Cliente.
- b) I-Servidor Web - II-Servidor Cliente - III-Servidor de Aplicação.
- c) I-Servidor Web - II-Servidor de Banco de Dados - III-Máquina Cliente.
- d) I-Servidor de Banco de Dados - II-Máquina Cliente - III-Servidor de Aplicação.
- e) I-Máquina Cliente - II-Servidor de Banco de Dados - III-Servidor Web.

### Comentários:

Servidor de Banco de Dados se associa com... Camada de Persistência;  
Servidor de Aplicação se associa com... Camada de Lógica de Negócio;  
Máquina Cliente se associa com... Camada de Apresentação.





**Gabarito:** A

**39.(FCC - 2012 – TRF/2ª - Analista de Sistemas)** São aspectos que podem caracterizar uma arquitetura cliente-servidor, estabelecida logicamente em 4 camadas:

I. A camada Cliente contém um navegador de Internet, caracterizado como cliente universal.

II. A camada de Lógica do Negócio se quebra em duas: camada de Aplicação e camada Web, em que o servidor Web representa a camada de Apresentação.

III. Na camada de Lógica do Negócio, o servidor de aplicação passa a utilizar middleware, para suporte a thin clients (PDA, celulares, smart cards, etc) e soluções baseadas em componentes, tais como, J2EE e .Net.

IV. Se, de um lado, a camada de Aplicação estabelece uma interface com a camada de Dados, do outro o faz com a camada Web e com os thin clients da camada Cliente.

Está correto o que consta em:

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

#### Comentários:

(I) Correto. Na Arquitetura em 4 Camadas, a Camada Cliente contém um navegador, caracterizado como cliente universal; (II) Errado. Camada de Lógica de Negócio é a Camada de Aplicação e a Camada Web não faz parte dela. Eu nunca ouvi falar nisso, mas a FCC disse que é correto! A diferença entre a Arquitetura em 4 Camadas e 3 Camadas é que a Apresentação não fica mais no cliente e, sim, na Camada Web. (III) Errado. Imagino que o middleware à que a questão se refere é o Servidor Web, mas ele ficaria na Camada Web fazendo o meio-campo para suportar thin clients e, não, na Camada de Lógica de Negócio – a FCC discorda! (IV) Correto. A Camada de Aplicação fica entre a Camada de Dados e a Camada Web.

**Gabarito:** E

**40.(FCC - 2010 – TRT/SE - Analista de Sistemas)** A arquitetura multicamadas divide-se em três camadas lógicas. São elas:

- a) Apresentação, Negócio e Acesso a Dados.



- b) Apresentação, Natureza e Acesso a Dados.
- c) Apresentação, Negócio e Alteração.
- d) Manipulação, Natureza e Acesso a Dados.
- e) Manipulação, Negócio e Acesso a Dados.

#### Comentários:

Easy! Apresentação, Negócio e Acesso a Dados.

**Gabarito:** A

**41. (FCC - 2010 – METRÔ/SP - Analista de Sistemas)** A arquitetura multicamadas divide-se em três camadas lógicas. São elas:

- a) Apresentação, Negócio e Alteração.
- b) Manipulação, Natureza e Acesso a Dados.
- c) Manipulação, Negócio e Acesso a Dados.
- d) Apresentação, Natureza e Acesso a Dados.
- e) Apresentação, Negócio e Acesso a Dados.

#### Comentários:

Easy! Apresentação, Negócio e Acesso a Dados. Vejam que as bancas se autocopiam – essa questão é quase idêntica a anterior.

**Gabarito:** E

**42. (FCC - 2008 – TRF/5ª - Analista de Sistemas)** Via de regra as divisões da arquitetura de software em três camadas orientam para níveis que especificam:

- a) os casos de uso, a estrutura dos dados e os processos de manutenção.
- b) a apresentação, as regras de negócio e os testes.
- c) a apresentação, os processos operacionais e a seqüência de execução.
- d) a apresentação, os componentes virtuais e a seqüência de execução.
- e) a apresentação, as regras de negócio e o armazenamento de dados.

#### Comentários:

Easy também, só mudaram as palavras! Apresentação, Regras de Negócio e Armazenamento de Dados.

**Gabarito:** E



**43. (FCC - 2015 – CNMP - Analista de Sistemas)** Há algumas variantes possíveis de arquitetura a serem utilizadas em um sistema de bancos de dados. Sobre essas variantes, é correto afirmar que:

- a) na arquitetura de 3 camadas, não há uma camada específica para a aplicação.
- b) a camada de apresentação da arquitetura de 2 camadas situa-se, usualmente, no servidor de banco de dados.
- c) na arquitetura de 3 camadas, a camada de servidor de banco de dados é denominada cliente.
- d) a arquitetura de 3 camadas é composta pelas camadas cliente, aplicação e servidor de banco de dados.
- e) na arquitetura de 2 camadas não há necessidade de uso de um sistema gerenciador de bancos de dados.

**Comentários:**

(a) Errado, é a camada intermediária; (b) Errado, fica na camada de apresentação; (c) Errado, é chamada camada de dados; (d) Correto, é exatamente isso; (e) Errado, é claro que há necessidade.

**Gabarito: D**



## LISTA DE EXERCÍCIOS

1. **(CESPE - 2013 – STF - Analista de Sistemas)** Quanto maior for o número de camadas, menor será o desempenho do software como um todo.
2. **(CESPE - 2013 – STF - Analista de Sistemas)** Cada camada tem comunicação (interface) com todas as demais camadas, tanto inferiores quanto superiores.
3. **(CESPE - 2013 – STF - Analista de Sistemas)** Em uma arquitetura em camadas, a camada de persistência é responsável por armazenar dados gerados pelas camadas superiores e pode utilizar um sistema gerenciador de banco de dados para evitar, entre outros aspectos, anomalias de acesso concorrente dos dados e problemas de integridade de dados.
4. **(CESPE – 2009 – UNIPAMPA - Analista de Sistemas)** Normalmente, a arquitetura em três camadas conta com as camadas de apresentação, de aplicação e de dados.
5. **(CESPE – 2009 – UNIPAMPA - Analista de Sistemas)** Em uma arquitetura em três camadas, na camada de aplicação, usualmente está um servidor de banco de dados que gerencia o conjunto de requisições.
6. **(CESPE – 2009 – UNIPAMPA - Analista de Sistemas)** O uso de middlewares é comum em aplicações de n camadas.
7. **(CESPE – 2009 – UNIPAMPA - Analista de Sistemas)** Na camada de persistência dos dados em aplicações n camadas, podem ser utilizados o banco de dados orientado a objetos e o banco de dados relacionais.
8. **(CESPE – 2010 – BASA - Analista de Sistemas)** Nessa arquitetura (arquitetura multicamadas), quando são consideradas três camadas, a primeira camada deve ser implementada por meio do servidor de aplicação.
9. **(CESPE – 2008 – IPEA - Analista de Sistemas)** Na arquitetura cliente-servidor com três camadas (three tier), a camada de apresentação, a camada de aplicação e o gerenciamento de dados ocorrem em diferentes máquinas. A camada de apresentação provê a interface do usuário e interage com o usuário, sendo máquinas clientes responsáveis pela sua execução. A camada de aplicação é responsável pela lógica da aplicação, sendo executada em servidores de aplicação. Essa camada pode interagir com um ou mais bancos de dados ou fontes de dados. Finalmente, o gerenciamento de dados ocorre em servidores de banco de dados, que processam as consultas da camada de aplicação e enviam os resultados.



10. **(CESPE – 2010 – BASA - Analista de Sistemas)** Em arquitetura multicamadas, o servidor de aplicação nada mais é do que um programa que fica entre o aplicativo cliente e o sistema de gerenciamento de banco de dados.
11. **(CESPE – 2010 – BASA- Analista de Sistemas)** Uma desvantagem dessa arquitetura (arquitetura multicamadas) é o aumento na manutenção da aplicação, pois alterações na camada de dados, por exemplo, acarretam mudanças em todas as demais camadas.
12. **(CESPE – 2011 – MEC – Analista de Sistemas)** O termo cliente é usado para designar uma parte distinta de um sistema de computador que gerencia um conjunto de recursos relacionados e apresenta sua funcionalidade para usuários e aplicativos.
13. **(CESPE – 2013 – FUB – Analista de Sistemas)** Aplicações cliente-servidor multicamadas são usualmente organizadas em três camadas principais: apresentação, lógica e periférico.
14. **(CESPE - 2008 – STJ - Analista de Sistemas)** A arquitetura de um sistema de software pode se basear em determinado estilo de arquitetura. Um estilo de arquitetura é um padrão de organização. No estilo cliente-servidor, o sistema é organizado como um conjunto de serviços, servidores e clientes associados que acessam e usam os serviços. Os principais componentes desse estilo são servidores que oferecem serviços e clientes que solicitam os serviços.
15. **(CESPE – 2009 – UNIPAMPA - Analista de Sistemas)** Nas aplicações cliente-servidor, em duas camadas, é simples acessar fontes de dados heterogêneas porque o legado de base de dados não precisa de drivers de conexões diferentes.
16. **(CESPE – 2003 – TRE/RS - Analista de Sistemas)** Aplicações com arquitetura cliente-servidor são assimétricas, no sentido de que o cliente e o servidor possuem papéis diferentes na arquitetura de comunicações.
17. **(CESPE – 2003 – TRE/RS - Analista de Sistemas)** O servidor, por possuir normalmente um hardware mais robusto, sempre deve executar a parte mais pesada do processamento.
18. **(CESPE – 2003 – TRE/RS - Analista de Sistemas)** Do ponto de vista das funcionalidades de usuários, o servidor não precisa necessariamente de uma interface de usuário.
19. **(CESPE – 2004 – SEAD/PA - Analista de Sistemas)** Em um modelo cliente-servidor em que o processamento é concentrado nos clientes e o armazenamento concentrado no servidor, observa-se uma baixa carga de tráfego na rede.
20. **(CESPE – 2004 – SERPRO - Analista de Sistemas)** Uma das vantagens da arquitetura cliente-servidor é que parte da carga de processamento é retirada do servidor e colocada nos vários clientes.



- 21. (CESPE – 2004 – STJ - Analista de Sistemas)** As camadas da arquitetura cliente-servidor de três camadas são: camada de interface de usuário, camada de regras de negócio e camada de acesso ao banco de dados.
- 22. (CESPE – 2004 – STJ - Analista de Sistemas)** Na arquitetura cliente-servidor multicamadas, uma alteração na camada de acesso aos dados não afeta a camada de interface de usuário, desde que essas camadas estejam na mesma máquina.
- 23. (CESPE – 2004 – STJ - Analista de Sistemas)** A arquitetura cliente-servidor multicamadas possui a vantagem de que a camada de interface de usuário pode se comunicar diretamente com qualquer outra camada, ou seja, não existe hierarquia entre camadas.
- 24. (CESPE – 2006 – DATAPREV - Analista de Sistemas)** Uma arquitetura cliente/servidor caracteriza-se pela separação do cliente, o usuário que acessa ou demanda informações, do servidor. Um exemplo típico é um navegador que acessa páginas na Internet. É uma arquitetura que permite o acesso a serviços remotos através de rede de computadores, e que tem como principal deficiência a falta de escalabilidade.
- 25. (CESPE – 2006 – DATAPREV - Analista de Sistemas)** Arquiteturas cliente/servidor podem ser decompostas em mais de duas camadas. Uma configuração muito utilizada é aquela em que os clientes acessam informações por meio de servidores de aplicação, que por sua vez acessam servidores de banco de dados. Este tipo de arquitetura é conhecida como arquitetura em 3 camadas, ou three-tier.
- 26. (CESPE – 2007– TCU - Analista de Sistemas)** A arquitetura cliente-servidor tem por motivação sincronizar a execução de dois processos que devem cooperar um com outro. Assim, dadas duas entidades que queiram comunicar-se, uma deve iniciar a comunicação enquanto a outra aguarda pela requisição da entidade que inicia a comunicação.
- 27. (CESPE – 2009 – ANTAQ - Analista de Sistemas)** Os principais componentes da arquitetura cliente-servidor, que é um modelo de arquitetura para sistemas distribuídos, são o conjunto de servidores que oferecem serviços para outros subsistemas, como servidores de impressão e servidores de arquivos, o conjunto de clientes que solicitam os serviços oferecidos por servidores, e a rede que permite aos clientes acessarem esses serviços.
- 28. (CESPE – 2010 – BASA – Analista de Sistemas)** Em arquiteturas cliente-servidor multicamadas, na maior parte das aplicações, o browser é adotado como cliente universal.
- 29. (CESPE – 2010 – BASA- Analista de Sistemas)** Em uma arquitetura cliente-servidor, os clientes compartilham dos recursos gerenciados pelos servidores, os quais também podem, por sua vez, ser clientes de outros servidores.



- 30. (CESPE – 2010 – BASA- Analista de Sistemas)** A Internet baseia-se na arquitetura cliente-servidor, na qual a parte cliente, executada no host local, solicita serviços de um programa aplicativo denominado servidor, que é executado em um host remoto.
- 31. (CESPE – 2010 – BASA- Analista de Sistemas)** A arquitetura cliente-servidor viabiliza o uso simultâneo de diferentes dispositivos computacionais, do seguinte modo: cada um deles realiza a tarefa para a qual é mais capacitado, havendo a possibilidade de uma máquina ser cliente em uma tarefa e servidor em outra.
- 32. (CESPE – 2011 – MEC – Analista de Sistemas)** A arquitetura cliente/servidor proporciona a sincronização entre duas aplicações: uma aplicação permanece em estado de espera até que outra aplicação efetue uma solicitação de serviço.
- 33. (CESPE – 2011 – MEC – Analista de Sistemas)** A arquitetura cliente/servidor enseja o desenvolvimento de um sistema com, no máximo, duas camadas, quais sejam, cliente e servidor.
- 34. (CESPE – 2013 – FUB – Analista de Sistemas)** Entre as desvantagens de se executar todas as camadas de uma aplicação cliente-servidor no lado do servidor se destaca a dificuldade de atualização e correção da aplicação.
- 35. (CESPE – 2017 – TRE/BA - Analista de Sistemas)** Com referência às arquiteturas multicamadas de aplicações para o ambiente web, assinale a opção correta.
- a) Se, na camada de dados, for realizada uma alteração no banco de dados, o restante das camadas também será afetado.
  - b) O modelo de três camadas recebe essa denominação caso um sistema cliente-servidor seja desenvolvido mantendo-se a camada de negócio do lado do cliente e as camadas de apresentação e dados no lado do servidor.
  - c) Cada camada é normalmente mantida em um servidor específico para tornar-se o mais escalonável e independente possível em relação a outras camadas, estando entre as suas principais características o eficiente armazenamento e a reutilização de recursos.
  - d) O objetivo das arquiteturas multicamadas consiste na junção de responsabilidades entre os componentes das aplicações web, de modo a atender aos requisitos funcionais e não funcionais esperados pela aplicação.
  - e) Na arquitetura de duas camadas — apresentação e armazenamento —, o computador que contiver a base de dados terá de ficar junto com os computadores que executarem as aplicações.
- 36. (FCC - 2012 – TST - Analista de Sistemas)** Uma arquitetura em camadas:





- a) possui apenas 3 camadas, cada uma realizando operações que se tornam progressivamente mais próximas do conjunto de instruções da máquina.
- b) tem, na camada mais interior, os componentes que implementam as operações de interface com o usuário.
- c) pode ser combinada com uma arquitetura centrada nos dados em muitas aplicações de bancos de dados.
- d) tem, como camada intermediária, o depósito de dados, também chamado de repositório ou quadro-negro.
- e) tem, na camada mais externa, os componentes que realizam a interface com o sistema operacional.

**37. (FCC - 2015 – TCM/GO - Analista de Sistemas - Adaptada)** Quanto à Arquitetura em 3 Camadas, é necessário um arranjo que possibilite a reutilização do código e facilite sua manutenção e seu aperfeiçoamento. Deve-se separar Apresentação, Regra de Negócio e Acesso a Dados. Busca-se a decomposição de funcionalidades de forma a permitir aos desenvolvedores concentrarem-se em diferentes partes da aplicação durante a implementação.

**38. (FCC - 2014 – TJ/AP - Analista de Sistemas)** Uma arquitetura muito comum em aplicações web é o Modelo Arquitetural 3 Camadas:

- I. Camada de Persistência.
- II. Camada de Lógica de Negócio.
- III. Camada de Apresentação.

Neste modelo, a correta associação dos componentes com as camadas é:

- a) I-Servidor de Banco de Dados - II-Servidor de Aplicação - III-Máquina Cliente.
- b) I-Servidor Web - II-Servidor Cliente - III-Servidor de Aplicação.
- c) I-Servidor Web - II-Servidor de Banco de Dados - III-Máquina Cliente.
- d) I-Servidor de Banco de Dados - II-Máquina Cliente - III-Servidor de Aplicação.
- e) I-Máquina Cliente - II-Servidor de Banco de Dados - III-Servidor Web.

**39. (FCC - 2012 – TRF/2ª - Analista de Sistemas)** São aspectos que podem caracterizar uma arquitetura cliente-servidor, estabelecida logicamente em 4 camadas:

- I. A camada Cliente contém um navegador de Internet, caracterizado como cliente universal.





II. A camada de Lógica do Negócio se quebra em duas: camada de Aplicação e camada Web, em que o servidor Web representa a camada de Apresentação.

III. Na camada de Lógica do Negócio, o servidor de aplicação passa a utilizar middleware, para suporte a thin clients (PDA, celulares, smart cards, etc) e soluções baseadas em componentes, tais como, J2EE e .Net.

IV. Se, de um lado, a camada de Aplicação estabelece uma interface com a camada de Dados, do outro o faz com a camada Web e com os thin clients da camada Cliente.

Está correto o que consta em:

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

**40. (FCC - 2010 – TRT/SE - Analista de Sistemas)** A arquitetura multicamadas divide-se em três camadas lógicas. São elas:

- a) Apresentação, Negócio e Acesso a Dados.
- b) Apresentação, Natureza e Acesso a Dados.
- c) Apresentação, Negócio e Alteração.
- d) Manipulação, Natureza e Acesso a Dados.
- e) Manipulação, Negócio e Acesso a Dados.

**41. (FCC - 2010 – METRÔ/SP - Analista de Sistemas)** A arquitetura multicamadas divide-se em três camadas lógicas. São elas:

- a) Apresentação, Negócio e Alteração.
- b) Manipulação, Natureza e Acesso a Dados.
- c) Manipulação, Negócio e Acesso a Dados.
- d) Apresentação, Natureza e Acesso a Dados.
- e) Apresentação, Negócio e Acesso a Dados.

**42. (FCC - 2008 – TRF/5ª - Analista de Sistemas)** Via de regra as divisões da arquitetura de software em três camadas orientam para níveis que especificam:

- a) os casos de uso, a estrutura dos dados e os processos de manutenção.
- b) a apresentação, as regras de negócio e os testes.
- c) a apresentação, os processos operacionais e a seqüência de execução.
- d) a apresentação, os componentes virtuais e a seqüência de execução.
- e) a apresentação, as regras de negócio e o armazenamento de dados.



**43. (FCC - 2015 – CNMP - Analista de Sistemas)** Há algumas variantes possíveis de arquitetura a serem utilizadas em um sistema de bancos de dados. Sobre essas variantes, é correto afirmar que:

- a) na arquitetura de 3 camadas, não há uma camada específica para a aplicação.
- b) a camada de apresentação da arquitetura de 2 camadas situa-se, usualmente, no servidor de banco de dados.
- c) na arquitetura de 3 camadas, a camada de servidor de banco de dados é denominada cliente.
- d) a arquitetura de 3 camadas é composta pelas camadas cliente, aplicação e servidor de banco de dados.
- e) na arquitetura de 2 camadas não há necessidade de uso de um sistema gerenciador de bancos de dados.

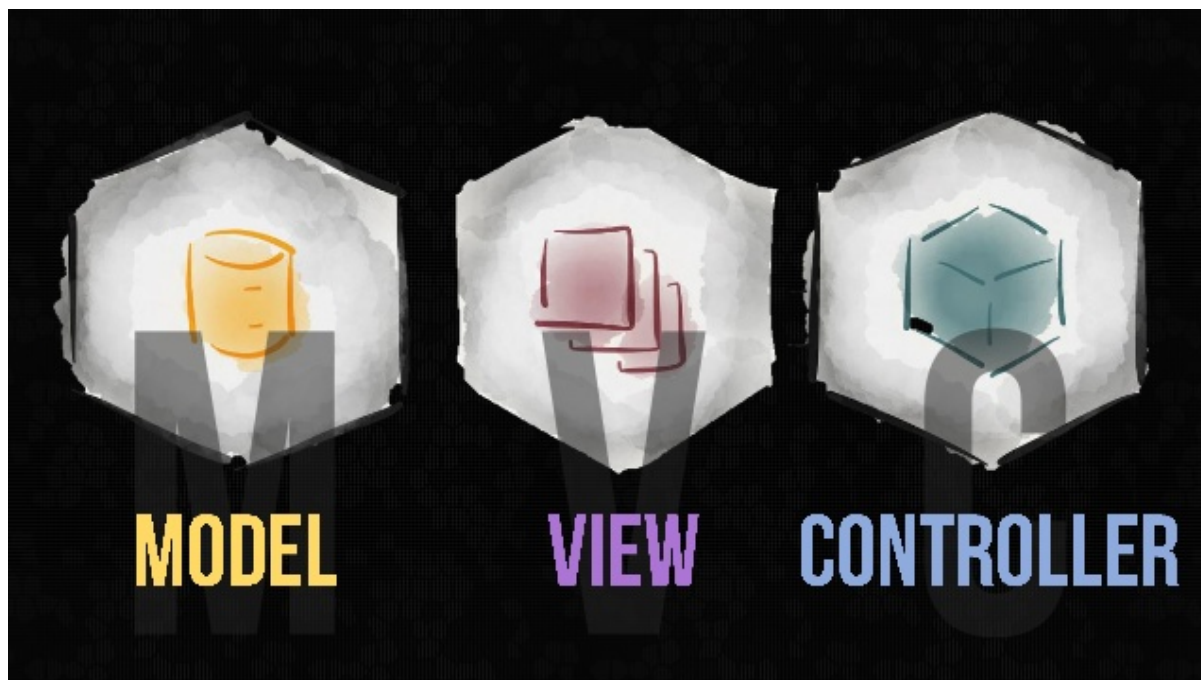


## GABARITO

- |             |             |
|-------------|-------------|
| 1. CORRETO  | 22. ERRADO  |
| 2. ERRADO   | 23. ERRADO  |
| 3. CORRETO  | 24. ERRADO  |
| 4. CORRETO  | 25. CORRETO |
| 5. ERRADO   | 26. CORRETO |
| 6. CORRETO  | 27. CORRETO |
| 7. CORRETO  | 28. CORRETO |
| 8. ERRADO   | 29. CORRETO |
| 9. CORRETO  | 30. CORRETO |
| 10. CORRETO | 31. CORRETO |
| 11. ERRADO  | 32. CORRETO |
| 12. ERRADO  | 33. ERRADO  |
| 13. ERRADO  | 34. ERRADO  |
| 14. CORRETO | 35. CORRETO |
| 15. ERRADO  | 36. LETRA C |
| 16. CORRETO | 37. LETRA C |
| 17. ERRADO  | 38. LETRA A |
| 18. CORRETO | 39. LETRA E |
| 19. CORRETO | 40. LETRA A |
| 20. CORRETO | 41. LETRA E |
| 21. CORRETO | 42. LETRA E |
|             | 43. LETRA D |



## 4 – ARQUITETURA MVC (MODEL VIEW CONTROLLER)



O Model-View-Controller (MVC) é um padrão arquitetural de software para implementar interfaces de usuário. **Ele divide uma aplicação de software em três partes interconectadas, de modo a separar representações internas de informação das formas em que a informação é apresentada para o usuário.** Galera, esse é um assunto que pode ser bastante aprofundado, vou tentar simplificá-lo nessa aula.

**Em uma linguagem bem simples e direta, ele é um padrão arquitetural de software que separa uma aplicação em três camadas.** Você pode entendê-lo como uma forma de organizar o código de uma aplicação de forma que sua manutenção fique mais fácil. Trata-se da separação de responsabilidades, sendo uma maneira de quebrar uma aplicação (ou parte dela) em camadas: Modelo, Visão e Controle.

**O MVC promove a estrita separação de responsabilidade entre componentes de uma interface gráfica** onde temos componentes responsáveis pela manutenção do estado da aplicação, denominado de Modelo, pela exibição de parte deste modelo para o usuário, ao que chamamos de Visão e pela coordenação entre atualizações no modelo e interações com o usuário, feita através do Controlador.

Durante a década de setenta, surgiu a necessidade de criação de uma arquitetura para ser utilizada em projetos de interface visual na linguagem de programação Smalltalk. **A ideia original era organizar o código, separar responsabilidades, aumentar a manutenibilidade, promover um baixo acoplamento e uma alta coesão, fomentar a reusabilidade do código e tornar o sistema escalável.**



Passou um bocadinho de tempo e, com o surgimento da WWW, algumas pessoas pensaram em adaptar esse padrão arquitetural para o mundo web. **Muitos frameworks de aplicação comerciais e não comerciais foram criados tendo como base esse modelo.** Estes frameworks variam em suas interpretações, principalmente no modo que as responsabilidades MVC são divididas entre o cliente e servidor.

### Camada de Modelo:

Essa é a camada responsável pela representação dos dados, provendo meios de acesso (leitura/escrita). **Cara, sempre que você pensar em manipulação de dados, (leitura, escrita ou validação de dados<sup>2</sup>), pense na Camada de Modelo!** Ela gerencia não só os dados, mas também os comportamentos fundamentais da aplicação – representados por regras de negócio (Sim, elas ficam na Camada de Modelo!).

**A Camada de Modelo encapsula as principais funcionalidades e dados do sistema.** Ela notifica suas visões e respectivos controladores quando surge alguma mudança em seu estado, isto é, ela é responsável pela manutenção do estado da aplicação. Estas notificações permitem que as visões produzam saídas atualizadas e que os controladores alterem o conjunto de comandos disponíveis.

### Camada de Controle:

Essa é a camada responsável por receber todas as requisições do usuário. Seus métodos – chamados *actions* – são responsáveis por uma página, controlando qual modelo usar e qual visão será mostrada ao usuário. **Ele é capaz de enviar comandos para o modelo atualizar o seu estado.** Ele também pode enviar comandos para a respectiva visão para alterar a apresentação da visão do modelo.

**A Camada de Controle atende às requisições do usuário e seleciona o modelo e a visão que o usuário usará para interagir com o modelo.** O usuário interage com controladores – por meio de visões – que interpretam eventos e entradas enviadas (*Input*), mapeando ações do usuário em comandos que são enviados para o modelo e/ou para a visão para efetuar as alterações apropriadas (*Output*).

Um controlador define o comportamento da aplicação, interpretando as ações do usuário e mapeando-as em chamadas do modelo. **Em um cliente de aplicações web, essas ações do usuário poderiam ser cliques em botões ou seleções de menus.** As ações realizadas pelo modelo poderiam ser ativar processos de negócio ou alterar o estado do modelo.

---

<sup>2</sup> Galera, a validação ocorre na Camada de Modelo. Pode ocorrer na Camada de Visão? Sim, eu posso utilizar um JavaScript p/ fazer algumas validações de dados, mas isso é inseguro e se trata de uma violação do modelo. Logo, aceitem que validações ocorrem na camada de modelo. Bacana?



Vocês já pensaram no porquê de essa camada ter esse nome? Porque ela controla o fluxo da aplicação, interpretando os dados de entrada e coordenando/orquestrando as manipulações do modelo e as interações com o usuário. Trata-se de uma camada intermediária entre a Visão e o Modelo. **Em geral, há um controlador para cada visão, apesar de poder existir várias controladoras para uma mesma visão.**

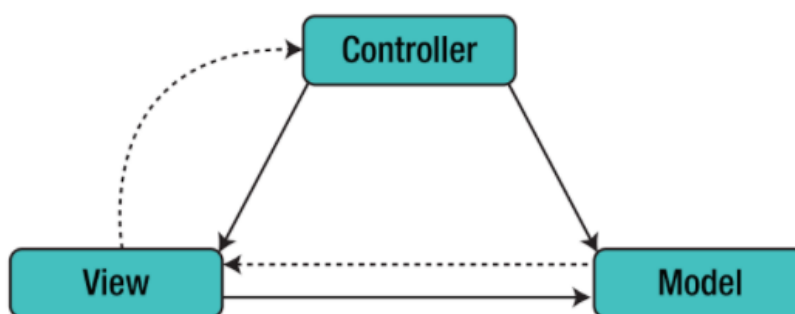
### Camada de Visão:

**Essa é a camada responsável pela interação com o usuário, sendo responsável apenas pela exibição de dados.** Trata-se de uma representação visual do modelo. Ela permite apresentar, de diversas formas diferentes, os dados para o usuário. A visão não sabe nada sobre o que a aplicação está fazendo atualmente, ela recebe instruções do controle, notifica o controle e recebe informações do modelo.

Prosseguindo com nossa linha de pensamento: geralmente, a visão contém formulários, tabelas, menus e botões para entrada e saída de dados. **Além disso, existem diversas visões para cada modelo.** Galera, agora um ponto importante que de vez em quando cai em prova e é importante saber bem para não confundir e acabar errando a questão por bobeira.

À primeira vista, a Arquitetura MVC parece não ter diferença alguma em relação à Arquitetura em Três-Camadas, com o Modelo substituindo a Camada de Dados, a Visão substituindo a Camada de Apresentação e o Controlador substituindo a Camada de Lógica de Negócio. **No entanto, essas duas arquiteturas são diferentes em relação a interação entre suas camadas.**

Na Arquitetura em Três-Camadas, a comunicação entre camadas é rigidamente linear, i.e., a Camada de Apresentação e a Camada de Dados só se conversam bidirecionalmente com a Camada de Lógica, mas nunca entre si. Já no MVC, a comunicação é triangular – **existem diversas implementações diferentes dessa arquitetura, uma comunicação típica é apresentada abaixo:**

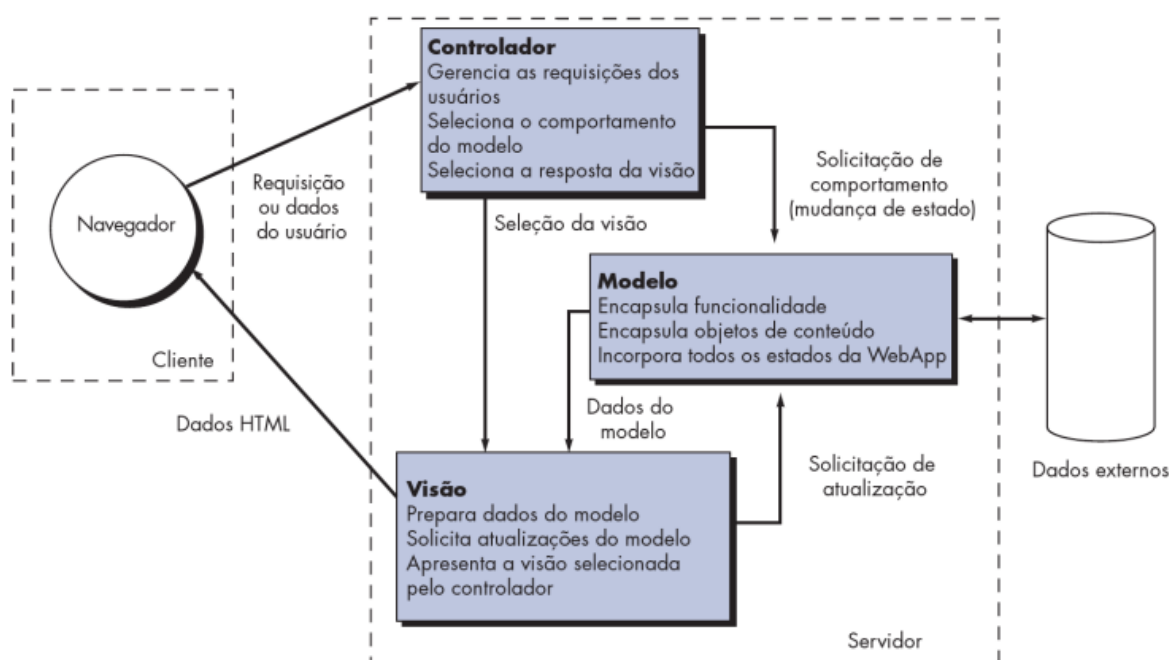


**Nesta figura, a Visão pode tanto gerar eventos a serem tratados pelo Controle quanto obter os dados a serem exibidos diretamente do modelo.** O Controle trata os eventos da Visão, mas também pode manipular diretamente o Modelo. Finalmente, o Modelo pode reagir diretamente tanto à Visão quanto ao Controle, mas também pode gerar eventos a serem tratados pela visão.



Essa última sentença é extremamente polêmica – vocês encontrarão muitos lugares dizendo que não é possível que a visão solicite diretamente o estado do modelo, mas é possível, sim! **Vamos supor que você esteja comprando cursos no site do Estratégia!** Colocou um no carrinho, colocou dois e colocou o terceiro. Você, então, clica no botão de listar os itens que estão no carrinho.

Nesse momento, a visão não precisa necessariamente fazer uma requisição para o controle, para que o controle peça a lista de itens para o modelo. Ora, ela pode pedir diretamente para o modelo! **Fiquem ligados também que existem diversas visões para cada modelo.** Na imagem abaixo, podemos ver as possíveis interações na Arquitetura MVC!



Para quem quiser conhecer melhor, recomendo os seguintes artigos:

- <http://www.cfgigolo.com/2008/01/mvc-model-view-controller-e-os-tres-macacos>
- <https://r.je/views-are-not-templates.html>
- <http://tableless.com.br/mvc-afinal-e-o-que>

**Por fim, não podemos confundir MVC com MVP (Model-View-Presenter).** O MVP é uma evolução do MVC que se comunica bidirecionalmente com as outras camadas, evitando que o Model tenha que se comunicar diretamente com a View sem passar pelo Controller e este último é fundamental para a interação com o usuário. O MVP desacopla as funções e torna a arquitetura ainda mais modular.

A camada Presenter é ciente de tudo o que ocorre nas outras duas camadas e deixa-as cientes do que ela está fazendo; a interação com usuário é feita primariamente pela View, e esta pode delegar ao Presenter determinadas tarefas; há uma relação um-para-um entre estas camadas. **Nesta relação, há uma referência do View para o Presenter mas não o oposto.**





Não devemos confundir também com o MVVM (Model-View-ViewModel). O MVVM é uma pequena evolução do MVP em um lado e um retrocesso em outro. Nele o ViewModel não está ciente do que ocorre no View, **mas este está ciente do que ocorre no ViewModel** (como no Padrão de Projeto Observer). No caso do Model, ambos estão cientes do que ocorre em cada um.

O nome se dá porque ele adiciona propriedades e operações ao Model para atender as necessidades do View, **portanto ele cria um novo modelo para a visualização**. É possível associar várias Views para um ModelView. É comum que as Views sejam definidas de forma declarativa (HTML/CSS, XAML, etc.). O Data Binding é feito entre a View e o ViewModel.

**Com esse padrão é possível reduzir a quantidade de código para manter.** Algumas automações são possíveis por ter todas as informações necessárias no ViewModel. ViewModel – é só um modelo mais adequado para uma visão específica (ou mais que uma). Pessoal, acredito que isso basta em relação a esses assuntos que só recentemente começaram a ser cobrados.





## EXERCÍCIOS COMENTADOS – CESPE

1. (CESPE – 2006 – CENSIPAM - Analista de Sistemas) O padrão MVC organiza um software em modelo, visão e controle. O modelo encapsula as principais funcionalidades e dados. As visões apresentam os dados aos usuários. Uma visão obtém os dados do modelo via funções disponibilizadas pelo modelo; só há uma visão para um modelo. Usuários interagem via controladoras que traduzem os eventos em solicitações ao modelo ou à visão; podem existir várias controladoras associadas a uma mesma visão.

### Comentários:

**A Camada de Controle atende às requisições do usuário e seleciona o modelo e a visão que o usuário usará para interagir com o modelo.** O usuário interage com controladores – por meio de visões – que interpretam eventos e entradas enviadas (Input), mapeando ações do usuário em comandos que são enviados para o modelo e/ou para a visão para efetuar as alterações apropriadas (Output).

Vocês já pensaram no porquê de essa camada ter esse nome? Porque ela controla o fluxo da aplicação, interpretando os dados de entrada e coordenando/orquestrando as manipulações do modelo e as interações com o usuário. Trata-se de uma camada intermediária entre a Visão e o Modelo. **Em geral, há um controlador para cada visão, apesar de poder existir várias controladoras para uma mesma visão.**

Prosseguindo com nossa linha de pensamento: geralmente, a visão contém formulários, tabelas, menus e botões para entrada e saída de dados. **Além disso, existem diversas visões para cada modelo.** Galera, agora um ponto importante que de vez em quando cai em prova e é importante saber bem para não confundir e acabar errando a questão por bobeira.

Conforme vimos em aula, pode haver várias visões para um modelo.

**Gabarito:** Errado

2. (CESPE – 2006 – CENSIPAM - Analista de Sistemas) No padrão MVC, se um usuário modifica o modelo, as visões que dependem desse modelo refletem essas modificações, pois o modelo notifica as visões quando ocorre uma modificação nos seus dados. Portanto, é usado um mecanismo para propagação de modificações que mantém um registro dos componentes que dependem do modelo.

### Comentários:

**A Camada de Controle atende às requisições do usuário e seleciona o modelo e a visão que o usuário usará para interagir com o modelo.** O usuário interage com controladores – por meio de visões – que interpretam eventos e entradas enviadas (Input), mapeando ações do usuário em



comandos que são enviados para o modelo e/ou para a visão para efetuar as alterações apropriadas (Output).

Conforme vimos em aula, mudanças em um modelo podem modificar as visões que dependem desse modelo, gerando uma rastreabilidade.

**Gabarito:** Certo

3. (CESPE – 2013 – BACEN - Analista de Sistemas) MVC (Model-View-Controller) é um modelo de arquitetura de software que separa, de um lado, a representação da informação e, de outro, a interação do usuário com a informação.

#### Comentários:

*O Model-View-Controller (MVC) é um padrão arquitetural de software para implementar interfaces de usuário. Ele divide uma aplicação de software em três partes interconectadas, de modo a separar representações internas de informação das formas em que a informação é apresentada para o usuário. Galera, esse é um assunto que pode ser bastante aprofundado, vou tentar simplificá-lo nessa aula.*

Conforme vimos em aula, a camada de visão trata da representação da informação e a camada de controle trata da interação do usuário com a informação (modelo).

**Gabarito:** Certo

4. (CESPE – 2008 – TJ/CE - Analista de Sistemas) A arquitetura MVC fornece uma maneira de dividir a funcionalidade envolvida na manutenção e apresentação dos dados de uma aplicação web.

#### Comentários:

*O MVC promove a estrita separação de responsabilidade entre componentes de uma interface gráfica onde temos componentes responsáveis pela manutenção do estado da aplicação, denominado de Modelo, pela exibição de parte deste modelo para o usuário, ao que chamamos de Visão e pela coordenação entre atualizações no modelo e interações com o usuário, feita através do Controlador.*

Conforme vimos em aula, a questão está perfeita!

**Gabarito:** Certo

5. (CESPE – 2008 – TJ/CE - Analista de Sistemas) A arquitetura MVC foi desenvolvida recentemente para mapear as tarefas complexas de saída do sistema do usuário.



### Comentários:

*Durante a década de setenta, surgiu a necessidade de criação de uma arquitetura para ser utilizada em projetos de interface visual na linguagem de programação Smalltalk. **A ideia original era organizar o código, separar responsabilidades, aumentar a manutenibilidade, promover um baixo acoplamento e uma alta coesão, fomentar a reusabilidade do código e tornar o sistema escalável.***

Conforme vimos em aula, não tem nada de recente!

**Gabarito:** Errado

6. (CESPE – 2008 – TJ/CE - Analista de Sistemas) Na arquitetura MVC, um controlador define o comportamento da aplicação, já que este é o responsável por interpretar as ações do usuário e as relaciona com as chamadas do modelo.

### Comentários:

*Um controlador define o comportamento da aplicação, interpretando as ações do usuário e mapeando-as em chamadas do modelo. **Em um cliente de aplicações web, essas ações do usuário poderiam ser cliques em botões ou seleções de menus.** As ações realizadas pelo modelo poderiam ser ativar processos de negócio ou alterar o estado do modelo.*

Conforme vimos em aula, ele realmente interpreta ações do usuário e direciona para as chamadas do modelo.

**Gabarito:** Certo

7. (CESPE – 2008 – TJ/CE - Analista de Sistemas) A arquitetura MVC não separa a informação de sua apresentação, porque, em sistemas web, informação e apresentação estão na mesma camada.

### Comentários:

***O MVC promove a estrita separação de responsabilidade entre componentes de uma interface gráfica** onde temos componentes responsáveis pela manutenção do estado da aplicação, denominado de Modelo, pela exibição de parte deste modelo para o usuário, ao que chamamos de Visão e pela coordenação entre atualizações no modelo e interações com o usuário, feita através do Controlador.*

Conforme vimos em aula, ela separa a informação (Modelo) da apresentação (Visão).

**Gabarito:** Errado



8. (CESPE – 2008 – TJ/CE - Analista de Sistemas) O desenvolvimento de sistemas web ocorre tipicamente em três camadas. A arquitetura MVC aumenta o escopo do desenvolvimento para, no máximo, quatro camadas, sendo a quarta camada o processamento dos dados do usuário.

#### Comentários:

Não, tipicamente temos três camadas. No entanto, realmente não há restrição de escopo para quatro camadas. Pode-se acrescentar outras camadas principalmente nas camadas de controle e visão.

**Gabarito:** Errado

9. (CESPE – 2009 – ANAC - Analista de Sistemas) O framework modelo visão controlador (MVC – model view controller) é muito utilizado para projeto da GUI (graphical user interface) de programas orientados a objetos.

#### Comentários:

*Durante a década de setenta, surgiu a necessidade de criação de uma arquitetura para ser utilizada em projetos de interface visual na linguagem de programação Smalltalk. **A ideia original era organizar o código, separar responsabilidades, aumentar a manutenibilidade, promover um baixo acoplamento e uma alta coesão, fomentar a reusabilidade do código e tornar o sistema escalável.***

Conforme vimos em aula, foi nesse contexto que ele foi criado e com esse objetivo.

**Gabarito:** Certo

10. (CESPE – 2009 – TCU - Analista de Sistemas) No MVC (model-view-controller), um padrão recomendado para aplicações interativas, uma aplicação é organizada em três módulos separados. Um para o modelo de aplicação com a representação de dados e lógica do negócio, o segundo com visões que fornecem apresentação dos dados e input do usuário e o terceiro para um controlador que despacha pedidos e controle de fluxo.

#### Comentários:

***O MVC promove a estrita separação de responsabilidade entre componentes de uma interface gráfica** onde temos componentes responsáveis pela manutenção do estado da aplicação, denominado de Modelo, pela exibição de parte deste modelo para o usuário, ao que chamamos de Visão e pela coordenação entre atualizações no modelo e interações com o usuário, feita através do Controlador.*

Conforme vimos em aula, nada a acrescentar!



**Gabarito:** Certo

11. (CESPE – 2010 – EMBASA - Analista de Sistemas) O MVC promove a estrita separação de responsabilidades entre os componentes de uma interface.

**Comentários:**

Essa questão foi anulada, porque ela não especificou qual tipo de interface. Considerando tratar-se de interface gráfica, a questão seria correta.

**Gabarito:** X

12. (CESPE – 2010 – EMBASA - Analista de Sistemas) No MVC, a visão é responsável pela manutenção do estado da aplicação.

**Comentários:**

*A Camada de Modelo encapsula as principais funcionalidades e dados do sistema. Ela notifica suas visões e respectivos controladores quando surge alguma mudança em seu estado, i.e., ela é responsável pela manutenção do estado da aplicação. Estas notificações permitem que as visões produzam saídas atualizadas e que os controladores alterem o conjunto de comandos disponíveis.*

Conforme vimos em aula, a questão trata da Camada de Modelo.

**Gabarito:** Errado

13. (CESPE – 2010 – EMBASA - Analista de Sistemas) O modelo no MVC tem como atribuição exibir a parte que é responsável pela manutenção da aplicação para o usuário.

**Comentários:**

*Essa é a camada responsável pela interação com o usuário, sendo responsável apenas pela exibição de dados. Trata-se de uma representação visual do modelo. Ela permite apresentar, de diversas formas diferentes, os dados para o usuário. A visão não sabe nada sobre o que a aplicação está fazendo atualmente, ela recebe instruções do controle, notifica o controle e recebe informações do modelo.*

Conforme vimos em aula, essa atribuição é da camada de visão.

**Gabarito:** Errado

14. (CESPE – 2010 – EMBASA - Analista de Sistemas) O controlador é responsável pela coordenação entre atualizações no modelo e interações com o usuário.



### Comentários:

*Vocês já pensaram no porquê de essa camada ter esse nome? Porque ela controla o fluxo da aplicação, interpretando os dados de entrada e coordenando/orquestrando as manipulações do modelo e as interações com o usuário. Trata-se de uma camada intermediária entre a Visão e o Modelo. **Em geral, há um controlador para cada visão, apesar de poder existir várias controladoras para uma mesma visão.***

Conforme vimos em aula, essa é realmente uma de suas responsabilidades.

**Gabarito:** Certo

**15.(CESPE – 2010 – EMBASA - Analista de Sistemas)** Por meio do MVC, é possível o desenvolvimento de aplicações em 3 camadas para a Web.

### Comentários:

*Passou um bocadinho de tempo e, com o surgimento da WWW, algumas pessoas pensaram em adaptar esse padrão arquitetural para o mundo web. **Muitos frameworks de aplicação comerciais e não comerciais foram criados tendo como base esse modelo.** Estes frameworks variam em suas interpretações, principalmente no modo que as responsabilidades MVC são divididas entre o cliente e servidor.*

Conforme vimos em aula, está perfeito!

**Gabarito:** Certo

**16.(CESPE – 2011 – CET - Analista de Sistemas)** No padrão de desenvolvimento modelo-visualização-controlador (MVC), o controlador é o elemento responsável pela interpretação dos dados de entrada e pela manipulação do modelo, de acordo com esses dados.

### Comentários:

*Vocês já pensaram no porquê de essa camada ter esse nome? Porque ela controla o fluxo da aplicação, interpretando os dados de entrada e coordenando/orquestrando as manipulações do modelo e as interações com o usuário. Trata-se de uma camada intermediária entre a Visão e o Modelo. **Em geral, há um controlador para cada visão, apesar de poder existir várias controladoras para uma mesma visão.***

Conforme vimos em aula, ele é responsável pelo fluxo da aplicação e, por isso, é o responsável pela interpretação dos dados de entrada e pela manipulação do modelo.

**Gabarito:** Certo



**17. (CESPE – 2011 – MEC - Analista de Sistemas)** O modelo MVC pode ser usado para construir a arquitetura do software a partir de três elementos: modelo, visão e controle, sendo definidas no controle as regras de negócio que controlam o comportamento do software a partir de restrições do mundo real.

#### Comentários:

*Essa é a camada responsável pela representação dos dados, provendo meios de acesso (leitura/escrita). **Cara, sempre que você pensar em manipulação de dados, (leitura, escrita ou validação de dados), pense na Camada de Modelo!** Ela gerencia não só os dados, mas também os comportamentos fundamentais da aplicação – representados por regras de negócio (Sim, elas ficam na Camada de Modelo!).*

Conforme vimos em aula, regras de negócio ficam no modelo; o controlador só orchestra as solicitações vindas da visão para o modelo.

**Gabarito:** Errado

**18. (CESPE – 2011 – MEC - Analista de Sistemas)** O controlador, no modelo MVC, realiza a comunicação entre as camadas de visão e modelo.

#### Comentários:

*Vocês já pensaram no porquê de essa camada ter esse nome? Porque ela controla o fluxo da aplicação, interpretando os dados de entrada e coordenando/orquestrando as manipulações do modelo e as interações com o usuário. Trata-se de uma camada intermediária entre a Visão e o Modelo. **Em geral, há um controlador para cada visão, apesar de poder existir várias controladoras para uma mesma visão.***

Conforme vimos em aula, a questão está perfeita!

**Gabarito:** Certo

**19. (CESPE – 2011 – MEC - Analista de Sistemas)** No MVC, o modelo representa o estado geral do sistema.

#### Comentários:

***A Camada de Modelo encapsula as principais funcionalidades e dados do sistema.** Ela notifica suas visões e respectivos controladores quando surge alguma mudança em seu estado, isto é, ela é responsável pela manutenção do estado da aplicação. Estas notificações permitem que as visões produzam saídas atualizadas e que os controladores alterem o conjunto de comandos disponíveis.*





Conforme vimos em aula, é lá que se encontram os dados (estado) do sistema.

**Gabarito:** Certo

**20. (CESPE – 2011 – MEC - Analista de Sistemas)** Apesar do seu amplo emprego em aplicações web, o MVC deve ser utilizado apenas em interfaces gráficas em função de sua arquitetura de componentes.

**Comentários:**

*Durante a década de setenta, surgiu a necessidade de criação de uma arquitetura para ser utilizada em projetos de interface visual na linguagem de programação Smalltalk. **A ideia original era organizar o código, separar responsabilidades, aumentar a manutenibilidade, promover um baixo acoplamento e uma alta coesão, fomentar a reusabilidade do código e tornar o sistema escalável.***

Conforme vimos em aula, quando ela começou nem existiam aplicações web. Ele é usado primariamente em interfaces gráficas, mas não se limita a isso!

**Gabarito:** Errado

**21. (CESPE – 2011 – MEC - Analista de Sistemas)** No MVC, é o modelo que permite apresentar, de diversas formas diferentes, os dados para o usuário.

**Comentários:**

***Essa é a camada responsável pela interação com o usuário, sendo responsável apenas pela exibição de dados.** Trata-se de uma representação visual do modelo. Ela permite apresentar, de diversas formas diferentes, os dados para o usuário. A visão não sabe nada sobre o que a aplicação está fazendo atualmente, tudo que ela faz é receber instruções do controle e informações do modelo e, então, exibi-las.*

Conforme vimos em aula, essa é uma função da visão!

**Gabarito:** Errado

**22. (CESPE – 2011 – MEC - Analista de Sistemas)** O controlador é o responsável pelas regras de negócio e pelos dados de uma aplicação no MVC.

**Comentários:**

*Essa é a camada responsável pela representação dos dados, provendo meios de acesso (leitura/escrita). **Cara, sempre que você pensar em manipulação de dados, (leitura, escrita ou validação de dados), pense na Camada de Modelo!** Ela gerencia não só os dados, mas também os*





*comportamentos fundamentais da aplicação – representados por regras de negócio (Sim, elas ficam na Camada de Modelo!).*

Conforme vimos em aula, essa é uma função do Modelo.

**Gabarito:** Errado

**23. (CESPE – 2013 – TCE/ES - Analista de Sistemas – Letra E)** No Padrão MVC, as regras do negócio que definem a forma de acesso e modificação dos dados são geridas pelo controlador.

#### Comentários:

*Essa é a camada responsável pela representação dos dados, provendo meios de acesso (leitura/escrita). **Cara, sempre que você pensar em manipulação de dados, (leitura, escrita ou validação de dados), pense na Camada de Modelo!** Ela gerencia não só os dados, mas também os comportamentos fundamentais da aplicação – representados por regras de negócio (Sim, elas ficam na Camada de Modelo!).*

Conforme vimos em aula, essa é uma função do Modelo.

**Gabarito:** Errado

**24. (CESPE – 2015 – STJ - Analista de Sistemas)** Na arquitetura em camadas MVC (modelo-visão-controlador), o modelo encapsula o estado de aplicação, a visão solicita atualização do modelo e o controlador gerencia a lógica de negócios.

#### Comentários:

***A Camada de Modelo encapsula as principais funcionalidades e dados do sistema.** Ela notifica suas visões e respectivos controladores quando surge alguma mudança em seu estado, isto é, ela é responsável pela manutenção do estado da aplicação. Estas notificações permitem que as visões produzam saídas atualizadas e que os controladores alterem o conjunto de comandos disponíveis.*

*Nesse momento, a visão não precisa necessariamente fazer uma requisição para o controle, para que o controle peça a lista de itens para o modelo. Ora, ela pode pedir diretamente para o modelo! **Fiquem ligados também que existem diversas visões para cada modelo.** Na imagem abaixo, podemos ver uma possível interação na Arquitetura MVC!*

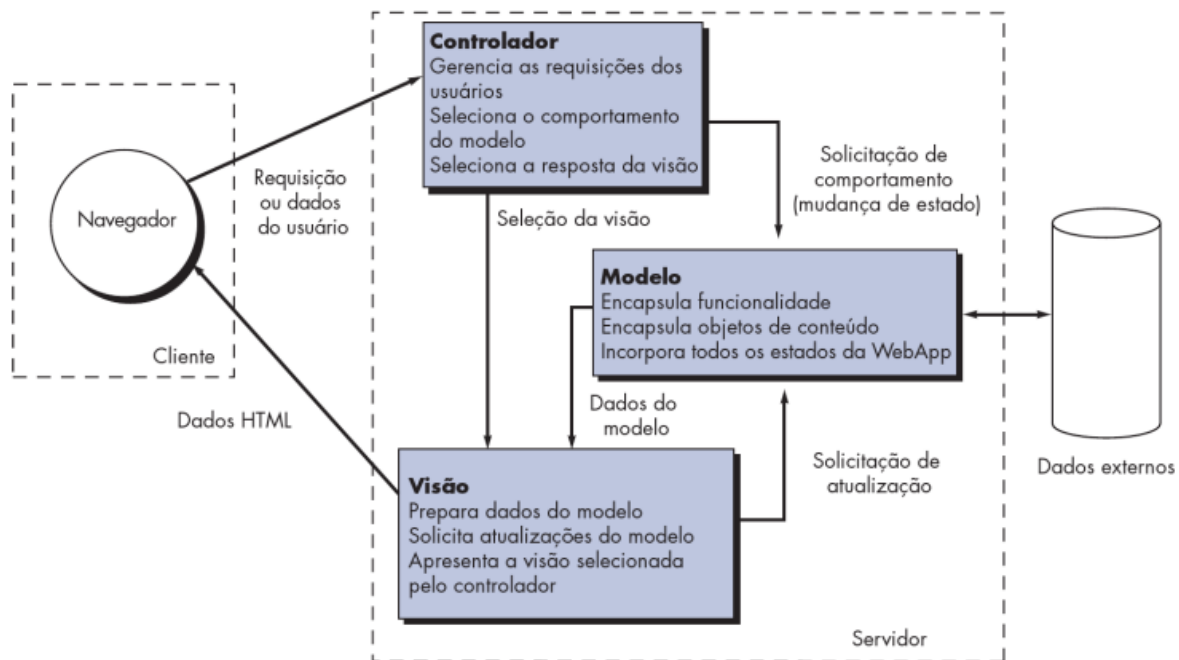
Conforme vimos em aula, a questão está errada! O modelo encapsula o estado da aplicação? Sim, isso é verdade. A visão solicita atualização do modelo? Sim, vimos que ela faz isso por meio de eventos que notificam o controlador. O Controlador gerencia a lógica de negócios? Não, quem gerencia a lógica de negócios é o modelo.



**Gabarito:** Errado

**25. (CESPE – 2015 – MEC - Analista de Sistemas)** O controlador gerencia as requisições dos usuários encapsulando as funcionalidades e prepara dados do modelo.

**Comentários:**



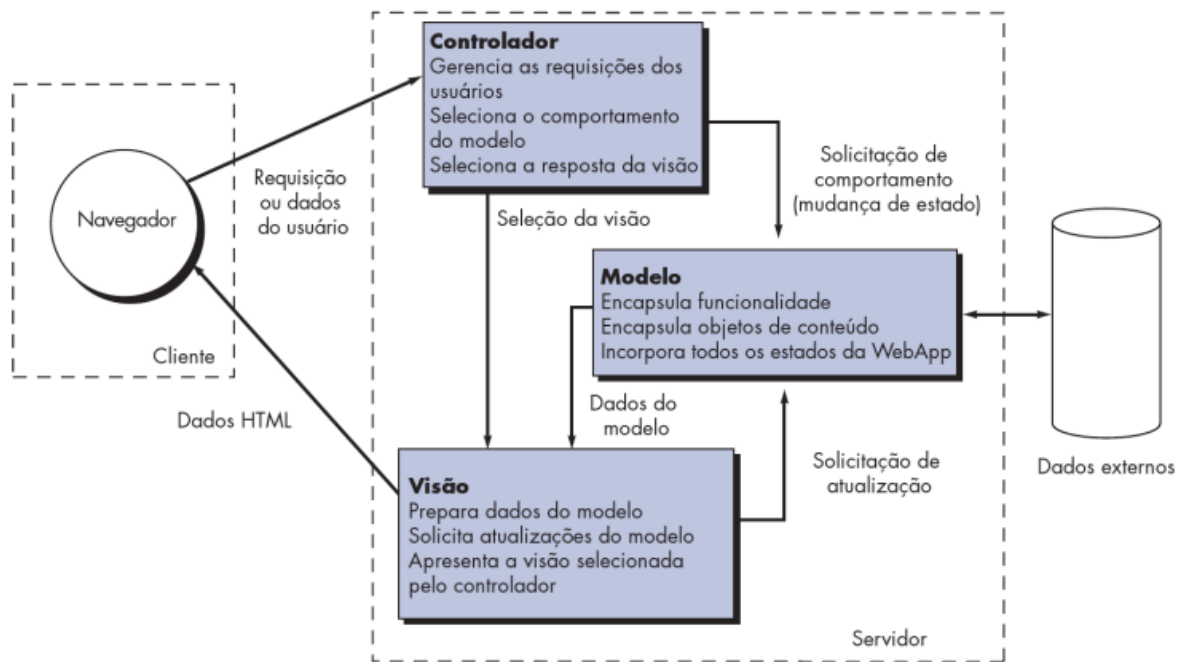
Conforme vimos em aula, o controlador gerencia as requisições dos usuários, o modelo encapsula funcionalidades e a visão prepara dados do modelo.

**Gabarito:** Errado

**26. (CESPE – 2015 – MEC - Analista de Sistemas)** A visão encapsula objetos de conteúdo, solicita atualizações do modelo e seleciona o comportamento do modelo.

**Comentários:**



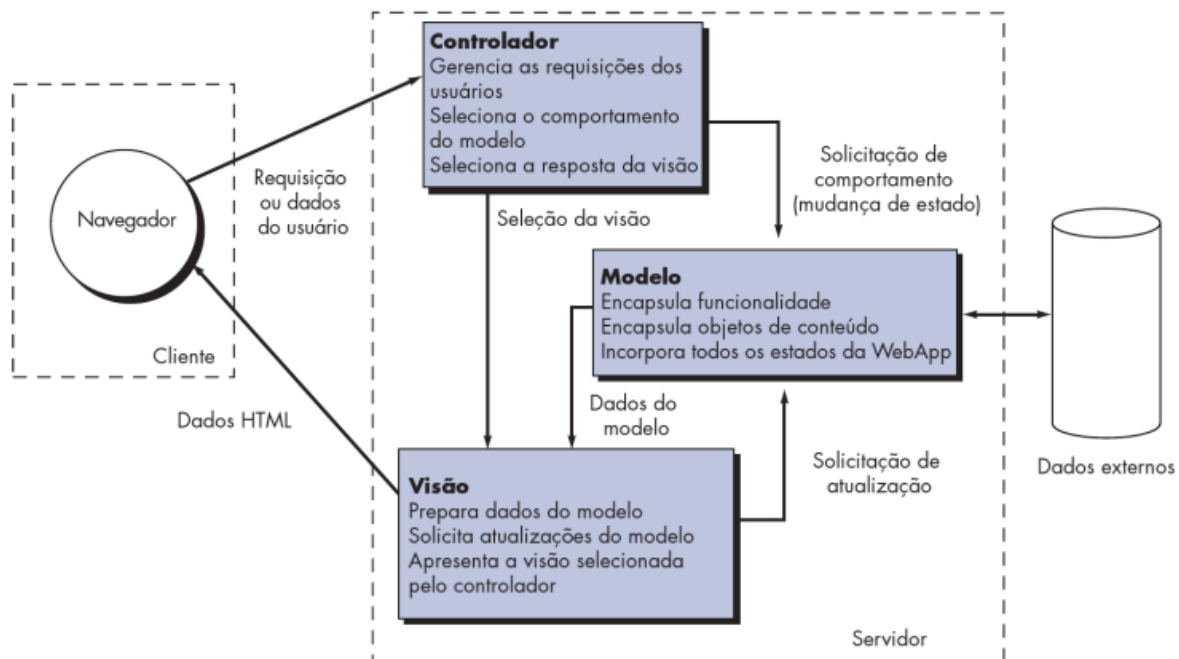


Conforme vimos em aula, o modelo encapsula objetos de conteúdo, a visão solicita atualizações do modelo e o controle seleciona o comportamento do modelo.

**Gabarito:** Errado

**27. (CESPE – 2015 – STJ - Analista de Sistemas)** No padrão em camadas modelo-visão-controlador (MVC), o controle é responsável por mudanças de estado da visão.

**Comentários:**



Conforme vimos em aula, o controle é responsável por solicitar mudanças de estado, mas quem realiza a mudança é o modelo.

**Gabarito:** Errado

**28.(CESPE – 2014 – ANTAQ - Analista de Sistemas)** O modelo MVC é um padrão de arquitetura que consiste na definição de camadas para a construção de softwares.

**Comentários:**

*O Model-View-Controller (MVC) é um padrão arquitetural de software para implementar interfaces de usuário. **Ele divide uma aplicação de software em três partes interconectadas, de modo a separar representações internas de informação das formas em que a informação é apresentada para o usuário.** Galera, esse é um assunto que pode ser bastante aprofundado, vou tentar simplificá-lo nessa aula.*

Conforme vimos em aula, está perfeito!

**Gabarito:** Certo

**29.(CESPE – 2014 – ANTAQ - Analista de Sistemas)** O controller tem a responsabilidade de armazenar e buscar os dados que deverão ser exibidos pelo view.

**Comentários:**

*Essa é a camada responsável pela representação dos dados, provendo meios de acesso (leitura/escrita). **Cara, sempre que você pensar em manipulação de dados, (leitura, escrita ou validação de dados), pense na Camada de Modelo!** Ela gerencia não só os dados, mas também os comportamentos fundamentais da aplicação – representados por regras de negócio (Sim, elas ficam na Camada de Modelo!).*

Conforme vimos em aula, trata-se do Model!

**Gabarito:** Errado

**30.(CESGRANRIO – 2014 – CEFET/RJ - Analista de Sistemas)** No contexto da Arquitetura de Sistemas, o MVC (model – view – controller) é um estilo arquitetural:

- a) interativo
- b) estrutural
- c) distribuído
- d) adaptável
- e) monolítico



### Comentários:

O Model-View-Controller (MVC) é um padrão arquitetural de software para implementar interfaces de usuário. **Ele divide uma aplicação de software em três partes interconectadas, de modo a separar representações internas de informação das formas em que a informação é apresentada para o usuário.** Galera, esse é um assunto que pode ser bastante aprofundado, vou tentar simplificá-lo nessa aula.

**Em uma linguagem bem simples e direta, ele é um padrão arquitetural de software que separa uma aplicação em três camadas.** Você pode entendê-lo como uma forma de organizar o código de uma aplicação de forma que sua manutenção fique mais fácil. Trata-se da separação de responsabilidades, sendo uma maneira de quebrar uma aplicação (ou parte dela) em camadas: Modelo, Visão e Controle.

Conforme vimos em aula, é um estilo arquitetural interativo, no sentido de que é um estilo para interface de usuário, fornecendo diversas visões diferentes para um mesmo modelo de dados.

**Gabarito:** A

**31. (ESAF – 2012 – CGU - Analista de Sistemas)** A definição de que um sistema deve ser desenvolvido em três níveis é feita pelo padrão de projeto:

- a) MVC (Model View Controller).
- b) MVC-Dev (Model Value Constructive Development).
- c) TMS (Time Milestones Setting).
- d) PMC (Project Main Controller).
- e) MCA (Model Classes Assignment).

### Comentários:

O Model-View-Controller (MVC) é um padrão arquitetural de software para implementar interfaces de usuário. **Ele divide uma aplicação de software em três partes interconectadas, de modo a separar representações internas de informação das formas em que a informação é apresentada para o usuário.** Galera, esse é um assunto que pode ser bastante aprofundado, vou tentar simplificá-lo nessa aula.

Conforme vimos em aula, trata-se do MVC (Model View Controller).

**Gabarito:** A

**32. (ESAF – 2010 – CVM - Analista de Sistemas)** Modelo MVC significa:

- a) Modo-View-Construtor.
- b) Modelo-View-Controlador.



- c) Modelo-Versão-Case.
- d) Módulo-Verificador-Controlador.
- e) Medida-Virtual-Concepção.

### Comentários:

*O Model-View-Controller (MVC) é um padrão arquitetural de software para implementar interfaces de usuário. **Ele divide uma aplicação de software em três partes interconectadas, de modo a separar representações internas de informação das formas em que a informação é apresentada para o usuário.** Galera, esse é um assunto que pode ser bastante aprofundado, vou tentar simplificá-lo nessa aula.*

Conforme vimos em aula, trata-se do Modelo-View-Controlador.

**Gabarito:** B

**33. (FCC – 2012 – TST - Analista de Sistemas)** No padrão MVC é possível definir grupos de componentes principais: o Model (Modelo), o View (Apresentação) e o Controller (Controle). Deve fazer parte do componente:

- a) Controller, uma classe que contém um método com a finalidade de calcular o reajuste de salário dos funcionários.
- b) View, uma classe que contém um método para persistir o salário reajustado de um funcionário.
- c) Controller, as animações desenvolvidas em Flash.
- d) View, as validações necessárias ao sistema, geralmente definidas através de um conjunto de comparações.
- e) Model, as classes com métodos conhecidos como setters e getters e que representam tabelas do banco de dados.

### Comentários:

(a) Errado, isso deve fazer parte do Model; (b) Errado, isso deve fazer parte do Model; (c) Errado, isso deve fazer parte da View; (d) Errado, isso deve fazer parte da Model; (e) Correto, esses métodos de fato devem estar na Model.

**Gabarito:** A

**34. (FCC – 2012 – MPE/AP - Analista de Sistemas)** Em uma Aplicação Web desenvolvida utilizando o design pattern MVC, as páginas HTML e as classes com métodos que acessam o



banco de dados e executam instruções SQL são representadas, respectivamente, nos componentes:

- a) Presentation e Business.
- b) View e Model.
- c) Controller e Model.
- d) Model e Business.
- e) View e Business.

#### Comentários:

Páginas HTML são representadas na View; classes com métodos que acessam o banco de dados e executam instruções SQL são representadas na Model.

**Gabarito:** B

**35. (FCC – 2012 – MPE/PE - Analista de Sistemas)** O padrão de projeto utilizado em aplicações WEB que permite separar as páginas e classes da aplicação em três grupos (muitas vezes chamados de camadas) conhecidos como Apresentação, Controle e Modelo é denominado de:

- a) 3-tier.
- b) DAO.
- c) MVC.
- d) DTO.
- e) DBO.

#### Comentários:

Só pode ser... MVC!

**Gabarito:** C

**36. (FCC – 2012 – TRT/PE - Analista de Sistemas)** O padrão de arquitetura MVC é um modelo de camadas que divide a aplicação em três componentes: Model (modelo), View (visualizador) e Controller (controlador). As funções de cada um destes três componentes são apresentadas abaixo:

- I. interpreta eventos de entrada e envia requisições para o modelo de dados; em seguida, processa os dados carregados a partir do modelo e envia para o visualizador.
- II. encapsula o acesso aos dados e funções básicas da aplicação, fornecendo ao usuário procedimentos que executam tarefas específicas.





III. exibe para o usuário os dados fornecidos pelo controle e estabelece uma interface para interação entre o usuário e a aplicação.

A associação correta do componente do padrão MVC com sua função está expressa, respectivamente, em

- a) (I) Controller; (II) Model; (III) View;
- b) (I) Model; (II) Controller; (III) View;
- c) (I) View; (II) Model; (III) Controller;
- d) (I) Controller; (II) View; (III) Model;
- e) (I) Model; (II) View; (III) Controller;

### Comentários:

(I) Quem faz essa orquestração de requisições é a Camada de Controle; (II) Acesso aos dados e funções básicas da aplicação são de responsabilidade da Camada de Modelo; (III) Exibição é sempre Camada de Visão!

**Gabarito: A**

**37. (FCC – 2012 – TJ/PE - Analista de Sistemas)** Com relação à arquitetura MVC, considere:

I. O MODEL representa os dados da empresa e as regras de negócio que governam o acesso e atualização destes dados.

II. O VIEW acessa os dados da empresa através do MODEL e especifica como esses dados devem ser apresentados. É de responsabilidade do VIEW manter a consistência em sua apresentação, quando o MODEL é alterado.

III. O CONTROLLER traduz as interações do VIEW em ações a serem executadas pelo MODEL. Com base na interação do usuário e no resultado das ações do MODEL, o CONTROLLER responde selecionando uma VIEW adequada.

IV. Permite uma única VIEW para compartilhar o mesmo modelo de dados corporativos em um fluxo de comunicação sequencial.

Está correto o que se afirma em:

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





### Comentários:

(I) Correto, ele representa os dados e as regras de negócio; (II) Correto, a View acessa os dados por meio do Model. Essa parte gera dúvida, mas é responsabilidade da View – após ser notificada – atualizar a apresentação quando há modificação na Model; (III) Correto, essa é a função do Controller, ele orquestra as requisições entre Model e View; (IV) Errado, na verdade permite quantas views forem necessárias para um mesmo Model;

**Gabarito: B**

### 38.(FCC – 2012 – MPE/PE - Analista de Sistemas) O componente Controller do MVC:

- a) Define o comportamento da aplicação, as ações do usuário para atualizar os componentes de dados e seleciona os componentes para exibir respostas de requisições.
- b) Envia requisições do usuário para o controlador e recebe dados atualizados dos componentes de acesso a dados.
- c) Responde às solicitações de queries e encapsula o estado da aplicação.
- d) Notifica os componentes de apresentação das mudanças efetuadas nos dados e expõe a funcionalidade da aplicação.
- e) É onde são concentradas todas as regras de negócio da aplicação e o acesso aos dados.

### Comentários:

(a) Correto, ele define o comportamento da aplicação, as ações do usuário para atualizar os componentes de dados e seleciona os componentes para exibir respostas de requisições; (b) Errado, essa responsabilidade é da View; (c) Errado, essa responsabilidade é do Model; (d) Errado, essa responsabilidade é do Model; (e) Errado, essa responsabilidade é do Model;

**Gabarito: A**

### 39.(FCC – 2011 – TRT/MT - Analista de Sistemas) No projeto de arquitetura modelo-visão-controle (MVC), o controlador:

- a) renderiza a interface de usuário a partir da visão, o modelo encapsula funcionalidades e objetos de conteúdo e a visão processa e responde a eventos e invoca alterações ao controlador.
- b) encapsula funcionalidades e objetos de conteúdo, o modelo processa e responde a eventos e invoca alterações ao controlador e a visão renderiza a interface de usuário a partir do modelo.



- c) encapsula funcionalidades e objetos de conteúdo, o modelo renderiza a interface de usuário a partir da visão e a visão processa e responde a eventos e invoca alterações ao controlador.
- d) processa e responde a eventos e invoca alterações ao modelo, o modelo encapsula funcionalidades e objetos de conteúdo e a visão renderiza a interface de usuário a partir do modelo.
- e) processa e responde a eventos e invoca alterações ao modelo, o modelo renderiza a interface de usuário a partir da visão e a visão encapsula funcionalidades e objetos de conteúdo.

### Comentários:

O Controlador processa e responde a eventos e invoca alterações ao modelo, o modelo encapsula funcionalidades e objetos de conteúdo e a visão renderiza a interface de usuário a partir do modelo.

**Gabarito:** D

**40.(FCC – 2010 – AL/SP - Analista de Sistemas)** Sobre as camadas do modelo de arquitetura MVC (Model- View-Controller) usado no desenvolvimento web é correto afirmar:

- a) Todos os dados e a lógica do negócio para processá-los devem ser representados na camada Controller.
- b) A camada Model pode interagir com a camada View para converter as ações do cliente em ações que são compreendidas e executadas na camada Controller.
- c) A camada View é a camada responsável por exibir os dados ao usuário. Em todos os casos essa camada somente pode acessar a camada Model por meio da camada Controller.
- d) A camada Controller geralmente possui um componente controlador padrão criado para atender a todas as requisições do cliente.
- e) Em aplicações web desenvolvidas com Java as servlets são representadas na camada Model.

### Comentários:

**Podemos afirmar, portanto, que a primeira é uma arquitetura linear, enquanto a segunda é uma arquitetura triangular.** A visão interage com o controle por meio de eventos; o controle notifica o modelo sobre uma mudança; o modelo modifica seu estado e notifica suas visões e



*controladores; as visões podem consultar diretamente o estado do modelo, sem interferência do controlador.*

(a) Errado, fica na Camada de Modelo; (b) Errado, quase tudo correto, mas as ações são executadas na Camada de Modelo; (c) Errado, trata-se de uma arquitetura triangular, logo pode ser acessada diretamente; (d) Correto, geralmente há um controlador padrão, sim (Ex: Servlets); (e) Errado, as Servlets são representadas na Camada de Controle.

**Gabarito:** D

**41. (FCC – 2009 – TJ/SE - Analista de Sistemas)** No modelo de três camadas MVC para web services, o responsável pela apresentação que também recebe os dados de entrada do usuário é a camada:

- a) View.
- b) Application.
- c) Controller.
- d) Data.
- e) Model.

**Comentários:**

*Apresentação?* Camada de Visão (View).

**Gabarito:** A

**42. (FCC – 2009 – TRT/MA - Analista de Sistemas)** Considere as funções:

- I. Seleção do comportamento do modelo.
- II. Encapsulamento dos objetos de conteúdo.
- III. Requisição das atualizações do modelo.

Na arquitetura Model-View-Control - MVC, essas funções correspondem, respectivamente, a:

- a) Model, View e Control.
- b) Control, View e Model.
- c) View, Model e Control.
- d) Control, Model e View.
- e) View, Control e Model.

**Comentários:**



A seleção do comportamento do modelo é feita pela Camada de Controle; Encapsulamento dos objetos de conteúdo é feito pela Camada de Modelo; Requisição das atualizações do modelo são feitas pela Camada de Visão.

**Gabarito:** D

**43.(UFG - 2017 – SANEAGO – Analista de Sistemas)** Dentro dos padrões arquiteturais de software, a arquitetura Model-View-ViewModel (MVVM) é próxima da arquitetura Model-View-Presenter (MVP), porém diferencia-se desta pelo fato de:

- a) ser desprovida de um componente controlador como existe no Model-View-Controller (MVC).
- b) implementar o padrão de projeto Observer na ligação entre dados (ViewModel) e tela (view).
- c) ligar diretamente as classes de tela (view) e dados (Model) dentro da estrutura do projeto.
- d) vincular a realização de atualizações de tela (view) à atualização de dados (ViewModel).

#### Comentários:

*Não devemos confundir também com o MVVM (Model-View-ViewModel). O MVVM é uma pequena evolução do MVP em um lado e um retrocesso em outro. Nele o ViewModel não está ciente do que ocorre no View, **mas este está ciente do que ocorre no ViewModel** (como no Padrão de Projeto Observer). No caso do Model, ambos estão cientes do que ocorre em cada um.*

*O nome se dá porque ele adiciona propriedades e operações ao Model para atender as necessidades do View, **portanto ele cria um novo modelo para a visualização**. É possível associar várias Views para um ModelView. É comum que as Views sejam definidas de forma declarativa (HTML/CSS, XAML, etc.). O Data Binding é feito entre a View e o ViewModel.*

Perfeito! É a aplicação do Padrão de Projeto Observer!

**Gabarito:** B

**44.(IBFC / EBSEH – 2017)** O modelo de três camadas físicas (3-tier), especificado nas alternativas, divide um aplicativo de modo que a lógica de negócio resida no meio das três camadas, foi adaptado como uma arquitetura para as aplicações Web em todas as linguagens de programação maiores. Muitos frameworks de aplicação comerciais e não comerciais foram criados tendo como base a arquitetura:

- a) MVC (Model-View-Controller)



- b) MDB (Model-Data-Business)
- c) UDC (User-Data-Controller)
- d) MDC (Model-Data-Controller)
- e) UVB (User-View-Business).

#### Comentários:

A divisão do aplicativo que separa em três camadas é a Arquitetura MVC (Model, View e Control).

**Gabarito:** Letra A

**45.(IBFC / TRE-AM – 2014)** Na arquitetura cliente-servidor, além dos dois principais componentes Cliente e o Servidor, existe um terceiro elemento intermediando os dois. Esse componente é chamado tecnicamente de:

- a) coreware.
- b) middleware.
- c) mainware.
- d) centerware.

#### Comentários:

O conceito de Middleware é amplamente utilizado em diversos contextos da computação. Como o próprio nome remete, ele é um "software do meio" ou "software intermediário". Possui diversas aplicações, como: intermediar a comunicação entre cliente e servidor (muito utilizado em ambientes distribuídos); intermediar a comunicação entre Softwares que utilizam protocolos ou plataformas diferentes; Intermediar a comunicação entre Sistema Operacional e Aplicações.

**Gabarito:** Letra B

**46.(IBFC / TRE-AM – 2014)** No desenvolvimento de sistemas dentro do conceito da arquitetura cliente-servidor de três camadas, temos as seguintes camadas:

1. Camada de Dados.
2. Camada de Apresentação.
3. Camada de Aplicações.
4. Camada de Negócio.

Estão corretas as afirmativas:

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



### Comentários:

A arquitetura cliente-servidor se divide em Camada de Dados, Camada de Apresentação e Camada de Negócio. Logo, somente 1, 2 e 4.

**Gabarito:** Letra A

---



## LISTA DE EXERCÍCIOS – CESPE

1. **(CESPE – 2006 – CENSIPAM - Analista de Sistemas)** O padrão MVC organiza um software em modelo, visão e controle. O modelo encapsula as principais funcionalidades e dados. As visões apresentam os dados aos usuários. Uma visão obtém os dados do modelo via funções disponibilizadas pelo modelo; só há uma visão para um modelo. Usuários interagem via controladoras que traduzem os eventos em solicitações ao modelo ou à visão; podem existir várias controladoras associadas a uma mesma visão.
2. **(CESPE – 2006 – CENSIPAM - Analista de Sistemas)** No padrão MVC, se um usuário modifica o modelo, as visões que dependem desse modelo refletem essas modificações, pois o modelo notifica as visões quando ocorre uma modificação nos seus dados. Portanto, é usado um mecanismo para propagação de modificações que mantém um registro dos componentes que dependem do modelo.
3. **(CESPE – 2013 – BACEN - Analista de Sistemas)** MVC (Model-View-Controller) é um modelo de arquitetura de software que separa, de um lado, a representação da informação e, de outro, a interação do usuário com a informação.
4. **(CESPE – 2008 – TJ/CE - Analista de Sistemas)** A arquitetura MVC fornece uma maneira de dividir a funcionalidade envolvida na manutenção e apresentação dos dados de uma aplicação web.
5. **(CESPE – 2008 – TJ/CE - Analista de Sistemas)** A arquitetura MVC foi desenvolvida recentemente para mapear as tarefas complexas de saída do sistema do usuário.
6. **(CESPE – 2008 – TJ/CE - Analista de Sistemas)** Na arquitetura MVC, um controlador define o comportamento da aplicação, já que este é o responsável por interpretar as ações do usuário e as relaciona com as chamadas do modelo.
7. **(CESPE – 2008 – TJ/CE - Analista de Sistemas)** A arquitetura MVC não separa a informação de sua apresentação, porque, em sistemas web, informação e apresentação estão na mesma camada.
8. **(CESPE – 2008 – TJ/CE - Analista de Sistemas)** O desenvolvimento de sistemas web ocorre tipicamente em três camadas. A arquitetura MVC aumenta o escopo do desenvolvimento para, no máximo, quatro camadas, sendo a quarta camada o processamento dos dados do usuário.
9. **(CESPE – 2009 – ANAC - Analista de Sistemas)** O framework modelo visão controlador (MVC – model view controller) é muito utilizado para projeto da GUI (graphical user interface) de programas orientados a objetos.



10. **(CESPE – 2009 – TCU - Analista de Sistemas)** No MVC (model-view-controller), um padrão recomendado para aplicações interativas, uma aplicação é organizada em três módulos separados. Um para o modelo de aplicação com a representação de dados e lógica do negócio, o segundo com visões que fornecem apresentação dos dados e input do usuário e o terceiro para um controlador que despacha pedidos e controle de fluxo.
11. **(CESPE – 2010 – EMBASA - Analista de Sistemas)** O MVC promove a estrita separação de responsabilidades entre os componentes de uma interface.
12. **(CESPE – 2010 – EMBASA - Analista de Sistemas)** No MVC, a visão é responsável pela manutenção do estado da aplicação.
13. **(CESPE – 2010 – EMBASA - Analista de Sistemas)** O modelo no MVC tem como atribuição exibir a parte que é responsável pela manutenção da aplicação para o usuário.
14. **(CESPE – 2010 – EMBASA - Analista de Sistemas)** O controlador é responsável pela coordenação entre atualizações no modelo e interações com o usuário.
15. **(CESPE – 2010 – EMBASA - Analista de Sistemas)** Por meio do MVC, é possível o desenvolvimento de aplicações em 3 camadas para a Web.
16. **(CESPE – 2011 – CET - Analista de Sistemas)** No padrão de desenvolvimento modelo-visualização-controlador (MVC), o controlador é o elemento responsável pela interpretação dos dados de entrada e pela manipulação do modelo, de acordo com esses dados.
17. **(CESPE – 2011 – MEC - Analista de Sistemas)** O modelo MVC pode ser usado para construir a arquitetura do software a partir de três elementos: modelo, visão e controle, sendo definidas no controle as regras de negócio que controlam o comportamento do software a partir de restrições do mundo real.
18. **(CESPE – 2011 – MEC - Analista de Sistemas)** O controlador, no modelo MVC, realiza a comunicação entre as camadas de visão e modelo.
19. **(CESPE – 2011 – MEC - Analista de Sistemas)** No MVC, o modelo representa o estado geral do sistema.
20. **(CESPE – 2011 – MEC - Analista de Sistemas)** Apesar do seu amplo emprego em aplicações web, o MVC deve ser utilizado apenas em interfaces gráficas em função de sua arquitetura de componentes.
21. **(CESPE – 2011 – MEC - Analista de Sistemas)** No MVC, é o modelo que permite apresentar, de diversas formas diferentes, os dados para o usuário.





22. **(CESPE – 2011 – MEC - Analista de Sistemas)** O controlador é o responsável pelas regras de negócio e pelos dados de uma aplicação no MVC.
23. **(CESPE – 2013 – TCE/ES - Analista de Sistemas – Letra E)** No Padrão MVC, as regras do negócio que definem a forma de acesso e modificação dos dados são geridas pelo controlador.
24. **(CESPE – 2015 – STJ - Analista de Sistemas)** Na arquitetura em camadas MVC (modelo-visão-controlador), o modelo encapsula o estado de aplicação, a visão solicita atualização do modelo e o controlador gerencia a lógica de negócios.
25. **(CESPE – 2015 – MEC - Analista de Sistemas)** O controlador gerencia as requisições dos usuários encapsulando as funcionalidades e prepara dados do modelo.
26. **(CESPE – 2015 – MEC - Analista de Sistemas)** A visão encapsula objetos de conteúdo, solicita atualizações do modelo e seleciona o comportamento do modelo.
27. **(CESPE – 2015 – STJ - Analista de Sistemas)** No padrão em camadas modelo-visão-controle (MVC), o controle é responsável por mudanças de estado da visão.
28. **(CESPE – 2014 – ANTAQ - Analista de Sistemas)** O modelo MVC é um padrão de arquitetura que consiste na definição de camadas para a construção de softwares.
29. **(CESPE – 2014 – ANTAQ - Analista de Sistemas)** O controller tem a responsabilidade de armazenar e buscar os dados que deverão ser exibidos pelo view.
30. **(CESGRANRIO – 2014 – CEFET/RJ - Analista de Sistemas)** No contexto da Arquitetura de Sistemas, o MVC (model – view – controller) é um estilo arquitetural:
- a) interativo
  - b) estrutural
  - c) distribuído
  - d) adaptável
  - e) monolítico
31. **(ESAF – 2012 – CGU - Analista de Sistemas)** A definição de que um sistema deve ser desenvolvido em três níveis é feita pelo padrão de projeto:
- a) MVC (Model View Controller).
  - b) MVC-Dev (Model Value Constructive Development).
  - c) TMS (Time Milestones Setting).
  - d) PMC (Project Main Controller).
  - e) MCA (Model Classes Assignment).



**32. (ESAF – 2010 – CVM - Analista de Sistemas)** Modelo MVC significa:

- a) Modo-View-Construtor.
- b) Modelo-View-Controlador.
- c) Modelo-Versão-Case.
- d) Módulo-Verificador-Controlador.
- e) Medida-Virtual-Concepção.

**33. (FCC – 2012 – TST - Analista de Sistemas)** No padrão MVC é possível definir grupos de componentes principais: o Model (Modelo), o View (Apresentação) e o Controller (Controle). Deve fazer parte do componente:

- a) Controller, uma classe que contém um método com a finalidade de calcular o reajuste de salário dos funcionários.
- b) View, uma classe que contém um método para persistir o salário reajustado de um funcionário.
- c) Controller, as animações desenvolvidas em Flash.
- d) View, as validações necessárias ao sistema, geralmente definidas através de um conjunto de comparações.
- e) Model, as classes com métodos conhecidos como setters e getters e que representam tabelas do banco de dados.

**34. (FCC – 2012 – MPE/AP - Analista de Sistemas)** Em uma Aplicação Web desenvolvida utilizando o design pattern MVC, as páginas HTML e as classes com métodos que acessam o banco de dados e executam instruções SQL são representadas, respectivamente, nos componentes:

- a) Presentation e Business.
- b) View e Model.
- c) Controller e Model.
- d) Model e Business.
- e) View e Business.

**35. (FCC – 2012 – MPE/PE - Analista de Sistemas)** O padrão de projeto utilizado em aplicações WEB que permite separar as páginas e classes da aplicação em três grupos (muitas vezes chamados de camadas) conhecidos como Apresentação, Controle e Modelo é denominado de:

- a) 3-tier.
- b) DAO.



- c) MVC.
- d) DTO.
- e) DBO.

**36. (FCC – 2012 – TRT/PE - Analista de Sistemas)** O padrão de arquitetura MVC é um modelo de camadas que divide a aplicação em três componentes: Model (modelo), View (visualizador) e Controller (controlador). As funções de cada um destes três componentes são apresentadas abaixo:

- I. interpreta eventos de entrada e envia requisições para o modelo de dados; em seguida, processa os dados carregados a partir do modelo e envia para o visualizador.
- II. encapsula o acesso aos dados e funções básicas da aplicação, fornecendo ao usuário procedimentos que executam tarefas específicas.
- III. exibe para o usuário os dados fornecidos pelo controle e estabelece uma interface para interação entre o usuário e a aplicação.

A associação correta do componente do padrão MVC com sua função está expressa, respectivamente, em

- a) (I) Controller; (II) Model; (III) View;
- b) (I) Model; (II) Controller; (III) View;
- c) (I) View; (II) Model; (III) Controller;
- d) (I) Controller; (II) View; (III) Model;
- e) (I) Model; (II) View; (III) Controller;

**37. (FCC – 2012 – TJ/PE - Analista de Sistemas)** Com relação à arquitetura MVC, considere:

- I. O MODEL representa os dados da empresa e as regras de negócio que governam o acesso e atualização destes dados.
- II. O VIEW acessa os dados da empresa através do MODEL e especifica como esses dados devem ser apresentados. É de responsabilidade do VIEW manter a consistência em sua apresentação, quando o MODEL é alterado.
- III. O CONTROLLER traduz as interações do VIEW em ações a serem executadas pelo MODEL. Com base na interação do usuário e no resultado das ações do MODEL, o CONTROLLER responde selecionando uma VIEW adequada.
- IV. Permite uma única VIEW para compartilhar o mesmo modelo de dados corporativos em um fluxo de comunicação sequencial.

Está correto o que se afirma em:



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

**38. (FCC – 2012 – MPE/PE - Analista de Sistemas)** O componente Controller do MVC:

- a) Define o comportamento da aplicação, as ações do usuário para atualizar os componentes de dados e seleciona os componentes para exibir respostas de requisições.
- b) Envia requisições do usuário para o controlador e recebe dados atualizados dos componentes de acesso a dados.
- c) Responde às solicitações de queries e encapsula o estado da aplicação.
- d) Notifica os componentes de apresentação das mudanças efetuadas nos dados e expõe a funcionalidade da aplicação.
- e) É onde são concentradas todas as regras de negócio da aplicação e o acesso aos dados.

**39. (FCC – 2011 – TRT/MT - Analista de Sistemas)** No projeto de arquitetura modelo-visão-controle (MVC), o controlador:

- a) renderiza a interface de usuário a partir da visão, o modelo encapsula funcionalidades e objetos de conteúdo e a visão processa e responde a eventos e invoca alterações ao controlador.
- b) encapsula funcionalidades e objetos de conteúdo, o modelo processa e responde a eventos e invoca alterações ao controlador e a visão renderiza a interface de usuário a partir do modelo.
- c) encapsula funcionalidades e objetos de conteúdo, o modelo renderiza a interface de usuário a partir da visão e a visão processa e responde a eventos e invoca alterações ao controlador.
- d) processa e responde a eventos e invoca alterações ao modelo, o modelo encapsula funcionalidades e objetos de conteúdo e a visão renderiza a interface de usuário a partir do modelo.
- e) processa e responde a eventos e invoca alterações ao modelo, o modelo renderiza a interface de usuário a partir da visão e a visão encapsula funcionalidades e objetos de conteúdo.



**40. (FCC – 2010 – AL/SP - Analista de Sistemas)** Sobre as camadas do modelo de arquitetura MVC (Model- View-Controller) usado no desenvolvimento web é correto afirmar:

- a) Todos os dados e a lógica do negócio para processá-los devem ser representados na camada Controller.
- b) A camada Model pode interagir com a camada View para converter as ações do cliente em ações que são compreendidas e executadas na camada Controller.
- c) A camada View é a camada responsável por exibir os dados ao usuário. Em todos os casos essa camada somente pode acessar a camada Model por meio da camada Controller.
- d) A camada Controller geralmente possui um componente controlador padrão criado para atender a todas as requisições do cliente.
- e) Em aplicações web desenvolvidas com Java as servlets são representadas na camada Model.

**41. (FCC – 2009 – TJ/SE - Analista de Sistemas)** No modelo de três camadas MVC para web services, o responsável pela apresentação que também recebe os dados de entrada do usuário é a camada:

- a) View.
- b) Application.
- c) Controller.
- d) Data.
- e) Model.

**42. (FCC – 2009 – TRT/MA - Analista de Sistemas)** Considere as funções:

- I. Seleção do comportamento do modelo.
- II. Encapsulamento dos objetos de conteúdo.
- III. Requisição das atualizações do modelo.

Na arquitetura Model-View-Control - MVC, essas funções correspondem, respectivamente, a:

- a) Model, View e Control.
- b) Control, View e Model.
- c) View, Model e Control.
- d) Control, Model e View.



e) View, Control e Model.

**43. (UFG - 2017 – SANEAGO – Analista de Sistemas)** Dentro dos padrões arquiteturais de software, a arquitetura Model-View-ViewModel (MVVM) é próxima da arquitetura Model-View-Presenter (MVP), porém diferencia-se desta pelo fato de:

- a) ser desprovida de um componente controlador como existe no Model-View-Controller (MVC).
- b) implementar o padrão de projeto Observer na ligação entre dados (ViewModel) e tela (view).
- c) ligar diretamente as classes de tela (view) e dados (Model) dentro da estrutura do projeto.
- d) vincular a realização de atualizações de tela (view) à atualização de dados (ViewModel).

**44. (IBFC / EBSERH – 2017)** O modelo de três camadas físicas (3-tier), especificado nas alternativas, divide um aplicativo de modo que a lógica de negócio resida no meio das três camadas, foi adaptado como uma arquitetura para as aplicações Web em todas as linguagens de programação maiores. Muitos frameworks de aplicação comerciais e não comerciais foram criados tendo como base a arquitetura:

- a) MVC (Model-View-Controller)
- b) MDB (Model-Data-Business)
- c) UDC (User-Data-Controller)
- d) MDC (Model-Data-Controller)
- e) UVB (User-View-Business).

**45. (IBFC / TRE-AM – 2014)** Na arquitetura cliente-servidor, além dos dois principais componentes Cliente e o Servidor, existe um terceiro elemento intermediando os dois. Esse componente é chamado tecnicamente de:

- a) coreware.
- b) middleware.
- c) mainware.
- d) centerware.

**46. (IBFC / TRE-AM – 2014)** No desenvolvimento de sistemas dentro do conceito da arquitetura cliente-servidor de três camadas, temos as seguintes camadas:

1. Camada de Dados.
2. Camada de Apresentação.
3. Camada de Aplicações.
4. Camada de Negócio.



Estão corretas as afirmativas:

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



## GABARITO - CESPE

- |             |             |
|-------------|-------------|
| 1. ERRADO   | 24. ERRADO  |
| 2. CORRETO  | 25. ERRADO  |
| 3. CORRETO  | 26. ERRADO  |
| 4. CORRETO  | 27. ERRADO  |
| 5. ERRADO   | 28. CORRETO |
| 6. CORRETO  | 29. ERRADO  |
| 7. ERRADO   | 30. LETRA A |
| 8. ERRADO   | 31. LETRA A |
| 9. CORRETO  | 32. LETRA B |
| 10. CORRETO | 33. LETRA A |
| 11. ANULADA | 34. LETRA B |
| 12. ERRADO  | 35. LETRA C |
| 13. ERRADO  | 36. LETRA A |
| 14. CORRETO | 37. LETRA B |
| 15. CORRETO | 38. LETRA A |
| 16. CORRETO | 39. LETRA D |
| 17. ERRADO  | 40. LETRA D |
| 18. CORRETO | 41. LETRA A |
| 19. CORRETO | 42. LETRA D |
| 20. ERRADO  | 43. LETRA B |
| 21. ERRADO  | 44. LETRA A |
| 22. ERRADO  | 45. LETRA B |
| 23. ERRADO  | 46. LETRA A |





## 5 – ARQUITETURA DISTRIBUÍDA

Os sistemas de computação estão passando por uma evolução. Desde 1945, quando começou a era moderna dos computadores, até aproximadamente 1985, os computadores eram grandes e caros. **Contudo, mais ou menos a partir de meados da década de 80, dois avanços tecnológicos começaram a mudar essa situação.** O primeiro foi o desenvolvimento de microprocessadores de grande capacidade.

**O segundo desenvolvimento foi a invenção de redes de computadores de alta velocidade.** Nesse cenário, surgem os sistemas distribuídos, os quais são plataformas formadas por um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente. Uma breve introdução histórica e agora podemos partir para o assunto...

A Arquitetura Mainframe (Grande Porte) é conhecida por ser altamente centralizada. Todo processamento ocorre no mainframe e há um bocado de terminal burro que o acessa. **Já a Arquitetura Distribuída inverte essa concepção, na medida em que o processamento é dispersado através da organização e seus desktops e servidores.** *Professor, quais são as vantagens dessa abordagem?*

**Cara, há ganhos de responsividade, escalabilidade, redundância, disponibilidade, compartilhamento de recursos computacionais, controle e envolvimento do usuário, entre outros.** Componentes podem ser hospedados em diferentes plataformas e tecnologias, além de se comunicarem por meio de uma rede – sendo que cada nó tem sua responsabilidade específica no processamento de uma tarefa comum.

*Galera, o usuário nota que o processamento é distribuído? Não, é tudo transparente! Para ele, é como se tudo estivesse sendo executado como um único sistema.* Os computadores em um sistema distribuído podem estar fisicamente próximos e conectados por uma rede local ou podem estar geograficamente distantes e conectados por uma rede remota.

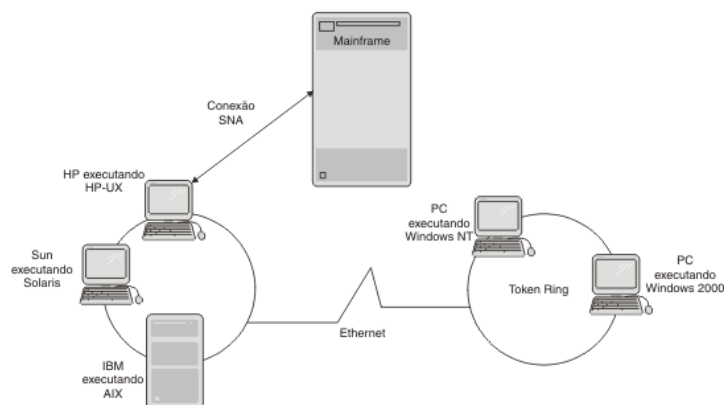
Uma arquitetura distribuída pode consistir em uma série de configurações possíveis, como mainframes, computadores pessoais, estações de trabalho, minicomputadores e assim por diante. **Como eu já mencionei anteriormente, a meta da computação distribuída é fazer um trabalho de rede como um computador único.** *Entendido? Prosseguindo...*

Os sistemas de computação distribuída podem ser executados no hardware fornecido por muitos fornecedores e podem utilizar diversos componentes de software baseados em padrões. **Esses sistemas são independentes do software subjacente.** Eles podem ser executados em vários sistemas operacionais e podem utilizar vários protocolos de comunicação.

Alguns hardware podem utilizar UNIX como o sistema operacional, enquanto outro hardware pode utilizar sistemas operacionais Windows. **Para comunicação entre máquinas, esse hardware pode utilizar SNA ou TCP/IP em Ethernet ou Token Ring.** A imagem abaixo (retirada do site da



IBM) apresenta uma possível arquitetura distribuída – percebam como é bastante descentralizada:



Esse sistema contém duas LANs conectadas entre si. Uma consiste em estações de trabalho UNIX (HP-UX, Solaris, AIX) de vários fabricantes (HP, Sun, IBM); já a outra consiste em PCs executando vários sistemas operacionais (NT e 2000) em Token Ring. Há uma LAN conectada a um mainframe por meio de uma conexão SNA. **Cliente/Servidor e P2P são exemplos famosos de Arquitetura Distribuída.**

**Galera, não confundam Arquitetura Distribuída com Arquitetura Paralela.** No primeiro caso, processos são executados concorrentemente em máquinas diferentes dentro de uma rede; é uma arquitetura fracamente acoplada; é mais imprevisível devido ao uso de redes de computadores e suas falhas; apresenta controle e acesso descentralizado dos recursos.

**No segundo caso, processos são executados concorrentemente em uma mesma máquina; é uma arquitetura fortemente acoplada** – podem compartilhar hardware ou se comunicar através de um barramento de alta velocidade; são mais previsíveis, porque falhas são menos comuns em barramentos de alta velocidade; apresenta controle e acesso centralizado dos recursos.

**As grandes vantagens dos Sistemas Distribuídos são: utilizam** melhor o poder de processamento; apresenta melhor desempenho; permitem compartilhar dados e recursos; podem apresentar maior confiabilidade; permitem reutilizar serviços já disponíveis; atendem um maior número de usuários; balanceamento de carga; escalabilidade; entre outros.

**Nem tudo são flores, há também desvantagens:** desenvolver, gerenciar e manter sistemas distribuídos; controlar o acesso concorrente a dados e a recursos compartilhados; evitar que falhas de máquinas ou da rede comprometam o funcionamento do sistema; garantir a segurança do sistema e o sigilo dos dados trocados entre máquinas; lidar com a heterogeneidade do ambiente; entre outros.

Um último conceito sobre esse assunto: Middleware. **Um Middleware é uma camada de software que permite que elementos de aplicações interoperem através de redes de**



**computadores** – imaginem um software que padroniza interfaces de programação para que você não se preocupe se existem protocolos, arquiteturas, sistemas operacionais ou bancos de dados diferentes.

Ele provê um modo para obter dados de um lugar para outro. **Além disso, é capaz de mascarar diferenças existentes entre sistemas operacionais, plataformas de hardware e protocolos de rede.** O Middleware oculta do desenvolvedor da aplicação a complexidade do processo de transporte da rede. Os exemplos mais comuns são: RPC, RMI, CORBA, DCOM, entre outros.

**Quando utilizamos linguagens orientadas a objetos em arquiteturas distribuídas, chegamos ao conceito de objetos distribuídos e invocação remota.** Como o nome bem diz, objetos distribuídos são instâncias que podem ser acessadas remotamente e trazem para as aplicações distribuídas todas as vantagens da orientação a objetos: maior reusabilidade, segurança, padronização.

Nas linguagens estruturadas, as funções remotas são utilizadas a partir de chamadas remotas, as famosas Remote Procedure Call (RPC). Os objetos não possuem funções, mas sim métodos, e não são “chamados”, mas invocados. *Alguém consegue descobrir o nome similar ao RPC para objetos distribuídos??* **Acertou quem disse Remote Method Invocation ou RMI.**

**O sistema RMI permite que um objeto executado em uma Java Virtual Machine (JVM) possa invocar métodos de outro objeto remoto, ou seja, executado em outra JVM.** RMI permite comunicação remota entre programas escritos na linguagem Java.

*Professor, quer dizer que, de uma máquina eu posso invocar um método de um objeto de outra máquina?*

*Mas se eu não tiver acesso ao código remoto?* Excelente pergunta, meu nobre padawan! Para isso, utilizamos um recurso muito importante da orientação a objetos: interfaces. **Desse modo, basta que o objeto que invoca o objeto distribuído tenha a interface com as assinaturas dos métodos remotos.** *Vocês pegaram essa parte aqui?*

Além da interface, existe um objeto local que implementa a interface remota, mas na verdade não faz nada além de repassar tudo ao objeto remoto (que também implementa a interface, obviamente). **O nome desse objeto local que implementa a interface remota é o proxy** *(qualquer semelhança com o padrão de projeto não é mera coincidência).*

*Ah, professor, o senhor QUASE me enganou! Como pode um objeto num local saber onde está o objeto remoto, e como saber qual das classes que implementam a interface remota estão disponíveis para uso?* Esse aluno vai passar! Meu padawan, você tem razão, de novo. **Para saber qual objeto implementa o serviço da interface remota a ser invocado e onde ele se encontra, existe um serviço chamado Binder!**



**O binder transforma o nome de um serviço remoto que o objeto local quer invocar numa referência real de um objeto distribuído.** Os servidores com os objetos distribuídos devem registrar os serviços no Binder para que os clientes possam acessá-los. Basta que, antes, haja um contrato para que os nomes dos serviços sejam bem conhecidos por todos: clientes, servidores com objetos remotos e Binder.

**Os objetos remotos do lado do servidor são dois para cada objeto distribuído: skeleton e dispatcher.** O skeleton implementa a interface remota, mas ainda não é quem faz o trabalho braçal (que pode ser uma classe “normal” que implementa a interface remota). Já o dispatcher faz parte do arcabouço para receber as invocações remotas e chamar o skeleton certo.

**Proxy (lado cliente), skeleton (lado servidor) e dispatcher (lado servidor) compõe o middleware de um sistema de objetos distribuídos.** Em algumas implementações, o skeleton e dispatcher são unidos numa classe. Finalmente, o objeto distribuído, que vai realizar cálculos, acessar bancos de dados e tudo o que o cliente precisar, é acessado pelo Skeleton, que recebe os resultados e faz todo o caminho de volta.



## EXERCÍCIOS COMENTADOS

1. (CESPE – 2008 – IPEA - Analista de Sistemas) A arquitetura distribuída é caracterizada pelo compartilhamento de recursos computacionais e serviços por meio da comunicação direta e descentralizada entre os sistemas envolvidos e inclui, entre outras coisas, a troca de informações, ciclos de processamento e espaço de armazenamento em disco.

### Comentários:

*Cara, há ganhos de responsividade, escalabilidade, redundância, disponibilidade, compartilhamento de recursos computacionais, controle e envolvimento do usuário, entre outros. Componentes podem ser hospedados em diferentes plataformas e tecnologias, além de se comunicarem por meio de uma rede – sendo que cada nó tem sua responsabilidade específica no processamento de uma tarefa comum.*

Conforme vimos em aula, ela se caracteriza por seu compartilhamento de recursos e serviços – além da comunicação direta e descentralizada entre os sistemas. Enfim, a questão está perfeita!

Gabarito: C

2. (CESPE – 2008 – SERPRO - Analista de Sistemas) Uma arquitetura distribuída permite a divisão de uma mesma tarefa em diferentes processadores em uma mesma CPU. Essa característica aumenta a velocidade de processamento de uma informação.

### Comentários:

*Galera, não confundam Arquitetura Distribuída com Arquitetura Paralela. No primeiro caso, processos são executados concorrentemente em máquinas diferentes dentro de uma rede; é uma arquitetura fracamente acoplada; é mais imprevisível devido ao uso de redes de computadores e suas falhas; apresenta controle descentralizado dos recursos.*

*No segundo caso, processos são executados concorrentemente em uma mesma máquina; é uma arquitetura fortemente acoplada – podem compartilhar hardware ou se comunicar através de um barramento de alta velocidade; são mais previsíveis, porque falhas são menos comuns em barramentos de alta velocidade; apresenta controle centralizado dos recursos.*

Conforme vimos em aula, não se deve confundir Arquitetura Distribuída com Arquitetura Paralela. Observem que, no caso apresentado, não há uma rede de computadores – tudo ocorre em uma mesma CPU! Logo, a questão trata de Arquitetura Paralela.

Gabarito: E



3. (CESPE – 2009 – ANATEL - Analista de Sistemas) Uma das vantagens da arquitetura distribuída é o compartilhamento de recursos, que permite que sistemas, aplicativos e dispositivos periféricos, como discos, impressoras, arquivos, estejam associados a diferentes computadores em uma rede. Uma segunda vantagem é a concorrência, uma vez que vários processos podem operar ao mesmo tempo em diferentes computadores na rede. E, por fim, uma terceira vantagem é a proteção, pois o acesso é feito de forma centralizada.

#### Comentários:

*Galera, não confundam Arquitetura Distribuída com Arquitetura Paralela. No primeiro caso, processos são executados concorrentemente em máquinas diferentes dentro de uma rede; é uma arquitetura fracamente acoplada; é mais imprevisível devido ao uso de redes de computadores e suas falhas; **apresenta controle e acesso descentralizado dos recursos**.*

Conforme vimos em aula, o acesso é feito de forma descentralizada.

**Gabarito:** E

4. (CESPE – 2009 – ANTAQ - Analista de Sistemas) Uma das desvantagens da arquitetura distribuída é sua complexidade, uma vez que é mais difícil compreender as propriedades emergentes dos sistemas que as dos sistemas centralizados.

#### Comentários:

*Nem tudo são flores, há também desvantagens: desenvolver, gerenciar e manter sistemas distribuídos; **controlar o acesso concorrente a dados e a recursos compartilhados**; evitar que falhas de máquinas ou da rede comprometam o funcionamento do sistema; garantir a segurança do sistema e o sigilo dos dados trocados entre máquinas; lidar com a heterogeneidade do ambiente; entre outros.*

Conforme vimos em aula, é realmente complexo compreender e controlar as propriedades de sistemas distribuídos. *Já imaginaram controlar as propriedades de concorrência e transação de um sistema que está distribuído?* Pois é!

**Gabarito:** C

5. (CESPE – 2009 – INMETRO - Analista de Sistemas) Em uma arquitetura distribuída, middleware é definido como uma camada de software cujo objetivo é mascarar a heterogeneidade e fornecer um modelo de programação conveniente para os programadores de aplicativos. Como exemplos de middlewares é correto citar: Sun RPC, CORBA, RMI Java e DCOM da Microsoft.

#### Comentários:



Um último conceito sobre esse assunto: Middleware. **Um Middleware é uma camada de software que permite que elementos de aplicações interoperem através de redes de computadores** – imaginem um software que padroniza interfaces de programação para que você não se preocupe se existem protocolos, arquiteturas, sistemas operacionais ou bancos de dados diferentes.

Ele provê um modo para obter dados de um lugar para outro. **Além disso, é capaz de mascarar diferenças existentes entre sistemas operacionais, plataformas de hardware e protocolos de rede.** O Middleware oculta do desenvolvedor da aplicação a complexidade do processo de transporte da rede. Os exemplos mais comuns são: RPC, RMI, CORBA, DCOM, entre outros.

Conforme vimos em aula, o item está perfeito!

Gabarito: C





## LISTA DE EXERCÍCIOS

1. **(CESPE – 2008 – IPEA - Analista de Sistemas)** A arquitetura distribuída é caracterizada pelo compartilhamento de recursos computacionais e serviços por meio da comunicação direta e descentralizada entre os sistemas envolvidos e inclui, entre outras coisas, a troca de informações, ciclos de processamento e espaço de armazenamento em disco.
2. **(CESPE – 2008 – SERPRO - Analista de Sistemas)** Uma arquitetura distribuída permite a divisão de uma mesma tarefa em diferentes processadores em uma mesma CPU. Essa característica aumenta a velocidade de processamento de uma informação.
3. **(CESPE – 2009 – ANATEL - Analista de Sistemas)** Uma das vantagens da arquitetura distribuída é o compartilhamento de recursos, que permite que sistemas, aplicativos e dispositivos periféricos, como discos, impressoras, arquivos, estejam associados a diferentes computadores em uma rede. Uma segunda vantagem é a concorrência, uma vez que vários processos podem operar ao mesmo tempo em diferentes computadores na rede. E, por fim, uma terceira vantagem é a proteção, pois o acesso é feito de forma centralizada.
4. **(CESPE – 2009 – ANTAQ - Analista de Sistemas)** Uma das desvantagens da arquitetura distribuída é sua complexidade, uma vez que é mais difícil compreender as propriedades emergentes dos sistemas que as dos sistemas centralizados.
5. **(CESPE – 2009 – INMETRO - Analista de Sistemas)** Em uma arquitetura distribuída, middleware é definido como uma camada de software cujo objetivo é mascarar a heterogeneidade e fornecer um modelo de programação conveniente para os programadores de aplicativos. Como exemplos de middlewares é correto citar: Sun RPC, CORBA, RMI Java e DCOM da Microsoft.





## GABARITO

1. CORRETO
2. ERRADO
3. ERRADO
4. CORRETO
5. CORRETO



# ESSA LEI TODO MUNDO CONHECE: PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.



1 Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.



2 Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).



3 Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).



4 Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).



5 Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).



6 Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).



7 Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.



8 O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.



Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.