ARQUITETURA DE SISTEMAS

Autor: Leonardo Costa Passos

Resumo com o que há de mais importante da disciplina de Arquitetura de Sistemas para provas de Concursos Públicos com foco na banca CESGRANRIO.

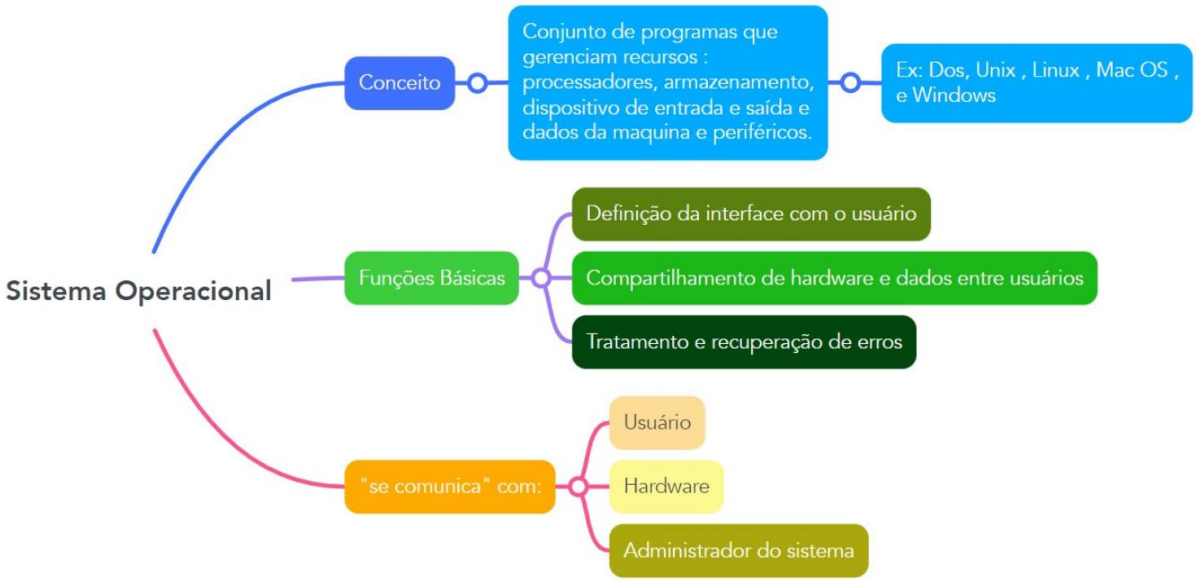
Se o conteúdo for útil na sua jornada de estudos, você pode me agradecer fazendo um PIX de um valor que considere justo para a seguinte chave:

leonardx@gmail.com

Table of Contents

1. CONCEITOS S.O.

O Kernel ou núcleo do sistema faz a interface entre o hardware e as aplicações, em outras palavras, se as aplicações e o hardware do sistema fossem cidadãos de diferentes países e precisam se comunicar entre si o Kernel agiria como mediador/tradutor da conversa.



Entre as diversas funções do Kernel, podemos citar:

- Gerência de memória, processador e dispositivos de entrada e saída;
- Realizar chamadas de sistema;

> As chamadas de sistema são os programas de usuários, ou aplicações, solicitando serviços ao Kernel.

Os SISTEMAS OPERACIONAIS EMBARCADOS: são executados em máquinas de pequeno porte, utilizadas para artefatos de função específica e que não aceitam softwares instalados por usuários.

- Esses sistemas foram criados para funções específicas para um sistema que ele consiga controlar. Além de possuir um baixo custo, não há possibilidade de instalação de softwares por parte de um usuário, por se tratar de um sistema dedicado a um tipo de função.

- Os três estados nos quais um processo pode se encontrar são:

  1 – *Em execução* (realmente *usando a CPU* naquele instante);

  2 – *Pronto* (*executável*; *temporariamente parado* para dar lugar para outro processo);

  3 – *Bloqueado* (*incapaz de executar* enquanto um evento externo não ocorra).

- Os processos escalonados deverão ser os que se encontram apenas no estado de PRONTO.

- Cache ou Caching é um componente que armazena dados para futuras solicitações para que os dados possam ser servidos mais rápido;
  - os dados armazenados no cache pode ser o resultado de um cálculo anterior, ou do duplicado dos dados armazenados em outro lugar.

- Um acerto de cache ocorre quando os dados solicitados podem ser encontrados em um cache, enquanto um erro de cache ocorre quando ele não pode.
- acessos ao cache são servidos por leitura de dados do cache, que é mais rápido do que recomputing um resultado ou ler a partir de um armazenamento de dados mais lenta; assim, os mais pedidos podem ser servidos a partir do cache, mais rápido o sistema executa.

- Spooling ou simplesmente Spool (do acrônimo Simultaneous Peripheral Operations On-line) refere-se a um processo de transferência de dados colocando-os em uma área de trabalho temporária onde outro programa pode acessá-lo para processá-lo em um tempo futuro.

- MULTIPLEXAÇÃO é dividir um determinado recurso entre várias fontes, por exemplo, processos e dispositivos de E/S ou o processador. Essa divisão é feita de forma que cada fonte faça uso do recurso de forma exclusiva por um período de tempo.

2. DEADLOCK

Deadlock é uma situação em que dois ou mais processos estão bloqueados, esperando uns pelos outros para liberar recursos que estão sendo mantidos. Para que um deadlock ocorra, quatro condições devem ser satisfeitas simultaneamente. Essas condições são conhecidas como as quatro condições de Coffman:

- Exclusão mútua: Refere-se à situação em que apenas um processo pode ter acesso a um recurso por vez. Quando um recurso é concedido a um processo, ele é mantido exclusivamente por esse processo até que seja liberado.

- Posse e espera: Ocorre quando um processo já está em posse de um recurso e solicita recursos adicionais, que estão sendo mantidos por outros processos. Nesse caso, o processo mantém os recursos que já possui e espera pelos recursos adicionais.

- Ausência de preempção: Significa que um recurso não pode ser retirado à força de um processo que o detém. Em outras palavras, um recurso só pode ser liberado voluntariamente pelo processo que o está utilizando, geralmente após concluir sua tarefa.

- Espera circular: Existe uma cadeia circular de processos, na qual cada processo da cadeia está esperando por um recurso que está sendo mantido pelo próximo processo na cadeia. Essa condição cria um ciclo de dependências entre os processos, fazendo com que todos fiquem bloqueados indefinidamente.

Se todas essas quatro condições ocorrerem ao mesmo tempo, um deadlock pode ser formado, levando ao bloqueio dos processos envolvidos. Para prevenir e gerenciar deadlocks, os sistemas operacionais implementam diferentes técnicas, como prevenção, detecção e recuperação de deadlock, além de evitar a ocorrência das condições de Coffman.

3. ESCALONAMENTO DE PROCESSOS

- Preempção: o ato do S.O. de utilizar as interrupções do relógio para retirar a CPU do processo em execução. Ou seja, o processo não pode monopolizar o processador!

- Quando o sistema não for preemptivo pode ocorrer uma situação denominada starvation (inanição): processo que nunca consegue chegar ao processador, fica eternamente aguardando, sempre tem alguém que "fura a fila".
    - alguns processos ou classes de processos permanecem sempre à espera do processador devido ao fato de que o escalonamento acaba sempre por privilegiar a escolha de outros processos.

- REGIÃO CRÍTICA é um conceito em programação concorrente onde dois ou mais processos acessam um recurso compartilhado, como uma variável ou estrutura de dados. Para evitar condições de corrida, onde o resultado final pode ser diferente dependendo da ordem em que os processos acessam o recurso, apenas um processo é permitido para entrar na região crítica de cada vez. Isso é conseguido através do uso de mecanismos de sincronização, como semáforos, mutexes ou monitores.
    - É importante notar que a região crítica não é uma situação em si, mas uma seção de código que precisa ser protegida para evitar problemas de concorrência.

3.1. Categorias de Algoritmos de Escalonamento:

- Lote: geralmente utilizado em computadores de grande porte, sem usuários esperando uma resposta rápida. São aceitáveis algoritmos não-preemptivos ou algoritmos preemptivos com longos períodos de tempo para cada processo. Isso reduz as trocas de processo, o que melhora o desempenho;

- Interativo: em um ambiente em que os usuários interagem a preempção é fundamental! Assim não ocorre uma monopolização da CPU por um processo, negando serviço a outros. Uso típico em computadores de propósito geral.

- Tempo real: por incrível que pareça a preempção pode ser desnecessária, pois os processos sabem que não podem ser executados por longos períodos de tempo. Normalmente os processos realizam suas atividades e são rapidamente bloqueados. Um sistema de tempo real executa apenas o que é necessário, ex.: radar que registra a velocidade do veículo e fotografa se ultrapassar um limite.

3.1.1. Escalonamento em Sistemas de Lote

- First-Come First-Served (FCFS): os processos recebem tempo de CPU na ordem em que solicitam. Basicamente há uma fila única e os processos vão "entrando" nela, os processos prontos são executados na ordem da fila e quando um processo é bloqueado (aguardando uma entrada, por exemplo), ele retorna para o fim da fila quando estiver pronto novamente. Ok, mas e se um processo começar a ser executado, nunca for bloqueado e tiver uma estimativa de execução de 4h, ele monopolizará o processador esse tempo todo? Isso mesmo! Esse algoritmo é do tipo não-preemptivo, uma característica de sistemas de lote.

- Shortest-Job First (SJF): mais um não-preemptivo, mas esse algoritmo presume que os tempos de execução são conhecidos previamente.

- Shortest Remaining Time Next (SRT): versão preemptiva do algoritmo SJF. Nesse algoritmo o escalonador sempre escolhe o job com tempo de execução mais curto (obviamente o tempo precisa ser conhecido previamente). Quando um novo job chega, seu tempo é comparado com o que falta para concluir o job corrente. Se o novo precisar de menos tempo, o processo corrente é suspenso e o novo é iniciado. Esse esquema permite que jobs novos curtos tenham uma boa prioridade.

3.1.2. Escalonamento em Sistemas Interativos

- Round-robin: é realizado um rodízio entre os processos, sendo que a cada processo é atribuído um intervalo de tempo (quantum), durante o qual ele pode ser executado. Se ao final do quantum o processo ainda estiver em execução é realizada a preempção da CPU e esta é alocada a um outro processo. Obviamente que se o processo tiver terminado antes do quantum ter expirado ou se tiver sido bloqueado, a troca da CPU é realizada neste momento. Um ponto interessante é a definição da duração do quantum, pois trocar de um processo para outro exige uma quantidade de tempo para salvar e carregar registradores e mapas de memória, atualizar tabelas e listas etc. Vamos supor que esse chaveamento de contexto demore 1ms e que o quantum seja de 9ms. Nesse caso 10% da CPU são desperdiçados em sobrecarga administrativa.

- Escalonamento por prioridade: cada processo recebe uma prioridade e o processo pronto, com a maior prioridade, tem a permissão para executar. Por exemplo, um daemon que envia um e-mail em segundo plano deveria ter uma prioridade mais baixa do que a de um processo responsável por uma videoconferência ao vivo. Para evitar que processos com alta prioridade monopolizem o uso da CPU, o escalonador pode diminuir a prioridade do processo em execução em cada interrupção de relógio. Se sua prioridade ficar abaixo da do próximo processo com maior prioridade, deve ocorrer um chaveamento de processo. Uma alternativa é ar um quantum a cada processo e, quando esse quantum esgotar, é dada a chance de execução ao próximo processo com maior prioridade.

- Escalonamento por múltiplas filas: basicamente os processos são agrupados em classes e cada classe possui uma prioridade: 1 quantum, 2 quanta (esse é o plural de quantum), 4, 8, e assim por diante. Na medida em que um processo utiliza todos os quanta destinados a ele, em seguida ele é movido para a classe seguinte (a que recebe mais quanta). Por exemplo, se um processo precisa de 50 quanta, primeiro ele recebe 1, depois 2, 4, 8, 16, 32. No total o processo teria recebido 63 quanta , ou seja, na última classe em que esteve (recebendo 32 quanta), na verdade só utilizou 19 e parou a execução, liberando o processador.

- Escalonamento garantido: a ideia é “fazer promessas realistas aos usuários sobre o desempenho, e cumpri-las!”. Como assim? Se houver N usuários trabalhando em uma CPU, cada um recebe cerca de 1/N do poder dessa CPU! Essa é uma promessa realista simples de cumprir. Para que essa promessa realmente seja cumprida, o sistema deve monitorar quanto da CPU cada processo de cada usuário recebeu e dar mais poder de CPU para quem teve menos. Assim ocorre uma compensação e aos poucos todos usuários terão um uso da CPU similar.

- Escalonamento por sorteio: o escalonamento garantido parece uma boa, mas é difícil implementar! Uma solução mais simples é o algoritmo de escalonamento por sorteio, o qual fornece “bilhetes de loteria” para os vários recursos do sistema (ex.: tempo de CPU). Quando uma decisão de escalonamento tiver que ser tomada, um “bilhete” é sorteado e o ganhador recebe o recurso. Os processos mais importantes podem receber “bilhetes” extras, aumentando as chances de serem sorteados.

- Shortest-job-first: A estratégia que associa, a cada processo, um valor baseado no tempo em que ele deverá ocupar a CPU e escolhe o de menor valor para a execução é denominada

- Semáforo mutex (ou simplesmente mutex, de "MUTual EXclusion") é um mecanismo de sincronização que permite implementar a exclusão mútua sem a deficiência da espera ocupada.

    - Um mutex é um tipo de semáforo usado para proteger recursos compartilhados. Quando um processo adquire um mutex (operação de bloqueio), outros processos que tentarem adquirir o mesmo mutex serão colocados em espera até que o processo original libere o mutex (operação de desbloqueio). Isso permite que os processos sincronizem seu acesso a recursos compartilhados e evitem condições de corrida.

    - Um mecanismo de sincronização simples, que permite implementar a exclusão mútua sem a deficiência da espera ocupada (busy wait)

3.2. A classificação de Flynn

é um sistema que categoriza arquiteturas de computadores com base no número de fluxos de instrução e dados que eles manipulam.

- SISD (Single Instruction, Single Data):
    - Esta é a arquitetura tradicional de um computador sequencial. Uma única instrução é aplicada a um único conjunto de dados por vez.

- SIMD (Single Instruction, Multiple Data):
    - Esta é a arquitetura usada nas GPUs, como mencionado na pergunta. Uma única instrução é aplicada a múltiplos conjuntos de dados ao mesmo tempo. Isso é útil para tarefas como processamento gráfico, onde a mesma operação precisa ser aplicada a muitos pixels ou vetores de uma vez.

- MISD (Multiple Instruction, Single Data):
    - Várias instruções são aplicadas a um único conjunto de dados. Esta categoria é rara e não é usada em muitos sistemas práticos.

- MIMD (Multiple Instruction, Multiple Data):
    - Múltiplas instruções são aplicadas a múltiplos conjuntos de dados. Esta arquitetura é usada em muitos sistemas de computação paralela e distribuída.

4. GERÊNCIA DE MEMÓRIA

- MMU (do inglês Memory Management Unit): Unidade de Gerenciamento de Memória ou é um dispositivo de hardware que traduz endereços virtuais em endereços físicos

- O processo de alocação e liberação das regiões da memória, dependendo da política escolhida, pode ocasionar situações em que pequenas regiões livres nos espaços entre regiões alocadas a outros processos se tornem difíceis de ser utilizadas, pois seu tamanho não comporta facilmente outros processos, configurando um fenômeno conhecido como FRAGMENTAÇÃO.

- MEMORY LEAK pode ocorrer em duas situações:

  1. Blocos de memória estão alocados e disponíveis para serem utilizados pelo programa, mas não são acessíveis porque a aplicação não tem nenhum ponteiro apontando para essa área de memória. Ou seja, tais blocos de memória não podem ser utilizados nem pelo programa nem por outros processos (ficam alocados e sem acesso!);

  2. Blocos de memória possuem dados que poderiam ser liberados por estarem inacessíveis e que, por algum "esquecimento", ainda são referenciados no código. Ou seja, mesmo sem estarem sendo usados, não podem ser liberados.

4.1. Técnicas de Alocação de Memória

- Alocação Contígua:

Nesta técnica, a memória é alocada em um bloco contíguo para um programa. Esta é uma abordagem simples, mas pode levar à fragmentação de memória.

- Alocação Encadeada (ou Ligada):
  - Cada bloco de memória contém um ponteiro para o próximo bloco de memória, formando uma cadeia de blocos. Esta técnica permite a alocação dinâmica de memória e ajuda a minimizar a fragmentação, mas o acesso é tipicamente sequencial.

- Alocação Indexada:
  - Em vez de ponteiros dentro dos blocos de memória, um bloco de índice contém ponteiros para cada bloco de memória. Isto permite acesso direto aos blocos de memória, mas pode haver desperdício de espaço se o bloco de índice for muito grande.

- Segmentação:
  - A segmentação é uma técnica que divide o espaço de endereço de um programa em partes ou segmentos que não precisam ser contíguos na memória. Cada segmento tem um propósito específico, como código, dados e pilha. A segmentação ajuda a melhorar a eficiência do uso da memória e a proteção de memória, pois cada segmento pode ter diferentes permissões de acesso.

- Paginação:
  - Divide a memória em blocos de tamanho fixo chamados páginas. A paginação minimiza a fragmentação e simplifica o gerenciamento de memória.

4.2. Lidando com pouca Memória RAM

Os sistemas operacionais utilizam várias técnicas para gerenciar a memória e garantir que os programas possam ser executados de forma eficiente, mesmo quando a memória principal (RAM) é limitada. Aqui estão algumas dessas técnicas:

- Swapping (Paginação de Memória Virtual):
  - swapping é uma técnica em que os processos são movidos entre a memória principal e a memória secundária (geralmente o disco rígido) para liberar espaço na memória principal. O espaço de memória secundária utilizado para o swapping é frequentemente chamado de memória virtual. Esta técnica pode ajudar a aumentar a quantidade total de memória disponível para os processos, mas a um custo de desempenho, já que o acesso à memória secundária é muito mais lento do que o acesso à memória principal.

- Paginação e Segmentação:
  - Estas duas técnicas dividem o espaço de memória do programa em unidades menores. Na paginação, a memória é dividida em páginas de tamanho fixo, enquanto na segmentação a memória é dividida em segmentos de tamanho variável. Estas técnicas permitem que diferentes partes do programa sejam carregadas na memória conforme necessário, economizando espaço e permitindo que vários programas compartilhem a memória de forma mais eficiente.

- Limpeza de Memória:
  - Alguns sistemas operacionais implementam algoritmos de limpeza de memória que buscam ativamente por partes da memória que não estão sendo usadas e as liberam para reutilização. Isto pode envolver a busca por páginas de memória que não foram acessadas recentemente, ou a busca por processos que terminaram, mas ainda não liberaram sua memória.

- Compactingação de Memória:
  - Esta técnica é usada para combater a fragmentação da memória. Quando os programas são carregados e descarregados da memória, eles podem deixar espaços pequenos e inutilizados. Ao longo do tempo, isso pode levar à fragmentação, onde há espaço livre suficiente na memória para um programa, mas não em um único bloco contíguo. A compactação de memória envolve mover os processos na memória para eliminar esses espaços e criar um bloco contíguo de espaço livre.

- Sobreposição e Multitarefa:
  - Com a sobreposição e a multitarefa, o sistema operacional pode permitir que outros processos usem a CPU enquanto um processo está esperando por memória. Isso pode dar a impressão de que vários processos estão sendo executados simultaneamente, melhorando o desempenho geral e a eficiência.

As técnicas específicas utilizadas dependem do sistema operacional e das características do hardware, bem como das necessidades específicas do ambiente (por exemplo, servidores versus sistemas de desktop, sistemas em tempo real versus sistemas de uso geral).

4.3. Técnicas de Gerenciamento de Memória

- Fragmentação externa: A fragmentação externa ocorre quando o espaço livre de memória está dividido em blocos não contíguos, dificultando a alocação de um processo que necessita de um espaço maior do que os blocos individuais disponíveis. Isso geralmente acontece em sistemas que utilizam alocação contígua de memória, onde os processos são alocados lado a lado, e os espaços livres se fragmentam à medida que os processos são alocados e liberados.

- Fragmentação interna: A fragmentação interna ocorre quando o espaço alocado para um processo excede o espaço realmente necessário, deixando espaço não utilizado dentro do espaço alocado. Isso é mais comum em sistemas que utilizam alocação de memória em blocos ou páginas de tamanho fixo, onde o tamanho do processo pode não se ajustar exatamente ao tamanho dos blocos ou páginas.

Técnica	Tipo de Sistema	Utilização de Disco	Tipo de Memória	Exemplos de Sistemas Operacionais
Overlay	Monoprogramado	Sim	Real	DOS, sistemas embarcados
Alocação Contínua Simples	Monoprogramado	Não	Real	Sistemas operacionais mais antigos
Alocação Particionada Estática	Monoprogramado	Não	Real	Sistemas operacionais mais antigos
Memória Virtual	Multiprogramado	Sim	Virtual	Windows 11, Linux, macOS, HP-UX, etc.
Swapping	Multiprogramado	Sim	Real/Virtual	Windows 11, Linux, macOS, HP-UX, etc.
Paginação	Multiprogramado	Sim	Real/Virtual	Windows 11, Linux, macOS, HP-UX, etc.

4.3.1. Técnica de overlay: (MONOPROGRAMADO/FIXO/no DISC)

- A técnica de overlay é um método de gerenciamento de memória que permite a execução de programas maiores do que a memória disponível para sistemas monoprogramados. O programa é dividido em módulos menores chamados overlays, que são carregados e descarregados na memória conforme necessário. Isso possibilita a execução de apenas uma parte do programa de cada vez, economizando espaço de memória.

- Gera fragmentação interna, pois cada módulo pode não ocupar completamente o espaço de memória reservado a ele. Não gera fragmentação externa, pois apenas um módulo é carregado na memória de cada vez.

4.3.2. Alocação contínua simples: (MONOPROGRAMADO/VARIÁVEL/DISCO)

- A alocação contínua simples é uma técnica de gerenciamento de memória utilizada em sistemas monoprogramados, onde todo o espaço de memória é alocado para um único processo. Como apenas um processo pode ser executado por vez, não há necessidade de gerenciar a memória entre múltiplos processos. Esse método é simples, mas limita a utilização do sistema a apenas um processo por vez.

- Gera fragmentação externa, pois os processos são alocados de forma contígua na memória, e a liberação de processos pode deixar espaços livres não contíguos. Não gera fragmentação interna, pois o espaço de memória é alocado exatamente de acordo com o tamanho do processo.

4.3.3. Alocação particionada estática (MONOPROGRAMADO/FIXO/no DISC):

- A alocação particionada estática é outra técnica de gerenciamento de memória utilizada em sistemas monoprogramados. Neste caso, a memória é dividida em partições de tamanho fixo que são alocadas para os processos. Essa técnica permite uma melhor organização e alocação de memória, mas ainda limita a execução a um único processo por vez.

- Gera fragmentação interna, pois as partições de tamanho fixo podem não se ajustar exatamente ao tamanho dos processos. Pode gerar fragmentação externa se as partições não forem preenchidas completamente e não puderem ser realocadas.

4.3.4. Técnica de memória virtual: (MULTIPROGRAMADO/FIXO/no DISC)

- A técnica de memória virtual é uma abordagem avançada de gerenciamento de memória que combina a memória principal e secundária (disco) para permitir a execução de programas maiores do que a memória disponível. O espaço de endereçamento virtual é dividido em unidades chamadas páginas, que são mapeadas para a memória física conforme necessário. Isso permite a execução de múltiplos processos simultaneamente, melhorando o desempenho e a utilização do sistema.

- Gera fragmentação interna quando combinada com a paginação, pois a última página de um processo pode não estar completamente preenchida. Elimina a fragmentação externa, pois as páginas podem ser alocadas de forma não contígua.

4.3.5. Técnica de swapping: MULTIPROGRAMADO/VARIÁVEL/no DISC

- A técnica de swapping é um método de gerenciamento de memória em sistemas multiprogramáveis, onde processos são trocados entre a memória principal e a secundária (disco). Quando um processo está bloqueado ou aguardando por algum recurso, ele pode ser movido para a memória secundária, liberando espaço na memória principal para outros processos. Isso melhora a eficiência e o desempenho do sistema, permitindo a execução simultânea de múltiplos processos.

- Gera fragmentação externa, pois os processos são alocados e liberados na memória, criando espaços livres não contíguos. Pode gerar fragmentação interna se combinado com a paginação ou outras técnicas de alocação de tamanho fixo.

4.3.6. Paginação:

- É uma forma de alocação não contígua de memória para o espaço de endereçamento de um processo.

- A paginação é uma técnica de gerenciamento de memória que divide o espaço de endereçamento de um processo em unidades de tamanho fixo chamadas páginas. Essas páginas são alocadas na memória de forma não contígua, o que facilita o gerenciamento de memória e elimina a fragmentação externa. A paginação permite a execução de múltiplos processos simultaneamente e é usada tanto em sistemas que empregam memória real quanto em sistemas que empregam memória virtual.

- A diferença entre a paginação e a segmentação é que, PAGINAÇÃO divide o programa em partes de tamanho fixo, sem qualquer ligação com a estrutura do programa, já SEGMENTAÇÃO permite uma relação entre a lógica do programa e sua divisão na memória.

- Paginação: técnica utilizada pela maioria dos sistemas com memória virtual. Em qualquer computador existe um determinado conjunto de endereços de memória que programas podem referenciar. Trata-se de endereços virtuais que formam o espaço virtual de endereçamento do processo. Em computadores sem memória virtual, o endereço virtual é colocado diretamente no barramento de memória uma palavra da memória física com mesmo endereço é lida ou escrita.

- A paginação normalmente gera fragmentação interna na última página, pois raramente um processo tem tamanho múltiplo da página! Um processo pode ter inúmeras páginas (depende o tamanho do processo e da página!). É utilizado em sistemas que utilizam memória virtual.

- Elimina a fragmentação externa, pois as páginas podem ser alocadas de forma não contígua.

4.3.7. Algoritmo de Substituição de Páginas

A substituição de páginas refere-se ao processo pelo qual um sistema operacional decide qual página armazenada em memória física deve ser substituída por outra página que está sendo trazida da memória secundária (disco rígido, por exemplo). Isso ocorre quando a memória física está cheia e é necessário liberar espaço para trazer páginas adicionais para a memória.

Portanto, enquanto a substituição de páginas trata da escolha de qual página substituir na memória física, o algoritmo de paginação trata da estratégia geral para mapear páginas virtuais em páginas físicas durante o gerenciamento de memória virtual.

- Menos Recentemente Usada (LRU - Least Recently Used):
    - Este algoritmo de substituição de páginas remove a página que não foi usada há mais tempo. Ele tenta explorar a localidade temporal de referência, ou seja, se uma página foi referenciada recentemente, é provável que seja referenciada novamente em um futuro próximo.

- Primeira a Entrar, Primeira a Sair (FIFO - First In, First Out):
    - Este algoritmo de substituição de páginas remove a página que foi carregada há mais tempo na memória, ou seja, a primeira que entrou. Este algoritmo é fácil de implementar, mas não considera se a página está sendo usada ou não.

- Segunda Chance (ou Algoritmo do Relógio):
    - Este é uma variante do algoritmo FIFO que dá uma "segunda chance" às páginas antes de substituí-las. Quando uma página é considerada para substituição, se a bit de referência da página estiver definida (indicando que a página foi recentemente acessada), a bit de referência é limpa e a página recebe uma "segunda chance". Se a bit de referência já estiver limpa, a página é substituída.

- Não Usada Recentemente (NUR - Not Recently Used):
    - Este algoritmo mantém um registro de duas informações sobre cada página: se a página foi referenciada recentemente e se a página foi modificada recentemente. As páginas são então priorizadas para substituição se não foram referenciadas ou modificadas recentemente.

- Ótimo (OPT ou MIN):
    - Este algoritmo substitui a página que não será usada pelo maior período de tempo no futuro. Embora forneça o menor número possível de faltas de página, é impraticável para a maioria dos sistemas operacionais, pois requer conhecimento prévio do padrão de acesso à memória.
- Relógio:
    - utiliza uma lista circular com um ponteiro apontando para a página mais antiga. Imagina-se funcionar como um relógio, onde o ponteiro vai variando de acordo com a substituição da página antiga para uma “nova” página antiga, e assim sucessivamente.

4.3.8. Dirty Bit

Na gerência de memória dos sistemas operacionais, o "Dirty Bit" é uma parte importante do mecanismo de controle que tem relação direta com a eficiência e a segurança do sistema.

O "Dirty Bit" é uma bandeira (ou flag), ou seja, um valor booleano que pode ser true (verdadeiro) ou false (falso), associado a cada página na memória. Este bit é usado para indicar se a página foi modificada (escrita) após ter sido carregada na memória.

Quando um processo altera uma página na memória, o sistema operacional configura o "Dirty Bit" dessa página para true, indicando que a página foi 'sujada' ou modificada. Por outro lado, se a página apenas foi lida e não modificada desde que foi carregada na memória, o "Dirty Bit" permanecerá como false, indicando que a página está 'limpa'.

O "Dirty Bit" é essencial no gerenciamento eficiente da memória e do disco, particularmente na estratégia de escrita de páginas de volta no disco (write-back). Se uma página precisa ser removida da memória (por exemplo, devido à necessidade de espaço para outras páginas), o sistema operacional verifica o "Dirty Bit". Se o bit estiver configurado como true, a página precisa ser escrita de volta no disco porque contém alterações que não foram salvas. Se o bit estiver como false, a página pode ser descartada sem a necessidade de escrita no disco, economizando assim operações de E/S de disco, que são relativamente caras em termos de tempo e recursos do sistema.

O "Dirty Bit" também tem um papel importante na segurança e na prevenção de erros, pois assegura que as alterações feitas na memória não sejam perdidas antes de serem salvas no disco.

5. GERÊNCIA DE ENTRADA/SAÍDA

DEADLOCK:

As quatro condições que devem ser verdadeiras para que ocorra um deadlock são:

- Condição de EXCLUSÃO MÚTUA: cada recurso ou está correntemente atribuído a exatamente um processo ou está disponível;
- Condição de POSSE E ESPERA: os processos que possuem recursos garantidos anteriormente podem solicitar novos recursos (um acumulador de recursos!);
- AUSÊNCIA DE PREEMPÇÃO: os recursos garantidos não podem ser retirados à força de um processo;
- Condição de ESPERA CIRCULAR: um encadeamento circular de dois ou mais processos, cada um esperando por um recurso mantido pelo próximo do encadeamento:

Algoritmo do avestruz: estratégia mais simples! O que um avestruz faz? Coloca a cabeça em um buraco... pois é, é isso mesmo! O algoritmo finge que nada aconteceu, simplesmente ignora.

- é a estratégia adotada pela maioria dos sistemas operacionais (Windows, Linux, entre outros). Por quê? Porque o preço seria alto em realizar muitas restrições ao usuário. Melhor deixar acontecer...e se for o caso, o usuário reinicia a máquina ou mata um processo, caso ocorra algum travamento.

- *MP3* (*MPEG Layer 3*) é um formato de compressão de áudio digital que *minimiza a perda de qualidade em músicas ou outros arquivos de áudio* reproduzidos no computador ou em dispositivo próprio. Para reproduzir arquivos neste formato, é necessário usar um aplicativo reprodutor de áudio, capaz de decodificar tal formato, como o *Media Player do Windows, iTunes do MacOS* e vários outros.

- o formato *jpg*, que é a mesma coisa que jpeg (*Joint Photographic Experts Group) é* usado para imagens com esquema de cores de 24 bits, o que permite trabalhar com até 16,8 milhões de cores. Este formato gera bem menores do que outros formatos de imagem, já que utiliza a compressão de imagens, que consiste na eliminação de dados redundantes dos arquivos. Quanto *maior o nível da compressão, menor é o tamanho do arquivo, e pior será a qualidade* da imagem.
  - *O nível de compressão é determinado por programas de tratamento de imagens*, e não se pode afirmar a percentagem da redução, que varia de arquivo para arquivo.

5.1. Implementação do armazenamento dos arquivos

1) ALOCAÇÃO CONTÍGUA (sequencial): armazena cada arquivo como uma sequência contígua de blocos. Se uma partição foi formatada com blocos de tamanho 4KB, um arquivo de 40KB deve alocar 10 blocos consecutivos. Como vantagens temos a simplicidade na implementação (basta saber o bloco de início e o de fim do arquivo), e o desempenho na leitura (o arquivo inteiro pode ser lido em uma única operação).

- E qual a desvantagem? Com o tempo, a partição se torna fragmentada. Imagine no exemplo anterior (arquivo de 40KB), que após gravado esse arquivo, outros tantos foram gravados na sequência. Depois esse arquivo foi excluído, deixando 10 blocos livres entre 2 arquivos quaisquer. Digamos que os arquivos a serem gravados posteriormente possuam sempre mais de 40KB, então essa lacuna permanecerá sem uso. Uma solução é utilizar algum algoritmo de compactação, para "empurrar" os arquivos para a "esquerda" ou "direita" para extinguir lacunas criadas com o tempo, mas essa operação é muito custosa!

2) ALOCAÇÃO ENCADEADA (aleatória): cada arquivo é mantido como uma lista encadeada de blocos de disco. Dessa forma todos os blocos podem ser utilizados! Além disso, é suficiente que a entrada de diretórios armazene apenas o endereço de disco do primeiro bloco e o restante pode ser encontrado a partir dele.

- O problema é que o acesso aleatório é mais lento que o sequencial. Outro problema é que o volume de armazenamento em um bloco não é mais uma potência de 2 (ex.: 512 bytes, 1KB, 2KB etc.), pois o ponteiro ocupa algum espaço. Isso exige mais operações como adquirir e concatenar informações de blocos do disco, o que gera uma maior sobrecarga.

- Os blocos de dados são encadeados entre si, justamente para que se torne mais eficiente o acesso. Ao invés de buscar cada bloco de dados em um acesso, o que seria mais lento, busca-se um bloco de referência, que possui um encadeamento para o próximo bloco e assim sucessivamente.

3) ALOCAÇÃO ENCADEADA COM TABELA NA MEMÓRIA (aleatória): no lugar de utilizar um ponteiro em cada bloco para apontar para o próximo, é utilizada uma tabela. Assim os blocos possuem um tamanho múltiplo de 2, o que facilita a vida do sistema de arquivos. Para definir o fim do arquivo deve-se utilizar um marcador especial, como por exemplo "-1". Essa tabela é chamada de FAT (File Alocation Table).

- A principal desvantagem é que a tabela inteira deve estar em memória, o que pode demandar de boa quantidade de espaço. Vamos a um exemplo: disco de 500GB, com tamanho de bloco 1KB (coloquei pequeno assim para facilitar o cálculo), a tabela precisa de 500 milhões de entradas (uma para cada bloco)! Cada entrada deve ter no mínimo 3 bytes para manter os endereços dos blocos, mas para facilitar a pesquisa as entradas possuem 4 bytes. Vamos ao cálculo: 500 milhões x 4 bytes = 2 bilhões (2 GB). Se o bloco tiver 4 KB, seria uma tabela de 500 MB. O MS-DOS e o Windows 95/98 utilizam apenas sistemas de arquivos FAT e as versões superiores do Windows também suportam.

4) I-NODES (aleatória): cada arquivo é associado a uma estrutura de dados denominada i-node (nó-índice), a qual lista os atributos e endereços de blocos do disco.

- A principal vantagem é que o i-node só precisa estar na memória quando o arquivo correspondente for aberto (não há uma tabela ou outra estrutura com todos os blocos na memória). O problema dos i-nodes é que, se cada um tiver um espaço para uma quantidade fixa de endereços de disco, o que fazer quando o tamanho de um arquivo passar desse limite? Uma solução é reservar o último endereço para um endereço de um bloco indireto simples (ao invés de um bloco de dados). Esse bloco indireto simples contém mais endereços de bloco de disco (como se fosse uma extensão). Essa ideia pode ir além, com o uso de blocos indiretos duplos e triplos, como vemos na figura:

- O ext2 (Second Extended File System) é um dos sistemas de arquivos suportados pelo sistema operacional Linux. Nesse sistema de arquivos, com exceção do nome do arquivo, todos os demais atributos, como permissões de acesso, identificação do dono do arquivo, identificação do grupo do arquivo, tamanho do arquivo, entre outros, são armazenados numa estrutura conhecida como I-NODES.

5.2. Sistemas de Arquivos

Sistema de Arquivos é a estrutura que indica como os arquivos devem ser *gravados e lidos* pelo sistema operacional do computador. É o sistema de arquivos que determina como as informações podem ser *guardadas, acessadas, copiadas, alteradas, nomeadas e até apagadas*. Ou seja, toda e qualquer manipulação de dados em um dispositivo de armazenamento *necessita* de um sistema de arquivos para que estas ações sejam possíveis. Sem um sistema de arquivos, os dados armazenados seriam apenas um conjunto de bits sem utilidade.

- EXT4:

    - EXT4 é nativo de distribuições Linux, baseado no sistema de arquivos ext3 e tem inúmeras melhorias. Entre elas se encontra o suporte para sistema de arquivos maiores e alocação de espaço de disco de arquivos maiores, mais rápido e mais eficiente, sem limite no número de subdiretórios dentro de um diretório, verificação de sistema de arquivos mais rápida e um agendamento mais robusto. Não é suportado pelo Windows.

        - Com o objetivo de evitar cópias desnecessárias dos blocos de dados de arquivos, os sistemas de arquivos EXT2, EXT3 e EXT4 do Linux permitem a criação de referências para arquivos através de hard links e symbolic links.

            - Hard links: De fato, um hard link refere-se diretamente ao inode do arquivo, o que significa que ele é efetivamente apenas outro nome para o arquivo. Um hard link não aloca um novo i-node. Todas as entradas de diretório que são hard links para o mesmo arquivo compartilham o mesmo i-node.

            - Symbolic links: Ao contrário dos hard links, os symbolic links não compartilham i-nodes. Em vez disso, eles são um arquivo separado que contém um caminho para outro arquivo. Este "caminho" é a referência ao arquivo original. Portanto, eles têm seu próprio i-node.

- FAT16

    - FAT: File Allocation Table (traduzindo: Tabela de Alocação de Arquivos). A primeira versão do FAT surgiu em 1977, para trabalhar com o sistema operacional MS-DOS, mas foi padrão até o Windows 95. Com o surgimento de dispositivos de armazenamento mais sofisticados e com maior capacidade, o sistema FAT foi ganhando revisões, identificadas pelos nomes

FAT12 e FAT16, sendo o primeiro quase um desconhecido e o último padrão dos sistemas operacionais da Microsoft por muito tempo.

- As versões surgem com o intuito de eliminar determinadas limitações do sistema de arquivos anterior. A Microsoft lançou, em 1996, o FAT32, que se tornou o sistema de arquivos do Windows 95 (versão OSR 2) e do Windows 98, sendo também compatível com versões lançadas posteriormente, como Windows 2000 e Windows XP, embora estes tenham um sistema de arquivos mais avançado, o NTFS.

- FAT32

  - o FAT32 se tornou o sistema de arquivos do Windows 95 (versão OSR 2) e do Windows 98, sendo também compatível com versões lançadas posteriormente, como Windows 2000 e Windows XP, mas não é o sistema de arquivos nativo do Windows 10, embora seja suportado por esta versão do Windows. Além disto, o tamanho máximo de arquivo neste sistema é de 4 GB.

  - o único sistema de arquivos suportado pelo SUSE SLES 15 SP2 que também é suportado pelo Windows 10 é o FAT32.

- NTFS (New Technology File System):

  - NTFS é um sistema de arquivos que surgiu com o lançamento do Windows NT. Sua confiabilidade e desempenho fizeram com que fosse adotado nos sistemas operacionais posteriores da Microsoft, como Windows XP, Windows Vista, Windows 7 e Windows 8.1 e é o sistema de arquivos padrão do Windows 10, entres outros. Uma característica marcante do NTFS é o seu esquema de permissões de acesso.

  - O Unix sempre foi considerado um sistema operacional seguro por trabalhar com o princípio de que todos os arquivos precisam ter variados níveis de permissões de uso para os usuários. O NTFS também é capaz de permitir que o usuário defina quem pode e como acessar pastas ou arquivos. Teoricamente, o tamanho de arquivo no NTFS pode chegar a 16 Exabytes.

5.3. Journaling

Trata-se do registro das atividades que o sistema de arquivos irá fazer antes que efetivamente o faça. Mas para que isso? Simples...se o sistema falhar antes da execução do trabalho planejado, é possível, após a reinicialização do sistema, recorrer ao log para descobrir o que estava acontecendo no momento da parada e retomar o trabalho. Ou seja, o journaling não evita problemas, mas tenta recuperar caso algo errado ocorra!

Dos sistemas de arquivos mais conhecidos é importante sabermos quais deles possuem ou não a implementação de journaling:

- FAT12, FAT16, FAT32: NÃO possuem journaling;
- NTFS: possui journaling;
- EXT2: NÃO possui journaling;
- EXT3, EXT4: possuem journaling.

5.4. Tipos de Acesso

- SEQUENCIAL:
  - inicialmente os sistemas operacionais armazenavam dados em fitas magnéticas e, neste caso, o acesso era restrito à leitura dos registros na ordem em que eram gravados e a gravação de novos registros só era possível no final do arquivo. Tal tipo de acesso é denominado acesso sequencial.
  - quando se usa um arquivo de acesso sequencial, é possível voltar ao início do arquivo no momento em que se deseje, de forma que o arquivo pode ser lido quantas vezes foram necessários. Não é necessário ler todos os registros para voltar ao início.
- ALEATÓRIO (ou DIRETO):
  - Nos computadores mais modernos (aos quais se refere a alternativa), existem dispositivos de armazenamento que permitem métodos de acesso muito mais eficientes, como ocorre com os discos magnéticos, como o HD, por exemplo, no qual se usa o acesso direto que permite a gravação de um registro diretamente na sua posição.
  - arquivos de acesso aleatório são muito eficientes para localizar um registro específico, já que permitem ler registros de um arquivo fora de ordem ou acessar registros pela chave, *sem precisar ler pela posição*, de forma tal que os registros podem ser lidos em qualquer ordem.
  - em arquivos de acesso aleatório é possível mudar a posição de leitura e escrita, já que em arquivos deste tipo, o conteúdo pode ser lido e gravado em qualquer ordem.
- ACESSO INDEXADO:
  - Esse é um caso especial de acesso direto em que um índice é usado para localizar os dados diretamente. O índice é como uma tabela de conteúdos que aponta para a localização dos dados.

5.5. Estratégia de Entrada e Saída

Programada: é uma opção de E/S em que há uma troca de dados "progressiva" entre o dispositivo responsável pela E/S e a CPU, que toma o controle do processo totalmente, incluindo as informações de status da transferência além da própria emissão de comandos de E/S. Nessa opção, os dados são copiados para o kernel do sistema operacional e a cada envio de instrução, a CPU verifica se o dispositivo está pronto para mais ou se já está terminado.

Assíncrona: é aquela operação de E/S onde o processo não fica bloqueado até que se termine toda a operação, ou seja, é possível alterar a operação enquanto o processo é executado.

Orientada à interrupção: funciona mais ou menos ao contrário da programada, onde ao invés da CPU realizar a verificação, ela espera que o dispositivo informe que ele está pronto para a próxima instrução. Esse informe é realizado por meio de uma interrupção.

Orientada a blocos de memória: também chamado de DMA, é um sistema que permite que os dispositivos de E/S acessem a memória diretamente, sem a intermediação da CPU, o que representa um ganho de desempenho direto.

De acesso indireto à memória: é basicamente o oposto do método DMA. Porém, essa definição não é adequada, porque apenas dizer que se trata de acesso indireto não significa nada. Existem vários métodos de várias abordagens que são de acesso indireto.

5.1. Sobrevivência de Dados Digitais

Os dados digitais precisam de intervenção humana para sobreviverem. Pois o maior problema na preservação de dados digitais é a rápida mudança tecnológica. Algumas técnicas são utilizadas para garantir a sobrevivência de dados digitais:

- refrescamento (transferência dos arquivos de um suporte para outro, sem alteração do conteúdo). Exemplo: transferir dados salvos em um disquete para um pendrive.
- migração (migração dos dados para os formatos suportados pelos novos softwares).
- emulação (ambientes operacionais que simulam o comportamento daquelas que se tornaram obsoletas).

6. RAID (REDUNDANT ARRAY OF INDEPENDENT DISKS)

O RAID permite a utilização de dispositivos comuns de armazenamento combinados de tal forma que a disponibilidade e/ou desempenho sejam aumentados. A controladora utiliza dois ou mais dispositivos e o computador "enxerga" como se fosse apenas um volume. Ex.: três discos em RAID em um sistema Windows e o usuário "enxerga" como volume "E:".

6.1. RAID-0 (Striping)

No RAID-0 os dados são divididos em blocos e escritos sequencialmente em cada um dos dispositivos do conjunto. A leitura e escrita ocorre simultaneamente em todos os dispositivos, apresentando melhor desempenho quando comparado a discos individuais. O problema é que, se um dos dispositivos de armazenamento do conjunto falhar, todos os dados são perdidos, pois não há redundância. Note que não há redundância no RAID-0, embora o "R" seja de "Redundant"!

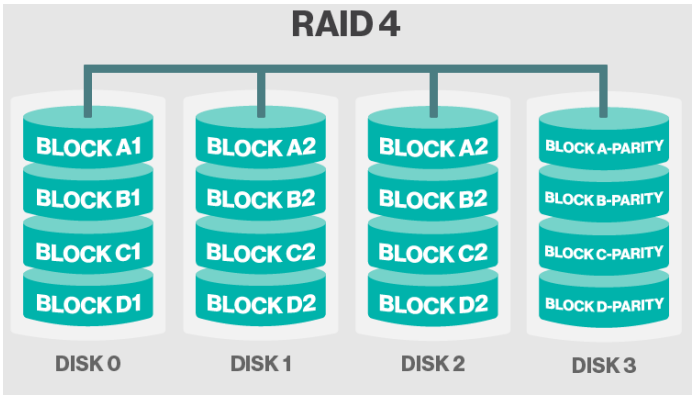
- A função de trocar aplicações e recursos de armazenamento de dados de um sistema que falhou para um sistema alternativo do cluster é denominada FAILOVER.
- Diante de uma situação de desastre com a perda de um sistema de arquivos, o administrador do sistema verifica que possui um backup completo (integral) e seis backups incrementais que foram executados consecutivamente, em intervalos de 24 horas. Para restaurar o sistema de arquivos, o administrador deve restaurar o backup completo e todos os seis backups incrementais.

6.2. RAID-1 (Mirror)

No RAID-1 começa de fato a acontecer a redundância. São necessários no mínimo dois discos (sempre em número par), onde todos os dados são gravados em todos os discos. No caso de um dos dispositivos de armazenamento falhar ou for removido, os dados presentes no outro dispositivo serão mantidos, apresentando maior segurança (redundância) do que apenas um dispositivo. O custo é o dobro do RAID-0, a gravação é feita aos pares e a leitura ocorre de forma paralela em todas as unidades do conjunto.

6.3. RAID-4

- No RAID-4 são necessários três discos no mínimo, sendo que um (fixo) será utilizado como paridade.
- O problema é que a escrita no disco onde são armazenados os dados de paridade é maior que nos outros discos, pois cada alteração nos discos acarreta também uma no disco de paridade.

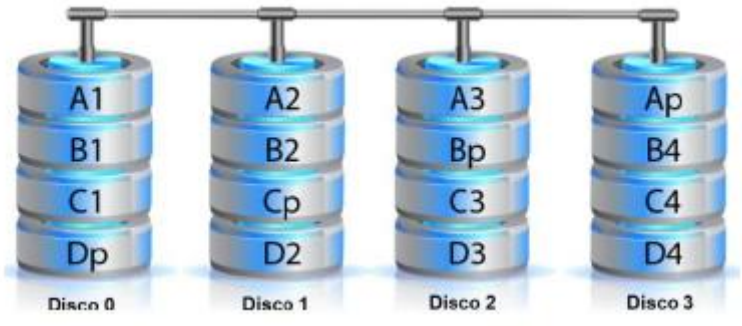


Vamos ver um exemplo: se forem utilizados 3 discos de 1 TB, um deles será de paridade, então o sistema "enxergará" apenas 2 TB! E se forem 4 discos de 1 TB? Aí teremos 3 TB disponíveis. No primeiro caso, a parte disponível seria 66,6% e no segundo seria 75%.

- Importante ressaltar que somente um disco é utilizado para paridade, mesmo que haja um esquema RAID-4 com 10 discos! E, mais uma vez, é importante lembrar que o disco de paridade é fixo, ou seja, é sempre o mesmo, diferente do que veremos no RAID-5, a seguir.

6.4. RAID-5

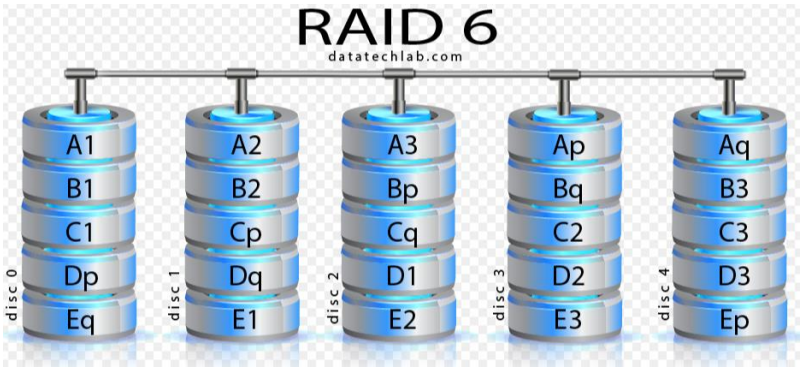
- O RAID-5 é uma evolução do RAID-4, pois a paridade é distribuída entre os discos.
- São necessários, no mínimo três discos, e o sistema continua a funcionar mesmo que um dos discos do conjunto falhe, pois os dados estão nos demais discos.
- Claro que há um tempo de espera para que um novo disco colocado no lugar do que falhou possa reconstruir os dados. Essa reconstrução é possível através dos outros discos.
- O RAID-5 tolera a falha de apenas um disco no sistema, assim como ocorre no seu antecessor, o RAID-4.



- Não há uma sobrecarga em um único disco de paridade, como ocorre no RAID-4.
- No caso apresentado como na figura, o aproveitamento para os dados é de 75%, pois o equivalente a 3/4 dos discos é utilizado para dados e o equivalente a 1/4 para a paridade. Se cada disco possui a capacidade de 2 TB, por exemplo, então 6 TB seriam destinados para o armazenamento de dados e 2 TB para a gravação dos cálculos de paridade.
- RAID 5: tamanho de armazenagem = n° de discos totais - 1 disco (paridade)

6.5. RAID-6

- usa dois cálculos de paridade diferentes, armazenando os resultados em blocos separados em discos distintos
- O RAID-6 é uma evolução do RAID-5, utilizando dupla paridade (o equivalente a 2 discos utilizados para isso).
- Dessa forma, permite que até dois discos falhem no mesmo conjunto, ao mesmo tempo, sem que haja a perda de dados.
- Obviamente que possui um menor aproveitamento para os dados quando comparado aos arranjos de paridade simples (RAID-4 ou RAID-5). Por exemplo, vamos comparar os arranjos RAID-5 e RAID-6, ambos com 4 discos de 1TB cada:
  - RAID-5: 4 HDs de 1TB, "1 disco" de paridade (1 TB), armazenamento = 3TB (75%).
  - RAID-6: 4 HDs de 1TB, "2 discos" de paridade (2 TB), armazenamento = 2TB (50%).

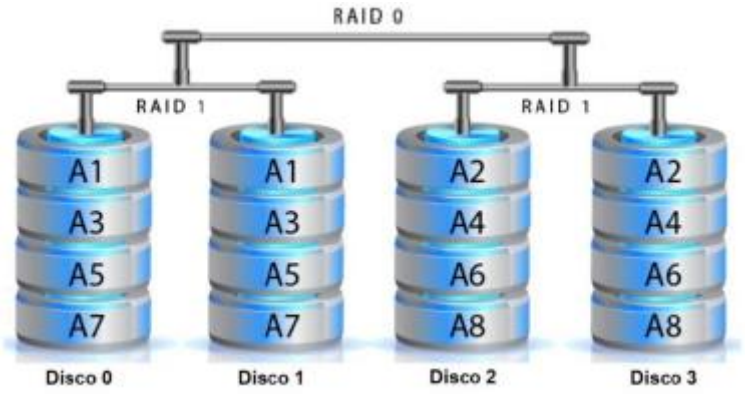


Os discos de paridade estão entre aspas porque na verdade não são discos fixos (como ocorre no RAID-4), mas o equivalente a 1 disco (RAID-5) ou a 2 discos (RAID-6), espalhados entre todos os discos do arranjo.
RAID 6: tamanho de armazenagem = nº de discos totais - 2 disco (paridade)

6.6. RAID-10

O RAID-10 é uma combinação do RAID-1 (parte interna) com RAID-0 (parte externa).

Veja na figura que se um arquivo tem os blocos A1 e A2, primeiro é aplicada a parte externa (RAID-0), então A1 é enviado para o lado esquerdo e A2 para o direito. Em cada um dos lados (parte interna) é aplicado o RAID-1, ou seja, A1 é escrito em ambos os discos (discos 0 e 1) e A2 é escrito em ambos os discos (discos 2 e 3).



Quantos discos são necessários no mínimo? Sabemos que RAID-0 precisa no mínimo de 2 e o RAID-1 também, então multiplicamos 2 por 2 e temos 4 como resposta! É extremamente importante saber montar (desenhar) na prova e a dica é: a parte interna (RAID-10) indica o RAID interno (RAID-1) e a externa (RAID-10) indica o RAID externo (RAID-0). Assim não tem erro! Da mesma forma acontece com outras combinações, como o RAID-50, RAID-100, entre outros.

Ou seja, basta saber como funcionam os arranjos RAID-0, RAID-1 e RAID-5, para que as combinações fiquem fáceis de serem montadas para responder questões em sua prova!

7. ARQUITETURA DE SOFTWARE

ACOMPLAMENTO: DEPENDÊNCIA DE COMPONENTES

- trata do nível de dependência entre módulos ou componentes de um software.
- O acoplamento está ligado ao grau de conexão entre módulos em uma estrutura de software.

COESÃO: DIVISÃO DE RESPONSABILIDADES

- trata do nível de responsabilidade de um módulo em relação a outros.
- 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, isto é, a força funcional relativa de um módulo ou componente de software.

Uma boa arquitetura de software deve ter componentes de projeto com baixo acoplamento e alta coesão.

- BAIXO ACOPLAMENTO: se os módulos pouco dependem um do outro, modificações de um não afetam os outros, além de não prejudicar o reuso.
- ALTA COESÃO: os módulos têm responsabilidades claramente definidas, eles serão altamente reusáveis, independentes e simples de entender.

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.

7.1. Arquitetura em Camadas

a) UMA CAMADA (monolítica): 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.

b) DUAS CAMADAS: Arquitetura Cliente-Servidor.

- Sistemas Cliente-Servidor em duas camadas foram dominantes durante aproximadamente toda a década de noventa e são utilizados até hoje.

VANTAGENS DE CLIENTES MAGROS	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.
DESVANTAGENS DE CLIENTES MAGROS	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; etc.

VANTAGENS DE CLIENTES GORDOS	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.
DESVANTAGENS DE CLIENTES GORDOS	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.

c) TRÊS CAMADAS: para minimizar o impacto de mudanças nas aplicações, decidiu-se separar a camada de negócio da camada de interface gráfica, gerando três camadas: Camada de Apresentação, Camada Lógica do Negócio e Camada de Acesso a Dados.

- 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.

    - Camada de Apresentação: É a chamada GUI (*Graphical User Interface*), ou simplesmente interface. Esta camada interage diretamente com o usuário, é através dela que são feitas as requisições como consultas, por exemplo.

    - Camada de Negócio: É responsável por implementar a lógica de negócio da aplicação. Nela estão todas as classes inerentes ao domínio da aplicação.

    - Camada de Dados: É responsável pela persistência e acesso aos dados da aplicação. Ela isola o resto da aplicação do meio de armazenamento usado de maneira que, se o meio de armazenamento for trocado, apenas as classes desta camada precisarão ser modificadas ou substituídas.

- Na arquitetura de três camadas, a camada intermediária, também conhecida como camada de lógica de negócios, geralmente é executada em um servidor de aplicativos.

- Na arquitetura de três camadas, a camada intermediária (lógica de negócios) geralmente atua como intermediária para a comunicação entre a camada de apresentação e a camada de dados.

CAMADA DE APRESENTAÇÃO	Também chamada de 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.

d) QUATRO CAMADAS:

- A) 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).

- B) quatro camadas em vez de três. Eles adicionam uma camada de infraestrutura à estrutura base da Arquitetura de 3 Camadas.

  MAIS EXTERNA:
  - Camada de Apresentação: Esta é a camada superior da arquitetura que os usuários veem e interagem. Ela fornece uma maneira para os usuários manipularem os dados e as funções do sistema. Na questão, essa camada é responsável pela interação com o ambiente do sistema.
  - Camada de Aplicação: Também conhecida como camada de lógica de negócios, esta é onde as funções que suportam as regras de negócio da aplicação são implementadas. Na questão, essa camada é responsável por conter pontos de entrada para as funcionalidades do sistema.
  - Camada de Domínio: Esta é uma extensão da arquitetura de três camadas, onde a lógica de negócios ou as regras de domínio são implementadas.
  - Camada de Dados (infraestrutura): Esta é a camada onde os dados são armazenados e recuperados de um banco de dados ou outro tipo de armazenamento. Na questão, a camada de desempenha este papel, sendo responsável por fornecer serviços técnicos, tais como transações e persistência.
- MAIS INTERNA:

Portanto, a Arquitetura de 4 Camadas é uma evolução da Arquitetura de 3 Camadas com uma camada adicional para separar as responsabilidades de infraestrutura da lógica de negócios.

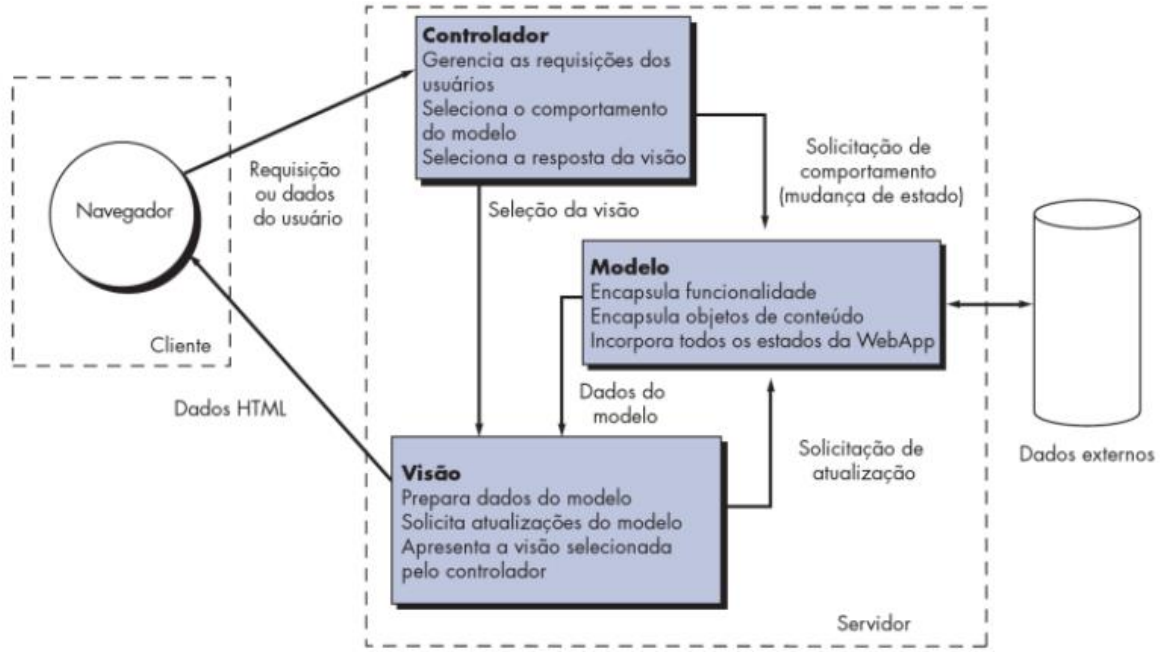
7.2. Arquitetura MVC

- 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.

- padrão arquitetural de software que separa uma aplicação em três camadas.

- forma de organizar o código de uma aplicação de forma que sua manutenção fique mais fácil.

MODEL	Essa classe também é responsável por gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas.
VIEW	Essa camada é responsável por apresentar as informações de forma visual ao usuário. Em seu desenvolvimento devem ser aplicados apenas recursos ligados a aparência como mensagens, botões ou telas.
CONTROLLER	Essa camada é responsável por intermediar as requisições enviadas pelo View com as respostas fornecidas pelo Model, processando os dados que o usuário informou e repassando para outras camadas.



8. ARQUITETURA ORIENTADA A SERVIÇOS (SOA)

- Um conjunto de princípios e melhores práticas para implementação e execução de processos de negócio automatizados em ambientes de tecnologia da informação heterogêneos.
- Uma forma de aproximar a linguagem do negócio e da tecnologia da informação, facilitando a integração de ambientes corporativos por meio de serviços.
- Um meio para organizar as soluções que promove o reúso, o crescimento e a interoperabilidade.
- Uma abordagem distribuída (não-monolítica) para integração de arquiteturas baseadas no conceito de serviço.
- Uma abordagem arquitetural corporativa que permite a criação de serviços de negócio interoperáveis, que podem ser reutilizados e compartilhados entre aplicações e empresas.
- No mercado atual, a plataforma de tecnologia mais associada com a realização de Arquitetura Orientada a Serviços são os Web Services.
- É muito importante visualizar e posicionar a Arquitetura Orientada a Serviços como um modelo de arquitetura que é agnóstico a qualquer plataforma de tecnologia.

- Não é uma tecnologia.
- Não é um produto.
- Não é um web service.
- Não é um projeto de TI.
- Não é um software.
- Não é um framework.
- Não é uma metodologia.
- Não é solução de negócio.
- Não é um middleware.
- Não é um serviço.
- Não é uma ferramenta.

VANTAGENS DO SOA	DESVANTAGENS DO SOA
Reutilização: o serviço pode ser reutilizado para outras aplicações.	**Performance:** a performance depende do servidor onde o serviço está publicado, como também da rede.
Produtividade: com o reuso, a equipe de desenvolvimento pode reutilizar serviços em outros projetos, diminuindo o tempo de desenvolvimento.	**Complexidade:** uma grande quantidade de serviços precisa ser gerenciada.
Flexibilidade: isolando a estrutura de um serviço as mudanças são feitas com maior facilidade.	**Robustez:** caso uma exceção acontecer não tem como reverter o processo.
Manutenibilidade: com baixo acoplamento, facilita a manutenção dos serviços.	**Disponibilidade:** uma queda na rede ou no servidor deixa todos os serviços indisponíveis.
Alinhamento com o negócio: a área de negócio visualiza os processos alinhados com a tecnologia.	**Testabilidade:** o debug no serviço é um problema para os desenvolvedores.
Interoperabilidade: disponibilizar serviços independentemente da plataforma e tecnologia.	**Governança:** exige-se que haja governança na organização.
Integração: a integração com outros serviços, aplicativos e sistemas legados.	**Design:** grande aumento na complexidade do design dos serviços.
Governança: gerenciamento dos processamentos de negócio.	**Segurança:** os serviços estão disponíveis na rede, qualquer aplicativo pode consumi-los; os dados são trafegados pela rede podendo ser interceptados.
Padronização: é baseado no uso de padrões (de design, de contratos, protocolos, etc).	
Abstração: serviço totalmente abstraído da sua implementação.	
Riscos: auxilia a mitigação de riscos de negócio.	

8.1. Princípios de Design SOA

Oito princípios básicos de design que norteiam toda modelagem lógica de serviços e aplicações:

1) Contrato de Serviço Padronizado (Service Contract)

- Todo serviço deve conter um contrato formal padronizado. O que é um contrato? Basicamente, trata-se de um documento que descreve o que o serviço faz, dentre outras coisas.
- Este princípio indica que os contratos de serviço, que definem como um serviço é usado, devem ser padronizados. Isso pode aumentar a reutilização e a interoperabilidade.

2) Baixo Acoplamento de Serviços (Service Loose Coupling)

- O baixo acoplamento de um serviço está relacionado com sua capacidade de ser independente de outros serviços para realizar sua tarefa.

- Este princípio sugere que os serviços devem ser tão independentes quanto possível, o que significa que uma mudança em um serviço não deve afetar os outros.

- Uma possível consequência do baixo acoplamento é a alta coesão (divisão de responsabilidades). Cada serviço tem sua responsabilidade bem definida e coerente: princípio da responsabilidade única.

- COMUNICAÇÃO ASSÍNCRONA: permite que os aplicativos enviem e recebam mensagens independentemente uns dos outros, o que reduz o acoplamento entre eles. Isso permite que um aplicativo continue trabalhando mesmo se outro aplicativo não estiver disponível ou demorar para responder. Isso é de particular importância em uma arquitetura orientada a serviços (SOA), onde diferentes serviços podem ter diferentes tempos de resposta e disponibilidade.

3) Abstração de Serviços (Service Abstraction)

- esconder a lógica de negócios complexa por trás de uma interface simples.
- serviços devem ser vistos como uma caixa-preta que compõe um sistema. Se eu quiser mudar a implementação, refatorar, manutenir, não há nenhum problema – desde que eu mantenha os padrões de entrada e saída definidos no contrato.

4) Reusabilidade de Serviços (Service Reusability)

- É de se esperar que um serviço seja reutilizável, isto é, eu posso utilizá-lo em diferentes contextos e tecnologias.

5) Autonomia de Serviços (Service Autonomy)

- É a capacidade de um serviço se auto-administrar. Um serviço autônomo é aquele que independe de um elemento externo para executar sua lógica.
- Se o serviço não depende de algum elemento externo para executar suas atividades, então ele é mais seguro, confiável e autônomo.
- Este princípio estabelece que os serviços devem ser autônomos, ou seja, capazes de funcionar independentemente de outros serviços. Eles devem ser capazes de serem modificados e evoluídos sem afetar outros serviços.

6) Independência de Estados de Serviços (Service Statelessness)

- Serviços são explicitamente projetados para minimizar o período durante o qual eles existem em uma condição de dependência de informações de estado. A independência de estado ocorre quando um serviço não precisa reter nenhum dado do estado/conexão para que outro serviço processe a sua lógica de negócio
- Os serviços devem evitar a alocação de recursos por muito tempo, logo é recomendada a construção de serviços stateless (sem estado).
- Independência de Estado de Serviço que preza por uma regra de negócio mais geral e que não causa dependência, por isso que se fala em regras de contexto. Veja uma descrição dele abaixo:
    - Os serviços não devem manter estado em memória (Stateless), desta forma são mais escaláveis e agnósticos de processos de negócio, o que aumenta o reúso.

7) Visibilidade de Serviços (Service Discoverability)

- Serviços são projetados para serem efetivamente descobertos e interpretados para que, na descoberta, seu propósito e capacidades sejam claramente entendidos.
- O objetivo principal é fazer com que o serviço se torne visível a todos, aumentando assim a agilidade, a produtividade e a confiabilidade dos consumidores. Os serviços são projetados para que sejam encontrados e seus propósitos e capacidades sejam claramente entendidos, ou seja, os contratos de serviços devem ter meta-informações extras que transmitem claramente o que o serviço realmente faz.

- O mecanismo de descoberta é necessário para evitar o desenvolvimento redundante de uma solução que já está contida em um serviço existente

8) Composição de Serviços (Service Composability)

- a capacidade de combinar serviços para criar um novo serviço;
- Serviços são projetados para que atuem como participantes eficazes de uma composição, independentemente do tamanho e da complexidade de composição.
- A composição de serviços permite que as capacidades de um serviço sejam combinadas várias vezes com as de outros serviços em novas configurações a fim de resolver problemas diferentes.

9) Interoperabilidade

- Este princípio indica que os serviços devem ser capazes de trabalhar juntos, independentemente de suas plataformas ou tecnologias subjacentes. A solução que permite que sistemas antigos e novos trabalhem juntos.

8.2. Composição De Serviços

Workflows de coordenação para implementar a composição de serviços:

1) Orquestração de Serviços

- A orquestração de serviços representa o processo em que diversos recursos dentro de uma organização podem ser coordenados para executar um conjunto de lógicas de processo de negócio.
- Essa tecnologia é mais comumente associada a uma plataforma central de gerenciamento de atividades que fornecem benefícios como transformação de dados e gerenciamento de transações.
- Em outras palavras, a orquestração representa um processo de negócio executável centralizado e único, chamado coordenador, responsável por controlar e coordenar as interações entre diferentes serviços. Ele é o cara que vai invocar e combinar os serviços isolados em serviços compostos.
- Lógica de processo de negócio – na orquestração – pertence à organização, isto é, trata-se de uma lógica privada e intra-organização.

2) Coreografia de Serviços

- Não possui um coordenador central.
- Cada serviço envolvido na coreografia sabe exatamente quando executar suas operações e com quem interagir.
- A coreografia é um esforço colaborativo, distribuído e descentralizado com foco na troca de mensagens em processos de negócio públicos.

- O processo negócio é público e interorganizações.

8.3. ESB (Enterprise Service Bus)

O Enterprise Service Bus (ESB) é um padrão de arquitetura de software que permite a integração de sistemas heterogêneos através de uma infraestrutura de comunicação comum. É uma peça fundamental em uma arquitetura orientada a serviços (SOA), proporcionando um meio para gerenciar a comunicação e a coordenação entre os serviços.

Um ESB é caracterizado por sua flexibilidade, confiabilidade, escalabilidade e capacidade de lidar com comunicação assíncrona. Ele é projetado para fornecer uma maneira consistente de conectar serviços e aplicativos, facilitando a reutilização de serviços, a redução de complexidade e o aumento da agilidade.

O ESB atua como um intermediário, que recebe mensagens de um aplicativo ou serviço e as encaminha para o destino apropriado, de acordo com as regras de roteamento definidas. Ele permite a transformação de mensagens para garantir que o formato da mensagem seja compatível com o sistema de destino. Além disso, um ESB também pode lidar com o enriquecimento de mensagens, adicionando informações adicionais conforme necessário.

Além disso, um ESB fornece recursos de segurança, como autenticação e autorização, para garantir que as mensagens sejam seguras e que apenas as partes apropriadas possam acessar ou enviar mensagens. Ele também pode lidar com transações, assegurando que todas as partes de uma transação sejam concluídas com sucesso, ou que a transação seja revertida se algo der errado.

O ESB também é útil para a orquestração de serviços, o que significa que ele pode coordenar a execução de vários serviços para completar um processo de negócios. Isso pode incluir a execução de serviços em uma sequência específica, o controle de fluxos de trabalho complexos e a gestão de exceções.

Em resumo, um ESB é uma ferramenta vital para empresas que lidam com uma variedade de sistemas e serviços, fornecendo uma maneira unificada e flexível de gerenciar a comunicação e a integração entre eles.

Para sua implementação, temos um conjunto básico de tecnologias, linguagens e protocolos, são eles:

- HTTP: Protocolo de transmissão de dados na web.
- SOAP ou REST: Linguagem de transporte de documentos XML na web
- XML: Linguagem padrão de troca de dados.
- WSDL: Tecnologia para descrição de interface em web services.
- UDDI: Tecnologia responsável pela descoberta dos serviços.

8.1. Serviços XML

- DTD (Document Type Definition)
    - DTD é usada para definir a estrutura e o tipo de dados em um documento XML.
- XSLT
    - XSLT (Extensible Stylesheet Language Transformations) é uma linguagem para transformar documentos XML em outros formatos de documento XML ou outros formatos como HTML, PDF, etc. É comumente usado em SOA e Web Services para

transformar a estrutura de dados de um serviço para a estrutura de dados esperada por outro serviço, ajudando a resolver discrepâncias entre eles.

- XQuery
  - XQuery é uma linguagem de consulta para documentos XML. Embora seja útil para pesquisar e extrair dados de documentos XML, não é projetada para transformar a estrutura dos dados.

- XLink
  - XLink é uma linguagem para criar links complexos entre documentos XML. Não é relevante para resolver discrepâncias de estrutura de dados em SOA ou Web Services.

- XSL-FO
  - XSL-FO (Extensible Stylesheet Language Formatting Objects) é uma linguagem para formatar documentos XML para saída, como impressão ou conversão em PDF.

9. WEB SERVICES

- Web Service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

- Os Web Services são componentes que permitem às aplicações enviar e receber dados. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, em um formato intermediário como XML, Json, CSV, etc.

- Web Service é a disponibilização de um serviço pela internet que pode ser acessado em qualquer lugar. Clientes enviam requisições com informações bem definidas e recebem respostas que podem ser síncronas ou assíncronas.

- Web Service é essencialmente a interoperabilidade entre programação e aplicações - especialmente quando eles usam linguagens, ferramentas ou plataformas diferentes.

- WEBSERVICE: um sistema de software projetado para suportar a interoperabilidade entre máquinas sobre rede / fazer a integração sistemas heterogêneos;

- Mecanismo que permite acessar um conjunto de recursos, no qual o acesso é fornecido por meio de uma interface descrita e exercitada consistentemente de acordo com restrições e políticas especificadas pela descrição do serviço;

- Web Services tratam da disponibilização de um serviço pela internet que pode ser acessado em qualquer lugar.

- Web Services são componentes de aplicativos baseados em XML, autocontidos e auto descritivos, que se comunicam usando protocolos abertos.

- Web Services são uma interface que descreve uma coleção de operações que são acessíveis pela rede através de mensagens XML padronizadas.

- Web Services tratam essencialmente da interoperabilidade entre programas e aplicações – especialmente quando eles usam linguagens, ferramentas ou plataformas diferentes.
- Web Services são um sistema de software projetado para permitir interoperabilidade na interação entre máquinas através de uma rede.
- Web Services são componentes de software com baixo fator de acoplamento, utilizado por meio de padrões de internet.
- Web Services representam uma função/lógica de negócio ou um serviço que pode ser acessado por uma outra aplicação na web, sobre redes públicas e, geralmente, disponibilizado por protocolos conhecidos.
- Web Services são soluções utilizadas na integração de sistemas e na comunicação entre aplicações diferentes, permitindo que elas enviem e recebam dados.
- Componentes de software que permitem a integração de aplicações distribuídas desenvolvidas com tecnologias heterogêneas;

CARACTERÍSTICA	DESCRIÇÃO
WEB SERVICES SÃO AUTOCONTIDOS	Isso significa que eles não necessitam ou dependem de outros componentes para ter uma existência própria.
WEB SERVICES SÃO AUTODESCRITIVOS	Isso significa que eles não necessitam de informações externas para expor suas funcionalidades.
WEB SERVICES UTILIZAM PROTOCOLOS ABERTOS	Isso significa que os protocolos não são de propriedade de nenhuma organização, são apenas protocolos padrões da internet.
WEB SERVICES SÃO FRACAMENTE ACOPLADOS	Isso significa que a interface do serviço pode mudar sem comprometer a capacidade do cliente de interagir com o serviço.
WEB SERVICES SÃO INDEPENDENTES DE TECNOLOGIA	Isso significa que eles são independentes de plataforma, sistema operacional, arquitetura de processador, linguagem de programação, entre outros.

9.1. Arquitetura de Web Services: Paradigma SOAP

- SOAP (Simple Object Access Protocol) é um protocolo baseado em XML utilizado para a troca de informações estruturadas e tipadas em uma rede, geralmente através de HTTP. Ele é comumente usado para implementar serviços da web, que são um componente-chave da arquitetura orientada a serviços (SOA).
- XML: É uma linguagem de marcação que permite definir um conjunto de regras para troca de dados, podendo ser compreendida tanto por humanos quanto por máquinas.
- O elemento "soap:Envelope" é o elemento raiz em um documento XML SOAP. Ele define o início e o fim de uma mensagem SOAP e é obrigatório em todas as mensagens SOAP.

A arquitetura de Web Services envolve basicamente três categorias de participantes:

- Provedor de Serviço (Service Provider): É qualquer aplicação que forneça serviços web;
- Solicitante do Serviço (Service Requester): É qualquer aplicação que deseje utilizar serviços web;

- Agente de Serviço (Service Broker): É qualquer aplicação que faça a intermediação entre provedor e solicitante;

A comunicação entre serviços web é por meio de três padrões fundamentais:

PADRÕES	DESCRIÇÃO
SOAP (SIMPLE/SINGLE OBJECT ACCESS PROTOCOL)	Baseado em XML, define uma organização para troca estruturada de dados entre Web Services.
WSDL (WEB SERVICES DESCRIPTION LANGUAGE)	Baseado em XML, define como as interfaces dos Web Services podem ser representadas.
UDDI (UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION)	Baseado em XML, trata-se do padrão de descobrimento que define como as informações podem ser organizadas.

9.1.1. SOAP (SIMPLE/SINGLE OBJECT ACCESS PROTOCOL)

- Trata-se de uma das formas de comunicação para encapsular dados transferidos no formato XML para Web Services;
- Trata-se de um formato baseado em XML para intercâmbio de mensagens – é utilizado para realizar o encapsulamento e o transporte de dados;
- Trata-se de um formato para envio e recebimento de mensagens independentemente de plataforma e tecnologia;
- Trata-se de um protocolo baseado em XML que define uma organização para a troca estruturada de dados entre Web Services;

ELEMENTOS	SITUAÇÃO	DESCRIÇÃO
ENVELOPE (ENVOLOPE)	OBRIGATÓRIO	Trata-se do elemento-raiz do documento XML. Ele identifica o documento XML como uma mensagem SOAP e funciona como um recipiente que contém os demais elementos da mensagem (Ex: Header e Body). Corresponde à descrição da mensagem e do que deve ser processado.
CABEÇALHO (HEADER)	OPCIONAL	Trata-se do elemento que carrega informações adicionais específicas para a aplicação a fim de estender as funcionalidades das Mensagens SOAP. Desta forma, é possível adicionar novas funcionalidades como autenticação, criptografia, autorização, assinatura digital, etc. Podem ser definidos vários cabeçalhos.
CORPO (BODY)	OBRIGATÓRIO	Trata-se do elemento que contém a carga útil (*payload*) da Mensagem SOAP. É aqui que estão as informações propriamente ditas do serviço web remetidas ao destinatário final da mensagem, incluindo a chamada ao procedimento ou o retorno de seu resultado.
FALHA (FAULT)	OPCIONAL	Trata-se do elemento contido no corpo responsável por relatar possíveis erros de envio ou processamento de mensagens, informando localização e formato dos erros encontrados. Quando estiver presente deve aparecer como um elemento filho do elemento Body.

```
POST /InStock HTTP/1.1
Host: www.dre.ufrj.br
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap- envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding" xmlns:tiposns="http://www.w3.org/2001/XMLSchema">

       <soap:Header>
             <m:atenticacao xmlns:m="http://www.dre.ufrj.br/ws/dre">21423edf69fgs</m:atenticacao>
       </soap:Header>

       <soap:Body>
             <m:retornaNome xmlns:m="http://www.dre.ufrj.br/ws/dre">
                   <numdre type="tiposns:int">123.456.789-00</drenum>
             </m:retornaNome>
       </soap:Body>

</soap:Envelope>
```

9.1.2. WSDL (WEB SERVICES DESCRIPTION LANGUAGE):

- WSDL (Web Services Description Language) é uma linguagem baseada em XML que é usada para descrever a funcionalidade oferecida por um serviço da web. Ele fornece uma maneira para os serviços da web descreverem a si mesmos, de modo que os clientes possam entender como interagir com eles.
- O uso de web services supõe a existência de uma arquitetura composta de um registro (registry), um provedor de serviço e um consumidor de serviço. Um provedor de serviços descreve seus serviços para o registro utilizando a linguagem WSDL.
- Trata-se de uma linguagem de descrição de web services, escrita em XML, para descrever serviços web, especificar as formas de acesso, as operações/métodos disponíveis.
- Trata-se de uma linguagem para descrever serviços de rede como endpoints (ou portas) que operam em mensagens que contêm informações orientadas à documento/procedimento.
- O elemento <portType> em um documento WSDL é usado para definir o conjunto abstrato de operações suportadas por um serviço web. Cada <portType> pode ter várias operações.
- Trata-se efetivamente de especificação que define como descrever serviços web em uma gramática XML.
- WSDL tem o propósito de descrever os serviços de rede como um conjunto de terminais que operam sobre mensagens recebidas.
- Trata-se de um protocolo baseado em XML para troca de informações em ambientes distribuídos e descentralizados (sim, alguns autores o consideram também como um protocolo).
- Documento que, resumidamente, descreve quais são as operações disponíveis em um serviço web com todas as particularidades de suas interfaces como métodos, parâmetros, protocolos, entre outros.

WSDL separa a descrição de um serviço em duas perspectivas:

1) Abstrata:
Trata-se da interface do serviço. Ou seja, descreve o que o serviço faz: parâmetros, tipos, operações, entradas, saídas, mensagens fault, etc.

2) Concreta:
Trata-se da implementação do serviço. Ou seja, descreve como o serviço é feito: protocolos de comunicação, codificação de dados, localização, portas, endereço de rede, etc. Ela também adiciona informações sobre como o serviço se comunicará e quem pode alcançá-lo.

Caso a implementação (perspectiva concreta) do serviço seja modificada por alguma razão, a interface (perspectiva abstrata) pode continuar a ser disponibilizada sem problemas – e até reutilizada para diversas implementações diferentes

WSDL define quatro padrões de mensagens suportadas para quatro tipos de operações:

TIPO	DEFINIÇÃO
ONE-WAY	A operação pode receber uma requisição, mas não retornará uma resposta.
REQUEST-RESPONSE	A operação pode receber uma requisição e retornará uma resposta.
SOLICIT-RESPONSE	A operação pode enviar uma requisição e esperará por uma resposta.
NOTIFICATION	A operação pode enviar uma mensagem, mas não esperará por uma resposta.

9.1.3. UDDI (UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION):

- O UDDI é uma especificação técnica, ou protocolo e um serviço de diretório onde empresas podem registrar, publicar, descrever, buscar, descobrir e integrar serviços web.
- UDDI contém informações genéricas sobre a provedores de serviços web, implementações e metadados básicas sobre eles.
- Trata-se de um serviço de diretório, baseado em XML, em que é possível registrar e localizar Web Services.
- Trata-se de uma especificação técnica que tem como objetivo descrever, descobrir e integrar Web Services.
- Trata-se de um diretório/registro para armazenamento de informações sobre Web Services – é um repositório de interfaces de Web Services descritas por WSDL.
- Trata-se de um protocolo que é um dos maiores blocos de construção requeridos para construir Web Services com sucesso (Sim, alguns o chamam de protocolo!).
- Trata-se de um padrão de descoberta que define como são organizadas as informações de descrição do serviço, permitindo que os solicitantes descubram os serviços.

As informações capturadas no contexto do UDDI podem ser classificadas em três categorias principais:

1) As Páginas Brancas contêm informações gerais sobre a organização que está oferecendo o serviço web, tais como nome e descrição do negócio (de preferência, em diversas línguas). Utilizando essas informações, é possível encontrar algum serviço sobre o qual já se pode conhecer algumas informações. Há também informações de contato do negócio (Ex: Endereço, Telefone, Fax, Identificadores, etc).

2) A Páginas Amarelas contêm uma classificação do serviço ou negócio disponíveis baseado em taxonomias padronizadas (Ex: SIC, NAICS, UNSPSC). Por exemplo: UNSPSC (United Nations Standard Products and Services Code) é uma taxonomia de produtos e serviços para e-commerce.

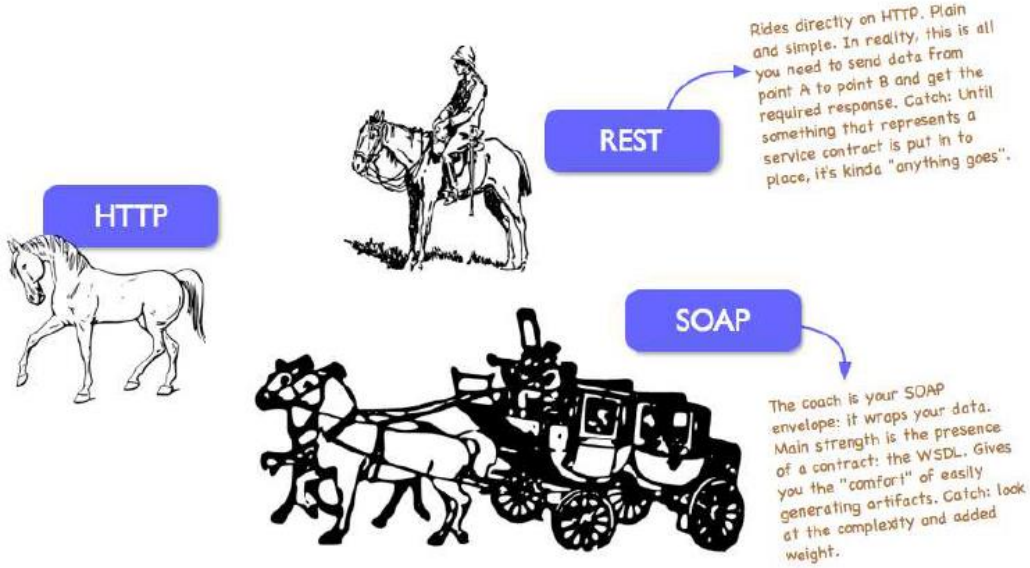
3) Páginas Verdes contêm informações mais técnicas sobre como acessar um serviço web. Elas são utilizadas para indicar os serviços oferecidos por cada negócio, incluindo todas as informações técnicas envolvidas na interação e acesso ao serviço. Em geral, essas informações incluem uma referência para uma especificação externa e um endereço para invocar o serviço.

9.1.4. Resumo Paradigma SOAP:

- Uma aplicação (Cliente/Solicitante de Serviço) utiliza um serviço de diretórios (Agente de Serviço) – como o UDDI – para localizar o serviço web oferecido por um servidor (Servidor/Fornecedor de Serviço).
- O UDDI fornece várias informações sobre o serviço web, incluindo a sua descrição por meio do WSDL.
- O cliente analisa o WSDL e envia uma Mensagem SOAP requisitando um serviço de acordo com as especificações.
- A mensagem é, então, enviada para o fornecedor do serviço, que interpreta a mensagem (realiza o parsing da mensagem) e invoca o método apropriado, passando os parâmetros fornecidos na mensagem.
- O método executado, então, retorna o resultado para o Servidor SOAP, que escreve uma Mensagem SOAP respondendo a requisição inicial com o resultado e envia para o Cliente SOAP.
- Por fim, este último lê a mensagem e repassa o resultado para a aplicação requisitante.

9.2. Arquitetura de Web Services: Paradigma REST

- REST (REpresentational State Transfer)



- Estilo arquitetural ou de desenvolvimento para projetar e construir aplicações de rede distribuídas fracamente acopladas.

- Se você apenas deseja chamar um serviço leve com parâmetros e respostas bem diretos, utilize o REST; se você deseja passar diversos parâmetros compostos e aninhados com respostas complexas, utilize o SOAP.

- REST é menos burocrático, tem menos overhead, suporta diversas linguagens e tem maior desempenho/escalabilidade.

- SOAP é mais burocrático, tem mais overhead, suporta apenas XML e tem menor desempenho/escalabilidade.

- SOAP exige o uso de XML para fornecer a resposta, enquanto REST pode ser usado com vários formatos, como JSON, RSS ou mesmo XML

- No SOAP, existe uma especificação que deve ser seguida para todas as requisições/respostas (chamada WSDL); no REST, não existe nenhuma especificação obrigatória.

SOAP [SIMPLE OBJECT ACCESS PROTOCOL]	REST [REPRESENTATIONAL STATE TRANSFER]
É um protocolo de comunicação baseado em XML.	É um estilo arquitetural ou de desenvolvimento independente de tecnologia.
Utiliza um Envelope enviado por geralmente por HTTP para transmitir dados.	Utiliza diretamente recursos oferecidos de forma nativa, em regra, pelo HTTP.
Suporta somente recursos no formato XML.	Suporta recursos no formato HTML XML, JSON, YAML, TXT, etc.
Permite invocar serviços por meio de Métodos RPC (Remote Procedure Calls).	Permite invocar serviços por meio da própria URI/URL.
Em geral, apresenta desempenho e escalabilidade menor, devido ao alto overhead.	Em geral, apresenta desempenho e escalabilidade maior, devido ao baixo overhead.
Não permite fazer *caching*.	Permite fazer *caching*.
Requer maior largura de banda para trafegar os dados.	Requer menor largura de banda para trafegar os dados.
Suporta recursos da WS-Security para incrementar a segurança.	Suporta apenas SSL/TLS e HTTPS para incrementar a segurança.
JavaScript pode chamar SOAP, mas é de difícil de implementação.	JavaScript pode facilmente chamar REST.

RESTRIÇÃO OU PRINCÍPIO	DESCRIÇÃO
CLIENTE/SERVIDOR	Responsabilidades devem ser separadas entre clientes e servidores. Isso permite que os componentes do cliente e do servidor evoluam de forma independente e, por sua vez, permite que o sistema seja escalável. Em outras palavras, busca-se separar a arquitetura e responsabilidades em dois ambientes. Dessa forma, o cliente não se preocupa com tarefas como a comunicação com banco de dados, gerenciamento de cache, log, entre outros; e o contrário também é válido, o servidor não se preocupa com tarefas como interface, experiência do usuário, entre outros. Permitindo, assim, a evolução independente das duas arquiteturas.
STATELESS (SEM ESTADO)	A comunicação entre cliente e servidor deve ser stateless (isto é, sem guardar estado). O servidor não precisa lembrar do estado do cliente. Em vez disso, os clientes devem incluir todas as informações necessárias na requisição para que o servidor possa entendê-la e processá-la. Em outras palavras, um mesmo cliente pode mandar várias requisições para o servidor, porém cada uma delas deve ser independente, ou seja, toda requisição deve conter todas as informações necessárias para que o servidor consiga entendê-la e processá-la adequadamente (qualquer informação de estado deve ficar no cliente).
CACHE	Respostas do servidor devem ser declaradas como *cacheable* ou *noncacheable*. Isso permite que o cliente ou seus componentes intermediários armazenem em *cache* respostas e reutilizem-nas para pedidos posteriores. Isto reduz a carga no servidor e ajuda a melhorar o desempenho. Isso significa que, quando um primeiro cliente solicita um determinado recurso ao servidor, esse processa a requisição e o cliente a armazena temporariamente em *cache*. Quando houver uma nova requisição, a resposta armazenada já está pronta para ser utilizada e nem precisará ser recuperada novamente.
INTERFACE UNIFORME	Todas as interações entre cliente, servidor e componentes intermediários são baseadas na uniformidade de suas interfaces. Isso simplifica a arquitetura geral, visto que componentes podem evoluir de forma independente à medida que implementem o que foi acordado em contrato. É basicamente um contrato para comunicação entre cliente e servidor. São regras para fazer um componente o mais genérico possível, tornando-o muito mais fácil de ser refatorado e melhorado. Obedece a quatro princípios: identificação de recursos; representação de recursos; respostas auto-explicativas/descritivas; e hypermídia.
CÓDIGO SOB DEMANDA	Esse princípio é opcional, na medida em que não faz parte da arquitetura em si. Ele trata da possibilidade de clientes poderem estender suas funcionalidades através do download e execução do código sob demanda. Exemplos incluem scripts Javascript, Applets Java, Silverlight, etc. Em outras palavras, permite que o cliente possa executar algum código sob demanda, ou seja, estender parte da lógica do servidor para o cliente, seja através de applets ou scripts. Assim, diferentes clientes podem se comportar de maneiras específicas mesmo que utilizando exatamente os mesmos serviços providos pelo servidor.

]

10. PADRÕES DE PROJETO (DESIGN PATTERNS)

PADRÕES CRIACIONAIS	DESCRIÇÃO
BUILDER	Esse padrão separa a construção de um objeto complexo da sua representação, de forma que o mesmo processo de construção possa criar diferentes tipos de representações.
4 ABSTRACT FACTORY	Esse padrão fornece uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.
PROTOTYPE	Esse padrão especifica os tipos de objetos para criar usando uma instância como protótipo e cria novos objetos copiando este protótipo.
2 SINGLETON	Esse padrão garante que uma classe tenha apenas uma instância e provê um ponto de acesso global a ela.
FACTORY METHOD	Esse padrão define uma interface para criar um objeto, mas deixa as subclasses decidirem qual classe instanciar.

PADRÕES ESTRUTURAIS	DESCRIÇÃO
1 ADAPTER	Esse padrão converte a interface de uma classe em outra interface que normalmente não poderiam trabalhar juntas por serem incompatíveis.
6 BRIDGE	Esse padrão desacopla uma interface de sua implementação, de forma que ambas possam variar independentemente.
COMPOSITE	Esse padrão compõe objetos em estruturas de árvore para representar hierarquias parte-todo, permitindo aos clientes tratarem objetos individuais e composições de objetos uniformemente.
DECORATOR	Esse padrão anexa responsabilidades adicionais a um objeto dinamicamente. Fornece uma alternativa flexível em relação à herança para estender funcionalidades.
3 FACADE	Esse padrão oferece uma interface unificada para um conjunto de interfaces em um subsistema, definindo uma interface de alto nível que facilita a utilização do subsistema.
FLYWEIGHT	Esse padrão utiliza compartilhamento para suportar eficientemente grandes quantidades de objetos de baixa granularidade.
PROXY	Esse padrão provê um substituto ou ponto através do qual um objeto pode controlar o acesso a outro objeto.

PADRÕES COMPORTAMENTAIS	DESCRIÇÃO
CHAIN OF RESPONSABILITY	Esse padrão evita o acoplamento do remetente de uma requisição ao seu receptor ao dar a mais de um objeto a chance de lidar com a requisição.
COMMAND	Esse padrão encapsula a requisição de um objeto, portanto permitindo que se parametrize os clientes com diferentes requisições.
5 ITERATOR	Esse padrão fornece uma maneira de acessar elementos de um objeto agregado sequencialmente sem expor sua representação interna.
MEDIATOR	Esse padrão define um objeto que encapsula a forma como um conjunto de objetos interagem, promovendo um fraco acoplamento ao evitar que objetos se refiram aos outros explicitamente.
MEMENTO	Esse padrão captura e externaliza o estado interno de um objeto, sem violar seu encapsulamento, de maneira que o objeto possa ser restaurado posteriormente.

11. ANÁLISE DE PONTOS DE FUNÇÃO

- Qual é a diferença entre análise de pontos de função e ponto de função?

  Análise de Pontos de Função é uma técnica, enquanto o ponto de função é o resultado da aplicação dessa técnica.

- O que mede a métrica de ponto de função? SOB A PERSPECTIVA DE QUEM?

  Ele mede o tamanho funcional de um software sob o ponto de vista do usuário – o que ele faz e, não, como ele faz.

- A AFP é dependente de alguma tecnologia específica?

  Não, ela é completamente independente de tecnologia, plataforma, linguagem de programação, etc.

- A AFP mede diretamente o esforço, produtividade ou custo de um software?

  Não, mas é possível correlacionar essas variáveis à métrica de pontos de função.

- Funciona como uma ferramenta para determinar o tamanho de um pacote adquirido, através da contagem de todas as funções incluídas (pode auxiliar no momento da venda de um sistema de software) e suporta a análise de produtividade e qualidade em conjunto com outras métricas como esforço, defeitos e custo.

- Provê auxílio aos usuários na determinação dos benefícios de um pacote para sua organização, através da contagem das funções que especificamente correspondem aos seus requisitos. Ao avaliar o custo do pacote, o tamanho das funções que serão utilizadas, a produtividade e o custo da própria equipe, é possível analisar se vale a pena comprar ou fabricar.

- Apoia o gerenciamento de escopo de projetos. Um desafio de todo gerente de projetos é controlar o aumento do escopo. Ao realizar estimativas e medições em cada fase do seu ciclo de vida, é possível determinar se os requisitos funcionais cresceram ou diminuíram; e se esta variação corresponde a novos requisitos ou aos existentes e que foram apenas mais detalhados.

- Complementa o gerenciamento dos requisitos auxiliando na verificação da solidez e completeza dos requisitos especificados. O processo de contagem favorece uma análise sistemática e estruturada da especificação de requisitos e traz benefícios semelhantes ao de uma revisão em pares – basta lembrar que a contagem é baseada em requisitos do usuário.

- Um meio de estimar custo e recursos para o desenvolvimento e manutenção de software. Através da realização de uma contagem ou estimativa de pontos de função no início do ciclo de vida de um projeto, é possível determinar seu tamanho funcional. Esta medida pode ser então utilizada como entrada para diversos modelos de estimativa de esforço, prazo e custo.

- Uma ferramenta para fundamentar a negociação de contratos. Pode-se utilizar pontos de função para gerar diversos indicadores de níveis de serviço em contratos de desenvolvimento e manutenção de sistemas. Além disso, permite o estabelecimento de contratos a preço unitário, onde a unidade representa um bem tangível para o cliente – reduzindo riscos.

- Um fator de normalização para comparação de software ou para a comparação da produtividade na utilização de diferentes técnicas. Diversas organizações, como o ISBSG, disponibilizam um imenso repositório de dados de projetos de software que permitem a realização de benchmarking com projetos similares do mercado e uma boa base de comparação.

FUNÇÕES DE DADOS E TRANSAÇÃO	COMPLEXIDADE FUNCIONAL		
	BAIXA	MÉDIA	ALTA
CONSULTA EXTERNA (CE)	3	4	6
ENTRADA EXTERNA (EE)	3	4	6
SAÍDA EXTERNA (SE)	4	5	7
ARQUIVO DE INTERFACE EXTERNA (AIE)	5	7	10
ARQUIVO LÓGICA INTERNA (ALI)	7	10	15

12. ERP (Enterprise Resource Planning)

ERP é um sistema interfuncional que atua como uma estrutura para integrar e automatizar muitos dos processos de negócios que devem ser realizados pelas funções de produção, logística, distribuição, contabilidade, finanças e de recursos humanos de uma empresa.

O Planejamento de Recursos Empresariais, conhecido pela sigla ERP (Enterprise Resource Planning), é uma ferramenta de gestão que integra todos os processos de uma empresa em um único sistema. O ERP é um software que permite a automação e a integração de processos de negócios, promovendo eficiência operacional, redução de custos e melhor tomada de decisão.

- Um dos principais benefícios do ERP é a centralização de informações. Isso significa que todos os dados gerados em diferentes departamentos da empresa, como vendas, produção, finanças, recursos humanos, entre outros, são armazenados em um único lugar. Isso facilita o acesso às informações, reduz a probabilidade de erros devido à duplicidade de dados e permite uma visão consolidada do negócio.

- Além disso, o ERP facilita a gestão de processos. Por meio de módulos dedicados a funções específicas, como controle de estoque, gestão de vendas, contabilidade, etc., a empresa pode monitorar e controlar suas operações de maneira mais eficiente. Isso permite uma maior agilidade na tomada de decisões e uma melhor alocação de recursos.

- O ERP também promove uma maior integração entre os departamentos da empresa. Com todos os dados em um único lugar, é possível ter uma visão mais completa do negócio, facilitando a colaboração entre diferentes áreas. Isso pode levar a uma maior eficiência operacional e a uma melhor tomada de decisões estratégicas.

Por fim, a implementação de um sistema ERP pode trazer desafios, como a necessidade de adaptação dos processos de negócios ao software, a resistência dos funcionários à mudança e o custo de implementação e manutenção. No entanto, quando bem implementado, o ERP pode trazer benefícios significativos para a empresa, como maior eficiência operacional, redução de custos e melhor tomada de decisão.

- Um Enterprise Resource Planning (ERP) pode agilizar o processo de tomada de decisão em uma empresa e diminuir suas margens de erro.

PORQUE

- Os ERP integram informações de diversos departamentos de uma empresa, possibilitando a automação das informações de negócios.

Vantagens:

- incorporação das melhores práticas de mercado aos processos internos da empresa.
- eliminação de redundâncias nas atividades empresariais.
- otimização do processo de tomada de decisão.
- redução do tempo de resposta ao mercado.

12.1. Implementação ERP

Implementar um sistema de planejamento de recursos empresariais (ERP) é uma tarefa complexa que requer planejamento cuidadoso e execução coordenada. Existem várias estratégias de implementação de ERP que as organizações podem escolher, dependendo de suas necessidades, recursos e tolerância ao risco. Aqui estão algumas das estratégias de implementação de ERP mais comuns:

- Implementação do tipo Big Bang:
  - Nesta abordagem, todos os módulos do ERP são implementados de uma só vez. A organização muda do sistema antigo para o novo sistema ERP em um único instante, geralmente após um período de treinamento intensivo. A vantagem dessa abordagem é que ela é rápida e permite que a organização comece a utilizar o novo sistema ERP o mais rápido possível. No entanto, também é arriscado, pois qualquer problema com o novo sistema pode afetar toda a organização de uma vez.

- Implementação em Fases:
  - Nesta estratégia, o ERP é implementado em estágios ou fases, ao longo de um período de tempo específico. Cada fase envolve a implementação de certos módulos ou funcionalidades do ERP. A principal vantagem dessa abordagem é que ela reduz o risco, já que qualquer problema que surja durante a implementação pode ser limitado a um módulo ou função específica. No entanto, essa abordagem leva mais tempo do que a implementação do tipo Big Bang.

- Implementação Paralela:
  - Nesta abordagem, o novo sistema ERP é executado em paralelo com o sistema antigo por algum tempo. Isso permite que os usuários se familiarizem com o novo sistema antes que o sistema antigo seja completamente desativado. A implementação paralela pode ser menos arriscada do que a implementação do tipo Big Bang, mas pode exigir mais recursos, já que a organização precisa manter e operar dois sistemas ao mesmo tempo.

- Implementação em Etapas:
  - Nesta abordagem, diferentes unidades de negócio ou locais geográficos implementam o ERP em diferentes momentos. Isso permite que a organização aprenda com a implementação em uma unidade ou local e aplique essas lições às implementações subsequentes. No entanto,

essa abordagem pode levar a inconsistências entre diferentes partes da organização.

Cada uma dessas estratégias tem suas vantagens e desvantagens, e a escolha entre elas depende de vários fatores, incluindo o tamanho e a complexidade da organização, os recursos disponíveis para a implementação do ERP, e a cultura e a tolerância ao risco da organização.

ESTRATÉGIA BIG BANG:

- A estratégia Big Bang na implementação de ERP envolve a transição de todo o sistema de uma só vez. A implementação é realizada em todas as partes da organização simultaneamente, ou seja, todos os módulos do ERP são colocados em operação na mesma data. Isso permite que todos na organização estejam na mesma página e também concentra todos os esforços de implementação em um único período de tempo.