FACVLDADE MAURÍCIO DE NASSAU
FAZENDO PARTE DA SUA HISTÓRIA

Arquitetura de Software

Configurações e Padrões Arquiteturais: principais padrões.

FACVLDADE MAURÍCIO DE NASSAU
FAZENDO PARTE DA SUA HISTÓRIA

- **Configuração**
  - **Configuração Arquitetural**: conjunto de associações específicas entre os componentes e os conectores de uma arquitetura de software;
  - Geralmente representada por um grafo onde nós são componentes e conectores e arestas ligações;
  - Indica uma possível comunicação entre componentes, mas não garante a real habilidade deles se comunicarem;
  - As ligações devem ser entre interfaces compatíveis. Caso contrário tem-se uma incompatibilidade arquitetural.

Configurações e Padrões Arquiteturais: principais padrões.

FACVLDADE MAURÍCIO DE NASSAU
FAZENDO PARTE DA SUA HISTÓRIA

- **Padrão Arquitetural**
  - Coleção identificada de decisões arquiteturais de projeto que são aplicáveis a um problema recorrente de desenvolvimento e parametrizadas de modo a serem aplicadas em qualquer contexto de desenvolvimento de software no qual o problema aparece;
  - Descreve uma relação entre organização e elementos que já deu certo em sistemas que foram desenvolvidos outras vezes;
  - Não são simplesmente “teorias”, e sim resultado prático de experimentos feitos em diferentes ambientes e sistemas.
  - Podem ser configurados com os componentes e conectores de um determinado sistema em questão.

Configurações e Padrões Arquiteturais: principais padrões.

- **Padrão Arquitetural**
  - Comumente empregado em sistemas de informação:
    - Data Store + Lógica de Negócio + Interface de Usuário
  - É facilmente mapeado em uma implementação distribuída com comunicação via *remote procedure call*;
  - Jogos multiplayer em rede;
  - Sistemas web.

Configurações e Padrões Arquiteturais: principais padrões.

- **Model-View-Controller (MVC)**
  - Em sistemas Web é usado como a base do gerenciamento de interação;
  - Pode também ser visto como um padrão de projeto;
  - Promove a separação e, a consequente independência de desenvolvimento, da informação manipulada pelo programa e interações do usuário com esta informação;

Configurações e Padrões Arquiteturais: principais padrões.

FACVLDADE MAURÍCIO DE NASSAU
FAZENDO PARTE DA SUA HISTÓRIA

- **Model-View-Controller (MVC)**

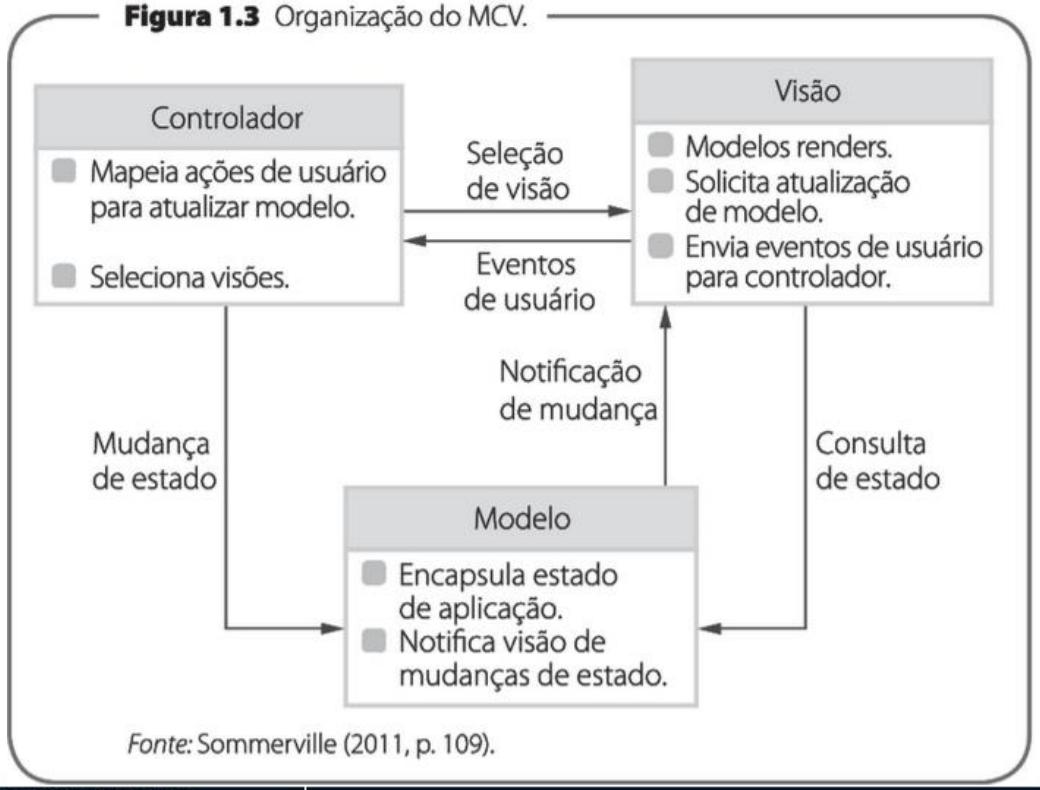
Nome	MVC (modelo-visão-controlador)
Descrição	Separa a apresentação e a interação dos dados do sistema. O sistema é estruturado em três componentes lógicos que interagem entre si. O componente modelo gerencia o sistema de dados e as operações associadas a esses dados. O componente visão define e gerencia como os dados são apresentados ao usuário. O componente controlador gerencia a interação do usuário (por exemplo, teclas, cliques do mouse etc.) e a passa para a visão e o modelo. Veja a Figura 1.3.
Exemplo	A Figura 1.4 mostra a arquitetura de um sistema aplicativo baseado na Internet organizado pelo uso do padrão MVC.
Quando é usado	É usado quando existem várias maneiras de se visualizar e interagir com dados. Também quando são desconhecidos os futuros requisitos de interação e apresentação de dados.
Vantagens	Permite que os dados sejam alterados de forma independente de sua apresentação, e vice-versa. Apoia a apresentação dos mesmos dados de maneiras diferentes, com as alterações feitas em uma representação aparecendo em todas elas.
Desvantagens	Quando o modelo de dados e as interações são simples, pode envolver código adicional e complexidade de código.

Fonte: Sommerville (2011, p. 109).

Configurações e Padrões Arquiteturais: principais padrões.

- **Model-View-Controller (MVC)**

Figura 1.3 Organização do MCV.

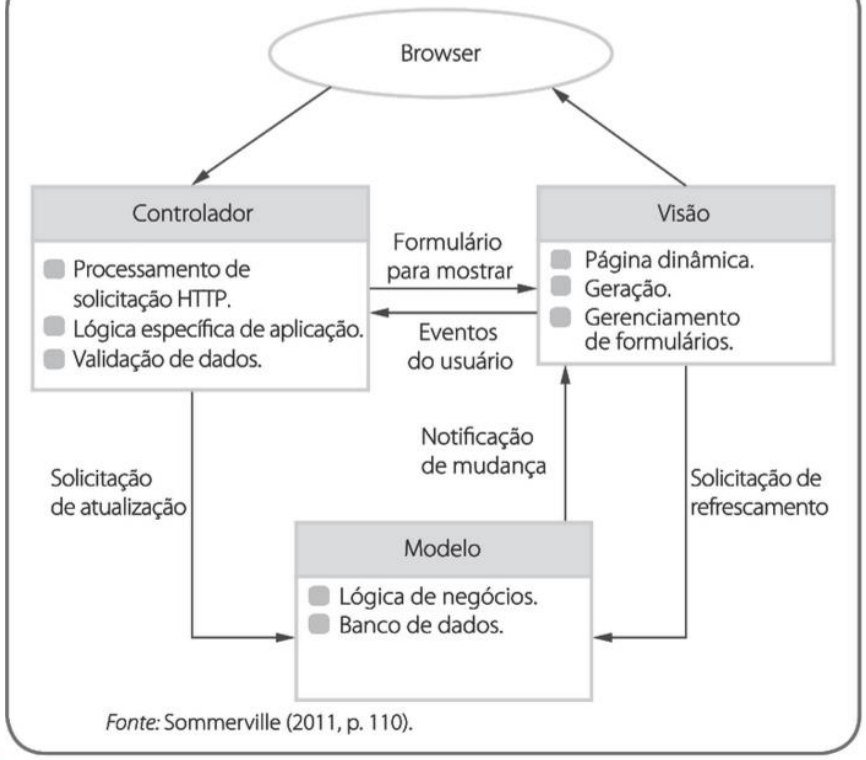


Fonte: Sommerville (2011, p. 109).

Configurações e Padrões Arquiteturais: principais padrões.

- **Model-View-Controller (MVC)**

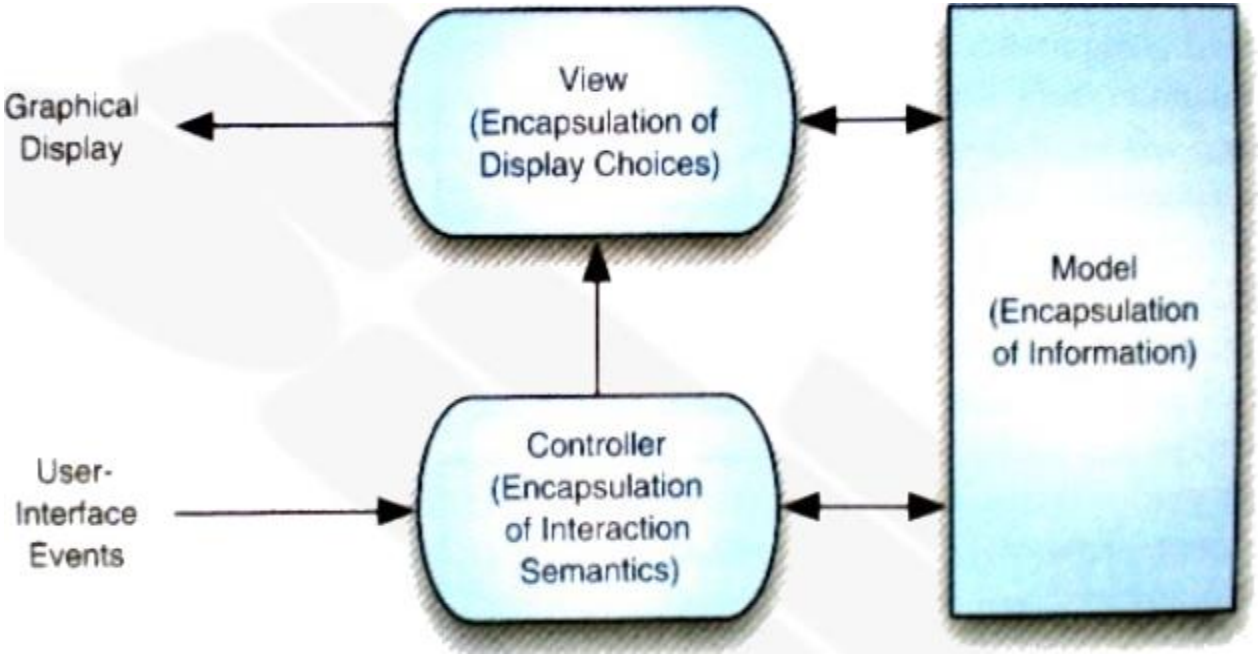
Figura 1.4 Arquitetura de aplicações Web utilizando o padrão MCV.



Fonte: Sommerville (2011, p. 110).

Configurações e Padrões Arquiteturais: principais padrões.

- **Model-View-Controller (MVC)**



Configurações e Padrões Arquiteturais: principais padrões.

- **Model-View-Controller (MVC)**
  - **Model**: encapsula a informação usada pela aplicação.
  - **View**: encapsula artefatos necessários à descrição gráfica da informação.
  - **Controller**: encapsula a lógica necessária à manutenção da consistência entre o model e o view. É responsável pelo processamento dos eventos do usuário.

Configurações e Padrões Arquiteturais: principais padrões.

- **Model-View-Controller (MVC)**
  - Colaborações entre os componentes (variações são consideradas na prática):
  - Quando a aplicação modifica um valor no model uma notificação é enviada à(s) view(s) de modo que a representação gráfica seja atualizada e exibida novamente;
  - Notificações também podem ser enviadas ao controller, que pode modificar a view se necessário;
  - A view pode solicitar dados adicionais ao model;
  - O sistema de janelas envia os eventos do usuário ao controller que pode consultar a view, obtendo informações que ajudam a determinar a ação a ser tomada;
  - Como consequência o controller atualiza o model.

Configurações e Padrões Arquiteturais: principais padrões.

- **Model-View-Controller (MVC)**
  - Geralmente existe um acoplamento forte entre as ações da view e do controller, eventualmente justificando um merge destes componentes;
  - MVC na World Wide Web:
    - **Model**: recursos web;
    - **View**: agente de renderização HTML do browser;
    - **Controller**: parte do browser que responde aos eventos do usuário e que interaje com o servidor web ou modifica o que é exibido no browser. Pode também agregar código obtido do servidor web (ex: JavaScript).

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura em Camadas**
  - Cada camada sujeita-se somente a recursos da que está logo abaixo dela – formando uma espécie de cadeia que se incrementa;
  - Uma camada pode ser substituída por outra, sem alterar sua interface, e mesmo quando a camada de interfaces muda, somente a camada paralela deve sofrer alterações;

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura em Camadas**

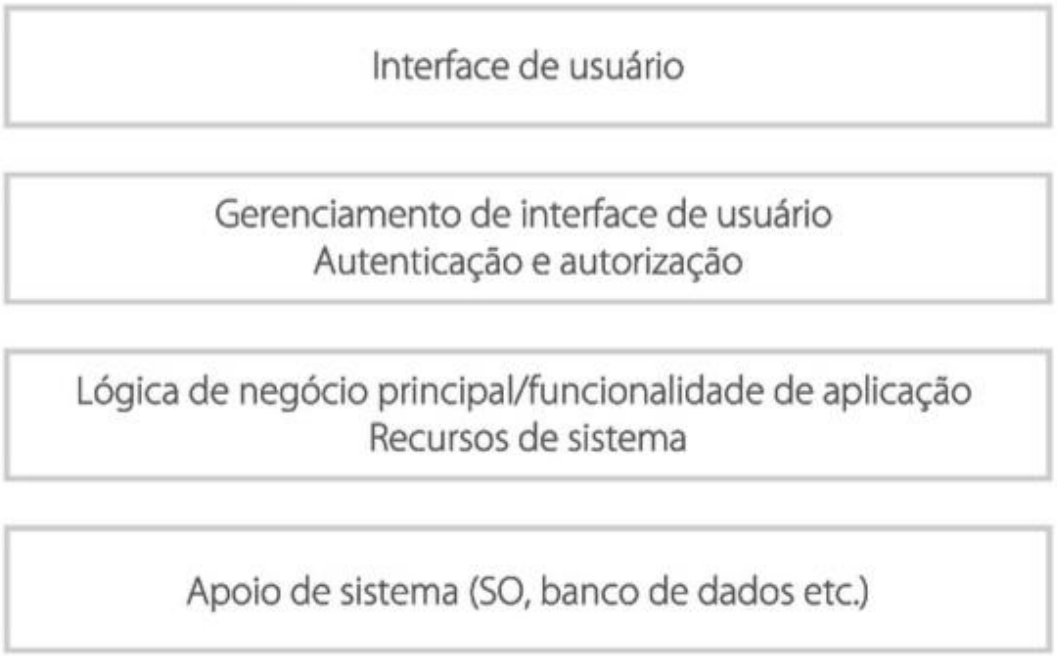
Nome	Arquitetura em camadas
Descrição	Organiza o sistema em camadas com a funcionalidade relacionada associada a cada uma delas. Uma camada fornece serviços à camada acima dela; assim, os níveis mais baixos de camadas representam os principais serviços suscetíveis de serem usados em todo o sistema. Veja a Figura 1.5.
Exemplo	Um modelo em camadas de um sistema para compartilhar documentos com direitos autorais, em bibliotecas diferentes, como mostrado na Figura 1.6.
Quando é usado	É usado na construção de novos recursos em cima de sistemas existentes; quando o desenvolvimento está espalhado por várias equipes, com a responsabilidade de cada equipe em uma camada de funcionalidade; quando há um requisito de proteção multinível.
Vantagens	Desde que a interface seja mantida, permite a substituição de camadas inteiras. Recursos redundantes (por exemplo, autenticação) podem ser fornecidos em cada camada para aumentar a confiança do sistema.
Desvantagens	Na prática, costuma ser difícil proporcionar uma clara separação entre as camadas, e uma de alto nível pode ter de interagir diretamente com camadas de baixo nível, em vez da imediatamente abaixo dela. O desempenho pode ser um problema por causa dos múltiplos níveis de interpretação de uma solicitação de serviço, uma vez que são processados em cada camada.

Fonte: Sommerville (2011, p. 110).

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura em Camadas**

Figura 1.5 Exemplo de uma arquitetura em camadas.

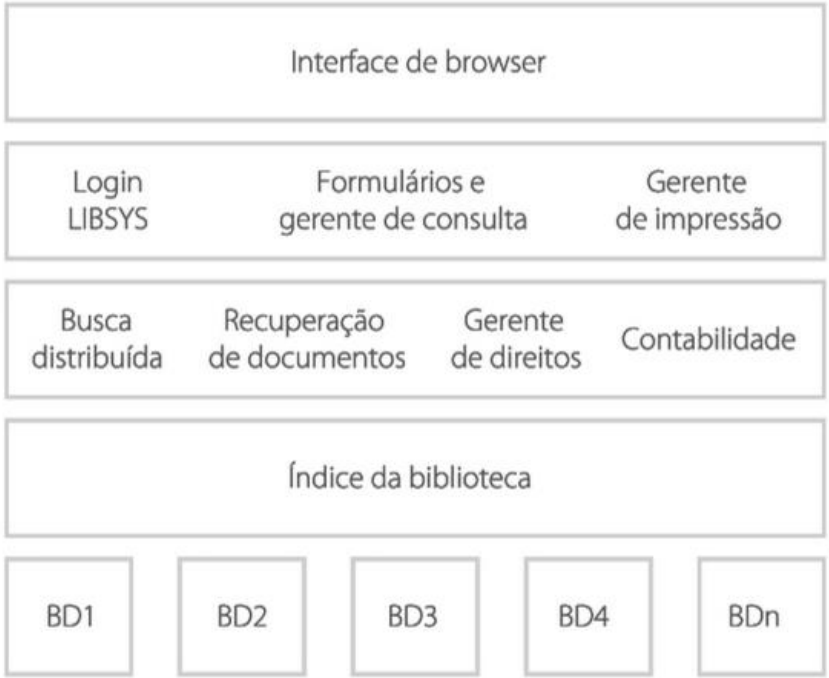


Fonte: Sommerville (2011, p. 111).

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura em Camadas**

Figura 1.6 Arquitetura do sistema LIBSYS.



Fonte: Sommerville (2011, p. 111).

Configurações e Padrões Arquiteturais: principais padrões.

FACVLDADE MAURÍCIO DE NASSAU
FAZENDO PARTE DA SUA HISTÓRIA

- **Arquitetura de Repositório**
  - Como um conjunto de componentes pode interagir para compartilhar dados entre si?
  - Importante para organizar estruturas em que os dados são gerados por um componente, mas utilizados por outros.
    - Ex: Sistemas CAD, de informação gerencial, de comando e controle.

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura de Repositório**

Nome	Repositório
Descrição	Todos os dados em um sistema são gerenciados em um repositório central, acessível a todos os componentes do sistema. Os componentes não interagem diretamente, apenas por meio do repositório.
Exemplo	A Figura 1.7 é um exemplo de um IDE em que os componentes usam um repositório de informações sobre projetos de sistema. Cada ferramenta de software gera informações que ficam disponíveis para uso por outras ferramentas.
Quando é usado	Você deve usar esse padrão quando tem um sistema no qual grandes volumes de informações são gerados e precisam ser armazenados por um longo tempo. Você também pode usá-lo em sistemas dirigidos a dados, nos quais a inclusão dos dados nos repositórios dispara uma ação ou ferramenta.
Vantagens	Os componentes podem ser independentes – eles não precisam saber da existência de outros componentes. As alterações feitas a um componente podem se propagar para os demais. Todos os dados podem ser gerenciados de forma consistente (por exemplo, backups feitos ao mesmo tempo), pois tudo está em um só lugar.
Desvantagens	O repositório é um ponto único de falha, assim, problemas nele podem afetar todo o sistema. Pode haver ineficiências na organização de toda a comunicação por intermédio do repositório. Distribuir o repositório a partir de vários computadores pode ser difícil.

Fonte: Sommerville (2011, p. 112).

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura de Repositório**

Figura 1.7 Exemplo de arquitetura de repositório para um IDE.

Fonte: Sommerville (2011, p. 112).

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura Cliente-Servidor**
  - Se organiza a partir de um conjunto de serviços e servidores associados em relação a clientes que acessam e utilizam esses serviços. E portanto, facilmente extensível;
  - Segundo Sommerville:
    1. Conjunto de servidores que oferecem serviços a outros componentes. Exemplos de servidores incluem: servidores de impressão que oferecem serviços de impressão; servidores de arquivos que oferecem serviços de gerenciamento de arquivos; e um servidor de compilação, que oferece serviços de compilação de linguagens de programação;
    2. Um conjunto de clientes que podem chamar os serviços oferecidos pelos servidores. Em geral, haverá várias instâncias de um programa cliente executando simultaneamente em computadores diferentes.
    3. Uma rede que permite aos cliente acessar esses serviços. A maioria dos sistemas cliente-servidor é implementada como sistemas distribuídos, conectados através de protocolos de internet.

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura Cliente-Servidor**

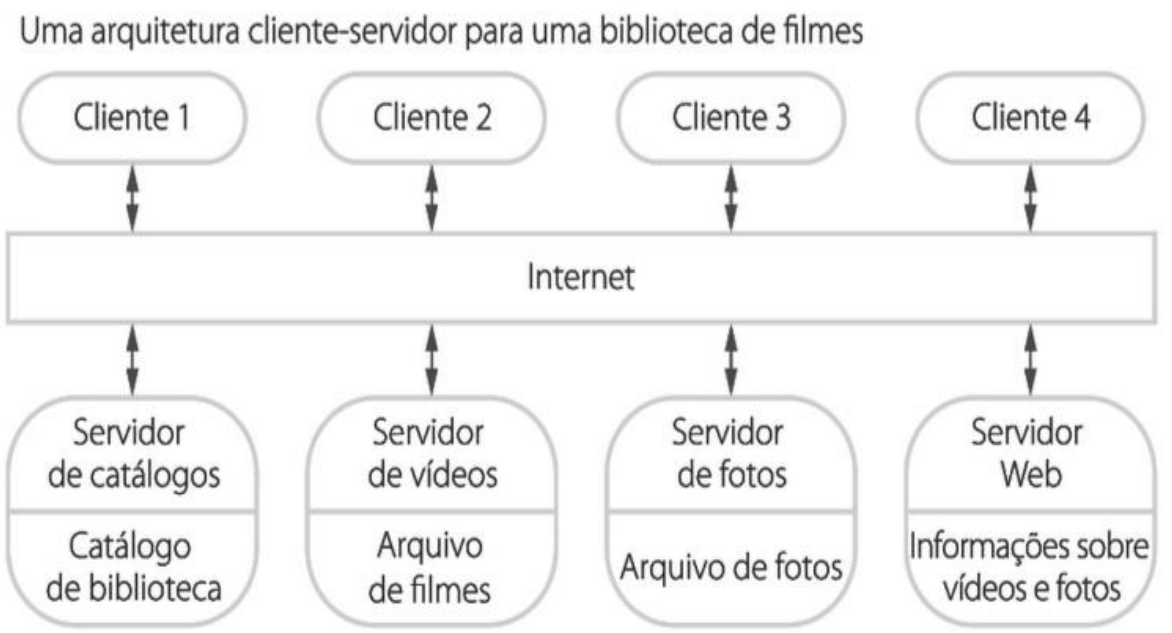
Nome	Cliente-servidor
Descrição	Em uma arquitetura cliente-servidor, a funcionalidade do sistema está organizada em serviços – cada serviço é prestado por um servidor. Os clientes são os usuários desses serviços e acessam os servidores para fazer uso deles.
Exemplo	A Figura 1.8 é um exemplo de uma biblioteca de filmes organizados como um sistema cliente-servidor.
Quando é usado	É usado quando os dados em um banco de dados compartilhado precisam ser acessados a partir de uma série de locais. Como os servidores podem ser replicados, também pode ser usado quando a carga em um sistema é variável.
Vantagens	A principal vantagem desse modelo é que os servidores podem ser distribuídos por meio de uma rede. A funcionalidade geral (por exemplo, um serviço de impressão) pode estar disponível para todos os clientes e não precisa ser implementada por todos os serviços.
Desvantagens	Cada serviço é um ponto único de falha suscetível a ataques de negação de serviço ou falha do servidor. O desempenho, bem como o sistema, pode ser imprevisível, pois depende da rede. Pode haver problemas de gerenciamento se os servidores forem propriedade de diferentes organizações.

Fonte: Sommerville (2011, p. 113).

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura Cliente-Servidor**

Figura 1.8 Exemplo do padrão cliente-servidor.



Fonte: Sommerville (2011, p. 113).

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura de Duto e Filtro**
  - Termos originários do sistema UNIX.
    - Dutos: Transfere um fluxo de dados de um processo ao outro;
    - Filtros: A cada transformação são filtrados os dados que podem ser operados considerando os dados de entrada;
  - Cada etapa do processamento só se implementa como uma transformação. Existe uma sequência de processamento entre os componente (que também pode ser executada paralelamente), e, enquanto os dados vão passando por ela, eles se transformam;
  - Segundo Sommerville (2011):
    - Quando as transformações são sequenciais com dados transformados em lotes, esse padrão arquitetural torna-se um modelo sequencial, comum para sistemas de processamento de dados.

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura de Duto e Filtro**

Nome	Duto e filtro
Descrição	O processamento dos dados em um sistema está organizado de modo que cada componente de processamento (filtro) seja discreto e realize um tipo de transformação de dados. Os dados fluem (como em um duto) de um componente a outro para processamento.
Exemplo	A Figura 1.9 é um exemplo de um sistema de duto e filtro usado para o processamento das faturas.
Quando é usado	Comumente, é usado em aplicações de processamento de dados (tanto as baseadas em lotes como as baseadas em transações) em que as entradas são processadas em etapas separadas para gerarem saídas relacionadas.
Vantagens	O reuso da transformação é de fácil compreensão e suporte. Estilo de *workflow* corresponde à estrutura de muitos processos de negócios. Evolução por adição de transformações é simples. Pode ser implementado tanto como um sistema sequencial quanto concorrente.
Desvantagens	O formato para transferência de dados tem de ser acordado entre as transformações de comunicação. Cada transformação deve analisar suas entradas e gerar as saídas para um formato acordado. Isso aumenta o *overhead* do sistema e pode significar a impossibilidade de reuso de transformações funcionais que adotam estruturas incompatíveis de dados.

Fonte: Sommerville (2011, p. 114).

Configurações e Padrões Arquiteturais: principais padrões.

- **Arquitetura de Duto e Filtro**

Figura 1.9 Exemplo do padrão duto e filtro.

Fonte: Sommerville (2011, p. 115).

Configurações e Padrões Arquiteturais: principais padrões.

FACVLDADE MAURÍCIO DE NASSAU
FAZENDO PARTE DA SUA HISTÓRIA

- **Exercícios de Fixação**
  1. O que é um padrão arquitetural?
  2. Faca um resumo em que constem as características básicas, quando são utilizados, vantagens e desvantagens dos seguintes padrões arquiteturais:
     a) MVC;
     b) Camadas;
     c) Repositório;
     d) Cliente-Servidor;
     e) Duto e Filtro;
  3. Explique o que são elementos de processamento.
  4. O que são elementos de dados ?