Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Computação (DACOM)
Curso de Bacharelado em Ciência da Computação

ARQUITETURA DE SOFTWARE: Estilos Arquiteturais

Prof. Dr. Alan Gavioli
alan@utfpr.edu.br

ESTILOS ARQUITETURAIS

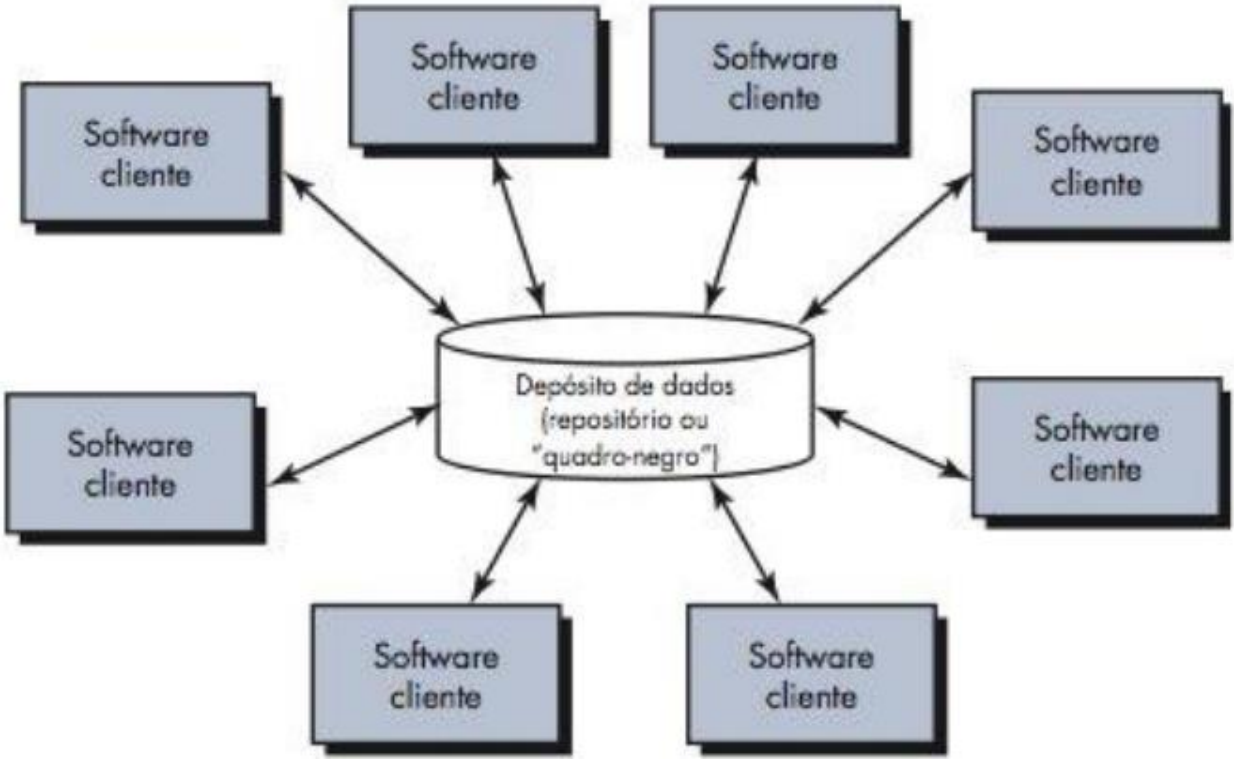
- No material estudado anteriormente, vimos que **estilos arquiteturais** definem um conjunto de regras de projeto que identificam **tipos de componentes**, seus **conectores** e **restrições** existentes para a sua composição, os quais podem ser usados para compor uma família de sistemas e subsistemas.

PRINCIPAIS ESTILOS ARQUITETURAIS

- Arquitetura em Camadas (detalhado no material anterior);
- Arquitetura Centralizadas em Dados;
- Arquitetura de Chamadas e Retornos;
- Arquitetura Cliente-servidor;
- Arquitetura de Fluxos de Dados.

ARQUITETURAS CENTRALIZADAS EM DADOS

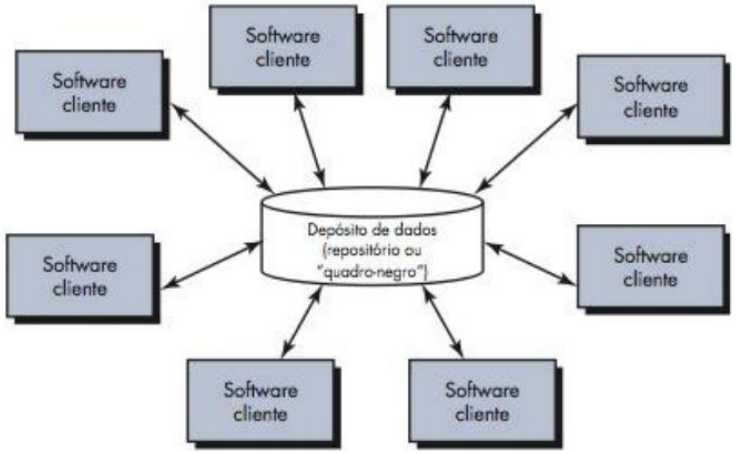
- Um repositório de dados (p. ex., um banco de dados ou arquivo) fica no centro da arquitetura e é acessado por outros componentes, que inserem, modificam ou eliminam dados do repositório (PRESSMAN, 2011).



ARQUITETURAS CENTRALIZADAS EM DADOS

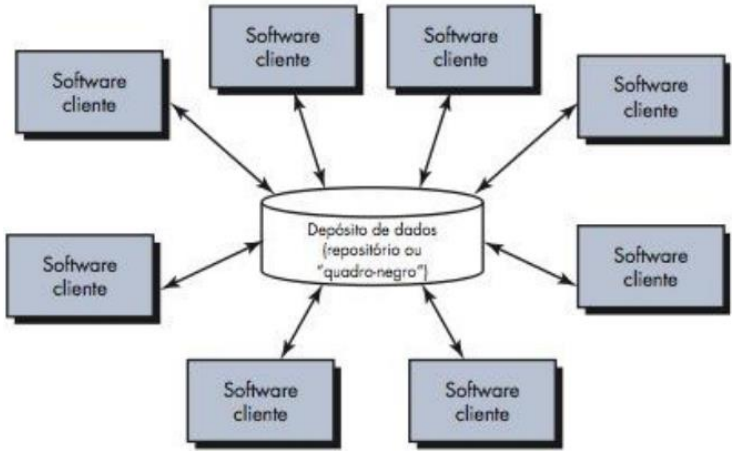
- Promove integrabilidade: clientes existentes podem ser alterados e novos componentes-clientes acrescentados à arquitetura sem preocupações, pois os componentes-clientes operam independentemente.

- Dados podem ser passados entre os clientes usando o mecanismo de quadro-negro (*blackboard*): este serve para coordenar a transferência de informações entre os clientes. Os componentes-clientes executam processos de maneira independente.



ARQUITETURAS CENTRALIZADAS EM DADOS

- Em alguns casos, o repositório de dados é passivo, ou seja, software-cliente acessa os dados independentemente de quaisquer alterações nos dados ou das ações de outros softwares-clientes.
- Uma variação dessa abordagem transforma o repositório em um "quadro-negro“, que envia notificações ao software-cliente quando os dados de seu interesse ficam disponíveis.

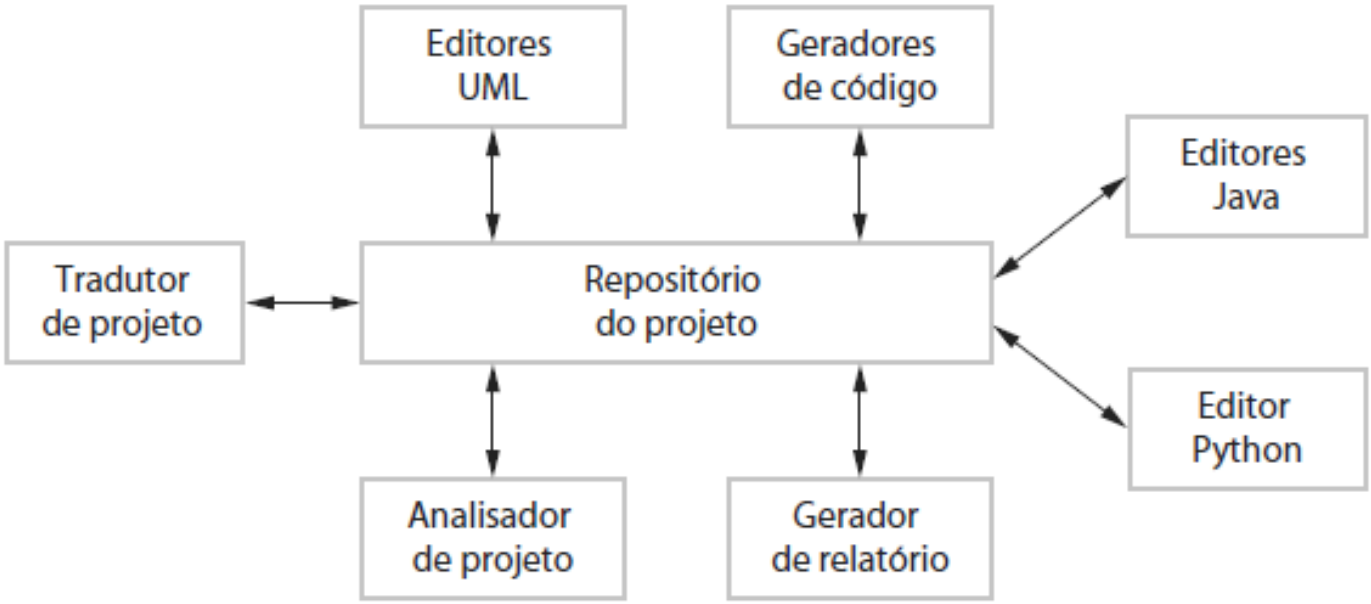


ARQUITETURAS CENTRALIZADAS EM DADOS

- Sommerville (2011) destaca que a maioria dos sistemas que usam grandes quantidades de dados é organizada em torno de um BD ou repositório compartilhado.
- “Arquitetura de Repositório” é adequada para aplicações em que os dados são gerados por um componente e usados por outro.
- Exs: sistemas de informações gerenciais, sistemas de CAD e IDEs.

ARQUITETURAS CENTRALIZADAS EM DADOS

- **Ex:** IDE que inclui ferramentas para suporte ao desenvolvimento dirigido a modelos. O repositório pode ser um ambiente de controle de versões, para gerenciar as alterações de software e possibilitar a reversão para versões anteriores.



ARQUITETURAS CENTRALIZADAS EM DADOS

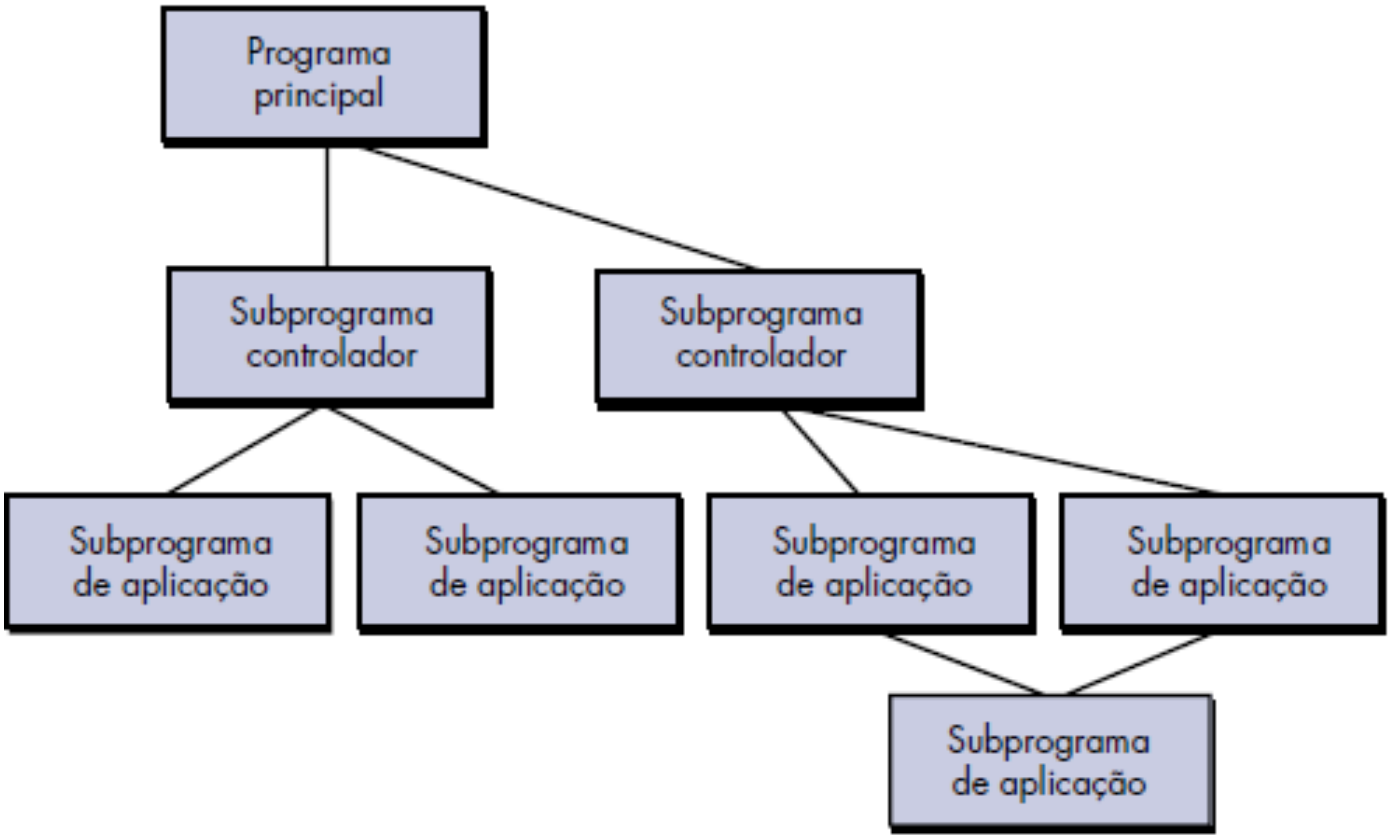
- **Vantagens:**
  - Os componentes podem ser independentes;
  - As alterações feitas a um componente podem propagar-se para todos os outros;
  - Todos os dados podem ser gerenciados de forma consistente, já que tudo está no mesmo repositório (ex: *backups* feitos ao mesmo tempo).
- **Desvantagens:**
  - Problemas no repositório podem afetar todo o sistema;
  - Possíveis ineficiências na organização de toda a comunicação através do repositório;
  - Dificuldade para distribuir o repositório através de vários computadores.

ARQUITETURAS DE CHAMADAS E RETORNOS

- **Segundo Pressman (2011), pode ser dividida em 2 subestilos:**
  - **Arquitetura de programa principal/subprograma:** clássica estrutura de programas, que decompõe a função em uma hierarquia de controle na qual um programa “principal” invoca uma série de componentes de programa que, por sua vez, podem invocar outros.
  - **Arquitetura de chamadas a procedimentos remotos:** os componentes de uma arquitetura de programa principal /subprogramas são distribuídos em vários computadores, em uma rede.

ARQUITETURAS DE CHAMADAS E RETORNOS

- **Ex. de arquitetura de programa principal/subprograma:**



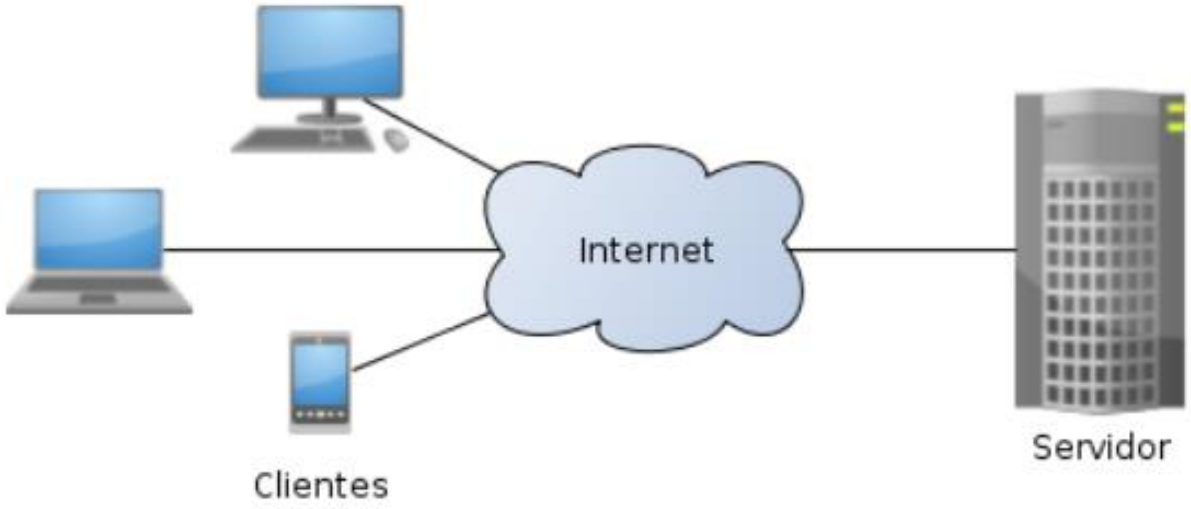
ARQUITETURA CLIENTE-SERVIDOR

- Muito usada para sistemas distribuídos;
- Na 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.
- Arquitetura usada quando os dados em um banco de dados compartilhado precisam ser acessados a partir de uma série de locais.

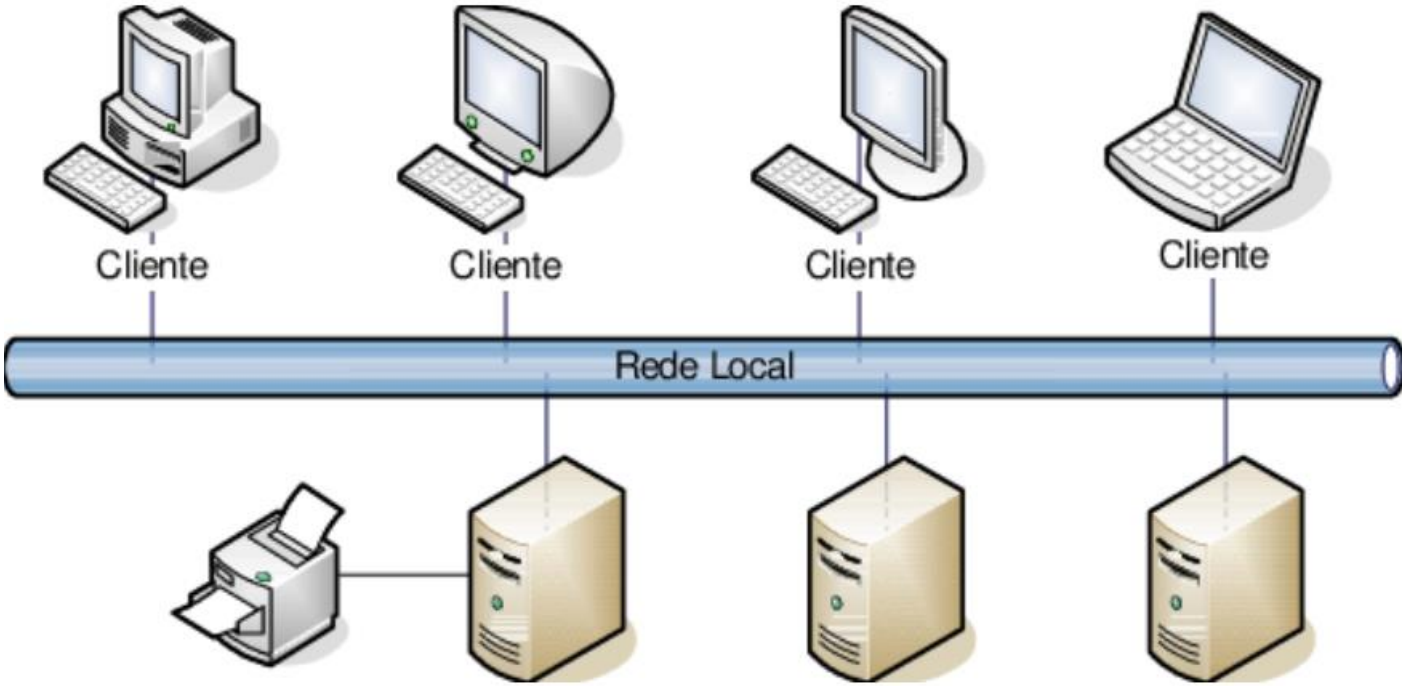
ARQUITETURA CLIENTE-SERVIDOR

- **Componentes dessa arquitetura:**
  - 1 - Conjunto de servidores para oferecer serviços a outros componentes. Exs.: servidores de impressão; servidores de arquivos que oferecem serviços de gerenciamento de arquivos; servidor de compilação, que oferece serviços de compilação de linguagens de programação.
  - 2 - Conjunto de clientes que podem chamar os serviços oferecidos pelos servidores. Normalmente, haverá várias instâncias de um programa cliente executando simultaneamente em computadores diferentes.
  - 3 - Rede que permite aos clientes acessar os serviços. A maioria dos sistemas cliente-servidor é implementada como sistemas distribuídos, conectados através de protocolos de Internet.

ARQUITETURA CLIENTE-SERVIDOR



ARQUITETURA CLIENTE-SERVIDOR



ARQUITETURA CLIENTE-SERVIDOR

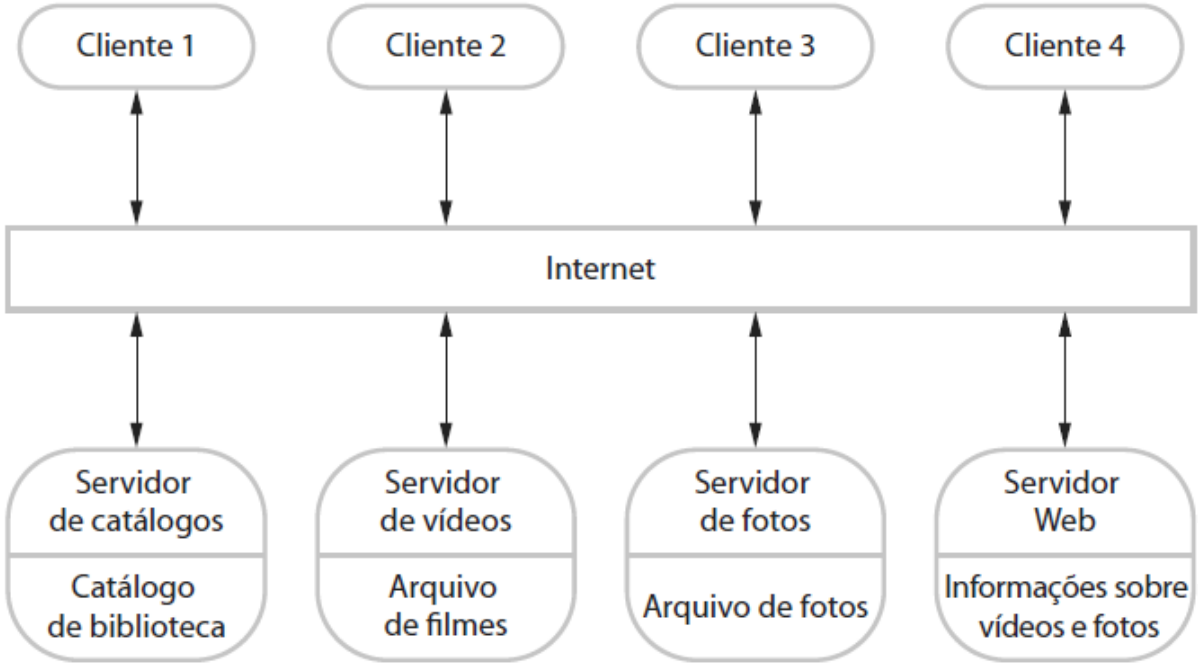
- Apesar de ser considerada uma arquitetura de sistemas distribuídos, o modelo lógico de serviços independentes rodando em servidores separados pode ser implementado em um único computador.
- Novamente, um benefício importante é a separação e a independência. Serviços e servidores podem ser modificados sem afetar outras partes do sistema.
- Os clientes podem ter de saber os nomes dos servidores e os serviços oferecidos; mas os servidores não precisam conhecer a identidade dos clientes ou quantos clientes estão acessando seus serviços.

ARQUITETURA CLIENTE-SERVIDOR

- Os clientes acessam os serviços de um servidor por meio de chamadas de procedimento remoto, usando um protocolo de solicitação-resposta, como o protocolo HTTP, usado na Internet.
- Essencialmente, um cliente faz uma solicitação a um servidor e espera até receber uma resposta.

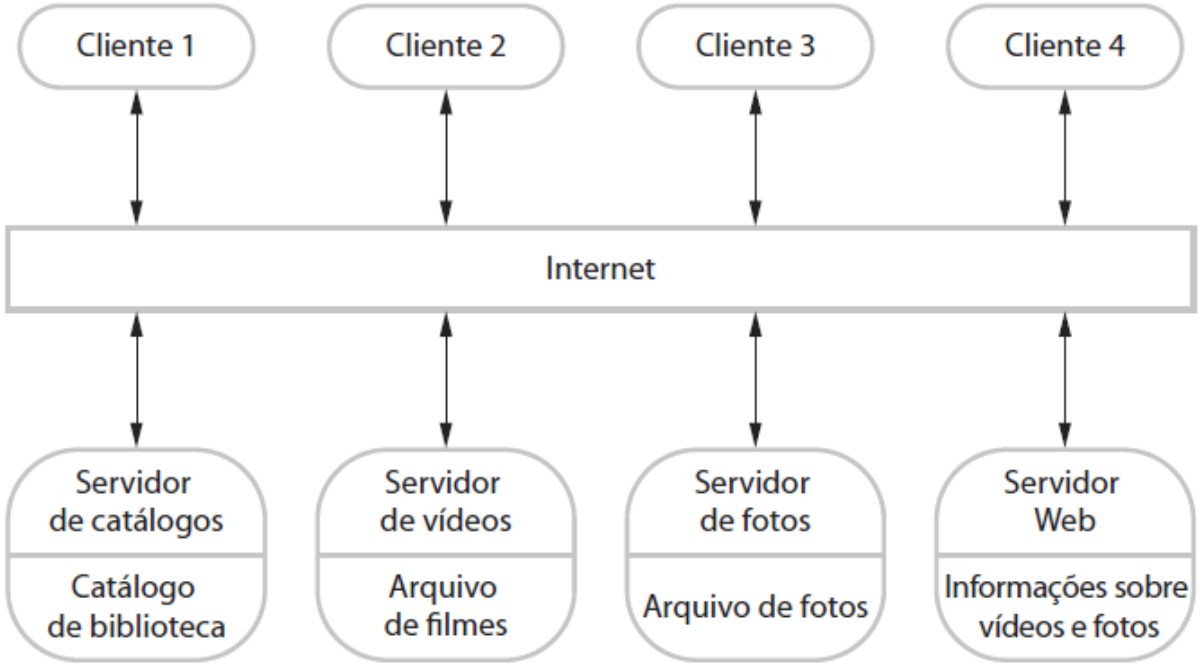
ARQUITETURA CLIENTE-SERVIDOR

- **Ex:** Sistema multiusuário baseado na Internet, para fornecimento de uma biblioteca de filmes e fotos. Nele, vários servidores gerenciam e apresentam os diferentes tipos de mídia. Os quadros de vídeo precisam ser transmitidos de forma rápida e em sincronia, mas em resolução relativamente baixa. Eles podem ser comprimidos em um arquivo, para que o servidor de vídeo possa lidar com a compressão e descompressão de vídeos. Mas as fotos devem permanecer em alta resolução; por isso, é interessante mantê-las em um servidor separado.



ARQUITETURA CLIENTE-SERVIDOR

- O catálogo deve ser capaz de lidar com uma variedade de consultas e fornecer links para o sistema de informação da Internet que incluam dados sobre os filmes e videoclipes, além de um sistema de comércio eletrônico que apoie a venda das fotografias, filmes e videoclipes.
- O programa do cliente é só uma interface de usuário integrada, construída usando-se um browser da Internet para acesso a esses serviços.

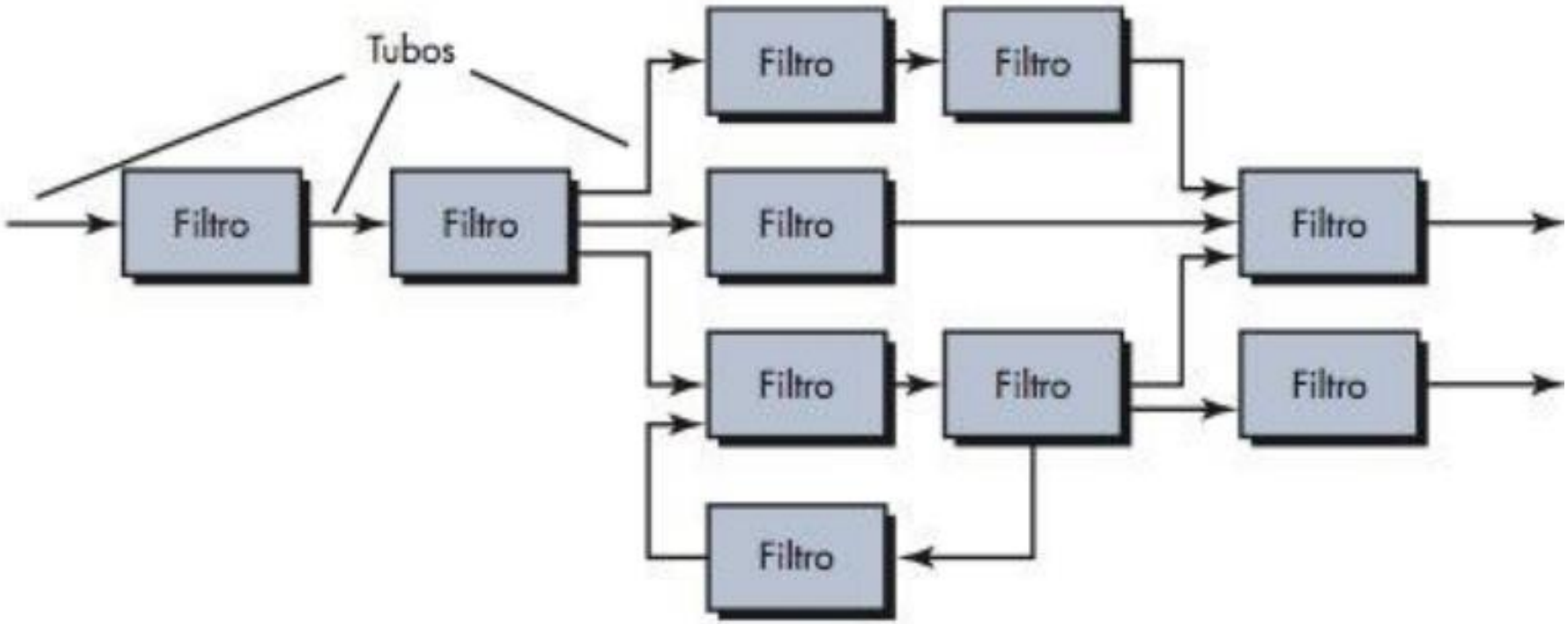


ARQUITETURA CLIENTE-SERVIDOR

- **Vantagens:**
  - A maior vantagem é que os servidores podem ser distribuídos em uma rede. A funcionalidade geral (p. ex., um serviço de impressão) pode estar disponível para todos os clientes e não precisa ser implementada por todos os serviços.
- **Desvantagens:**
  - Cada serviço está em um servidor, suscetível a ataques de negação de serviço ou de falha do servidor, que podem interromper o serviço.
  - O desempenho, bem como o sistema, pode ser imprevisível, pois depende da rede.
  - Podem ocorrer problemas de gerenciamento, caso os servidores sejam propriedade de diferentes organizações.

ARQUITETURAS DE FLUXO DE DADOS

- Segundo Pressman (2011), é útil quando dados de entrada devem ser transformados por meio de uma série de componentes computacionais em dados de saída.
- Um padrão tubos-e-filtro (*pipes-and-filters*) tem um conjunto de componentes, denominados filtros, conectados por tubos que transmitem dados de um componente para o seguinte.



ARQUITETURAS DE FLUXO DE DADOS

- **Cada filtro:**
  - É independente dos componentes anteriores e posteriores a ele;
  - É projetado para aguardar a entrada de dados de certa forma, e produzir saída para o filtro seguinte de uma determinada forma;
  - Não precisa conhecer o funcionamento de seus filtros vizinhos.

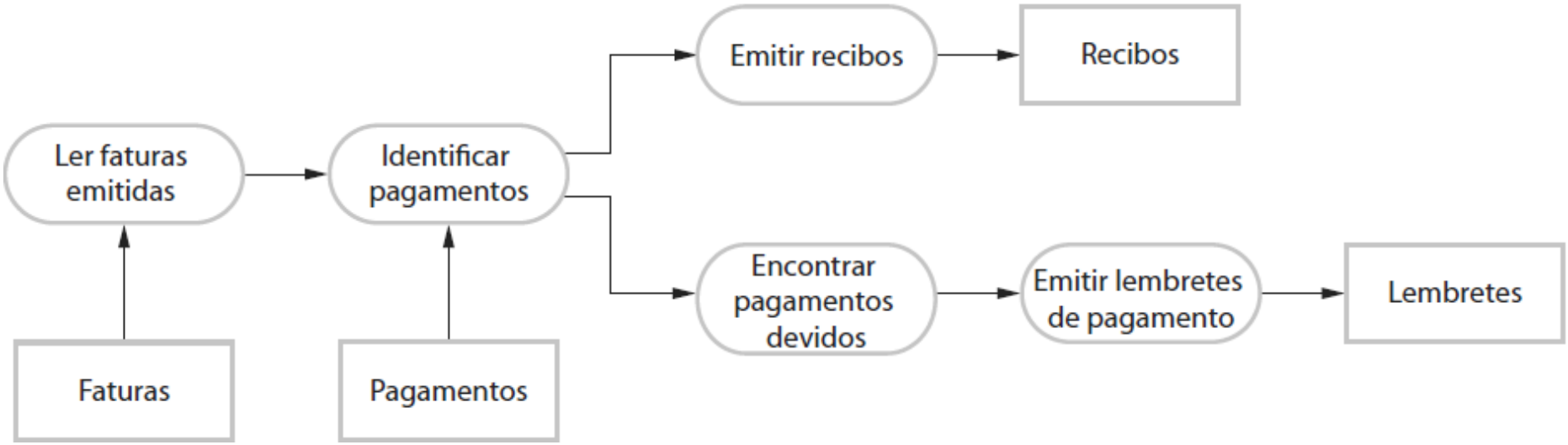
- Se o fluxo de dados ocorrer em uma única linha de transformações, é chamado de sequencial por lotes. Esta é uma arquitetura comum para sistemas de processamento de dados como, p. ex., sistemas de faturamento de pedidos; também é usada para implementação de compiladores de linguagens de programação.

ARQUITETURAS DE FLUXO DE DADOS

- **Ex:** Compilador de linguagem de programação:

ARQUITETURAS DE FLUXO DE DADOS

- **Ex:** Uma empresa emitiu faturas para os clientes. Semanalmente, os pagamentos realizados são conferidos com as faturas. Para cada pedido pago, um recibo é emitido; mas para as faturas que não foram pagas dentro do prazo, é emitido um lembrete:



ARQUITETURAS DE FLUXO DE DADOS

- **Vantagens:**
  - 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.
- **Desvantagens:**
  - 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 previamente combinado.

BIBLIOGRAFIAS UTILIZADAS

- PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: Uma Abordagem Profissional**. 8 ed. Porto Alegre: AMGH Editora Ltda, 2016.

- SOMMERVILLE, I. **Engenharia de Software**. 10 ed. São Paulo: Pearson, 2019.