ARQUITETURA DE SW

Modelos Arquiteturais

Análise e Desenvolvimento de Sistemas

Prof. Ms. Renato Leite
renato@sectras.edu.br

OBJETIVOS

- Revisão: O que define uma arquitetura?
- Entender o que é um modelo;
- Padrões arquiteturais;
- Exemplos de mercado.

O QUE DEFINE?

- Elementos:
  - de processamento – são eles que operam os dados, transformando suas entradas em saídas logicamente necessárias.
  - de dados – os dados que serão a matéria-prima a ser utilizada pelo sistema e processada.
  - de conexão – "amarram" os elementos na estrutura. Eles conectam qualquer forma de elemento entre si, de modo a formar um conjunto funcional operante.

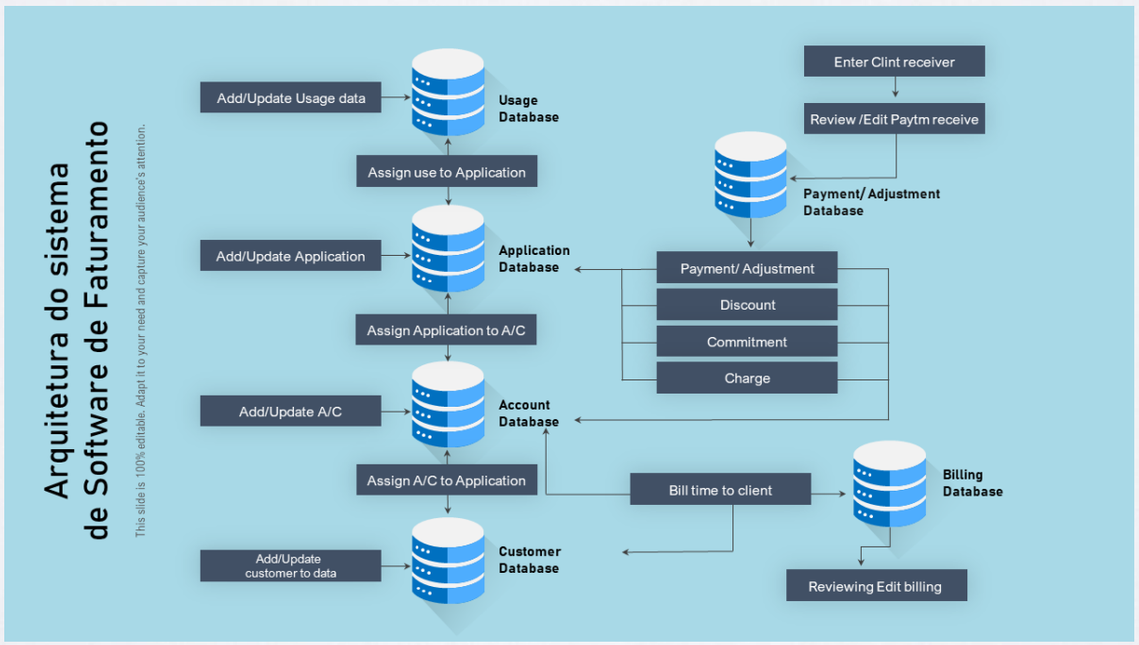
O QUE É UM MODELO?

- Modelo (nome masculino): "Imagem, desenho ou objeto que serve para ser imitado (desenhando ou esculpindo)."; "Molde, exemplar."; "[Figurado] Coisa ou pessoa que é ou merece ser imitada = exemplo." (Dicionário Priberam)

- Um modelo em Arquitetura de Software é uma representação abstrata e de exemplo, que define como os componentes de um sistema podem ser organizados e como interagem entre si.

PADRÕES ARQUITETURAIS

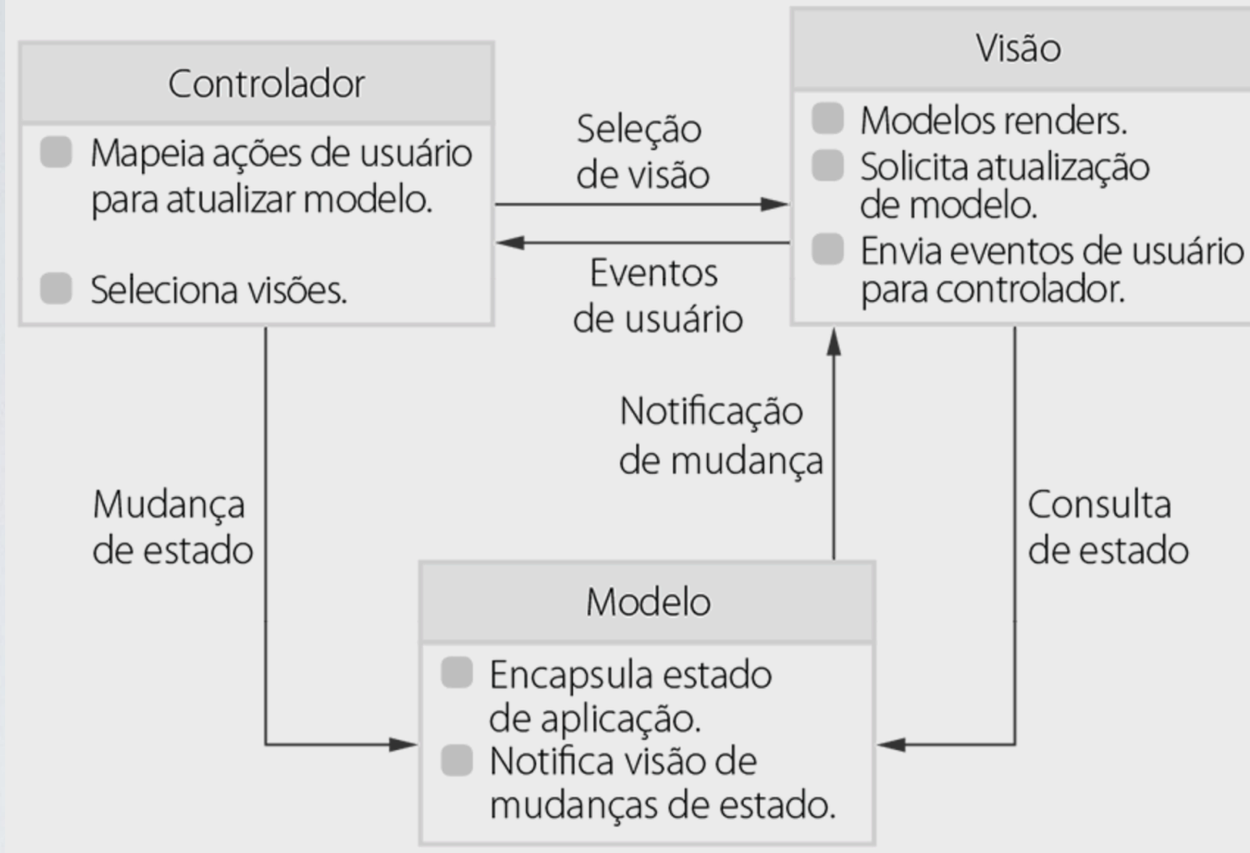
- São referências obtidas a partir de experiências bem sucedidas em sistemas já implementados;
- São diagramas e tabelas que apresentam de forma abstrata o que compõem uma arquitetura e suas descrições;

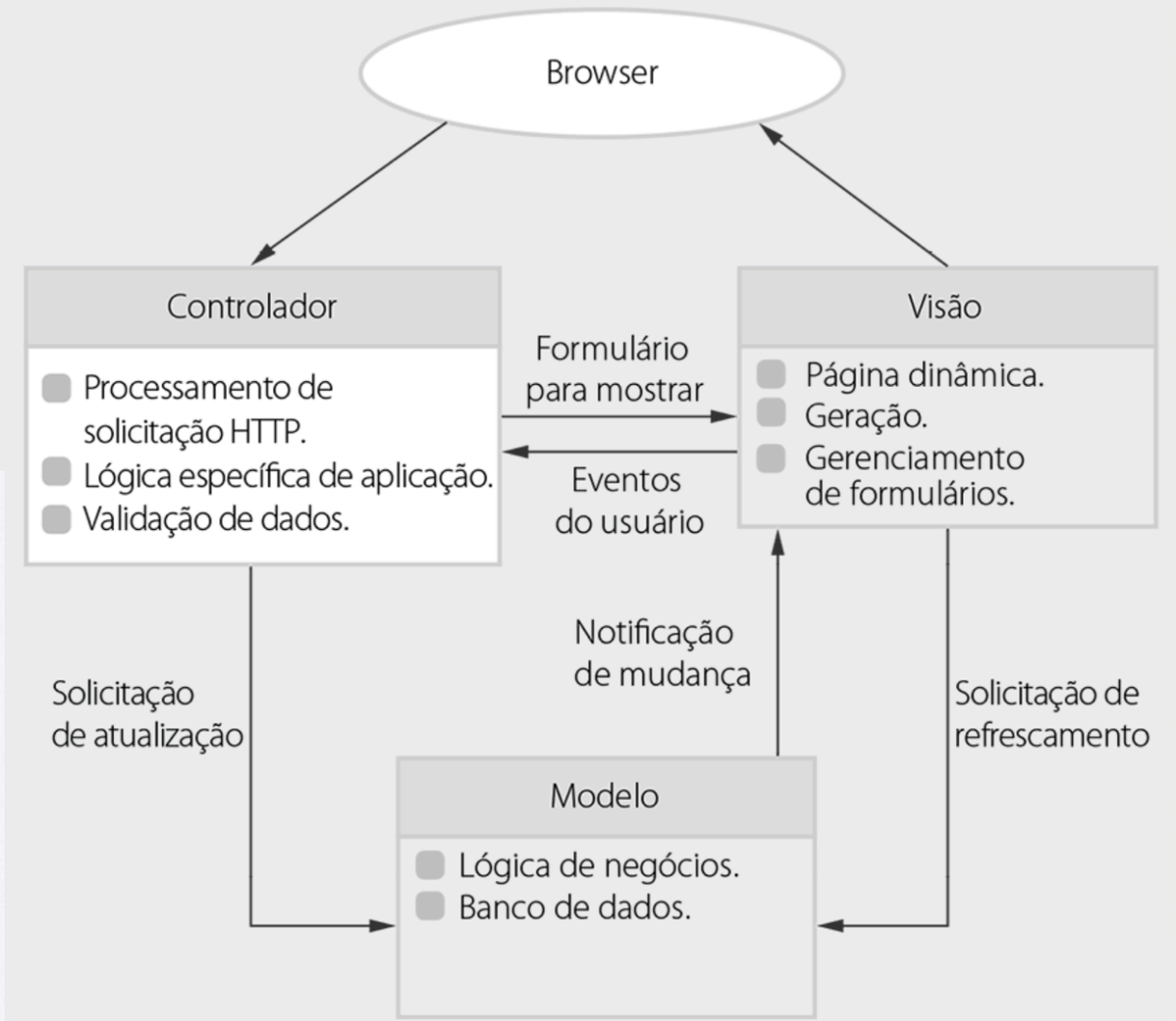


PADRÃO MVC

- 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.
- É usado quando existem várias maneiras de se visualizar e interagir com dados, ou 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.

PADRÃO MVC





Fonte: Sommerville (2011, p. 110).

PADRÃO LAYERED

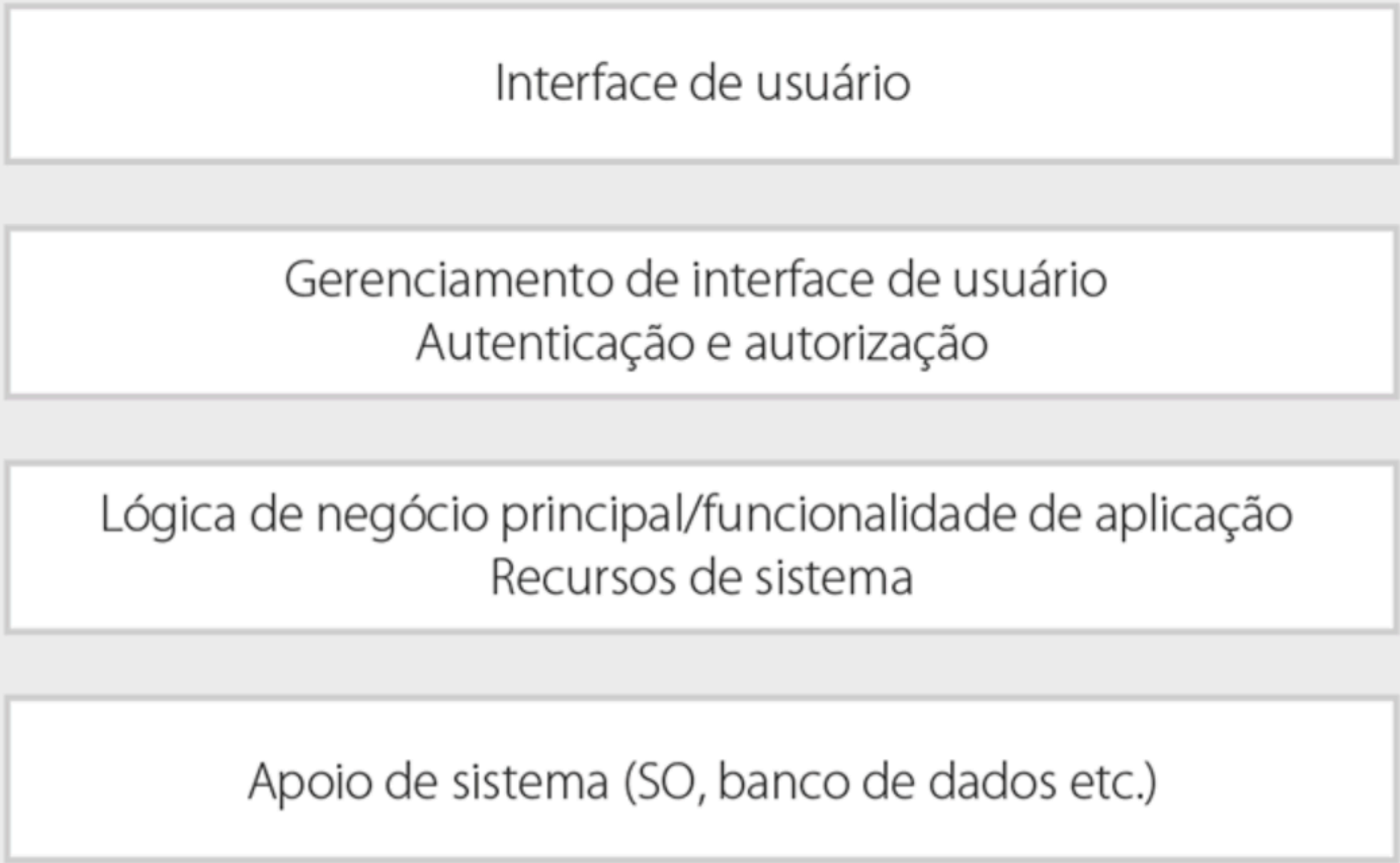
- Organiza o sistema em camadas com a funcionalidade relacionada associada a cada uma delas. Uma camada fornece serviços à camada acima dela; assim, os níveis mais baixos de camadas representam os principais serviços suscetíveis de serem usados em todo o sistema.

- 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.

PADRÃO LAYERED



Fonte: Sommerville (2011, p. 111).

PADRÃO REPOSITORY

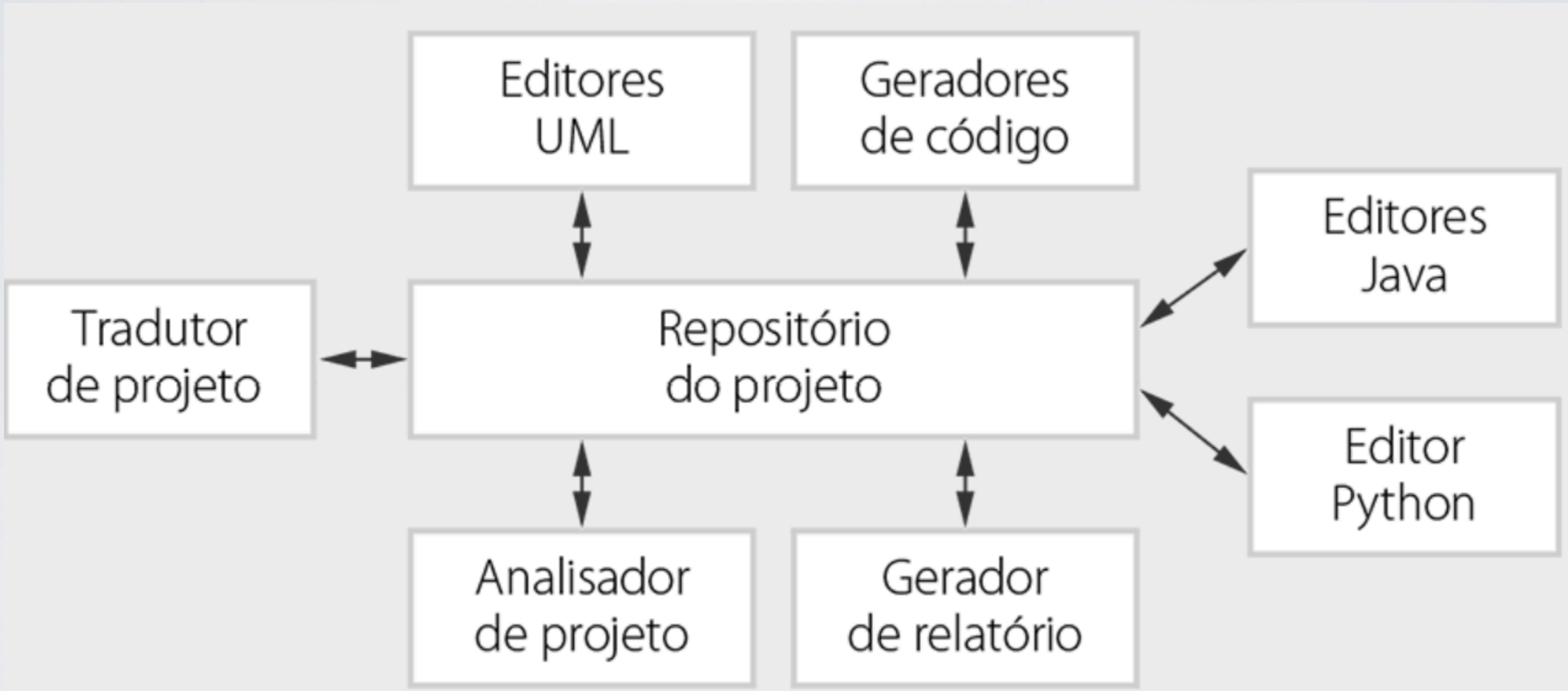
- Todos os dados em um sistema são gerenciados em um repositório central, acessível a todos os componentes do sistema. Os componentes não interagem diretamente, apenas por meio do repositório.

- 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. Gestão centralizada do dado.

- Desvantagens: O repositório é um **ponto único de falha**, **distribuir** o repositório a partir de vários computadores **pode ser difícil**.

PADRÃO REPOSITORY



Fonte: Sommerville (2011, p. 112).

PADRÃO CLIENT-SERVER

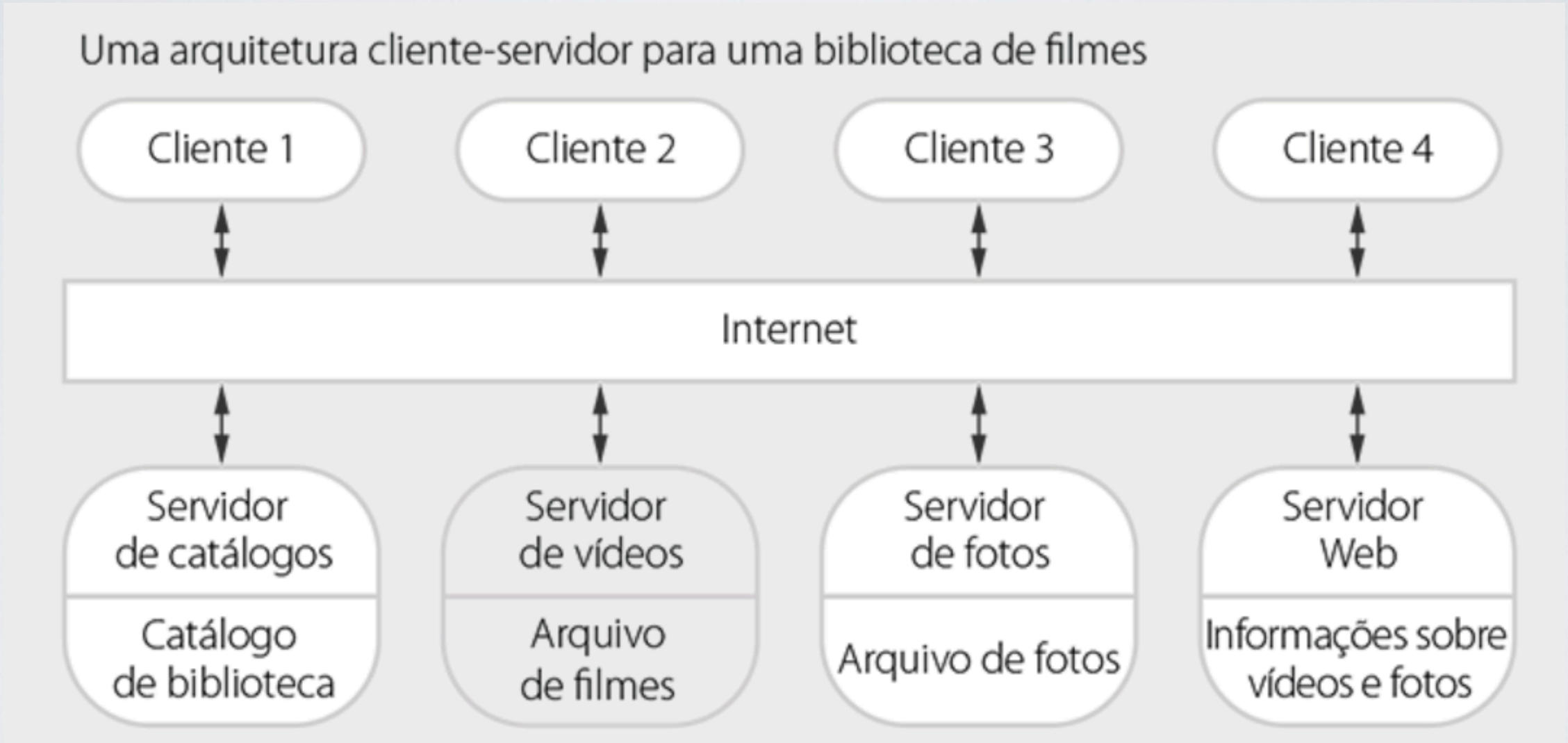
- Em uma arquitetura cliente-servidor, a funcionalidade do sistema está organizada em serviços – cada serviço é prestado por um servidor. Os clientes são os usuários desses serviços e acessam os servidores para fazer uso deles.

- 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.

- Vantagem: 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.

- Desvantagem: **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.

PADRÃO CLIENT-SERVER



PADRÃO DUCT-FILTER

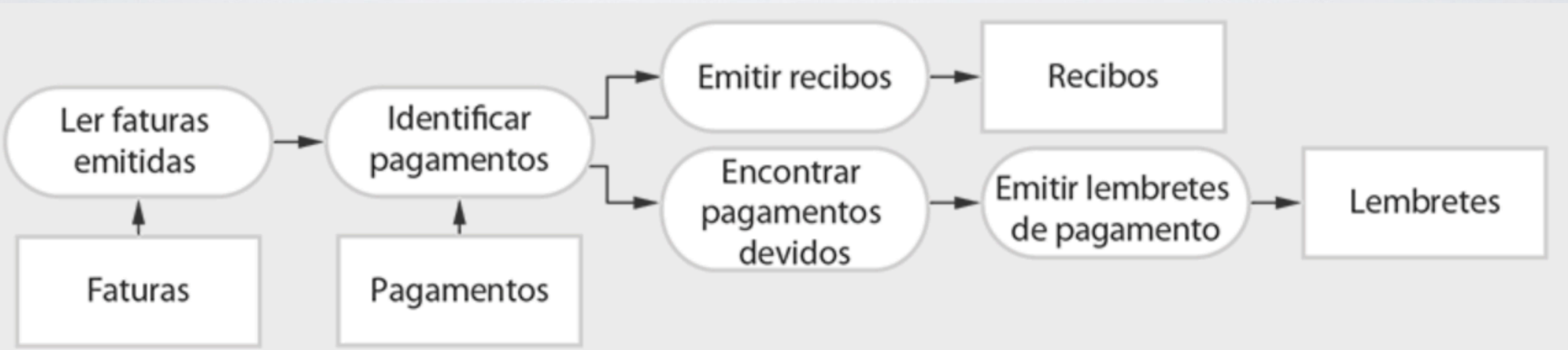
- O processamento dos dados em um sistema está organizado de modo que cada componente de processamento (filtro) realize um tipo de transformação de dados. Os dados fluem (como em um cano) de um componente a outro para processamento.

- Comumente, é usado em aplicações de transformação de dados (*ETL - Extract, Transform, Load*) 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.

PADRÃO DUCT-FILTER



Fonte: Sommerville (2011, p. 115).

NA PRÁTICA

Padrão Arquitetural	Exemplo de Mercado	Justificativa
MVC (Model–View–	Ruby on Rails – usado por GitHub, Shopify e Basecamp	Separa aplicação em Model, View e Controller, facilitando manutenção e testes.
Repository	Spring Data JPA – utilizado em bancos (Santander, Itaú)	Encapsula o acesso a dados, permitindo trocar tecnologia de persistência sem impactar o domínio.
Layered (Arquitetura em Camadas)	Internet Banking – Bradesco, Itaú, Caixa	Organiza responsabilidades em apresentação, aplicação, domínio e infraestrutura, garantindo escalabilidade.
Client–Server	WhatsApp	Cliente envia requisições ao servidor central que processa e redireciona para outros clientes.
Pipe-and-Filter (duct-filter)	Apache Spark – usado por Netflix, Uber, Amazon	Processa dados em pipelines, onde cada filtro transforma o fluxo e os pipes conectam as etapas.

BIBLIOGRAFIA

- SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011

- GALLOTTI, G. M. A. Arquitetura de Software. 1ª Ed. São Paulo: Pearson, 2016. [acervo digital]

- JUNIOR, A. K. Computação em Nuvem. 1ª Ed. Curitiba: Contentus, 2020. [acervo digital]

- ZANETTI, M. Arquitetura de TI e modelos de negócios. 1ª Ed. Curitiba: Contentus, 2020. [acervo digital]

DÚVIDAS?