Universidade Federal de Viçosa
Instituto de Ciência Exatas e Tecnológicas
CCF 321 - Projeto de Sistemas Para Web

Trabalho 01

Arquiteturas de Software

Pedro Cardoso de Carvalho Mundim Matrícula: 3877

Professor:
Jeverson Ricardo Nery Silva dos Santos

19 de maio de 2022

1 Resumo sobre Arquitetura de Software

A arquitetura de software diz respeito à estrutura interna de um sistema. Dessa forma, é possível observar que, quando há mudanças ocorrendo em um software, a sua arquitetura também sofrerá mudanças. Essas mudanças são importantes pois, a evolução do projeto espera que o software seja flexível, extensível, portável e reutilizável.

1.1 Os elementos da Arquitetura de Software

No modelo proposto por Perry e Wolf [Gallotti, 2016], é dito que a arquitetura se baseia em como os seus elementos se organizam, ou seja, ela pode ser vista como um conjunto de elementos, organização e decisões. Dentro da organização, são estabelecidos padrões para que os elementos se relacionem uns com os outros. Esses elementos são os seguintes:

- Elementos de processamento: operam os dados.
- Elementos de dados: matéria-prima utilizada pelo sistema.
- Elementos de conexão: conectam os elementos entre si.

No exemplo da Figura 1 [Gallotti, 2016], podemos observar um sistema de informações fictício baseado na Netflix, o qual gerencia locações de vídeos e realiza streaming, via Web.

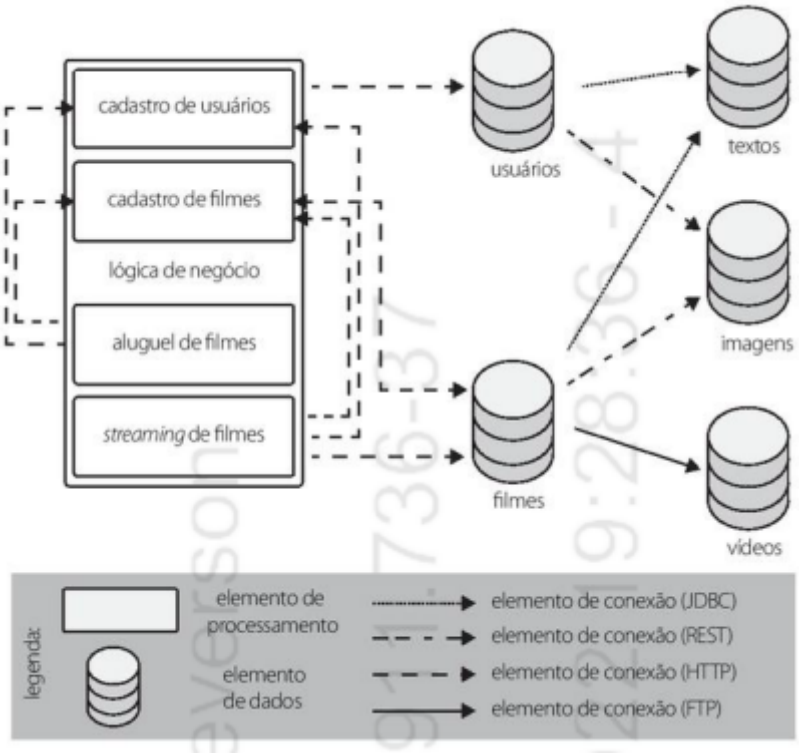

Figura 1: Exemplo da organização dos elementos. [Gallotti, 2016]

2 Configurações e Padrões Arquiteturais

Um padrão arquitetural irá descrever relações entre a organização dos elementos, anteriormente comentados, que deram certo em sistemas que já foram desenvolvidos. Vejamos alguns desses padrões.

2.1 Arquitetura MVC

O **padrão MVC** ou Model-view-controller é estruturado em três componentes que interagem entre si.

- **Model** irá gerenciar o sistema de dados.
- **View** irá definir como esses dados serão apresentados ao usuário.
- **Controller** irá gerenciar a interação com o usuário.

Mas como ocorre a interação entre esses componentes? Começa com a interação do usuário na camada view. Então, o controlador coleta essas informações e as envia para o model que ficará responsável por avaliar aqueles dados e transmitir uma resposta.

Algumas das **vantagens** do padrão MVC são: segurança, organização, eficiência, tempo, dentre outras. Ou seja, é possível alterar os dados sem se preocupar com sua apresentação; o desenvolvedor ficará com maior facilidade em entender o que e como foi construído o software, facilitando assim a correção de erros; o projeto se torna escalável, dentre outros fatores.

No entanto, também existem **desvantagens** no MVC. Uma delas é que, quando o modelo de dados e as interações são simples, isso pode acabar envolvendo código adicional e de maior complexidade.

Vejamos na Figura 2 [Gallotti, 2016] um exemplo de um sistema que utiliza o padrão MVC.

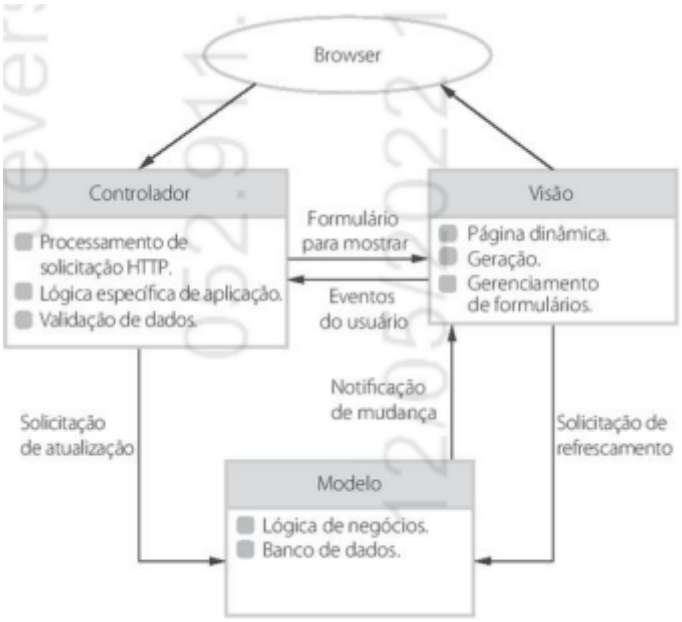

Figura 2: Arquitetura de aplicações Web com MVC.[Gallotti, 2016]

Temos nesta figura um exemplo prático de MVC, em que ele é utilizado em sistemas baseados em Web para gerenciar interações.

2.2 Arquitetura em Camadas

Vimos que o MVC aplica dois conceitos importantes quando estamos falando de arquitetura. Estes são a separação e a independência. No entanto, há outras arquiteturas que fazem o mesmo. A **arquitetura em camadas** é uma delas. Nesse padrão, cada camada depende dos recursos que estão na camada abaixo dela, ou seja, é uma arquitetura incrementável. Nesse sentido, a arquitetura em camadas visa a criação de aplicativos modulares, fazendo com que uma camada seja dependente apenas da camada imediatamente abaixo.

Esse tipo de arquitetura é utilizado, por exemplo, quando o desenvolvimento está dividido dentro das equipes. Uma de suas **vantagens** é a separação do código, em que é possível substituir camadas inteiras. Entretanto também existem **desvantagens** nesse padrão. Uma delas é que, normalmente é difícil proporcionar um boa separação das camadas.

Vejamos na Figura 3 [Gallotti, 2016] um exemplo de um sistema que utiliza o padrão em camadas.

Figura 3: Arquitetura do sistema LIBSYS.[Gallotti, 2016]

Neste exemplo, temos a possibilidade de que um grupo de bibliotecas acesse materiais protegidos. O LIBSYS [Gallotti, 2016] é dividido, como mostra a figura, em cinco camadas. Na camada mais baixa se encontram os bancos de dados, seguidos pelo índice na camada superior.

2.3 Arquitetura de Repositório

Neste modelo de arquitetura, todos os dados do sistema são gerenciados por um **repositório**. Os componentes não interagem diretamente, apenas por meio deste repositório. Sistemas baseados na arquitetura de repositório normalmente contam com volumes significativos de dados. O repositório entra em jogo para auxiliar no compartilhamento desses dados.

Algumas **vantagens** desse tipo de arquitetura: os componentes podem ser independentes; gerenciamento centralizado de atividades; facilidade de integração com outras ferramentas; dentre outros. Por outro lado, temos algumas **desvantagens**: os subsistemas devem concordar com o modelo de dados do repositório; a evolução dos dados é difícil e cara; há uma dificuldade em distribuir o repositório em uma série de máquinas; dentre outros.

Vejamos na Figura 4 [Gallotti, 2016] um exemplo de um sistema que utiliza o padrão de repositório.

Figura 4: Exemplo de arquitetura de repositório para um IDE.[Gallotti, 2016]

Neste exemplo, o repositório atua como um ambiente controlado. As alterações feitas no

programa ficam sob controle, o que torna possível retornar para versões antigas caso seja necessário.

2.4 Arquitetura Cliente-servidor

A **arquitetura cliente-servidor** é baseada em sistemas distribuídos por tempo de execução. Ela mostra como os dados e o processamento são distribuídos em uma série de processadores. Ou seja, esse padrão está organizado em serviços, nos quais os clientes os acessam para fazer uso dos mesmos.

Os componentes mais importantes dessa arquitetura são:

- Um conjunto de servidores.
- Um conjunto de clientes.
- Uma rede.

Nessa arquitetura, o cliente faz uma solicitação para algum dos servidores e aguarda sua resposta.

Algumas **vantagens** dessa arquitetura são: é uma arquitetura distribuída, o que permite uso efetivo do sistema; é fácil incluir um novo servidor; o hardware normalmente não é caro; dentre outros. Algumas **desvantagens** são: não há modelo de dados compartilhado; gerenciamento de cada servidor pode ser redundante; o desempenho pode ser imprevisível.

Vejamos na Figura 5 [Gallotti, 2016] um exemplo de um sistema que utiliza o padrão cliente-servidor.

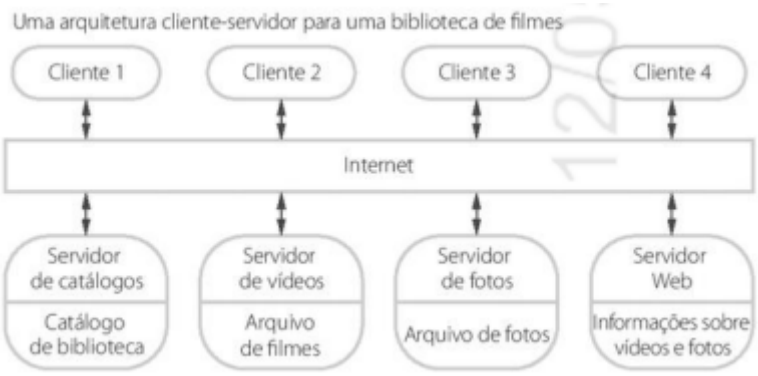

Figura 5: Exemplo do padrão cliente-servidor.[Gallotti, 2016]

Este exemplo ilustra uma biblioteca de filmes e fotos organizados com padrão cliente-servidor.

2.5 Arquitetura de Duto e Filtro

A **arquitetura de duto e filtro** se organiza também em tempo de execução. É composto por uma cadeia de elementos de processamento. A saída de cada elemento é a entrada do próximo. Os filtros (programas) são conectados por meio de dutos que agem como buffers, armazenando a saída de um filtro enquanto ela não é lida pelo próximo filtro.

Os termos duto e filtro são originários do sistema Unix. Um exemplo clássico do padrão duto e filtro é o shell do Linux.

Algumas **vantagens** dessa arquitetura: o reuso da transformação é de fácil compreensão e suporte; evoluções são simples; pode ser um sistema sequencial ou corrente; dentre outras. Algumas **desvantagens** são: o formato de dados tem que ser de acordo com as transformações de comunicação; cada transformação deve analisar as entradas e gerar saídas em formato de acordo; dentre outras.

Vejamos na Figura 6 [Gallotti, 2016] um exemplo de um sistema que utiliza o padrão de duto e filtro.

Figura 6: Exemplo do padrão duto e filtro.[Gallotti, 2016]

Nesse exemplo, uma loja emitiu as faturas de seus clientes. A cada semana, as faturas são conferidas de acordo com os pagamentos feitos. É então emitido um recibo. Caso alguma fatura ainda não esteja paga, mas dentro do prazo, é emitido um lembrete.

3 Minha Experiência com Arquiteturas no Curso

Em relação à minha experiência com padrões de arquitetura no curso, posso citar principalmente o uso do MVC, durante a disciplina de Programação Orientada a Objetos, visto tanto na teoria quanto na prática.

Também tive contato teórico com a Arquitetura em Camadas e Cliente-servidor na disciplina de redes de computadores.

Durante todo o curso, também "utilizei"a Arquitetura de repositório, principalmente para salvar e gerenciar os trabalhos práticos das disciplinas. A principal ferramenta utilizada nesse propósito foi o GitHub.

Referências

[Gallotti, 2016] Gallotti, G. (2016). Arquitetura de software. *Publicadora Pearson Education do Brasil Ltda.*