Anotações de ES3A3

Aula 03 – Padrões e Diagramas de Arquitetura (Aula 03)

Padrões de Arquitetura: É um conceito criado em meados da década de 1990, com o nome de “Estilos de Arquitetura”. No entanto, atualmente, os termos “padrão” e “estilo” não representam distinções significativas. Ou seja, são basicamente “sinônimos”.

Recapitulação *(com base no documento de anotações anterior)*
Padrão de Arquitetura: é uma descrição abstrata de como será a organização de todos os componentes do sistema, fornecendo orientações e boas práticas para sua construção. Além disso, um sistema, dependendo de seu tamanho e contexto, poderá utilizar mais de um padrão de arquitetura.

Adendo do professor sobre o trecho marcado: atualmente, a maioria dos sistemas usa vários padrões de arquitetura ao mesmo tempo.

Descrição de um Padrão de Arquitetura: Ao elaborar um documento que visa descrever um Padrão de Arquitetura, é necessário abordar alguns tópicos: nome do padrão, descrição conceitual, exemplo do tipo de sistema que pode utilizar o padrão, vantagens e desvantagens e etc.

Padrão de Arquitetura - Divisão em Camadas: nessa organização, o sistema é organizado em diversas camadas. No entanto, cada uma depende dos recursos e serviços do seu nível (camada) inferior. Além disso, cada camada é responsável por um conjunto específico de funcionalidades.

Geralmente, a divisão em camadas é composta por três camadas principais:

1. **Camada de Apresentação (ou interface de usuário):** Essa camada é responsável pela interação do usuário e apresentação dos dados ao mesmo. Geralmente, consiste na interface gráfica do usuário ou linha de comando.
2. **Camada de aplicação (ou camada lógica de negócio):** Essa camada é responsável pela lógica de negócio do sistema. Ela processo solicitações do usuário e executa funções essenciais do sistema. Normalmente ela inclui validações, regras de negócio e processamento de dados.
3. **Camada de Dados:** Essa camada é responsável pelo armazenamento e acesso dos dados do sistema.

Adendo do professor: As camadas podem ser divididas de várias formas. Essa é uma delas.

Padrão de Arquitetura — Repositório: representa um conjunto de componentes, que interagem entre si, e podem compartilhar dados entre si.

Observação: sistemas que utilizam grande quantidade de dados são organizados em torno de banco de dados ou repositórios compartilhados. Arquitetura ideal para aplicações em que um componente gera dados que são usados por outros componentes. Presumo que serviços como github utilizam arquiteturas do tipo repositório.
Adendo do professor: Na verdade, o Github seria o próprio repositório. Ele pode ser usado como tal em uma arquitetura de repositório, conforme o exemplo mencionado em aula, das IDEs conectadas a seu repositório de projetos.

A desvantagem dessa arquitetura é que, dependendo da modificação ~~de determinado componente~~ (modificação no repositório)*, essa mesma modificação pode afetar componentes dependentes (como relatado em uma questão da lista de exercício da aula 03).*

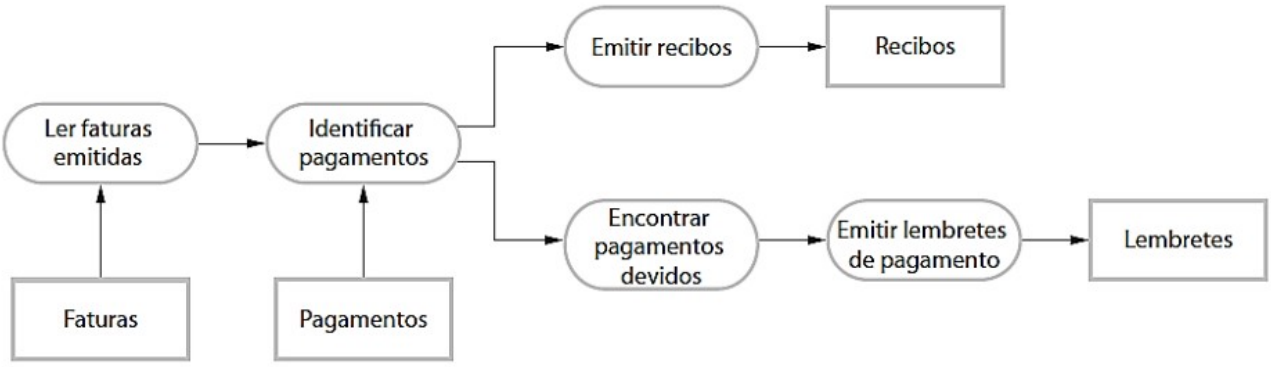
Padrão de Arquitetura — Cliente Servidor: representa sistemas distribuídos, onde suas funcionalidades são disponibilizadas por servidores. Amplamente utilizado na internet.
Tal arquitetura se organiza em 3 componentes:

1. Servidores: oferecem serviços de páginas web, de banco de dados e aplicação.
2. Clientes: que realizam chamadas aos serviços oferecidos pelos servidores.
3. Rede: permitem que os clientes acessam tais serviços oferecidos através de protocolos de comunicação.

Observação: Uma vantagem dessa arquitetura é que podemos ter um serviço X, em um determinado servidor, que pode ser disponibilizado a diversos clientes; sem a necessidade desse serviço ser implementado em outros servidores (ou em todas as máquinas clientes).
A desvantagem, no entanto, pode ser a vantagem relatada anteriormente, tendo em vista que os serviços podem estar armazenados em servidores únicos, facilitando ataques. Além disso, essa arquitetura é extremamente dependente da rede.

Padrão de Arquitetura — Dutos e Filtros: Em uma arquitetura de dutos e filtros, os dutos são canais que transportam dados de um filtro para outro, enquanto os filtros são componentes que processam ou transformam os dados de entrada em dados de saída.

Observação: cada filtro, ou componente, é projetado para receber – via dutos - um tipo específico de dado e, a partir dele, realizar modificações necessárias para o próximo filtro – lembra o processo de "decantação". Além disso, cada filtro trabalha de forma independente, sem a necessidade de conhecer funcionalidades de outros filtros, ou seja, aplicando o conceito de independência funcional.
A desvantagem dessa arquitetura é que cada processamento deve ~~processar~~ (validar – verificar se as entradas estão no formado esperado) o formato de suas entradas, via dutos, e gerar saídas previamente definidas.



O primeiro filtro lê as faturas emitidas para os clientes. Uma vez por semana, o filtro seguinte compara os pagamentos realizados com as faturas emitidas. Um terceiro filtro emite um recibo para cada fatura paga. Em paralelo, outro filtro detecta as faturas que não foram pagas dentro do prazo e, na sequência, um filtro emite um lembrete para essas faturas.

Diagramas: A UML oferece diferentes representações gráficas que podemos utilizar para representar pontos específicos de uma determinada arquitetura, sendo eles: diagrama de comunicação, de pacotes, de componentes e de implantação.

1. **Diagrama de Comunicação:** Representa a visão de processo da Arquitetura. Permite posicionar os participantes livremente, desenhar vínculos para mostrar como eles se conectam e usar números para mostrar a sequência de mensagens.
2. **Diagrama de Pacote:** Auxilia na representação de sistemas complexos, que utilizam muitas classes. Nesse diagrama classes são

agrupadas em pacotes.

3. **Diagrama de Componentes:** Representa os relacionamentos dos componentes por meio de interfaces. Interfaces são “contratos” que definem como algo externo pode fazer uso do componente.
4. **Diagrama de Implantação**: mostra o layout físico do sistema, exibindo quais partes do software é executado em determinada parte do hardware.