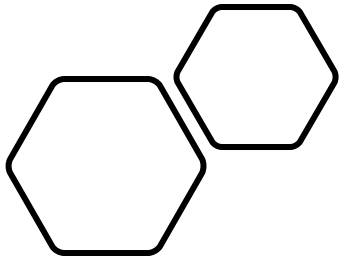
Padrões Arquiteturais Clássicos

Prof. Bernardo Copstein

Introdução a arquitetura de software



Leituras recomendadas:

- Martin, Robert C.. Clean Architecture (Robert C. Martin Series) (p. 2). Pearson Education. Edição do Kindle.
- Sommerville, Ian. Engineering Software Products – An Introduction to Modern Software Engineering. Pearson, 2019.

Relembrando: porque a arquitetura de um software é importante?

Somerville

- Influencia as propriedades não funcionais de um software
  - Responsividade
  - Confiabilidade
  - Disponibilidade
  - Segurança
  - Usabilidade
  - **Manutebilidade**
  - Resiliência

Robert Martin

- Minimizar a quantidade de recursos necessários para desenvolver e **manter** o sistema

Martin: Software produz dois tipos de valores para o stakeholder:

Comportamento: faz o que o stakeholder espera que ele faça da melhor forma possível

Arquitetura: permite que o software evolua com facilidade

- Correções
- Alterações
- Novas funcionalidades

Relembrando:
o projeto
arquitetural

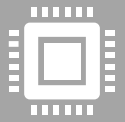
Envolve compreender as questões que irão afetar à arquitetura de um produto em particular

Envolve conflitos; impossível resolver todos

- Manutebilidade x performance
- Segurança x usabilidade
- Disponibilidade x custo e “time to Market”

Do começo

Arquitetura de software começa com código



Codificar implica conhecer os paradigmas de programação

Paradigmas de programação são formas de programar relativamente não relacionadas a linguagens

Um paradigma de programação define que estruturas usar e quando usá-las

Três paradigmas

Estruturado

- Impõem disciplina sobre a transferência direta de controle
- Fluxo de controle de um programa

Orientado a objetos

- Impõem disciplina sobre a transferência indireta de controle
- Polimorfismo

Funcional

- Impõe disciplina sobre as atribuições
- Imutabilidade das variáveis
- Alterações seguem uma disciplina rígida
- Orienta a separação de componentes imutáveis e alteráveis
  - Especialmente útil em aplicações concorrentes
  - Evita problemas de condições de corrida
- Armazenar apenas transações ao invés de variáveis modificáveis

Paradigmas x Arquiteturas

Usa-se polimorfismo como mecanismo para cruzar os limites arquiteturais

Usa-se programação funcional para impor disciplina na localização e acesso aos dados

Usa-se programação estruturada como base algorítmica dos módulos

Em resumo → preocupações de qualquer arquitetura de software:

- Funções
- Separação de componentes
- Gerenciamento de dados

Padrões arquiteturais clássicos

Camadas

MVC

Pipes and filters

Broker

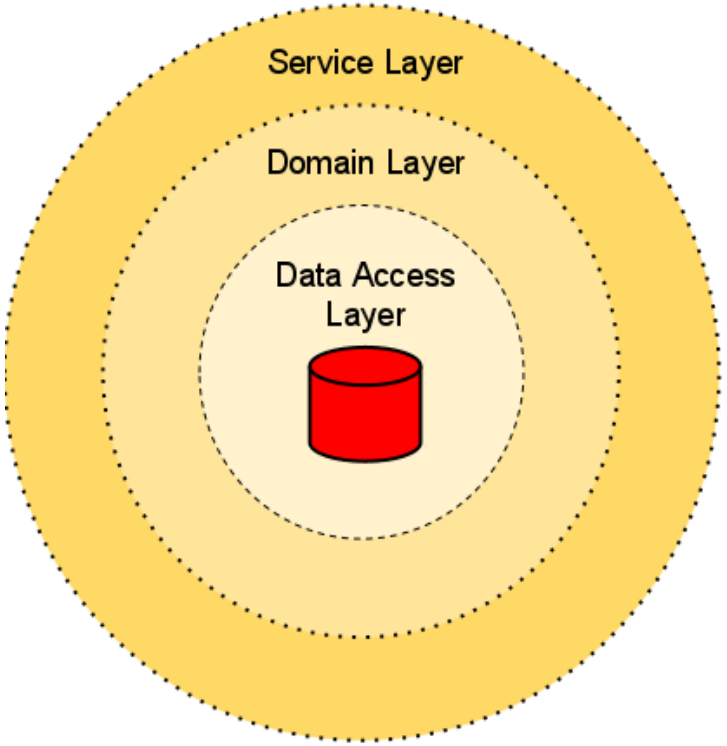
Games

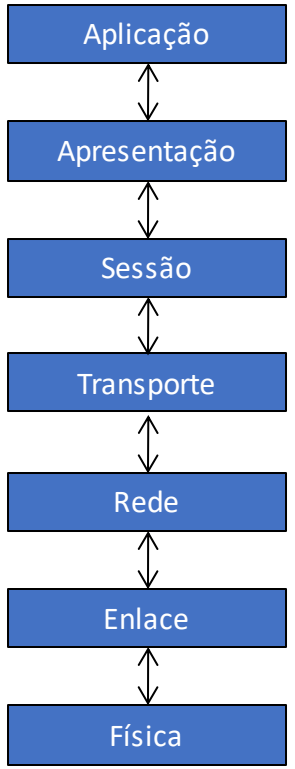
Sistemas inteligentes

Padrão arquitetural camadas

- **Descrição**: auxilia a estruturar aplicações que podem ser decompostas em grupos de subtarefas, nas quais cada grupo de subtarefas é um nível particular de abstração.

Aplicação coorporativa em camadas



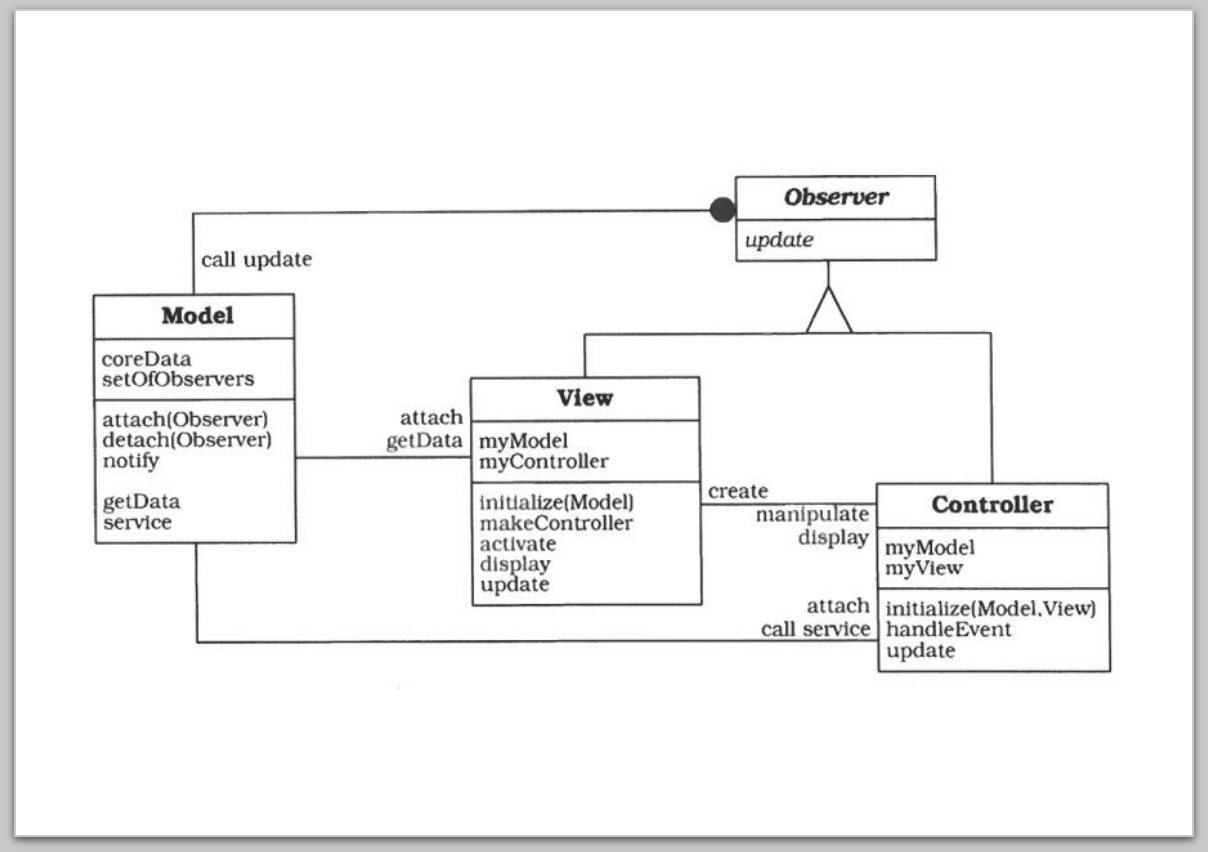


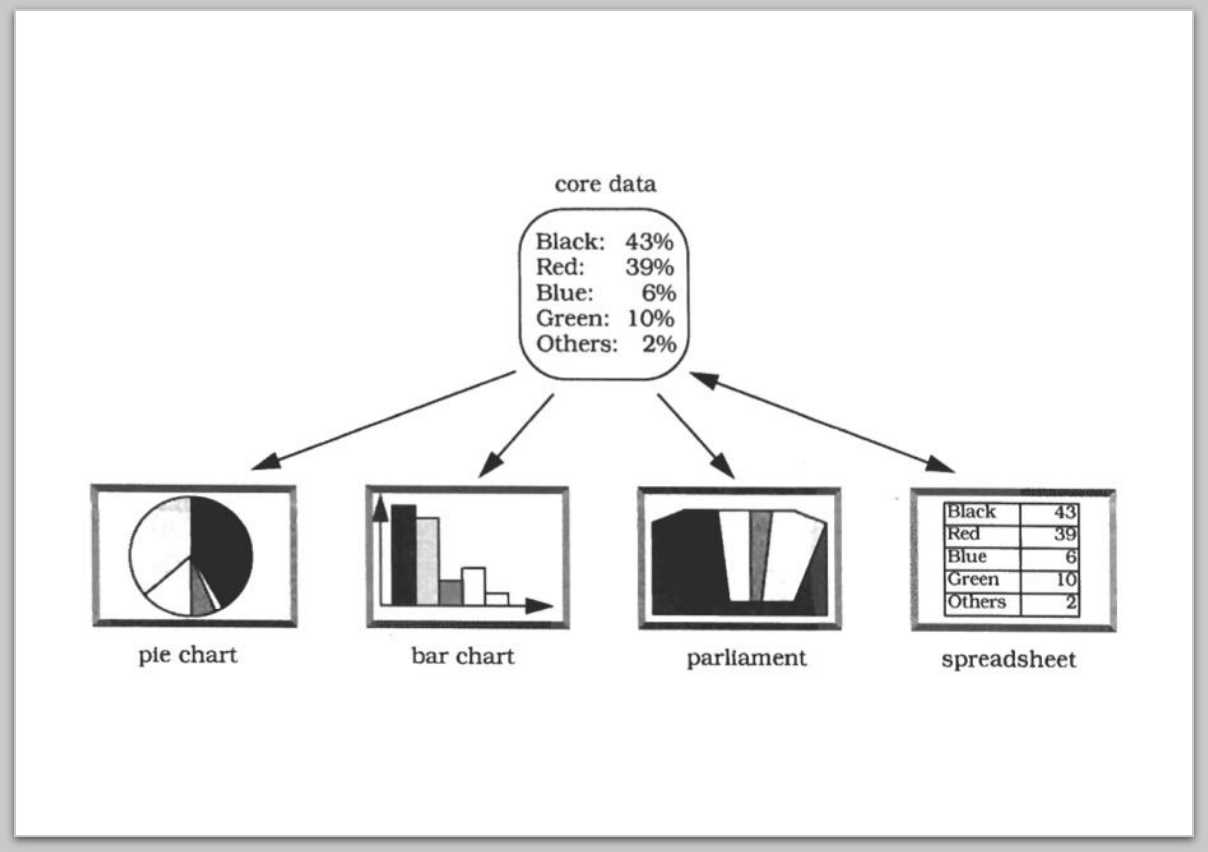
Modelo de referência OSI (Open Systems Interconnection)

Padrão arquitetural: Model View Controler

- **Descrição:** é um padrão utilizável em aplicações interativas que exigem uma interface com o usuário flexível. O MVC divide uma aplicação interativa em 3 componentes: o modelo contém os dados e as principais funcionalidades. As visões exibem informação para o usuário enquanto que os controllers capturam as entradas do usuário. As visões e os controllers compõem a interface com o usuário, enquanto que um mecanismo de propagação de alterações (normalmente implementado usando o padrão de projeto Observer) informa sobre alterações no modelo.

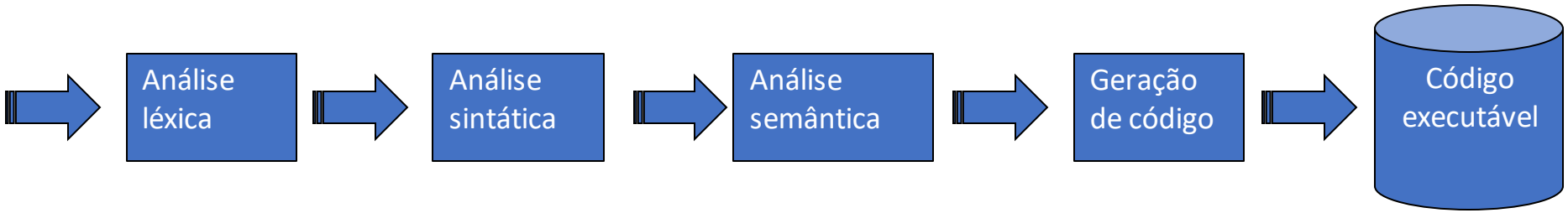
MVC: estrutura



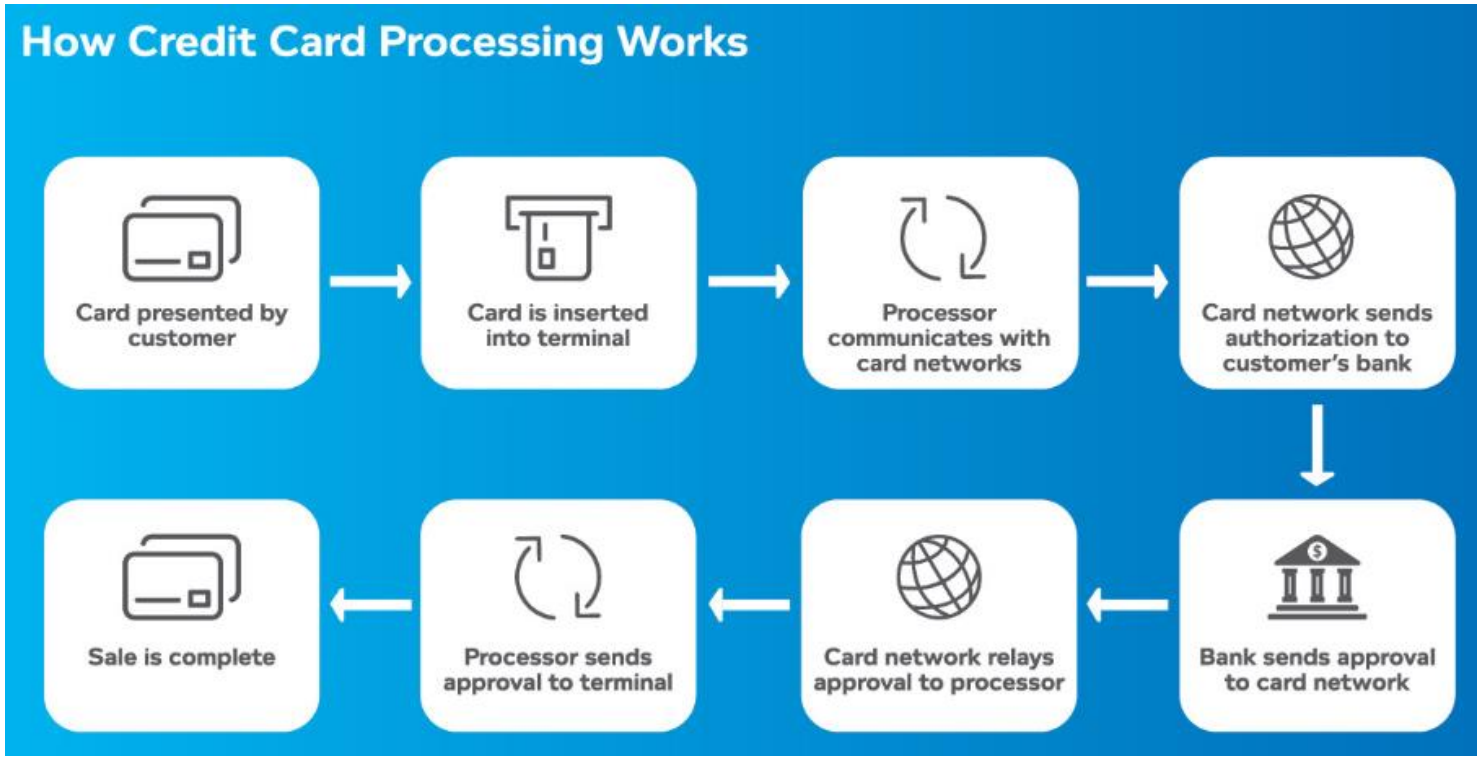


Padrão arquitetural: pipes and filters

- **Descrição**: provê uma estrutura para sistemas que processam uma cadeia de dados. Cada passo do processamento é encapsulado em um componente-filtro. Dados são passados por *pipes* entre filtros adjacentes. A recombinação de filtros permite a construção de sistemas correlatos.
- Exemplo clássico: compiladores:



Pipes and filters: outros exemplos

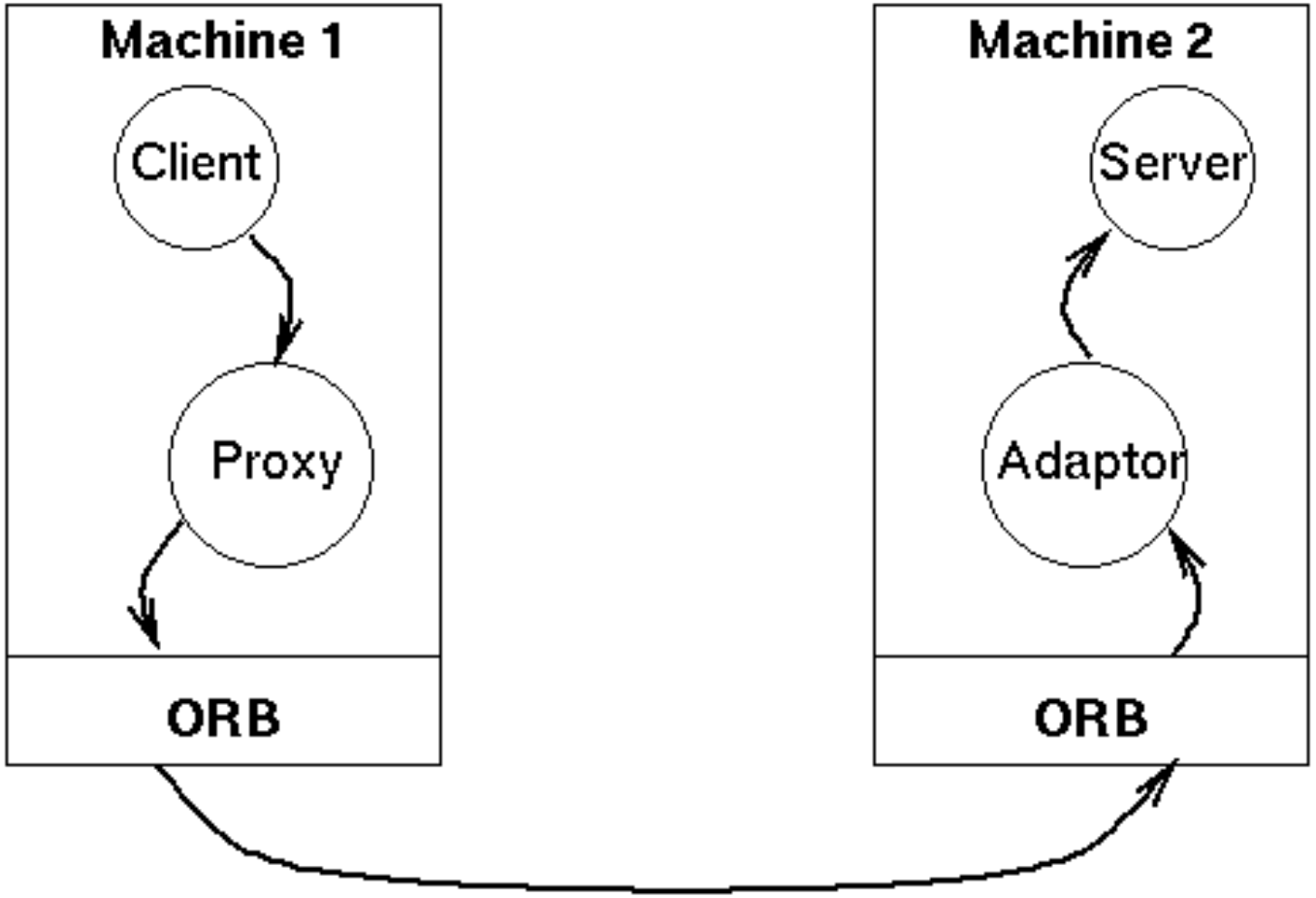


O comando "pipe" (|) do linux

- Pipe = cano
- Concatena a saída de um programa na entrada de outro
  - Exemplo:
    - ls –las | grep bash
    - Procura a palavra bash na saída gerada pelo comando "ls"

Padrão arquitetural: broker

- **Descrição: pode ser usado para estruturar sistemas de software distribuídos com componentes pouco acoplados que interagem por invocações de serviços remotos. Um componente *broker* é responsável por coordenar a comunicação, tais como repassar requisições, bem como, transmitir resultados e condições de erro.**
- **Exemplo: sistema WEB com sistema legado**



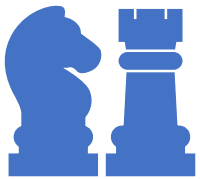
Outros exemplos

Arquitetura para jogos:
https://www.researchgate.net/publication/228882646_A_software_architecture_for_games

Sistemas inteligentes:
https://ieeexplore.ieee.org/abstract/document/385968?reload=true

Exercício

Pesquise um exemplo de arquitetura de jogos e um exemplo de arquitetura de sistemas inteligentes.

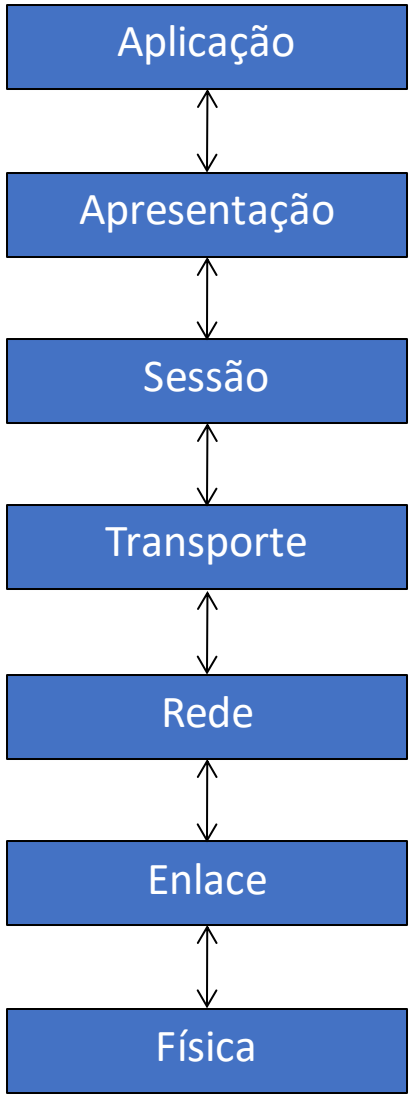


Nosso foco: aplicações corporativas

- Aplicações corporativas são aquelas que procuram automatizar as atividades do dia a dia das empresas
- Normalmente caracterizam-se por:
  - Diferentes perfis de usuário
  - Diferentes formas de interface com o usuário para acessar os sistemas
  - Usuários e sistemas distribuídos
  - Grandes volumes de dados
  - Grandes volumes de transações
  - Algoritmos de missão crítica

O padrão camadas: origens

- O padrão arquitetural clássico para o desenvolvimento de aplicações corporativas é o padrão camadas
- Simples de entender e aplicar procura organizar o código de uma aplicação (monolito) em camadas coesas, onde cada uma tem uma funcionalidade específica
- Suas origens estão na área de redes de computadores e seu modelo de referência OSI



Modelo de referência OSI (Open Systems Interconnection)

Padrão camadas: detalhamento

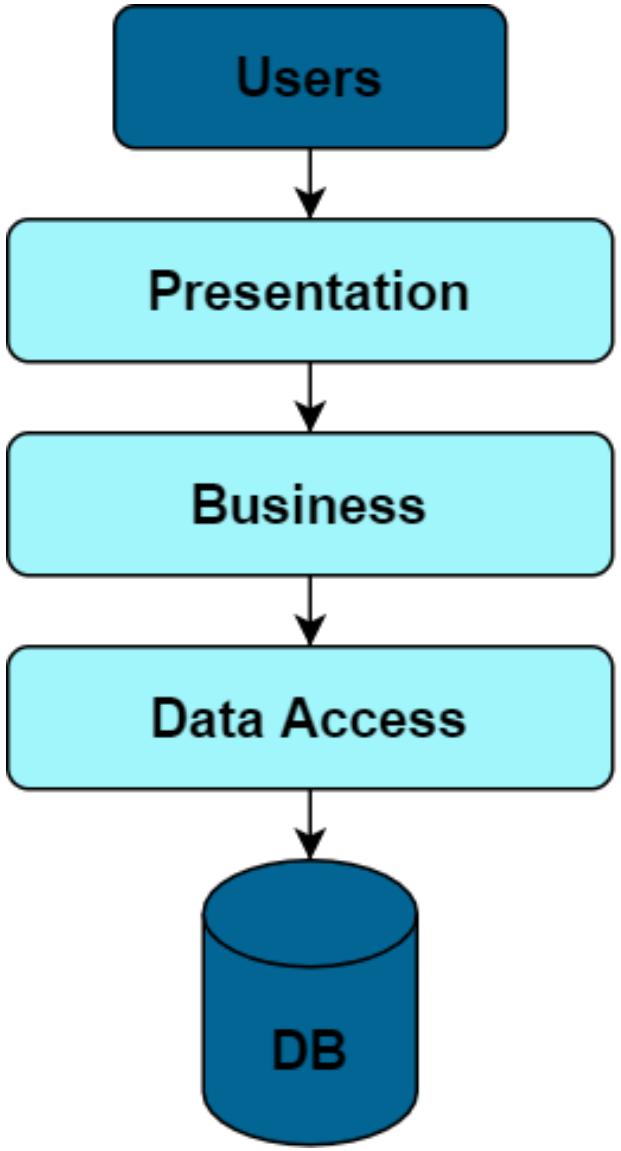
- Camadas são as abstrações horizontais das partes de uma aplicação.
- Seus limites são ortogonais ao fluxo dos dados.
- As camadas representam diferentes níveis e tipos de abstrações das responsabilidades que acompanham o desenvolvimento de software
- Dividir um software em camadas:
  - Garante o princípio da responsabilidade única
  - Garante a separação de responsabilidades
  - Facilita suportar diferentes tecnologias
  - Fatia a aplicação em partes mais fáceis de manter

Quantas camadas?

- O padrão “n-tier” é o padrão arquitetural mais comum
- É o modelo padrão para a maioria dos arquitetos e desenvolvedores que trabalham com orientação a objetos
- Cada camada cobre um objetivo específico da aplicação
- Na maioria das vezes distinguimos dois modelos:
  - A arquitetura clássica de 3 camadas
  - A arquitetura baseada em domínio (DDD) que utiliza 4 camadas

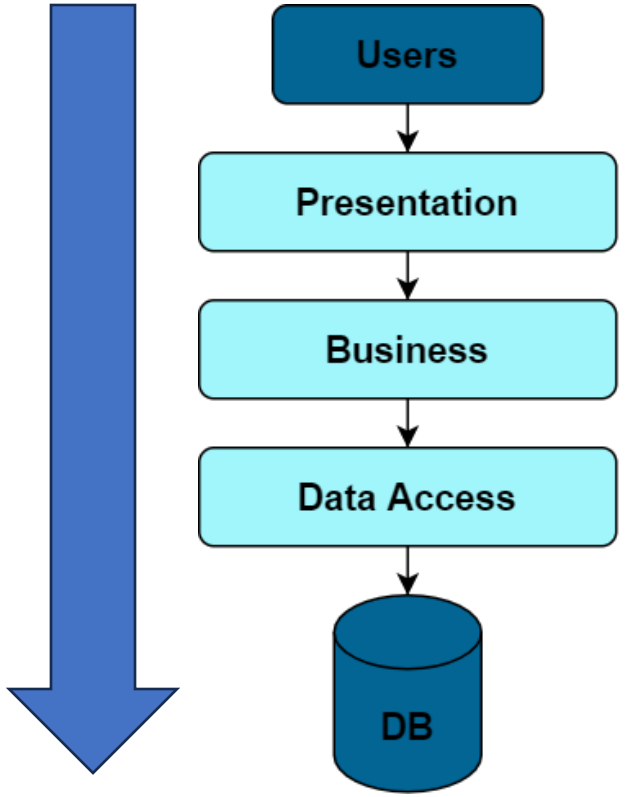
Detalhando o modelo de 3 camadas

- O nível de “apresentação” contém única e exclusivamente código e tecnologias relacionadas com a experiência do usuário
- A camada de negócio abriga o código responsável por implementar regras de negócio específicas (ex: regras de logística, cálculo de impostos, lógica de promoção etc)
- O nível de “acesso a dados” é responsável por fazer a conexão com as tecnologias de persistência de dados.



Regra fundamental

- O fluxo dos dados é sempre da camada de apresentação em direção a camada de acesso a dados
- Nada em uma camada “inferior” pode conhecer algo de uma camada “superior”
  - Os objetos da camada de apresentação podem acionar os objetos da camada de negócio ou acesso a dados.
  - O contrário não é permitido



Contexto histórico

- A organização de software corporativo em camadas surge mais ou menos ao mesmo tempo que se popularizam as tecnologias de SGBDs
- Diferentes bibliotecas oferecem diferentes abstrações para se fazer acesso aos bancos de dados relacionais
- O padrão que mais se popularizou junto as linguagens orientadas a objetos é o que retorna uma abstração de tabela como resposta a uma consulta SQL. A partir deste "objeto de retorno" é possível navegar na tabela resposta exibindo os resultados.
- No final dos anos 1990 começaram a surgir componentes capazes de exibirem diretamente os resultados desse tipo de consulta

Construindo aplicações de 3 camadas

- O projeto da aplicação começa pela modelagem do banco de dados relacional
- A camada de acesso a dados é construída a partir do projeto do banco de dados
    - Tipicamente os métodos retornam os resultados das consultas (abstrações das tabelas resposta)
- Classes que tratam das regras de negócio são acrescentadas na camada de negócios
- Por fim cria-se a camada de apresentação que consome serviços das classes das camadas de negócio ou de acesso a dados conforme o caso.

Dinâmica

Seguindo as orientações apresentadas, apresente a modelagem de classes, organizadas segundo uma arquitetura de três camadas, visando a implementação do sistema que segue:

- O sistema de uma loja online mantém um catálogo de produtos e o controle de estoque. Sobre cada produto mantém-se o código, a descrição e o preço de venda. No estoque mantém-se o código do produto, a quantidade disponível, o custo médio de aquisição e o estoque mínimo.
- O backend desse sistema deve oferecer endpoints para:
  - Apresentar catálogo: listar código, descrição, preço de venda e quantidade em estoque de todos os produtos disponíveis para venda (estoque > 0)
  - Efetuar uma venda: recebe o código do produto e a quantidade desejada. Retorna o custo da venda e dá baixa no estoque.
  - Entrada no estoque: recebe código do produto e quantidade e atualiza o estoque. Atualiza o custo médio de aquisição ((custo anterior * quantidade disponível + custo novo * quantidade adquirida)/(quantidade disponível + quantidade adquirida)).
  - Compras necessárias: retorna a lista dos códigos dos produtos que estão com o estoque abaixo do mínimo.

MARISTA

Dinâmica

Discuta com o grande grupo as soluções desenvolvidas para o sistema de vendas:

- Analise as dependências das camadas superiores em relação ao banco de dados. Considere:
  - Organização do banco
  - Tipo de banco: SQL ou NoSQL
  - Tecnologia (MySQL, Oracle, Postgresql etc)
- Análise o respeito as dependências entre as camadas
  - Se existe alguma dependência invertida
  - Se existem casos em que as camadas foram “puladas”
- Análise a necessidade de se modelarem entidades
  - Que tipo de resposta a camada de acesso a dados retorna: o que são DTOs?

MARISTA

Quando usar o modelo de 3 camadas

- O modelo de 3 camadas classico é o mais indicado para aplicações CRUD
- Aplicações CRUD são aquelas onde a maioria dos casos de uso está relacionado com criar, consultar, atualizar ou remover alguma coisa (Create, Read, Update ou Delete).
- Para sistemas mais complexos sugere-se o uso de DDD (será visto na sequência)