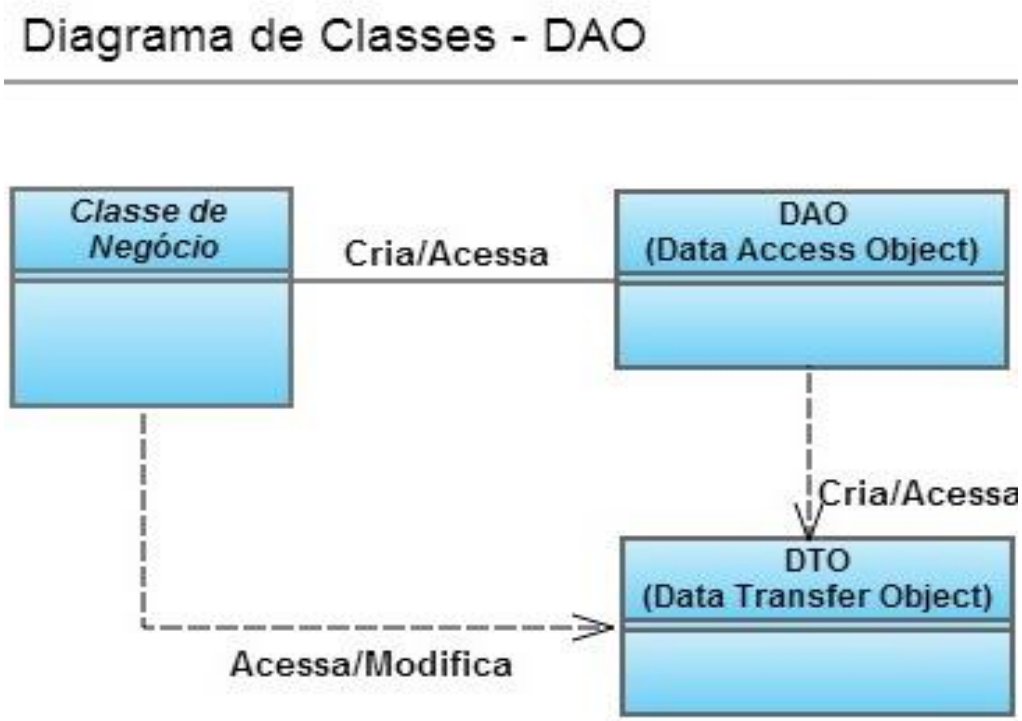
Definindo um padrão para arquitetura Web

Padrões de Projeto

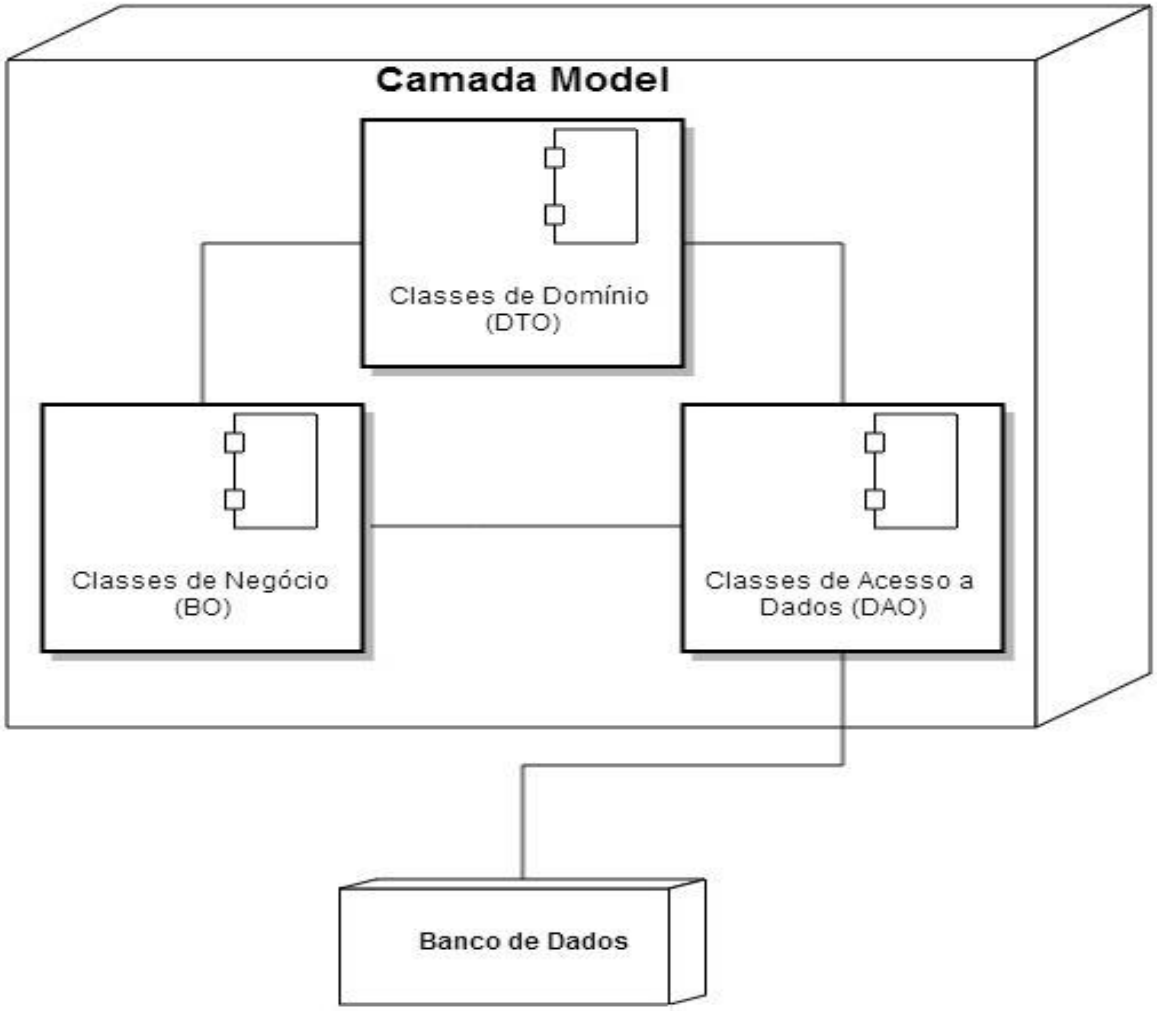
- Soluções reutilizáveis para situações ou problemas encontrados comumente em desenvolvimento de software orientado a objetos.
- Livros
  - Design Patterns: Elements of Reusable Object-Oriented Software (Gangue dos Quatro - *Gang of Four* ou *GoF*)
  - Core J2EE Patterns

DAO (Data Access Object)

- Objetivo: Encapsular o acesso a dados em uma classe separada da aplicação.



DAO (Data Access Object)

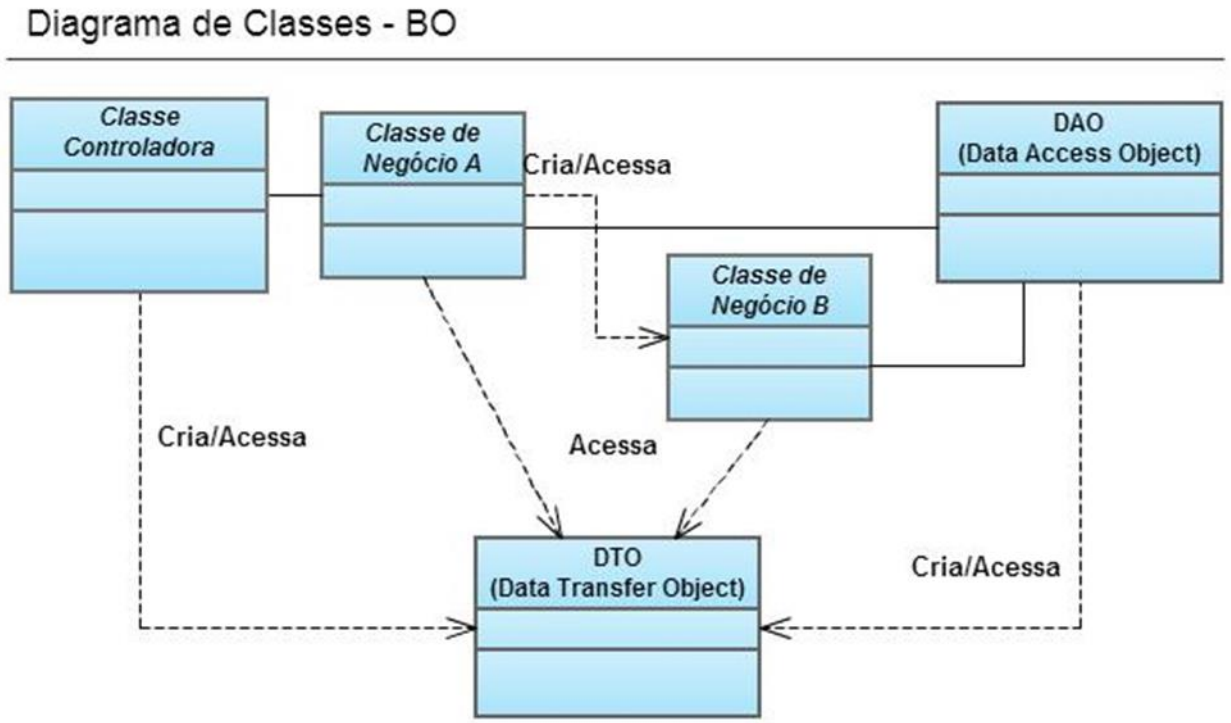


DAO (Data Access Object)

- Características
  - Centralização do código de acesso/manipulação de dados da aplicação.
  - Separação da lógica de negócio da persistência.
  - Tornar transparente o acesso aos dados nas aplicações.
  - Possibilitar acesso a diferentes fontes de dados de forma transparente para o usuário.
    - O padrão DAO em conjunto com padrões de projeto que atuam como fábricas de objetos (Factory e Abstract Factory) possibilita a implementação de acesso para diferentes mecanismos de persistência.

Business Object (BO)

- Objetivo: Centralizar as regras de negócio da aplicação.

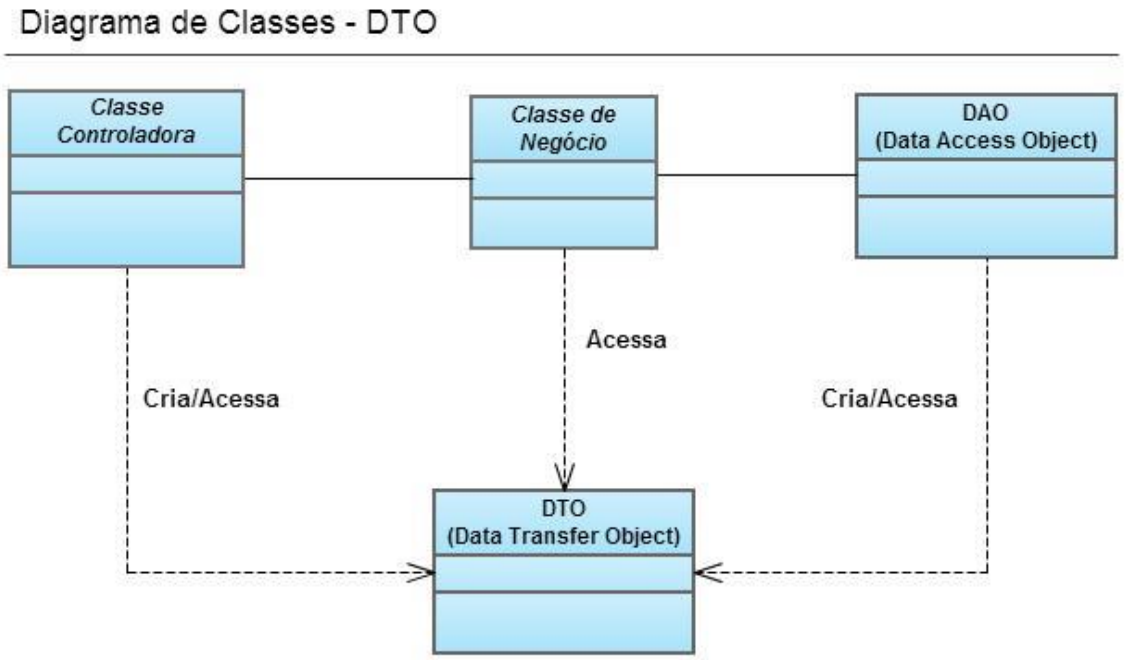


Business Object (BO)

- Características:
  - Centralização da lógica de negócio da aplicação.
  - Separação da lógica de negócio do acesso a dados.
  - Reutilização de código.
  - Possibilidade de uso em outras arquiteturas.
    - por exemplo, em SOA (Service-Oriented Architecture), modificando apenas as chamadas dos métodos de negócio para a acessarem serviços, e não mais métodos de classes de acesso a dados.

Data Transfer Object (DTO)

- Objetivo: Transportar os dados de uma única vez entre as camadas da aplicação

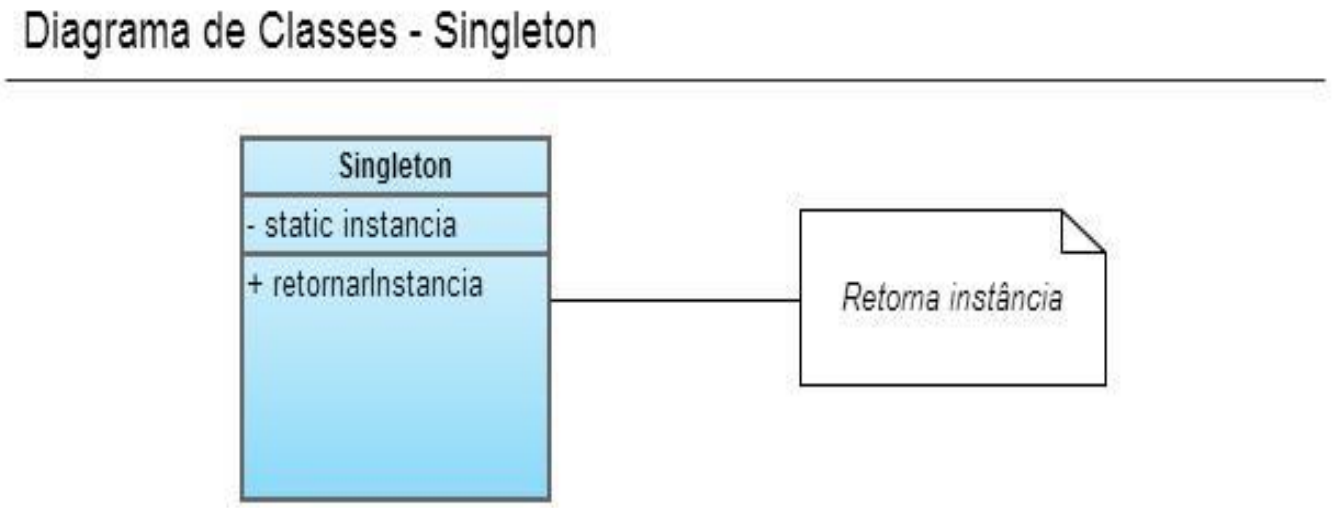


Data Transfer Object (DTO)

- Características:
  - Reduzir o tráfego na rede.
  - Transportar dados entre camadas.
  - Simplificação de especificação e codificação de interfaces.

SINGLETON

- Objetivo: Criar apenas um objeto de uma determinada classe.

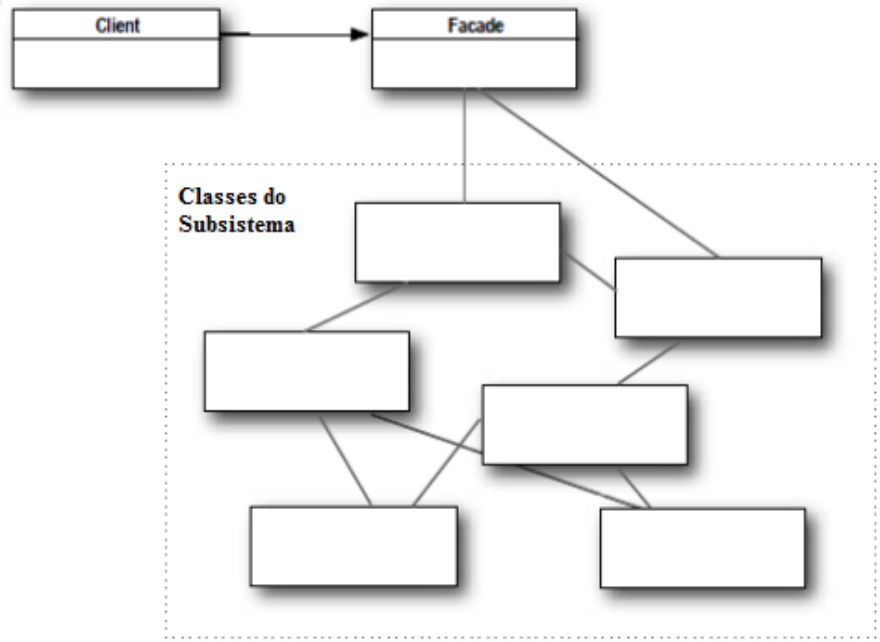


SINGLETON

- Características:
  - Reutilização de objetos:

Facade

- Objetivo: ocultar a complexidade de uma ou mais classes, simplificando o uso de um subsistema implementando apenas uma classe que forneça uma interface única e mais razoável.



Facade

- Características:
  - fornece uma interface unificada para um conjunto de interfaces em um subsistema.
  - define uma interface de nível mais alto que facilita a utilização do subsistema”
  - Um subsistema pode ter diversos Facades.

MODELO MVC

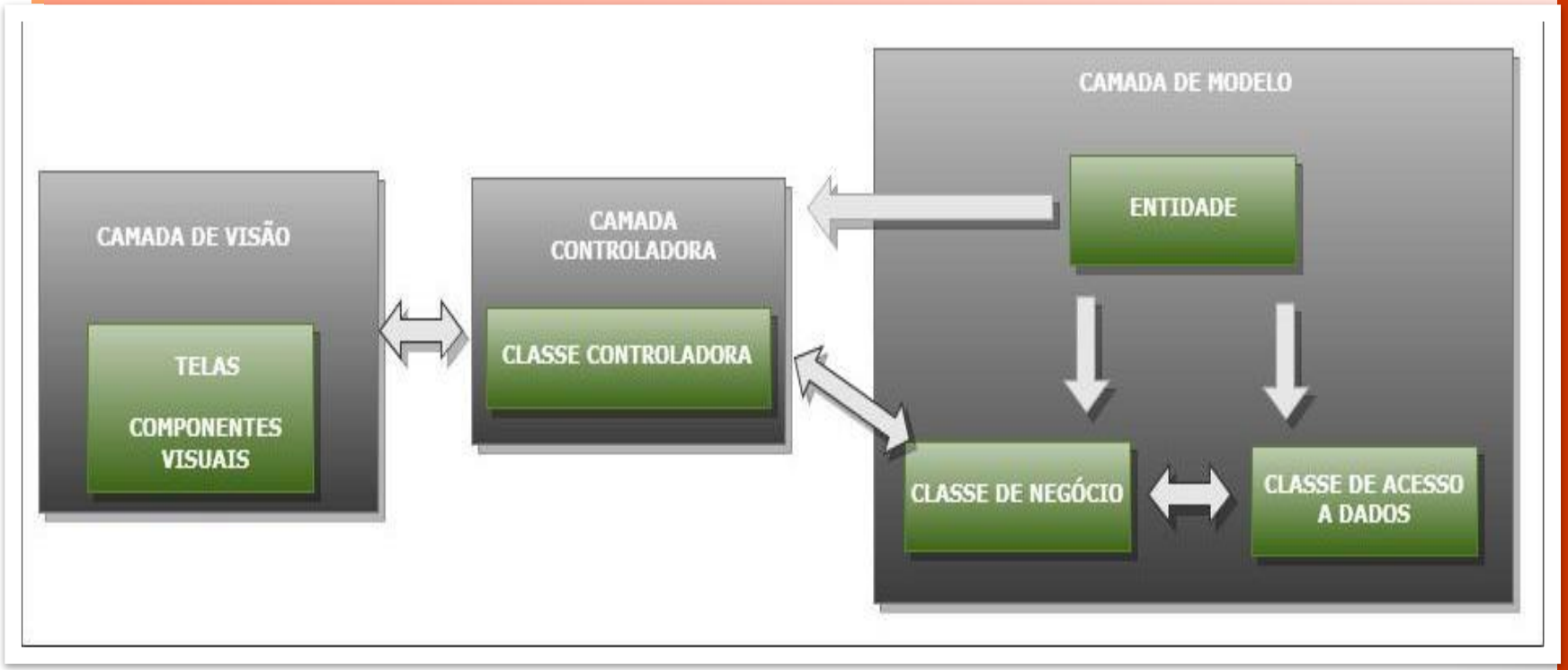
MVC

- Padrão de desenvolvimento de software baseado em 03 camadas:
  - Modelo (Model)
  - Visão (View)
  - Controladora (Controller)
- Objetivo: separar a lógica da aplicação da apresentação das informações ao usuário.

MVC - Camadas

- Modelo
  - concentra as classes de domínio (entidades) da aplicação, além das classes de negócio e de acesso a dados;
- Visão
  - responsável pelo layout da aplicação (telas em HTML, por exemplo) e seus componentes visuais;
- Controladora
  - direciona o fluxo de dados entre as camadas de visão e de modelo da aplicação.
    - pode-se alterar as telas ou componentes visuais do sistema sem modificar as classes responsáveis pela a lógica da aplicação (modelo e controladoras), e vice-versa.
    - diminui-se o tempo de manutenção de funcionalidades devido a alta coesão (classes com responsabilidades e objetivos bem definidos) e ao baixo acoplamento (pouca dependência de outras classes).

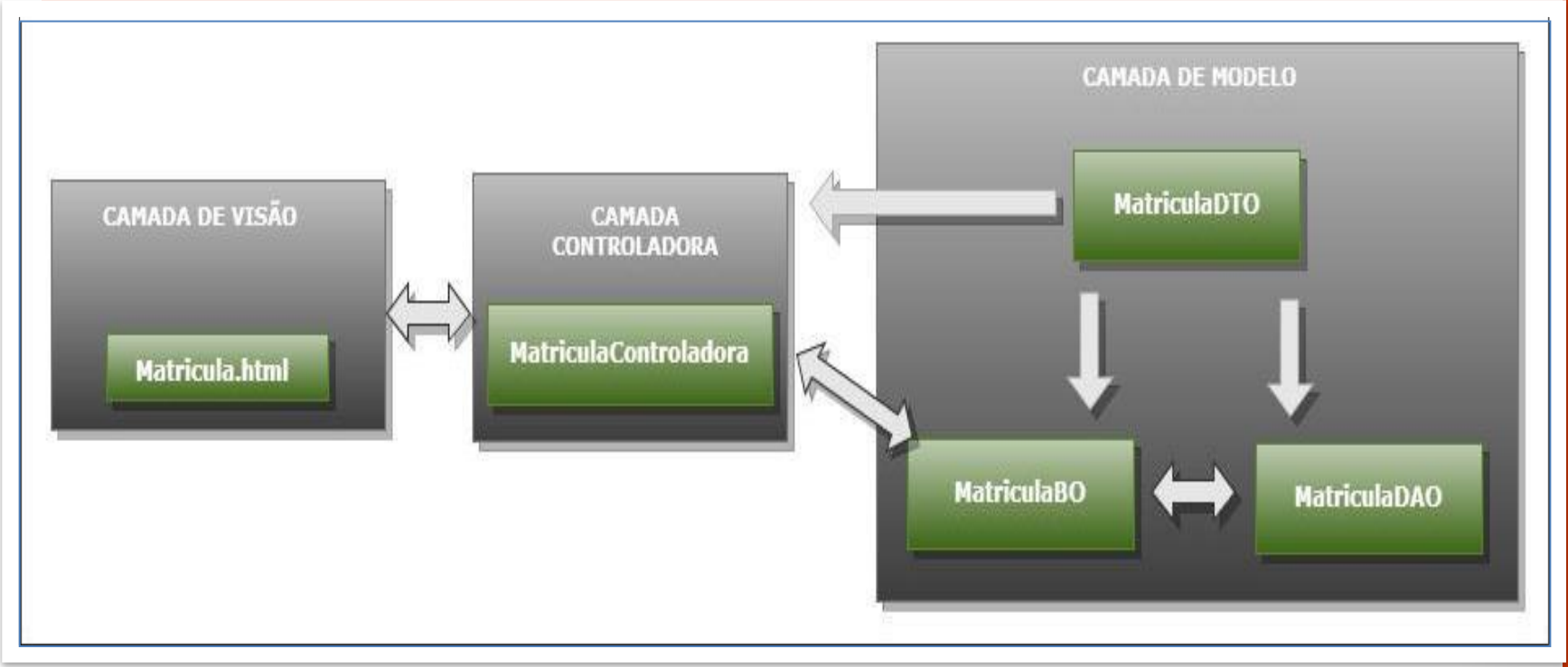
MVC - Camadas


CAMADA DE MODELO
ENTIDADE
CAMADA DE VISÃO
TELAS
COMPONENTES VISUAIS
CAMADA CONTROLADORA
CLASSE CONTROLADORA
CLASSE DE NEGÓCIO
CLASSE DE ACESSO A DADOS

MVC

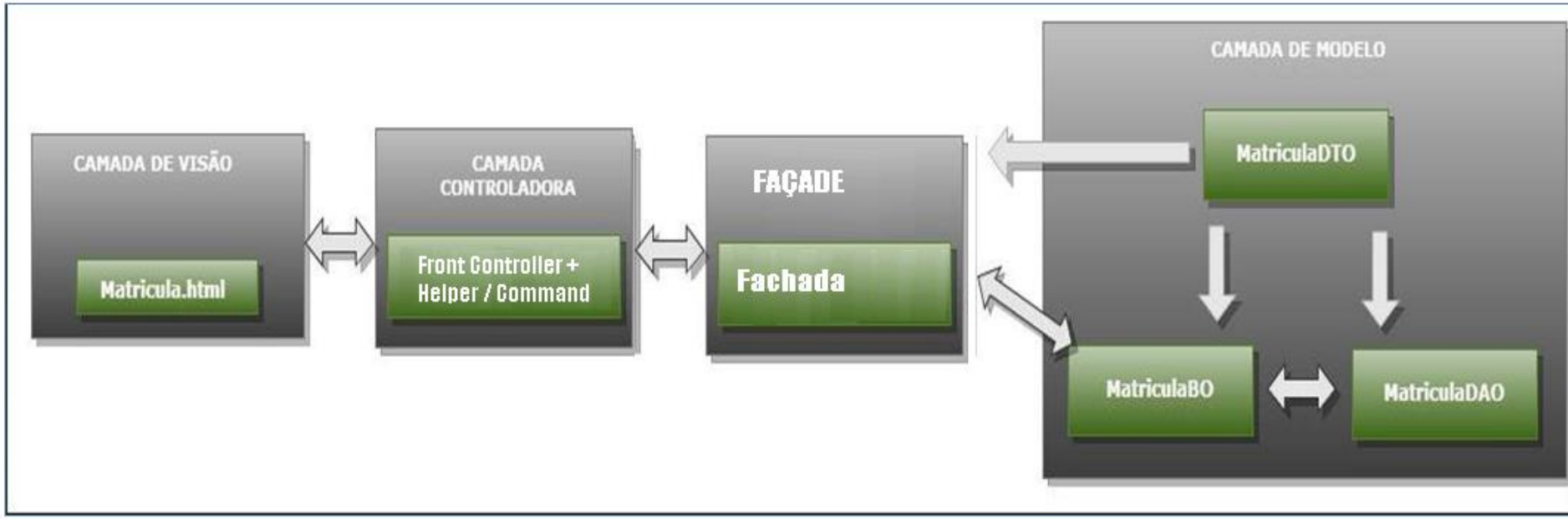
- a cada nova funcionalidade será necessário criar as respectivas camadas de visão, controladora e modelo
- Pode-se combinar o MVC com outros padrões de projeto:
  - DAO (Data Access Object),
  - DTO (Data TransferObject),
  - BO (Business Object) e
  - Singleton.
    - Aplica-se DTO em Matricula(MatriculaDTO);
    - MatriculaRN recebe o padrão BO (MatriculaBO) e
    - DAO e Singleton são aplicados em MatriculaPersistente (MatriculaDAO).

MVC - Exemplo



- Tela de Matrícula (Matricula.html)
- Classe controladora de Matrícula (MatriculaControladora)
- Entidade de Matrícula (Matricula)
- Regras de negócio de Matrícula (MatriculaRN (Regras de Negócio))
- Persistência de Matrícula (MatriculaPersistente)

MVC com fachada de negócio



Exercício

Implemente uma aplicação de leilão em MVC com as seguintes funcionalidades:

1. Administrador
   - Iniciar Leilão: informar o produto e lance mínimo
2. Usuário
   - Na página principal são listados os leiloes ativos e os inativos, apresentando a foto do produto e o preço. Se estiver ativo, o botão do lance.
   - Cadastrar-se: ao se cadastrar o cliente recebe 50 bagos
   - Registrar Lances: ao clicar no lance, um centavo é incluído como lance do produto e um bago retirado do saldo co cliente.
   - Comprar bagos: escolher o lote de 50, 80 ou 100 bagos para comprar

Se ficar 30 segundo sem nenhum usuário dar lance no produto, o último lance é considerado vencedor.