MVC

Separa responsabilidades:

Model: dados e regras de negócio;

View: interface com o usuário;

Controller: lógica que recebe dados da View e interage com o Model.

Isso facilita manutenção, testes e evolução do sistema.

Camadas

Modulariza o sistema em camadas distintas: apresentação (UI), lógica de negócio e persistência.

Garante organização e separação clara, o que aumenta a escalabilidade e manutenibilidade.

Estilo Arquitetural Secundário: Arquitetura em Camadas

O sistema também é organizado em camadas bem definidas:

Camada de Apresentação (UI): Responsável pela interface com o usuário.

Camada de Lógica de Negócio: Onde ficam as regras e validações.

Camada de Persistência: Responsável por salvar e recuperar dados do banco.

Justificativa:

Reforça a separação de responsabilidades.

Garante organização e facilita testes unitários por camada.

Aumenta a manutenibilidade e a escalabilidade.

Cliente-Servidor

Permite que diferentes clientes (web, mobile) acessem um backend centralizado.

Mantém backend e frontend desacoplados, facilitando integração e atualizações independentes.

Complementos

RESTful APIs no Controller para integração com outros sistemas e facilidade de comunicação entre cliente e servidor.

Event-driven architecture para comunicação em tempo real, como notificações via WebSocket para atualizar status de pedidos sem refresh.

A arquitetura em camadas reforça a organização, facilitando a divisão de responsabilidades e testes unitários.

Diagrama com Visão Geral do Sistema - Arquitetura MVC (Sistema estilo iFood)

Camada Model (Negócio + Dados)

- `Cliente`
- `Restaurante/Loja`
- `Produto`
- `Pedido`
- `Pagamento`
- `Entrega`
- `Entregador`

Funções:

- Contêm regras de negócio
- Lógica de persistência
- Integração com banco de dados

Camada View (Interface do Usuário)

- App Web/Mobile Cliente:
    - Tela de login
    - Tela de busca de restaurantes
    - Cupom
    - Cartão e pagamento
    - Tela de carrinho
    - Tela de pedido em andamento

- App Restaurante:
    - Tela de recebimento de pedidos
    - Tela de status do pedido
    - Faturamento
    - Controle de Lojas
    - Produtos
    - Preço
- App Entregador:
    - Tela de pedidos disponíveis
    - Tela de entrega ativa
    - Rotas de entregas

Funções:

- Interface amigável para usuários
- Responsiva, atualiza com status em tempo real
- Moderna
- Com Dicas para induzir o usuário a consumir mais

Camada Controller (Orquestrador)

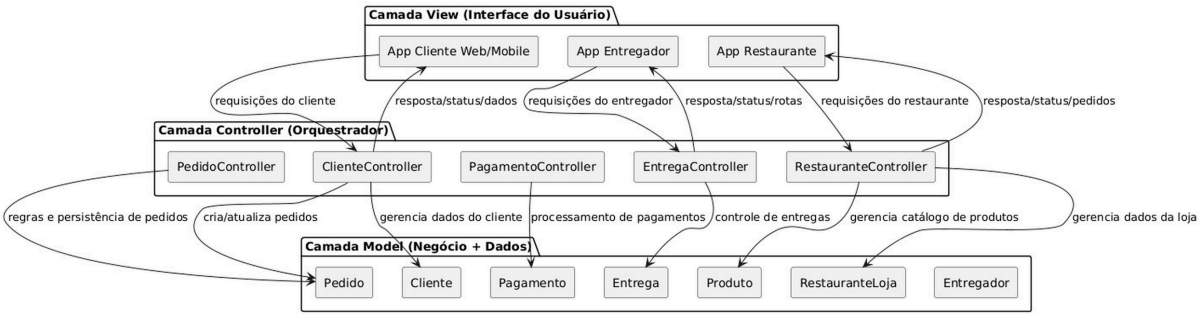
- `PedidoController`
- `PagamentoController`
- `EntregaController`
- `ClienteController`
- `RestauranteController`

Funções:

- Recebe requisições da view
- Valida dados
- Chama os métodos do model
- Retorna resposta para a view

Fluxo resumido (MVC aplicado ao fluxo "Fazer Pedido")

1. **View**: Cliente seleciona produtos e clica em "Confirmar Pedido"
2. **Controller**: `PedidoController` recebe os dados, valida e cria o pedido
3. **Model**: `Pedido` é salvo no banco, status inicial é "Aguardando confirmação"
4. **View (Restaurante)**: recebe notificação em tempo real do novo pedido
5. **View (Cliente)**: mostra "Pedido enviado"


Camada View (Interface do Usuário)
App Cliente Web/Mobile
App Entregador
App Restaurante
requisições do cliente
resposta/status/dados
requisições do entregador
resposta/status/rotas
requisições do restaurante
resposta/status/pedidos
Camada Controller (Orquestrador)
PedidoController
ClienteController
PagamentoController
EntregaController
RestauranteController
regras e persistência de pedidos
cria/atualiza pedidos
gerencia dados do cliente
processamento de pagamentos
controle de entregas
gerencia catálogo de produtos
gerencia dados da loja
Camada Model (Negócio + Dados)
Pedido
Cliente
Pagamento
Entrega
Produto
RestauranteLoja
Entregador