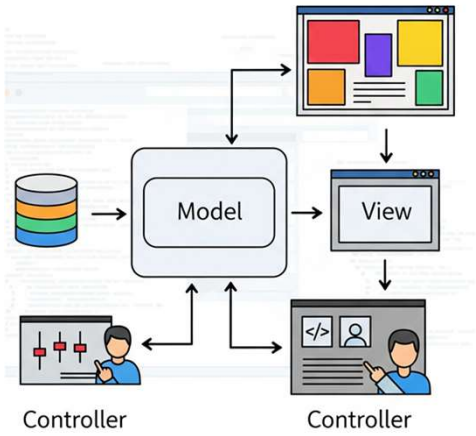


MVC

MVC (Model-View-Controller)

O padrão MVC (Model-View-Controller), que adaptaremos para Controller-Service-View, é uma forma de organizar o código que divide a aplicação em três partes principais.

O objetivo é a "Separação de Preocupações" (Separation of Concerns).

Service: Cuida dos DADOS e das REGRAS DE NEGÓCIO.

View: Cuida da APRESENTAÇÃO.

Controller: Cuida do FLUXO / LÓGICA DE CONTROLE.

O Service (O Cérebro da Aplicação)

Responsabilidade Única: Gerenciar os dados e a lógica de negócio.

O que ele faz?

- Conecta-se e conversa com o banco de dados.
- Contém as regras de negócio (ex: "um contato não pode ser salvo sem um nome").
- Busca, valida, insere, atualiza e deleta informações.

O que ele NÃO faz?

- Ele NÃO sabe como os dados serão exibidos.
- Ele NÃO contém HTML ou qualquer código de apresentação.
- Pense no Service como o "especialista no negócio". Você pede "salve este novo contato", e ele valida os dados, formata o que for preciso e realiza a gravação.

A View (A Vitrine da Aplicação)

Responsabilidade Única: Exibir os dados para o usuário.

O que ela faz?

- É o nosso arquivo HTML.
- Recebe os dados que o Controller envia e os organiza na página.
- É a camada com a qual o usuário interage diretamente (botões, formulários, etc.).

O que ela NÃO faz?

- Ela NÃO busca dados no banco.
- Ela NÃO possui lógica complexa. A View é "burra", ela apenas exibe o que mandam.
- Pense na View como o "designer de interface". Ela pega os dados brutos e os deixa bonitos e organizados para o usuário final.

O Controller (O Maestro da Orquestra)

Responsabilidade Única: Orquestrar o fluxo da aplicação.

O que ele faz?

- É o intermediário entre o Service e a View.
- Recebe a requisição do usuário (ex: site.com/contatos).
- Pede os dados ao Service.
- Envia esses dados para a View correta.
- Recebe os dados de um formulário ($_POST) e manda o Service salvá-los.

O que ele NÃO faz?

- Ele NÃO executa queries SQL (isso é trabalho do Service/DAO).
- Ele NÃO contém HTML (isso é trabalho da View).
- Pense no Controller como o "gerente de projetos". Ele ouve o cliente (usuário), pede o trabalho para a equipe de especialistas (Service) e entrega o resultado para a equipe de design (View).

Fluxo de uma Requisição

1. O Usuário acessa site.com/contatos/listar.
2. O Roteador (nosso index.php) direciona a chamada para o método listar() do ContatoController.
3. O Controller diz: "Service, preciso de todos os contatos!".
4. O Service vai ao banco de dados, busca os contatos e retorna um array de dados para o Controller.
5. O Controller recebe o array e diz: "View, pegue esses dados e mostre para o usuário!".
6. A View (lista_contatos.php) recebe os dados, monta o HTML com um foreach e exibe a página final.

Mão na massa

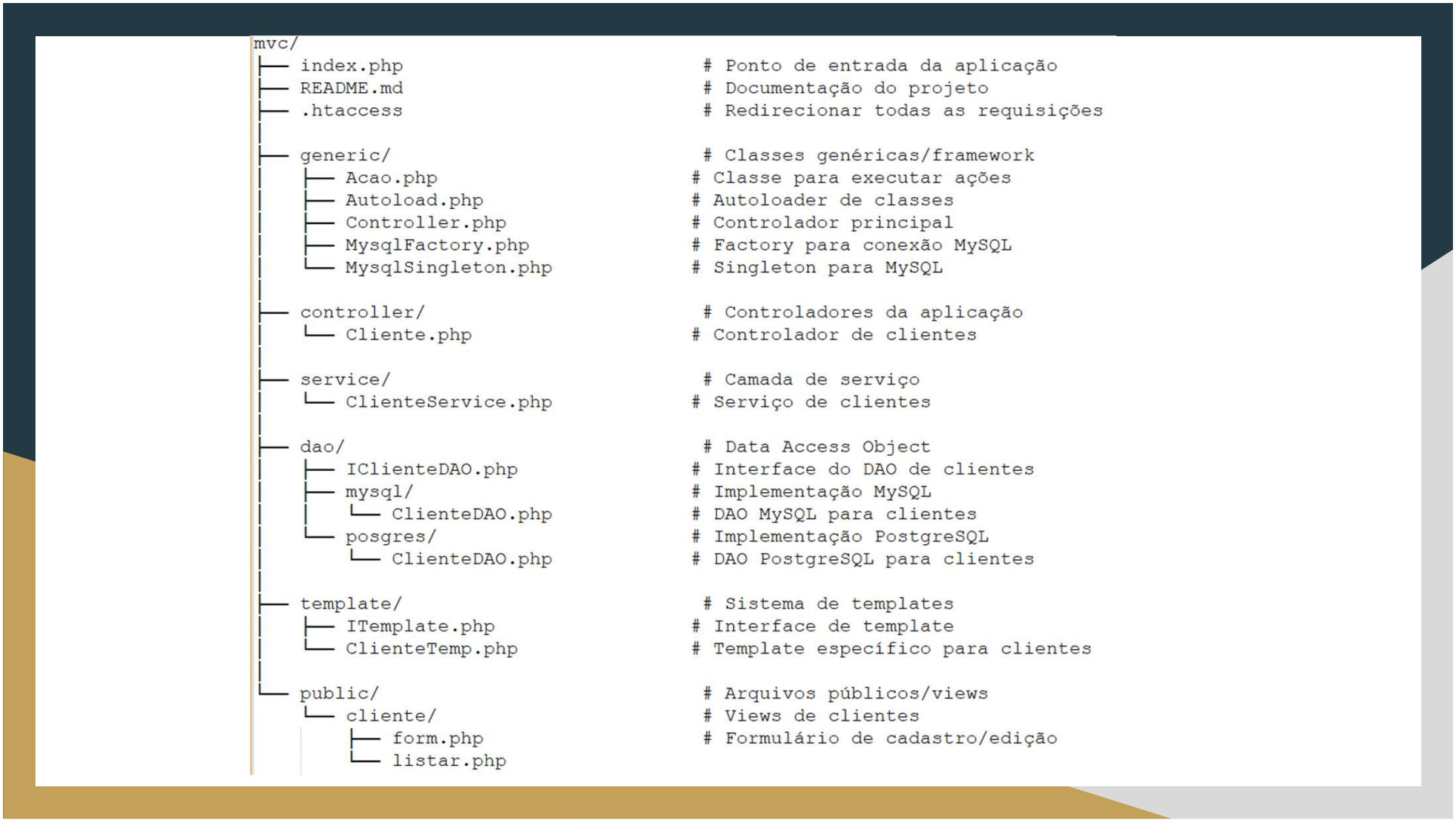
Construir uma aplicação simples para gerenciar clientes.

Funcionalidades de hoje:

Criar a estrutura de pastas do projeto.

Listar os contatos existentes.

Adicionar um novo cliente através de um formulário.


mvc/
├── index.php # Ponto de entrada da aplicação
├── README.md # Documentação do projeto
├── .htaccess # Redirecionar todas as requisições
│
├── generic/ # Classes genéricas/framework
│ ├── Acao.php # Classe para executar ações
│ ├── Autoload.php # Autoloader de classes
│ ├── Controller.php # Controlador principal
│ ├── MysqlFactory.php # Factory para conexão MySQL
│ └── MysqlSingleton.php # Singleton para MySQL
│
├── controller/ # Controladores da aplicação
│ └── Cliente.php # Controlador de clientes
│
├── service/ # Camada de serviço
│ └── ClienteService.php # Serviço de clientes
│
├── dao/ # Data Access Object
│ ├── IClienteDAO.php # Interface do DAO de clientes
│ ├── mysql/ # Implementação MySQL
│ │ └── ClienteDAO.php # DAO MySQL para clientes
│ └── posgres/ # Implementação PostgreSQL
│ └── ClienteDAO.php # DAO PostgreSQL para clientes
│
├── template/ # Sistema de templates
│ ├── ITemplate.php # Interface de template
│ └── ClienteTemp.php # Template específico para clientes
│
└── public/ # Arquivos públicos/views
└── cliente/ # Views de clientes
├── form.php # Formulário de cadastro/edição
└── listar.php

.htaccess

.htaccess implementa um padrão muito comum em aplicações web com PHP, conhecido como "Front Controller". A ideia principal é redirecionar todas as requisições que não sejam para arquivos ou diretórios existentes para um único arquivo PHP (neste caso, index.php).

Isso permite criar URLs amigáveis (como /contato ou /produtos/1) em vez de URLs "feias" (como /index.php?param=contato).

.htaccess: RewriteEngine on

Esta linha é o interruptor principal. Ela liga o motor de reescrita de URLs do Apache (mod_rewrite). Sem ela, nenhuma das regras abaixo funcionaria. É como dar a partida no carro antes de começar a dirigir.

RewriteEngine: A diretiva que controla o motor de reescrita.

on: O comando para ligar.

.htaccess: RewriteCond

Antes de aplicar a regra, o Apache faz duas verificações:

- **RewriteCond %{REQUEST_FILENAME} !-f**
  - Tradução: "A URL que o usuário pediu NÃO É um arquivo (-f) que existe de verdade no servidor?"
  - !-f → NÃO é um File (arquivo).

- **RewriteCond %{REQUEST_FILENAME} !-d**
  - Tradução: "A URL que o usuário pediu NÃO É um diretório (-d) que existe de verdade no servidor?"
  - !-d → NÃO é um Directory (diretório).

Conclusão: A regra só será aplicada se a requisição NÃO for para um arquivo (.css, .jpg, .js) ou uma pasta existente.

.htaccess: RewriteRule

RewriteRule ^(.*)$ index.php?param=$1 [L,QSA]

Vamos quebrar em 3 partes:

- Padrão: **^(.*)$**
  - É uma Expressão Regular (Regex) que "captura" tudo o que vem na URL.
  - contato, produtos/tenis/42, sobre-nos... tudo é capturado!
- Destino: **index.php?param=$1**
  - Para onde a requisição será enviada internamente.
  - $1 é uma variável que contém exatamente o que foi capturado no Padrão.
- Flags (Opções): **[L,QSA]**
  - L (Last): Se esta regra for usada, pare de processar outras regras.
  - QSA (Query String Append): Se a URL original tiver parâmetros (?busca=teste), ANEXE eles ao final.

.htaccess

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php?param=$1 [L,QSA]
```