**SUMÁRIO**

[1. ARCHITECTURAL PATTERNS 1](#_Toc60276488)

[1.1. DIFERENÇA ENTRE DESIGN PATTERN E ARCHITECTURAL PATTERN 1](#_Toc60276489)

[2. arquitetura mvc (model view controle) 1](#_Toc60276490)

[2.1. CAMADA CONTROLLER 2](#_Toc60276491)

[2.2. CAMADA MODEL 2](#_Toc60276492)

[2.3. CAMADA VIEW 2](#_Toc60276493)

[3. SERVIDOR PHP EMBUTIDO 3](#_Toc60276494)

[3.1. EXECUTANDO APLICAÇÕES NO SERVIDOR EMBUTIDO 5](#_Toc60276495)

[4. ESTRUTURA DE DIRETÓRIOS para projetos 6](#_Toc60276496)

[4.1. CRIANDO UMA ESTRUTURA INICIAL 7](#_Toc60276497)

[5. composer e autoload 8](#_Toc60276498)

# ARCHITECTURAL PATTERNS

Um **padrão de arquitetura** é uma solução geral e reutilizável para um problema que ocorre com frequência em arquitetura de software dentro de um determinado contexto.[1] **Padrões de arquitetura são similares aos padrões de projeto de software (Design Patterns)**, mas possuem um escopo mais amplo.

Os padrões de arquitetura abordam vários problemas de engenharia de software, como limitações de desempenho de hardware de computador, alta disponibilidade e minimização de um risco de negócio. Alguns padrões arquiteturais foram implementados dentro de frameworks de software.

## DIFERENÇA ENTRE DESIGN PATTERN E ARCHITECTURAL PATTERN

Enquanto a arquitetura do software é responsável pelo esqueleto e pela infraestrutura de alto nível de um software, o design do software é responsável pelo design do nível de código, como o que cada módulo está fazendo, o escopo das classes e os objetivos das funções, etc.

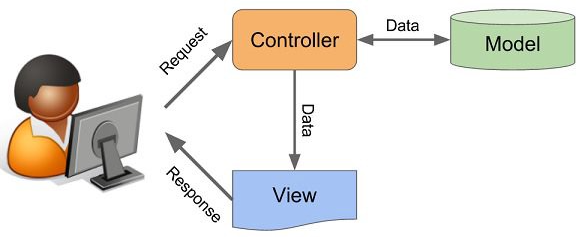
**Em termos de engenharia de software**, os padrões de arquitetura nos permitem pensar nos requisitos não funcionais, e de alto nível de uma aplicação, enquanto que os padrões de projetos se voltam para o aspecto algorítmico, de baixo nível da aplicação, ou, em outras palavras, para a implementação dos seus requisitos funcionais.

Para saber mais sobre Design Patterns e Arquitectural Patterns: <https://oieduardorabelo.medium.com/arquitetura-de-software-a-diferen%C3%A7a-entre-arquitetura-e-design-964ab38ea61a>

# arquitetura mvc (model view controle)

O MVC é uma sigla do termo em inglês Model (modelo) View (visão) e Controller (Controle) que facilita a troca de informações entre a interface do usuário aos dados no banco, fazendo com que as respostas sejam mais rápidas e dinâmicas. Embora existam vários que podem ser utilizados, **o padrão MVC é o mais conhecido e empregado** entre os desenvolvedores profissionais.

A arquitetura MVC é dividida em três componentes, ou camadas essenciais: Model, Controller e View, que dizem respeito, como é nítido, a sua sigla.



## CAMADA CONTROLLER

A camada de controle é responsável por intermediar as requisições enviadas pelo View com as respostas fornecidas pelo Model, realizando um papel de interface entre as próprias requisições, o processamento e a resposta.

## CAMADA MODEL

Essa camada também é conhecida como Business Object Model (objeto modelo de negócio). Sua responsabilidade é gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas.

Pode soar estranho, o termo alternativo utilizado para se referenciar a essa camada, mas existe porque originalmente o padrão MVC foi criado fora do contexto de desenvolvimento Web. Portanto, em uma linguagem clara e compreensível para o contexto da Web, **a camada model apenas manipula os dados do banco de dados**.

## CAMADA VIEW

Essa camada é responsável por apresentar as informações de forma visual ao usuário. Em seu desenvolvimento devem ser aplicados apenas recursos ligados a aparência como mensagens, botões ou telas.

Seguindo UM processo de comparação, o View está na linha de frente da comunicação com usuário e é responsável transmitir questionamentos ao controller e entregar as respostas obtidas ao usuário. É a parte da interface que se comunica, disponibilizando e capturando todas as informações do usuário.

# SERVIDOR PHP EMBUTIDO

O PHP possui um servidor web embutido, que passou a existir desde a sua versão 5.4, e apesar de ser mais comum, e ensinado, o uso de AMP’S, como WAMP, XAMPP, e outros, que são pacotes prontos, e úteis de início, o uso da funcionalidade de servidor do próprio PHP se demonstra uma alternativa mais produtiva, eficiente e segura, que dispensa, inclusive, o uso do próprio Apache.

Diferentemente do XAMPP, e outros, com o PHO server entra-se no diretório desejado e simplesmente inicia-se o server. O mesmo será acessível a partir do endereço e porta que você definir no momento de inicializar.

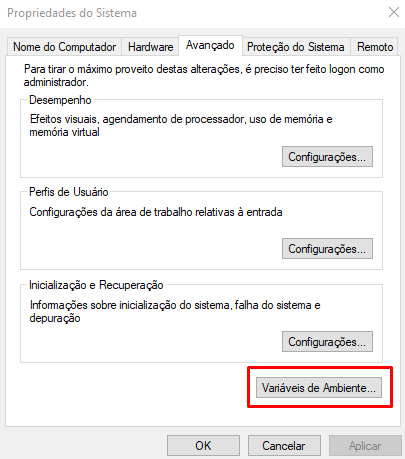
A operação se dá pelo prompt de comando, e por isso, pode parecer ser mais complexo, ou custoso; uma comum falsa impressão.

**O primeiro passo** é realizar o download do PHP, em sua versão mais recente (neste caso, para Windows): [https://windows.php.net/download#php-8.0](https://windows.php.net/download%23php-8.0) . Será baixado um ZIP que deverá ser extraído dentro de uma pasta escolhida, que pode ser chamada de “php” e estar localizada do diretório C.

**O segundo passo** é encontrar, nos arquivos extraídos, o chamado “php.ini-development”, que existe em duas versões, para desenvolvedor, que é esta, e para produtor. Devemos mudar o seu nome para “**php.ini**”, apenas.

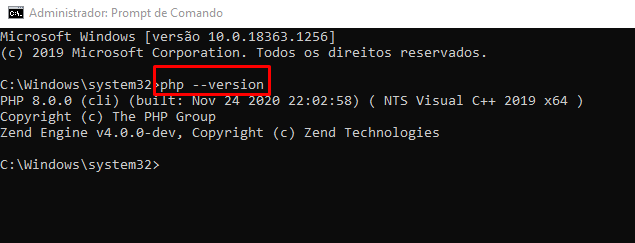
Ele é um arquivo de configurações que é lido pelo interpretador do PHP quando iniciado. Possui diretivas de como o PHP deve ser comportar quanto a uso de memória, caminhos, módulos instalados, entre outras configurações.

**O terceiro passo** é clicar em “meu computador” com o botão direito, propriedades, “configurações avançadas do sistema” em “variáveis de ambiente”.



**O quarto passo** é copiar o caminho do diretório “php” criado, e que contém os arquivos extraídos do Download do PHP. **Em seguida**, na interface aberta, após clicado em “variáveis de ambiente”, clicar em “path”, em “novo”, e adicionar, no novo campo, o caminho copiado. Clicar enter, e sair.

**As variáveis de ambiente são** uma espécie de atalho visível por todo o sistema, onde o Windows e outros programas procuram por executáveis. Com o passo anterior, o PHP é reconhecido como um comando executável, no prompt, e a partir de qualquer local do sistema. Tente:



**O quinto passo** consiste em abrir um bloco de notas como administrador, e após isto, clicar em “arquivo” e “abrir”. O local que deve ser acessado é no diretório C, pasta Windows, System 32, Drivers, etc, e dentro desta última não mostrará nenhum arquivo. A extensão dos arquivos mostrados deve ser mudada para “todos os arquivos”. Feito isto, irá mostrar alguns arquivos, clicar no chamado “hosts”.

Aberto o arquivo, **em uma das linhas estará escrito o IP 127.0.0.1**, com um “#” a esquerda. Deve-se retirar o asterisco.

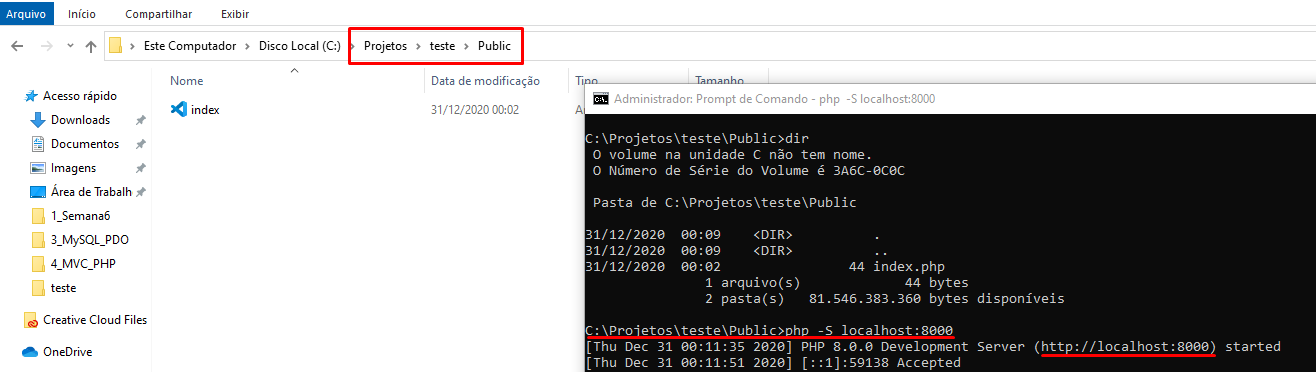
## EXECUTANDO APLICAÇÕES NO SERVIDOR EMBUTIDO

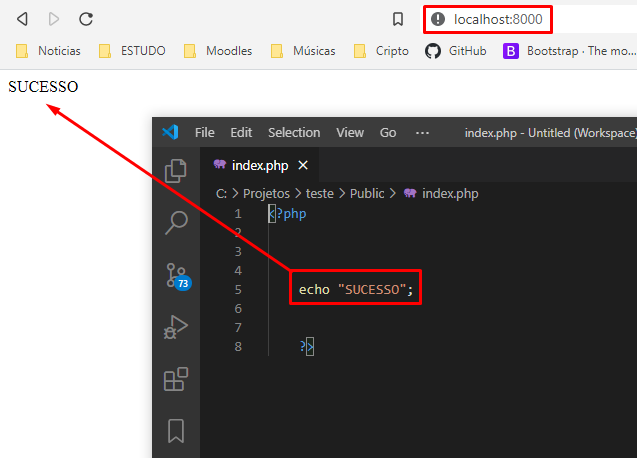
Realizados os passos anteriores, o PHP estará prontamente configurado para executar as aplicações no seu servidor.

**Para rodar uma aplicação**, pelo prompt de comando, se deve acessar o diretório da aplicação, por exemplo “cd C:\projetos\primeiro\_projeto\public”. Neste caso, estaria acessando o projeto de nome “primeiro projeto”, e a sua pasta “public” que contém o arquivo executável “index.php”.

Após isso, para executar a aplicação index, contida na pasta acessada, se escreveria php -S localhost:**porta**, em que **porta** é a escolhida para que a aplicação seja executada. Por exemplo, poderia ser, sendo comuns estas, a porta 8000 ou 8080. Escrito o comando, clicaria-se enter, e em seguida seria impresso na tela o endereço em que a aplicação estaria sendo executada.

O endereço impresso seria copiado e inserido na barra de navegação do Browser, e por fim, a aplicação seria mostrada na tela.





# ESTRUTURA DE DIRETÓRIOS para projetos

A estrutura de diretórios é exatamente o que parece ser: a forma como organizamos as pastas para a estruturação de nossos projetos.

Projetos profissionais, ou em geral, no mercado profissional de desenvolvimento Web, a organização dos diretórios é de suma importância, e apesar de variar, e ser, em seu princípio, uma escolha do desenvolvedor, existem padrões adotados e difundidos que, se aprendidos e assimilados, definitivamente valem como um conhecimento a mais.

Existem excelentes referências no mercado, e por ser, este, em relação ao PHP, cito o guia de estruturação de diretórios do Laravel, um dos principais Frameworks da linguagem. Para saber mais, ver: [https://laravel.com/docs/5.3/structure#the-root-directory](https://laravel.com/docs/5.3/structure%23the-root-directory)

O que não significa, porém, que por ser o foco em PHP, e o guia ser do Laravel, que estes padrões, arquitetural e de diretórios, sirvam apenas para implementações com esta linguagem – pelo contrário, servem para projetos em todos os seus segmentos.

## CRIANDO UMA ESTRUTURA INICIAL

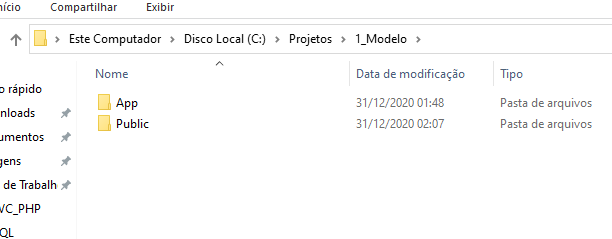
Ao invés de sempre criar projetos do zero, agora, será criada e utilizada uma estrutura base, um “mini framework”, cuja utilidade será total para todos os demais projetos Web possíveis de serem criados. Existem, como citado, outras estruturas, e esta mesma pode ser ampliada e aprimorada com o tempo, a depender da necessidade.

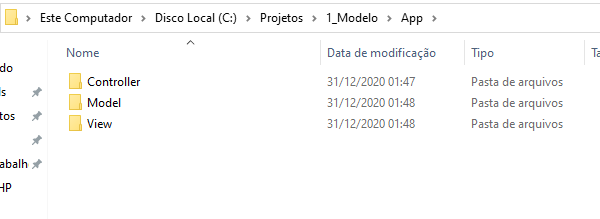
**Deverá ser criada uma pasta para projetos**, preferencialmente no disco local C. Dentre desta, haverão as pastas de cada projeto, e dentro de cada um haverá duas outras pastas, e serão estas a estrutura base a que me refiro – a organização base dos projetos.

**A primeira subpasta se chamará “App”,** e terá dentro de si outras três, sendo cada uma correspondente a uma camada do padrão MVC. Assim, terá a pasta “Controller”, “Model” e “View”.

**A segunda subpasta se chamará “Public”**, e terá dentro de si arquivo “index”, que é o ponto de entrada para todas as solicitações que entram no aplicativo.

**Esta é a principal forma de organização para aplicar em todos os projetos**. As outras pastas adicionais, para ampliar a organização, como descrito no site do Laravel, são implementadas de acordo com a vontade e necessidade do desenvolvedor.





# composer e autoload