**MODEL VIEW CONTROLLER**

<https://www.treinaweb.com.br/blog/o-que-e-mvc>

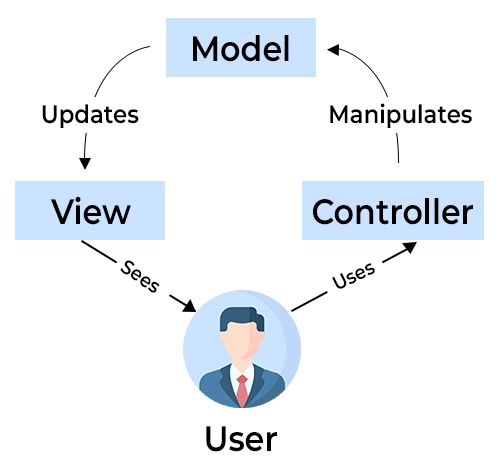
O MVC é um padrão de arquitetura de software. O MVC sugere uma maneira para você pensar na divisão de responsabilidades, principalmente dentro de um software web. O princípio básico do MVC é a divisão da aplicação em três camadas: a camada de interação do usuário (view), a camada de manipulação dos dados (model) e a camada de controle (controller).

**- MODEL** = A responsabilidade dos models é representar o negócio. Também é responsável pelo acesso e manipulação dos dados na sua aplicação (conectar com a fonte de dados, receber os dados, tratar os dados, validar os dados, etc.).

**- VIEW** = A view é responsável pela interface que será apresentada, mostrando as informações do model para o usuário. É basicamente a camada de interação com o usuário (interface de interação com o usuário final do sistema).

**- CONTROLLER** = É a camada de controle, responsável por ligar o model e a view, fazendo com que os models possam ser repassados para as views e vice-versa. Receber as requisições do usuário, processá-las e devolvê-las adequadamente (requere os dados necessários do model e os exibe na view, assim como recebe dados da view, os processa e em seguida os passa para o model).





**EXEMPLO DO SITE**

Vamos utilizar o exemplo de uma página web, onde o usuário pode realizar o cadastro de clientes. Neste caso, provavelmente você teria uma classe chamada cliente.php que contém as informações do cliente que você deseja guardar (como nome, endereço, cidade, etc.). Essa classe seria o seu model.

Aqui, ainda poderíamos acoplar aspectos de manipulação de bancos de dados, concentrando nesta estrutura os métodos para inserir, alterar, excluir e listar os clientes a partir de uma tabela em um banco de dados.

A página HTML seria nossa view, que mostraria, por exemplo, a lista de usuários cadastrados ou mesmo o próprio formulário para cadastro de novos usuários.

O controller faz o meio de campo entre o model e a view. Ele é necessário porque as estruturas presentes com view não deveriam acessar diretamente os models, já que isso poderia criar um acoplamento entre as estruturas de apresentação e definição de negócio: é necessária uma estrutura intermediária para ligá-los.

E aqui entra o controller, que age como uma ponte entre os dois. Você pode ter uma classe dentro do seu projeto PHP para fazer o papel de um controller, realizando a ligação entre models e views.

**EXEMPLO DO VÍDEO**

O que descreverei agora é sobre o código que foi usado de exemplo em um vídeo pelo qual eu estava estudando. Primeiramente, no arquivo main.py, temos a função main que é responsável por criar um model, uma view (que recebe o model como parâmetro) e um controller (que recebe o modelo e o view como parâmetro), em seguida, a função roda o controller. O model se conecta ao banco de dados e possui funções básicas como criação de tabela, inserir elemento, deletar elemento, mostrar elementos. A view cria uma interface gráfica do usuário e recebe o model como parâmetro porque é a partir dele que a view consegue exibir informações na tela que são controladas pelo model. O controller, por fim, recebe tanto a model como view de parâmetro porque ele é a ponte entre os dois, ele vai estar continuamente atualizando a view para aparecer novas informações, adicionar ou deletar um elemento utilizando o model e atualizar na view, entre outras coisas. Acredito que o papel de cada tenha ficado claro usando o exemplo do vídeo como base.