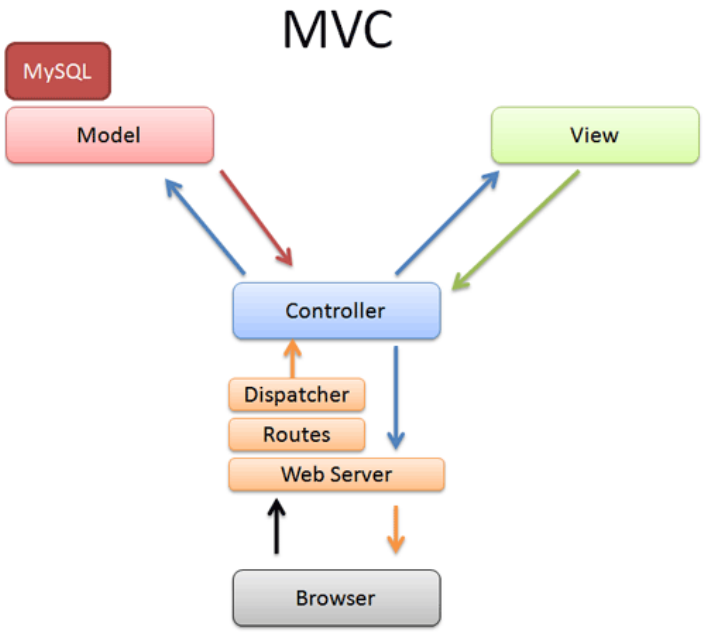
MVC com PHP: Entenda o padrão Model-View-Controller

- Esse curso capacita a:
  - Entender o padrão Model-View-Controller;
  - Filtrar e validar dados do formulário;
  - Usar session e cookies para autenticação;
  - Aplicar boas práticas e usar PSRs;
  - Saber o que são WebServices e como implementar;

Aulas:

1. **Uma entrada para as requisições:**
   - É uma boa prática ter uma entrada única na aplicação web;
   - Se na URL não foi especificado um arquivo, o servidor PHP automaticamente chamará o arquivo index.php (**php –S localhost:8000 –t public**);
   - É boa prática usar URLs amigáveis (mais legíveis);
   - PHP possui uma variável "super global" chamada **$_SERVER** que tem várias informações úteis sobre a requisição, e dela podemos obter a URL: **$_SERVER['PATH_INFO'];**

2. **Entendendo Model View Controller:**
   - Padrão arquitetural MVC:
     - Model: classes com a lógica de negócio e persistência;
     - View: arquivos com o código HTML;
     - Controller classes que ligam a Model e View;
   - FrontController (ou Dispatcher), representa a entrada da aplicação e recebe todas as requisições e decide qual controller específico usar;



3. **HTTP, Formulários e Validação:**
    - **filter_input()** [Filtra dados provenientes das requisições HTTP, como $_GET, $_POST, $_COOKIE, $_SERVER. **filter_var** [Filtra variáveis comuns];
    - **header()** [Permite enviar qualquer tipo de cabeçalho HTTP para o cliente da aplicação];
        - Para ler os dados enviados da requisição existem variáveis "super globais" como **$_REQUEST, $_POST** e **$_GET**;
        - Como funciona o redirecionamento, isto é, chamar automaticamente uma nova URL pelo navegador, para tal, o servidor precisa devolver o cabeçalho **Location** para o navegador usando a função **header**;

4. **Completando o cadastro:**
    - A função **isset(..)** para testar se uma variável existe;
    - A função **extract(..)** que recebe um array associativo e define variáveis para cada chave;
    - Podemos ativar a buffer de saída do buffer com o **ob_start()**;
    - O método **ob_get_contents()** devolve conteúdo do buffer;
    - O método **ob_clean()** limpa o buffer;
    - O método **ob_get_clean** devolve conteúdo e limpa o buffer;

5. **Autenticando usuários:**
    - Função indicada para criptografar senhas e armazena-las no banco: **password_hash('123456', PASSWORD_ARGON2I);**
        - Como gerar uma senha segura usando o algoritmo ARGON2I;
        - Usar no código PHP a função **password_verify($senhaPura, $senhaHash);**
        - Como inserir dados com Doctrine através de SQL;
        - Como validar um email usando a função **filter_input (FILTER_VALIDATE_EMAIL);**

6. **Trabalhando com Sessão:**
    - Por padrão o servidor não guarda informações ou dados entre requisições, por causa do protocolo HTTP que é stateless (sem manter estado);
    - Para armazenar dados entre requisições precisamos usar uma SESSION (sessão);
    - Uma session tem um ID (**PHPSESSID)** associado que fica salvo dentro de um arquivo de texto chamado Cookie;
    - O cookie por sua vez fica salvo no navegador;
    - O navegador automaticamente envia o cookie em cada requisição;
    - Uma sessão precisa ser inicializada explicitamente no PHP pelo comando **session_start()**;
    - **session_start()** precisa ser chamada antes de qualquer saída;

7. **PSRs e Boas Práticas:**
    - Existem vários padrões definidos através de PSRs;
    - Seguindo as PSRs aumentamos a compatibilidade do nosso código entre frameworks e bibliotecas;
    - PSRs:
        - PSR-4: Autoloading;
        - PSR-7: HTTP message interfaces;
        - PSR-11: Container interface;
        - PSR-15: HTTP Server Request Handlers;

8. **Introdução a WebServices:**
    - Um Webservice permite que plataformas e linguagens diferentes se integrem (funcionem em conjunto) e troquem informações;
    - WebServices podem utilizar o protocolo HTTP para acessar uma funcionalidade de uma outra aplicação;
    - O retorno de um WebService normalmente é definido nos formatos XML ou JSON (no lugar de HTML);
    - O cliente de um WebService lê os dados e apresenta da forma que quiser;