DIO
CEZAR
GO

PHP MVC PDO

O QUE É?

- é a utilização da linguagem PHP para a criação de uma estrutura com padrão de projeto MVC com persistência através de PDO;

RECURSOS UTILIZADOS

- composer para *autoloading* das classes;
- diretório Configurations para salvar as configurações do projeto;
  - a classe `Config.php` lê os dados do arquivo `Config.json` que contém as configurações de conexão ao banco de dados;
- namespaces para organização das classes;
- PDO para acesso ao banco de dados;
- JSON para armazenamento das configurações;
- mas no fim... é php vanilla;

EXEMPLO CONFIG.JSON

CONFIG.JSON

```
{
	"config": {
		"database" : {
			"type" : "",
			"host" : "",
			"user" : "",
			"pass" : "",
			"base" : ""
		}
	}
}
```

EXEMPLO CONFIG.PHP

CONFIG.PHP

```
<?php
      namespace App\Configurations;
      class Config{
            public static $config_json = "./Config.json";
            public static $config;
            public static function start(){
            Config::$config =
            json_decode(file_get_contents(__DIR__ . "/" . Config::$config_json),
      true)['config'];
            }
      }
      Config::start();
?>
```

DATABASE

- a classe `DataBase` foi refatorada para ser mais GENÉRICA;
- agora temos uma estrutura que insere, atualiza, recuperar e remove dados de qualquer tabela (passada por parâmetro);
- as funções foram definidas como estáticas, pois não é necessário criar um objeto desta classe;

DATABASE

- `public static function get($table, $id = false)`
  - método para recuperar um ou vários registros (dependendo do id) de uma tabela do banco de dados;
- `public static function save($table, $data)`
  - método para salvar (insert ou update) em uma tabela;
  - a condição para inserção é o dado $data[‘id’] que deve conter ou não o id de um registro;
  - se houver, tenta-se atualizar um item, se não tenta salvá-lo

DATABASE

- `public static function get($table, $id = false)`
  - método para recuperar um ou vários registros (dependendo do id) de uma tabela do banco de dados;
- `public static function save($table, $data)`
  - método para salvar (insert ou update) em uma tabela;
  - a condição para inserção é o dado $data[‘id’] que deve conter ou não o id de um registro;
  - se houver, tenta-se atualizar um item, se não tenta salvá-lo;
- `public static function delete($table, $id)`
  - método para excluir um item;

DATABASE

- `public static function sql($sql, $binds = false)`
  - método para executar um SQL;
  - binds pode ser falso (sem parâmetros a serem colocados no SQL)
  - ou pode conter um array com a mesma ordem dos SQL's a serem utilizados e seus respectivos parâmetros;
  - `DB::sql("SELECT * FROM users WHERE id = :id", array('id' => 1));`

CONTROLLERS

- faz a integração entre as views e as models;
- nesse caso, temos apenas a `ControllerUser.php`;
- nesse caso temos três métodos:
  - `index` → utilizado para renderizar a index que mostra todos os usuários;
  - `save` → utilizado para salvar um usuário específico chamado por alguma página do PHP;
  - `query` → utilizado para exibir apenas um usuário com um único ID;

MODELS

- aplica toda a regra de negócios fazendo a comunicação com a classe DB;
- neste exemplo é uma única model `ModelUser.php`;
- possui as funções:
  - `getData` → obtém todos os registros de usuários;
  - `saveData` → salva um dado de um usuário;
  - `query` → seleciona um único usuário com um id específico;

VIEWS

- formata como será a exibição de uma coleção de dados que chega da model;
- nesse caso, temos apenas `ViewUser.php`;
- possui apenas uma função: `render()` que imprime uma lista de usuários baseando-se no *array* passado em seu parâmetro;

show me the code