**App Lista de Tarefas**

**– Introdução**

- Aplicação que utiliza os processos de CRUD no BD;

- Nos atentaremos a segurança também (regra de negócio e querys).

**Iniciando projeto**

Criando tabelas e inserindo registros;

Criando classe Tarefa.

***C:\xampp\xampp\app\_lista\_tarefas > tarefa.model.php***

class Tarefa

{

    private $id;

    private $id\_status;

    private $tarefa;

    private $data\_cadastro;

    public function \_\_get($atributo)

    {

        return $this->$atributo;

    }

    public function \_\_set($atributo, $valor)

    {

        $this->$atributo = $valor;

    }

}

**Criando a classe TarefaService**

- Será utilizada para a instância de um objeto que vai intermediar a gravação de uma tarefa no banco de dados.

<?php

class TarefaService

{

    public function inserir() //create

    {

    }

    public function recuperar() //read

    {

    }

    public function atualizar() //update

    {

    }

    public function remover() //delete

    {

    }

}

**Enviando dados para o Back-END**

Enviar dados do Front, para o Back através do formulário contido no arquivo nova\_tarefa.php;

Form será submetido via POST.

**nova\_tarefa.php**

<form method="post" action="tarefa\_controller.php">

                                <!--método post + destino(para onde dados serão enviados)-->

                                <div class="form-group">

                                    <input type="text" name="tarefa"

</div>

                            </form>

No **tarefa\_controller.php** no repositório público:

<?PHP

require\_once '../../app\_lista\_tarefas/tarefa\_controller.php'

    ?>

**tarefa\_controller.php** repositório privado>

<?php

echo '<pre>';

print\_r($\_POST);

echo '</pre>';

?>

**Inserindo registros**

**tarefa.model.php**, **tarefa.service.php** e **conexao** terão que ser executados em **tarefa\_controller.php** do diretório público, logo faremos um require.

require “../../app-listatarefas/tarefa.model.php”;

require “../../app-listatarefas/tarefa.service.php”;

require “../../app-listatarefas/conexao.php”;

Ainda em **tarefa\_controller.php** na pasta privada, cria-se instância do objeto tarefa:

$tarefa = new *Tarefa*();

Acessa método mágico \_\_set, definido em tarefa.model, para que na SuperGlobal post no índice *[tarefa]* setar o *atributo tarefa do objeto*:

$tarefa->\_\_set(’tarefa’ , $\_POST[’tarefa’]);

Criar instância de conexão (com bd)

$conexao = new *Conexao*();

Instância de tarefaService, pois é esse objeto que recupera o objeto de tarefa e a conexão para a realização das operações junto ao banco de dados:

$tarefaService = new *TarefaService*();

Criando em tarefa.service, o constructor do objeto que receberá como parâmetro variável de tarefa e a conexão que serão atribuídos aos atributos internos.

private $conexao;

private $tarefa;

public *function \_\_construct*(*Conexao $conexao*, *Tarefa $tarefa*) {

*$this*->conexao = $conexao;

*$this*->tarefa = $tarefa;

}

*Conexao* e *Tarefa*

Para fins de maior segurança.

TarefaService($conexao, $tarefa);

Parâmetros recebidos são atributos criados na class TarefaService.

 $this->conexao = $conexao ->conectar(); //conectar() ----- link de conexão com o banco de dados(necessário)

public function inserir() //create

    {

        $query = 'insert into tb\_tarefas(tarefa)values(:tarefa)';

Criando variável onde ‘tarefa’ recebe um parâmetro nomeado ‘:tarefa’.

        $stmt = $this->conexao->prepare($query);

Criando variável **(stmt)** que recebe variável **(conexao)** que recebe a instância de PDO;

Executa método prepare (Prepara uma instrução SQL para execução), passando a query (tratada como uma variável que faz uma solicitação de informações a um banco de dados).

        $stmt->bindValue(':tarefa', $this->tarefa->\_\_get('tarefa')); //vinculando valor(tarefa do método get) com variável(tarefa) dentro da query

Vincula valor (:tarefa) do método GET, com variável (tarefa) dentro da query.

        $stmt->execute();

Executando query.

header('Location: nova\_tarefa.php?inclusao=1');

Assim que formulário for enviado, usuário é encaminhado para esta url.

<?php

    if (isset($\_GET['inclusao']) && $\_GET['inclusao'] == 1) {

**Isset** – verifica se a variável existe.

Verificando se na super b=global GET existe o valor incluso e se ele é igual a um.

?>

        <div class="bg-success pt-2 text-white d-flex justify-content-center">

            <h5>Tarefa inserida com sucesso! </h5>

        </div>

    <?php } ?>

**Listando registros – 1**

Será feito uma **refatoração**, **para** que **tarefa\_controller.php** seja capaz de **tratar** as **ações de inclusão** (que atualmente é o que ele já faz) **e listagem das tarefas**;

***nova\_tarefa.php***

no action do formulário:

<form method="post" action="tarefa\_controller.php?acao=inserir">

***tarefa\_controller.php*** só é **executado**, **caso** **exista 'acao'** e se **'acao** for **= existir**'.

if (isset($\_GET['acao']) && $\_GET['acao'] == 'inserir') {

Aplicando condição if/else em tarefa\_controller.php.

Criando variável ação para clareza no código

$acao = isset($\_GET['acao']) ? $\_GET['acao']: $acao;

Se houver um índice ação definido na super global GET $acao = isset($\_GET['acao']), esse respectivo valor será trabalhado $\_GET['acao'];

Caso o contrário, aplicação **espera** uma **variável** chamada **ação** : $acao;.

$acao = isset($\_GET['acao']) ? $\_GET['acao']: $acao;

if ($acao == 'inserir') {

}else if ($acao == 'recuperar') {}