**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') {}

**Listando registros – 2**

***tarefa.service.php***

public function recuperar() //read

    {

        $query = 'select id, id\_status, tarefa from tb\_tarefas'; //query recupera 3 colunas específicas de tb\_tarefas

        $stmt = $this->conexao->prepare($query); // stmt recebe conexao (contém-instancia de PDO) e executa método prepare(que prepara uma instrução SQL para execução), passando a query(faz solicitação de informações a um bd)

        $stmt->execute(); //executando query

        return $stmt->fetchAll(PDO::FETCH\_OBJ); //recupera tudo na variável como objeto

    }

***tarefa\_controller.php***

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

    $tarefa = new Tarefa(); //criando instancia do obj tarefa

    $conexao = new Conexao(); //instancia de conexao

    $tarefaService = new TarefaService($conexao, $tarefa); //instancia de tarefa service pois é o obj tarefaservice que recupera objeto e conexao para relização das operações junto ao banco de dados

    $tarefaService->recuperar();//return do método será exibido

}

Dinamizando tarefas em ***todas\_tarefas.php***

<?php foreach ($tarefas as $indice => $tarefa) { ?>

                                <div class="row mb-3 d-flex align-items-center tarefa">

                                    <div class="col-sm-9">

<?= $tarefa->tarefa ?>(<?= $tarefa->id\_status?>)

                                    </div>

</div>

<?php } ?>

Exibindo texto de status da tarefa:

***todas\_tarefas.php***

<?= $tarefa->tarefa ?>(<?= $tarefa->status?>)

***tarefa.service.php***

public function recuperar() //read

    {

        $query = 'select

                    t.id, s.status, t.tarefa

                 from

                    tb\_tarefas as t

                    left join

tb\_status as s on (t.id\_status = s.id)

          ';

tb\_tarefas: t ---- tb\_tarefas as t

priorizando coluna a esquerda (que recupera dados da tabela) --- left Join

tb\_status: s --- tb\_status as s

juntando id\_status de tb\_tarefas com id de tb\_status: para recuperar informações escritas do status --- on (t.id\_status = s.id)

t.id, s.status, t.tarefa --- **id** e **tarefa** vem de **tb\_tarefas**; **status** vem de **tb\_status**.

**Atualizando registros – 1**

Adicionando evento de click no elemento HTML de edição

<i class="fas fa-edit fa-lg text-info" onclick="editar()"></i>

Criando **script** (em header) com função **editar ()** e dentro dela criar elementos.

<head>

    <script>

function editar() {

            //criar form de edição

    let form = document.createElement('form')

            //criar input para entrada do texto

    let inputTarefa = document.createElement('input')

            //criar um button para o envio do form

    let button = document.createElement('button')

        }

    </script>

</head>

Incluindo inputTarefa e button no form:

            form.appendChild(inputTarefa)

            form.appendChild(button)

Foram inclusos padrões nos elementos, como classe, tipo....

button.innerHTML = 'Atualizar'**//adicionando texto ao elemento('dentro do HTML/HTML interno)**

Adicionando ID a DIV de tarefa, para ser possível atuar sobre ela.

<div class="col-sm-9" id="tarefa\_<?= $tarefa->id ?>">

Foi adicionado o id de tarefa, para que ele seja único.

Adicionando ia ao método editar:

<i class="fas fa-edit fa-lg text-info" onclick="editar(<?= $tarefa->id ?>)"></i>

function editar(id) {}

Função recebe ID como parâmetro

Selecionando DIV tarefa (tarefa clicada)

let tarefa = document.getElementById('tarefa\_' + id)//selecionando div pelo ID

Limpando texto da tarefa para a inclusão do form

tarefa.innerHTML=''

Incluir form na página

tarefa.insertBefore(form, tarefa[0])

            //insertBefore -- permite que uma árvore de elementos HTML seja inserida dentro de um outro elemento já renderizado

insertBefore – insert após a renderização da página (método nativo da API do DOM).

Parâmetros: árvore de elementos que será adicionada, nó dentro do elemento selecionado (tarefa) que a árvore de elementos será adicionada

(form, tarefa[0])

Input e button na mesma linha:

            form.className = 'row'

            inputTarefa.className = 'col-9’

            button.className = 'col-3'

Exibindo tarefa no elemento de edição:

onclick="editar(<?= $tarefa->id ?>, '<?= $tarefa->tarefa ?>')"></i>

Encapsulada dentro de aspas simples, pois é uma STRING, diferente do primeiro valor que é INT.

Passando referência da descrição do produto (id);

Criando input Hidden para guardar o id da tarefa:

let inputId = document.createElement('input')

            inputId.type = 'hidden'//'oculto'

            inputId.name = 'Id'

            inputId.value = id

Incluindo elemento no form:

            form.appendChild(inputId)

**Atualizando registros – 2 (BACK-END)**

Adicionando método atualizar ao CONTROLLER:

else if ($acao == 'atualizar') {

    echo 'oi';

}

Encaminhando envio do FORM para a CONTROLLER:

            form.action = 'tarefa\_controller.php?acao=atualizar'

Instanciando tarefa, adicionando id e tarefa e como valor id e tarefa vindos da super global $\_POST:

$tarefa = new Tarefa(); //instância de Tarefa

    $tarefa->\_\_set('id', $\_POST['id']); //id da super global POST como valor

    $tarefa->\_\_set('tarefa', $\_POST['tarefa']); // atribuindo como valor [‘tarefa’] da super global POST

Instância conexão:

    $conexao = new Conexao();//nstanciando conexao em uma variável

No método **atualizar()**:

        $query = 'update tb\_tarefas set tarefa = : tarefa where id = :id';

//$query -- usada para armazenar a consulta SQL

'update tb\_tarefas set tarefa = : tarefa where id = :id';

A própria consulta SQL

'update tb\_tarefas’

Atualizando registros em **tb\_tarefas**

set tarefa = :tarefa where id = :id'

Redefinindo nome da coluna ‘tabela’ para ‘:tabela’ onde ‘id’ é = ‘:id’.

Confirando variável **$stmt** dentro de **atualizar()**.

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

$stmt->bindValue(':tarefa', $this->tarefa->\_\_get('tarefa'));

$stmt->bindValue(':id', $this->tarefa->\_\_get('id'));

return $stmt->execute(); //executando query

Se retornar qualquer valor **superior a 0** é **TRUE**;

Se retornar **0 é FALSE**.

***tarefa\_controller.php***

if ($tarefaService->atualizar()) { //se retorno da atualização é true(1)

        header('location: todas\_tarefas.php');//reencaminha usuário para página de tarefas

    }

Se retorno da atualização for **TRUE** (**1**), **reencaminha usuário** para página de tarefas (**todas\_tarefas.php**).

**Prepare com marcadores ‘?’ e métodos setters com return $this**

No método prepare, podemos no update alterar, ao invés de marcadores nomeados utilizamos: ?;

A ligação (**bind**) será feita a partir da indicação de qual é ‘?’ será substituído (ordem sequencial: 1, 2...).

        $query = 'update tb\_tarefas set tarefa = ? where id = ?';

        $stmt->bindValue(1, $this->tarefa->\_\_get('tarefa'));

  $stmt->bindValue(2, $this->tarefa->\_\_get('id'));

Método \_\_set retorna ele mesmo.

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

    {

        $this->$atributo = $valor;

        return $this;

    }

Código mais enxuto na hora de usar método POST:

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

**Removendo registros**

Adicionando função de click no ícone:

<i class="fas fa-trash-alt fa-lg text-danger" **onclick="remover(<?= $tarefa->id ?>)"></i>**

Função remover;

Redirecionando user para página com todas as tarefas e concatenando com id passado por parâmetro.

function remover(id) {

            location.href = 'todas\_tarefas.php?acao=remover&id' + id;}

Condição de remoção na tarefa\_controller.php:

else if ($acao == 'remover') {

    $tarefa = new Tarefa(); //criando instancia do obj tarefa

    $tarefa->\_\_set('id', $\_GET['id']);

    $conexao = new Conexao(); //instanciando conexao em uma variável

    $tarefaService = new TarefaService($conexao, $tarefa); //instância tarefaService

    $tarefaService->remover();

}

Verificando se ação é igual a ‘remover’;

Instanciando ‘Tarefa’;

Buscando id com método GET;

Instância de conexão;

Instância de tarefaService com conexão e tarefa como parâmetros;

Executando método remover.

***tarefa.service.php***

public function remover()

    { //delete

        $query = 'delete from tb\_tarefas where id = :id';

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

        $stmt->bindValue(':id', $this->tarefa->\_\_get('id'));

        $stmt->execute();

    }

Removendo tarefas do banco de dados:

- **query** com código mySQL;

- variável **stmt** **recebe** **conexão** que **executa** **método** prepare que recebe a query;

- **stmt** faz o **bind** – **atribui** ‘id’ – vindo do get à ‘:id’ – vinda da query;

- executa a query.