**App Send Mail – Iniciando projeto**

**C:**>**xampp**>**xampp**>**ht\_docs**>**app\_send\_mail**

***Enviando dados do front para o back***

                    <form action="processa\_envio.php" method="post">

<input name="para"

<input name="assunto"

<textarea name="mensagem"

Verificando se dados contidos no objeto são válidos, para dar ou não continuidade no processamento da lógica;

Lógica simples: verifica se atributo estão ou não preenchidos.

public function mensagemValida()

    {

        if (empty($this->para) || empty($this->assunto) || empty($this->mensagem)) {//empty verifica se variável está vazia

        //verifica se pelo menos um campo estiver vazio

            return false; //mensagem não é válida

        }

        return true; //se não, mensagem é válida

    }

***Adicionando a biblioteca PHPMailer ao projeto***

- O **processo de envio de emails** é muito comum, muitas **aplicações implementam esse** tipo de **recurso**. Por isso, **existem** muitas **iniciativas prontas**;

- Muitas **bibliotecas podem ser baixadas e utilizadas** dentro do código;

- Essas bibliotecas prontas **encapsulam** toda a **complexidade** **por trás desse processo.**

- Entre tantas, utilizaremos a **PHPMailer** para auxiliar nesse processo.

//importando biblioteca PHPMailer

require "./bibliotecas/PHPMailer/Exception.php";

require "./bibliotecas/PHPMailer/OAuthTokenProvider.php";

require "./bibliotecas/PHPMailer/OAuth.php";

require "./bibliotecas/PHPMailer/PHPMailer.php";

require "./bibliotecas/PHPMailer/POP3.php"; //-->especificações do protocolo de recebimento de email

require "./bibliotecas/PHPMailer/SMTP.php"; //-->especificações do protocolo de envio de email

importando a biblioteca

//namespaces - importante para a configuração de envio de email

use PHPMailer\PHPMailer\PHPMailer;

use PHPMailer\PHPMailer\Exception;

//quando extraímos esses namespaces que reservam classes para a biblioteca PHPMailer (para não haver nenhum tipo de conflito em relação a

nome de recursos dentro da aplicação)

//podemos utilizar esses recursos ao longo do código

Namespaces

if (!$mensagem->mensagemValida()) { //se mensagem for válida (todos os campos estiverem preenchidos)

    echo 'Mensagem é válida';

    die(); //mata processamento do script no ponto em que a instrução é lida

}

try {

<lógica do phpmailer>

} catch (Exception $e) {

    echo "Não foi possível enviar este e-mail! Por favor tente novamente mais tarde";

    echo "Detalhes do erro: {$mail->ErrorInfo}";

}

$mail = new PHPMailer(true);//produz instância de PHPMailer

//criando objeto com base na clasee

**Configurando PHP e enviando e-mails**

Para fazer o envio de mensagens, precisamos de um **Servidor SMTP**;

**SMTP** –Funciona de uma maneira muito parecida com os correios:

É despachada uma determinada correspondência, ele recebe e organiza sua entrega;

Há diversos servidores SMTP disponíveis gratuitamente, como por exemplo o próprio Gmail do Google.

**Front-end se comunica com Back-end** através do protocolo HTTP, enquanto a **aplicação implementa** uma **biblioteca** que**utiliza**um **serviço disponível na internet**;

Através da internet, **utilizando o protocolo SMTP**, **será feita** uma **autenticação no Gmail**, **para** a partir da nossa **aplicação** **disparar um Email** (sendo o Gmail, o **responsável por fazer essa entrega ao destinatário**).

-Login no Gmail

DNS – endereço do Gmail na internet:

$mail->Host = 'smtp.gmail.com';

Equitação e porta do gmail:

$mail->SMTPSecure = PHPMailer::ENCRYPTION\_SMTPS; //Enable implicit TLS encryption

    $mail->Port = 587; //TCP port to connect to; use 587 if you have set `SMTPSecure = PHPMailer::ENCRYPTION\_STARTTLS`

Configurações de remetente e destinatário:

$mail->setFrom('testeanaalmeida@gmail.com', 'Ana Almeida Remetente'); //remetente do email (quem o envia)

    $mail->addAddress('testeanaalmeida@gmail.com', 'Ana Almeida Destinatário'); //destinatário do email (quem o recebe)

setFrom – seta remetente;

addAddress – seta destinatário;

Enviaremos e-mail para nosso próprio e-mail, para fins de teste.

Configs de remetente e destinatário (não será utilizado nesse app).

//$mail->addReplyTo('info@example.com', 'Information');//contato padrão caso o destiatário deseje responder o remetente

    //$mail->addCC('cc@example.com');//adiciona destinatário de cópias

    //$mail->addBCC('bcc@example.com');//cópia oculta

No caso de adicionar anexo ao e-mail (não será utilizado nesse app).

//Attachments - possibilidade de adicionar anexo ao email

    //$mail->addAttachment('/var/tmp/file.tar.gz'); //Add attachments

    //$mail->addAttachment('/tmp/image.jpg', 'new.jpg'); //Optional name

Configurações de conteúdo do e-mail

//Content

    $mail->isHTML(true); //Set email format to HTML

    $mail->Subject = 'Oi, eu sou o assunto.'; //assunto do email

    $mail->Body = 'Oi, eu sou o conteúdo do <strong>e-mail</strong>'; //conteúdo

        $mail->AltBody = 'Oi, eu sou o conteúdo do e-mail';

Formato do e-mail (HTML);

Assunto;

Corpo/Conteúdo do e-mail – pode-se definir um body que contenha tags HTML;

Praticamente todos os clientes de e-mail suportam elementos HTML na página, é uma técnica comum;

Porém, também existem clients que não suportam a renderização de tags HTML, ou até mesmo é possível desabilitar a opção de renderização de tags HTML no client de e-mail;

Por esse motivo, existe a possibilidade de inclusão de um body alternativo onde não exista marcação HTML;

No nosso caso será feito o **teste com marcação HTML**.

Servidores SMTP geralmente são fechados para aplicações exernas, sendo necessário autorizar previamente que apps externos ao domínio tenham acesso a realizar autenticação com aqueles usuários.

**Ajustando configurações de acesso ao SMTP do Gmail**

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Site

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

**A partir daí e-mail será enviado normalmente.**



**Enviando e-mails com base nos parâmetros do front-end**

    $mail->addAddress($mensagem->\_\_get('para')); //destinatário do email (quem o recebe)

    $mail->Subject = $mensagem->\_\_get(‘assunto’); //assunto do email

    $mail->Body = $mensagem->\_\_get('mensagem'); //conteúdo

    $mail->AltBody = 'É necessário  utilizar um client que suporte HTML para ter acesso total ao conteúdo dessa mensagem';

**Desnecessário nesse nosso app**

 $mail->send();

    echo 'Email enviado com sucesso';

**mensagem de sucesso**

**Lógica da aplicação está pronta.**

**Melhorando feedback visual**

Para melhor navegação do usuário final.

header('Location: index.php'); //redirecionando usuário(caso acesso seja inválido) para página inicial do app

encaminhando usuário para página principal no caso de acesso indevido;

public $status = ['codigo\_status' => null, 'descricao\_status' => ''];

try{

$mensagem->status['codigo\_status'] = 1;

    $mensagem->status['descricao\_status'] = 'Email enviado com sucesso';

} catch (Exception $e) {

    $mensagem->status['codigo\_status'] = 2;

    $mensagem->status['descricao\_status'] = 'Não foi possivel enviar este e-mail! Por favor tente novamente mais tarde. Detalhes do erro: ' . $mail->ErrorInfo;

    //lógica que armazene o erro para posterior análise por parte do programador

Atribuindo valores aos índices do array de status da mensagem

<div class="row">

            <div class="col-md-12">

                <?php

                if ($mensagem->status['codigo\_status'] == 1) { ?>

                    <div class="container">

                        <h1 class="display-4 text-success">Sucesso</h1>

                        <p>

                            <?= $mensagem->status['descricao\_status'] ?>

                        </p>

                        <div>

                            <a href="index.php" class="btn btn-success btn-lg mt-5 text-white">Voltar</a>

                        </div>

                    </div>

                <?php } ?>

                <?php

                if ($mensagem->status['codigo\_status'] == 2) { ?>

                    <div class="container">

                        <h1 class="display-4 text-danger">Ops!</h1>

                        <p>

                            <?= $mensagem->status['descricao\_status'] ?>

                        </p>

                        <a href="index.php" class="btn btn-success btn-lg mt-5 text-white">Voltar</a>

                    </div>

                <?php } ?>

            </div>

        </div>

Exibindo mensagem de sucesso ou erro de envio de mensagem;

Com botão que redireciona para página principal.

**Segurança do lado do back-end da aplicação**

Todos os arquivos do projeto estão dentro do diretório público do Apache ***C:\xampp\xampp)***;

Diretório público do HTTP (seja ele qual for) está disponível para a internet (disponível para o mundo);

Qualquer computador, pode fazer requisições HTTP, com acesso ao conteúdo contido dentro do diretório público;

Caso haja algum problema de segurança, erro no servidor ou qualquer outro problema, é possível que os arquivos contidos dentro do diretório público fiquem expostos;

Por esse motivo, arquivos que contenham informações sigilosas (e-mail, senha) precisam estar protegidos, com uma camada de segurança.

Recortar **processa\_envio.php** (onde existem informações sigilosas). E colocar na nova pasta: **C:\xampp\xampp\app\_send\_mail**;

Dentro de ***C:\xampp\xampp\htdocs\app\_send\_mail*** criar o seguinte arquivo:

***processa\_envio.php***

Com o seguinte conteúdo:

<?php

require "../../app\_send\_mail/processa\_envio.php";

?>

Deslocar pasta bibliotecas dentro de **C:\xampp\xampp\app\_send\_mail**;

Ajustar requires – centralizar em: **C:\xampp\xampp\htdocs\app\_send\_mail\processa\_envio.php**

<?php

//importando biblioteca PHPMailer

require "../../app\_send\_mail/bibliotecas/PHPMailer/Exception.php";

require "../../app\_send\_mail/bibliotecas/PHPMailer/OAuthTokenProvider.php";

require "../../app\_send\_mail/bibliotecas/PHPMailer/OAuth.php";

require "../../app\_send\_mail/bibliotecas/PHPMailer/PHPMailer.php";

require "../../app\_send\_mail/bibliotecas/PHPMailer/POP3.php"; //-->especificações do protocolo de recebimento de email

require "../../app\_send\_mail/bibliotecas/PHPMailer/SMTP.php"; //-->especificações do protocolo de envio de email

require "../../app\_send\_mail/processa\_envio.php";

?>

Antes do uso das bibliotecas (em ***processa\_envio.php***), fazer o require do que é necessário em relação a aquelas respectivas bibliotecas

Acerto do caminho dos arquivos da biblioteca