**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.