Para enviar correos electrónicos desde Laravel sin tener un dominio de correo, es posible utilizar servicios de terceros que permiten el envío de correos electrónicos a través de API. Uno de los servicios más populares es Mailgun.

Para usar Mailgun en Laravel, sigue estos pasos:

1. Registra una cuenta en Mailgun y confirma tu dirección de correo electrónico.
2. En el panel de control de Mailgun, ve a la sección de "Dominios" y agrega un nuevo dominio. El dominio puede ser ficticio si no tienes uno propio.
3. Después de agregar el dominio, Mailgun proporcionará una clave de API. Copia la clave API y guárdala en un lugar seguro.
4. En tu aplicación Laravel, instala el paquete guzzlehttp/guzzle con Composer usando el siguiente comando:

javascript

composer require guzzlehttp/guzzle

1. En el archivo .env de tu aplicación Laravel, agrega las siguientes variables de entorno:

makefile

MAIL\_DRIVER=mailgun

MAILGUN\_DOMAIN=tu-dominio-mailgun.com

MAILGUN\_SECRET=tu-clave-de-api-mailgun

MAIL\_FROM\_ADDRESS=tu-direccion-de-correo@tu-dominio.com

MAIL\_FROM\_NAME="Tu Nombre"

Reemplaza tu-dominio-mailgun.com con el dominio que has agregado en Mailgun y tu-clave-de-api-mailgun con la clave de API que has copiado anteriormente. También, actualiza tu-direccion-de-correo@tu-dominio.com y Tu Nombre con tu dirección de correo electrónico y tu nombre.

1. En el archivo config/mail.php, agrega la siguiente configuración para el driver de Mailgun:

bash

'mailgun' => [

'transport' => 'mailgun',

'domain' => env('MAILGUN\_DOMAIN'),

'secret' => env('MAILGUN\_SECRET'),

],

1. Ahora puedes enviar correos electrónicos desde tu aplicación Laravel usando el servicio Mailgun. Por ejemplo, puedes enviar un correo electrónico de prueba en un controlador con el siguiente código:

php

use Illuminate\Support\Facades\Mail;

use App\Mail\PruebaCorreo;

class CorreoController extends Controller

{

public function enviarCorreo()

{

$datos = [

'titulo' => 'Prueba de correo electrónico',

'contenido' => 'Este es un correo electrónico de prueba enviado desde Laravel utilizando Mailgun.',

];

Mail::to('destinatario@dominio.com')->send(new PruebaCorreo($datos));

return 'El correo electrónico ha sido enviado.';

}

}

Asegúrate de reemplazar destinatario@dominio.com con la dirección de correo electrónico del destinatario y crear la clase PruebaCorreo para definir el contenido del correo electrónico.

aquí está el código de la clase PruebaCorreo de ejemplo:

php

<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Mail\Mailable;

use Illuminate\Queue\SerializesModels;

class PruebaCorreo extends Mailable

{

use Queueable, SerializesModels;

public $datos;

/\*\*

\* Create a new message instance.

\*

\* @return void

\*/

public function \_\_construct($datos)

{

$this->datos = $datos;

}

/\*\*

\* Build the message.

\*

\* @return $this

\*/

public function build()

{

return $this->subject($this->datos['titulo'])

->view('correos.prueba')

->with('contenido', $this->datos['contenido']);

}

}

En este ejemplo, la clase PruebaCorreo toma un arreglo de datos en su constructor y lo almacena en una propiedad pública $datos. La función build() define cómo se construirá el correo electrónico y define el asunto y la vista del correo electrónico. En este caso, el asunto es el título del correo electrónico almacenado en $datos['titulo'] y la vista del correo electrónico se encuentra en el archivo correos/prueba.blade.php. En la vista, el contenido del correo electrónico se pasa a través de la variable $contenido, que se encuentra en $datos['contenido'].

aquí está el código de ejemplo para la vista correos/prueba.blade.php:

php

<!DOCTYPE html>

<html>

<head>

<title>{{ $datos['titulo'] }}</title>

</head>

<body>

<h1>{{ $datos['titulo'] }}</h1>

<p>{{ $contenido }}</p>

</body>

</html>

En esta vista, se utiliza la sintaxis Blade de Laravel para imprimir el título y el contenido del correo electrónico. El título se imprime como un encabezado <h1> y el contenido se imprime como un párrafo <p>. Note que la variable $datos que se utiliza en la vista es la misma que se pasa al constructor de la clase PruebaCorreo y que contiene el título y el contenido del correo electrónico.