**Laboratoire : Gestion Complète des Emails avec Laravel**

**Objectif**

Apprendre à configurer et utiliser Laravel pour envoyer des emails via différents pilotes (SMTP, Mailgun, Sendmail, etc.). Explorer toutes les fonctionnalités avancées, telles que les pièces jointes, les copies cachées, les emails Markdown et les files d'attente.

**Étape 1 : Configuration Initiale**

**1.1 Fichier .env**

Configurez le fichier **.env** pour définir les paramètres d'envoi d'email. Voici une configuration typique :

plaintext

Copier

MAIL\_MAILER=smtp

MAIL\_HOST=smtp.mailtrap.io

MAIL\_PORT=2525

MAIL\_USERNAME=your-mailtrap-login

MAIL\_PASSWORD=your-mailtrap-password

MAIL\_ENCRYPTION=tls

MAIL\_FROM\_ADDRESS="admin@contact.com"

MAIL\_FROM\_NAME="${APP\_NAME}"

* **MAIL\_MAILER** : Choisissez le pilote (**smtp**, **sendmail**, **mailgun**, etc.).
* **MAIL\_HOST** , **MAIL\_PORT** , **MAIL\_USERNAME** , **MAIL\_PASSWORD** : Informations spécifiques au service choisi.
* **MAIL\_ENCRYPTION** : Utilisez **tls** ou **ssl** pour sécuriser la connexion.
* **MAIL\_FROM\_ADDRESS** , **MAIL\_FROM\_NAME** : Adresse et nom par défaut de l'expéditeur.

**1.2 Installation de Dépendances**

Si vous utilisez un service comme Mailgun ou Postmark, installez les packages nécessaires via Composer :

bash

Copier

composer require laravel/mailgun-sdk

**Étape 2 : Création d'une Classe Mailable**

**2.1 Génération de la Classe**

Créez une classe Mailable pour représenter votre email :

bash

Copier

php artisan make:mail ContactEmail

Cela crée un fichier **ContactEmail.php** dans **app/Mail**.

**2.2 Structure de la Classe**

Voici un exemple complet de la classe **ContactEmail** :

php

Copier

namespace App\Mail;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Mail\Mailable;

use Illuminate\Queue\SerializesModels;

use Illuminate\Mail\Mailables\Address;

use Illuminate\Mail\Mailables\Envelope;

use Illuminate\Mail\Mailables\Content;

use Illuminate\Mail\Mailables\Attachment;

class ContactEmail extends Mailable

{

use Queueable, SerializesModels;

public $data;

/\*\*

\* Create a new message instance.

\*

\* @param array $data

\*/

public function \_\_construct(array $data)

{

$this->data = $data;

}

/\*\*

\* Get the message envelope.

\*

\* @return \Illuminate\Mail\Mailables\Envelope

\*/

public function envelope()

{

return new Envelope(

from: new Address('admin@contact.com', 'John Lennon'),

subject: 'Nouvelle Demande de Contact',

);

}

/\*\*

\* Get the message content definition.

\*

\* @return \Illuminate\Mail\Mailables\Content

\*/

public function content()

{

return new Content(

view: 'emails.contact',

with: [

'name' => $this->data['name'],

'email' => $this->data['email'],

'message' => $this->data['message'],

],

);

}

/\*\*

\* Attachments for the message.

\*

\* @return array

\*/

public function attachments()

{

return [

Attachment::fromPath('/path/to/file.pdf')->as('document.pdf')->withMime('application/pdf'),

];

}

}

* **Propriété $data** : Stocke les données transmises à l'email.
* **Méthode envelope()** : Configure l'expéditeur et le sujet.
* **Méthode content()** : Spécifie la vue utilisée pour le contenu et passe les données.
* **Méthode attachments()** : Ajoute des pièces jointes.

**Étape 3 : Création de la Vue Email**

Créez une vue Blade pour le contenu de l'email dans **resources/views/emails/contact.blade.php** :

html

Copier

<!DOCTYPE html>

<html>

<head>

<title>Nouvelle Demande de Contact</title>

</head>

<body>

<h1>Une nouvelle demande de contact a été reçue :</h1>

<p><strong>Nom :</strong> {{ $name }}</p>

<p><strong>Email :</strong> {{ $email }}</p>

<p><strong>Message :</strong> {{ $message }}</p>

</body>

</html>

**Étape 4 : Envoi d'Email depuis le Contrôleur**

Modifiez votre contrôleur pour envoyer l'email lorsqu'un utilisateur soumet une demande de contact :

php

Copier

use Illuminate\Support\Facades\Mail;

use App\Mail\ContactEmail;

public function sendContactEmail(Request $request)

{

// Récupérer les données du formulaire

$data = $request->validate([

'name' => 'required|string',

'email' => 'required|email',

'message' => 'required|string',

]);

// Envoyer l'email

Mail::to('administrateur@chezmoi.com')->send(new ContactEmail($data));

// Rediriger vers une page de confirmation

return view('confirm');

}

**Étape 5 : Options Avancées**

**5.1 Copie et Copie Cachée**

Ajoutez des copies ou des copies cachées à l'email :

php

Copier

Mail::to('administrateur@chezmoi.com')

->cc('support@chezmoi.com')

->bcc('backup@chezmoi.com')

->send(new ContactEmail($data));

**5.2 Emails Markdown**

Laravel permet de créer des emails en Markdown pour un rendu simplifié. Créez un fichier Markdown dans **resources/views/emails/markdown/contact.md** :

markdown

Copier

@component('mail::message')

**# Nouvelle Demande de Contact**

Nom : {{ $name }}

Email : {{ $email }}

Message : {{ $message }}

@component('mail::button', ['url' => 'http://votre-site/contact'])

Voir le Site

@endcomponent

Merci,<br>

{{ config('app.name') }}

@endcomponent

Puis modifiez la méthode **content()** dans la classe Mailable :

php

Copier

public function content()

{

return new Content(

markdown: 'emails.markdown.contact',

with: [

'name' => $this->data['name'],

'email' => $this->data['email'],

'message' => $this->data['message'],

],

);

}

**Étape 6 : Test de l'Email**

**6.1 Mode Log**

En mode développement, configurez le pilote **log** dans **.env** :

plaintext

Copier

MAIL\_MAILER=log

Les emails ne seront pas envoyés mais enregistrés dans les logs. Vérifiez-les avec la commande :

bash

Copier

tail -f storage/logs/laravel.log

**6.2 MailTrap**

Utilisez MailTrap pour tester les emails sans les envoyer réellement. Configurez-le dans **.env** :

plaintext

Copier

MAIL\_DRIVER=smtp

MAIL\_HOST=sandbox.smtp.mailtrap.io

MAIL\_PORT=2525

MAIL\_USERNAME=your-mailtrap-login

MAIL\_PASSWORD=your-mailtrap-password

MAIL\_ENCRYPTION=null

Accédez ensuite au tableau de bord MailTrap pour visualiser les emails reçus.

**6.3 Route de Test**

Créez une route pour prévisualiser l'email directement dans le navigateur :

php

Copier

Route::get('/test-email', function () {

return new \App\Mail\ContactEmail([

'name' => 'Durand',

'email' => 'durand@chezlui.com',

'message' => 'Votre site est magnifique !',

]);

});

**Étape 7 : Optimisation avec les Files d'Attente**

Pour éviter de ralentir l'application lors de l'envoi d'emails volumineux, utilisez les files d'attente :

php

Copier

Mail::to('administrateur@chezmoi.com')

->queue(new ContactEmail($data));

Assurez-vous d'avoir configuré un système de file d'attente (Redis, Beanstalkd, etc.) et lancez le worker :

bash

Copier

php artisan queue:work

**Conclusion**

Ce laboratoire vous a permis d'explorer toutes les fonctionnalités d'envoi d'emails avec Laravel, y compris :

* La configuration des paramètres.
* La création de classes Mailable.
* Les pièces jointes, copies et copies cachées.
* Les emails Markdown.
* Les outils de test (Log, MailTrap).
* L'utilisation des files d'attente pour optimiser les performances.