Guide d'installation de Mailer Laravel avec QR Code - Notification par Email de Connexion Laravel

QUANTA PIXEL

06/03/2025

Introduction

Ce guide vous expliquera comment configurer une notification par email dans Laravel, avec un QR code attaché. Il comprend les étapes nécessaires pour configurer vos paramètres d'email, installer les packages nécessaires et créer une classe Mailable pour envoyer des emails avec des QR codes en pièce jointe.

Prérequis

Avant de commencer, assurez-vous d'avoir les éléments suivants :

- **Projet Laravel installé** : Vous devez avoir un projet Laravel déjà configuré.
- **Mot de passe d'application Gmail généré** : Si vous souhaitez utiliser Gmail pour envoyer des emails, vous devez avoir généré un **mot de passe d'application** pour l'authentification SMTP. Générez un mot de passe d'application Gmail ici.

Étape 1 : Configurer les paramètres de l'email

Afin d'envoyer des emails depuis votre application Laravel, vous devez configurer les paramètres d'email dans le fichier `.env`.

1.1 Mettez à jour le fichier `.env`

Ouvrez le fichier `.env` de votre projet et configurez les paramètres d'email comme suit :

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=quantapixel@gmail.com
MAIL_PASSWORD=qwer azer mcxv kufy  # Utilisez votre mot de passe d'application Gmail
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=quantapixel@gmail.com
MAIL_FROM_NAME="${APP_NAME}"
```

Remplacez `MAIL_USERNAME` et `MAIL_PASSWORD` par vos informations Gmail (utilisez le mot de passe d'application si vous utilisez Gmail). `MAIL_FROM_ADDRESS` correspond à l'adresse email depuis laquelle les messages seront envoyés et `MAIL_FROM_NAME` est le nom qui apparaîtra dans l'email.

Étape 2 : Installer le package nécessaire pour la génération de QR Code

Vous devez installer le package `endroid/qr-code` pour générer des QR codes.

Exécutez la commande suivante pour l'installer :

```
composer require endroid/qr-code
```

Ce package vous permettra de générer des QR codes à inclure dans l'email.

Étape 3 : Générer la classe Mailable

Ensuite, vous devrez créer une **classe Mailable** qui s'occupera de l'envoi des emails avec le QR code en pièce jointe.

3.1 Créez la classe Mailable

Exécutez la commande Artisan suivante pour générer la classe `UserInfoQRMail` :

```
php artisan make:mail UserInfoQRMail
```

Cela générera un nouveau fichier dans `app/Mail/UserInfoQRMail.php`. Vous pouvez maintenant ouvrir et modifier ce fichier pour inclure la génération du QR code et son ajout en pièce jointe.

Étape 4 : Modifier la classe Mailable

Mettez à jour la classe Mailable générée pour générer un QR code et l'ajouter à l'email.

Code Exemple :

```
<?php

namespace App\Mail;

use App\Models\User;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;
use Endroid\QrCode\QrCode;
use Endroid\QrCode\Writer\PngWriter;
use Illuminate\Support\Facades\Storage;

class UserInfoQRMail extends Mailable
{
    use Queueable, SerializesModels;

    public $user;

    /**
     * Crée une nouvelle instance de message.
     *
     * @param User $user
     */
    public function __construct(User $user)
    {
        $this->user = $user;
    }

    /**
     * Construire le message.
     *
     * @return $this
     */
    public function build()
    {
        // Données pour générer le QR code
        $qrData = "Nom: {$this->user->f_name} {$this->user->l_name}\nEmail: {$this->user->email}\nLocat

        // Créer une nouvelle instance de QR Code
        $qrCode = new QrCode($qrData);
```

```
        // Utiliser PngWriter pour générer l'image du QR code
        $writer = new PngWriter();

        // Définir le chemin où enregistrer l'image QR code
        $qrFilePath = storage_path('app/public/qrcodes/user_' . $this->user->id . '_qr.png');

        // Créer le répertoire s'il n'existe pas
        $directory = storage_path('app/public/qrcodes');
        if (!file_exists($directory)) {
            mkdir($directory, 0755, true);
        }

        // Générer et enregistrer l'image du QR code
        $qrImageData = $writer->write($qrCode);
        file_put_contents($qrFilePath, $qrImageData->getString());

        // Construire l'email avec le QR code en pièce jointe
        return $this->subject('QR Code avec les informations de l\'utilisateur')
                    ->view('userInfoQR') // Assurez-vous que cette vue existe
                    ->attach($qrFilePath, [
                        'as' => 'user_info_qr.png',
                        'mime' => 'image/png',
                    ]);
    }
}
```

Explication du Code

- **Génération du QR Code** : Le QR code est généré en utilisant la classe `QrCode`, qui encode les informations de l'utilisateur telles que le nom, l'email et la localisation.
- **PngWriter** : Nous utilisons le `PngWriter` pour générer l'image du QR code au format PNG.
- **Chemin du fichier** : L'image QR code générée est enregistrée dans le répertoire `storage/app/public/qrcodes/`.
- **Pièce jointe** : Le QR code est attaché à l'email via la méthode `attach()`, il sera envoyé en tant que pièce jointe au format PNG.

Étape 5 : Créer la Vue de l'Email

Assurez-vous de créer la vue (`resources/views/userInfoQR.blade.php`) pour le contenu de l'email. Cette vue sera utilisée dans le corps de l'email.

Exemple de Vue Email (`userInfoQR.blade.php`)

```
<!DOCTYPE html>
<html>
<head>
    <title>QR Code des informations de l'utilisateur</title>
</head>
<body>
    <h1>Informations de l'utilisateur</h1>
    <p>Nom : {{ $user->f_name }} {{ $user->l_name }}</p>
    <p>Email : {{ $user->email }}</p>
    <p>Localisation : {{ $user->location }}</p>
    <p>Le QR code avec les informations de l'utilisateur est en pièce jointe.</p>
</body>
</html>
```

Cette vue affiche les informations de base de l'utilisateur et informe le destinataire qu'un QR code est attaché à l'email.

Étape 6 : Envoyer l'Email

Une fois que tout est configuré, vous pouvez envoyer l'email avec le QR code en pièce jointe. Vous pouvez déclencher l'envoi de l'email depuis un contrôleur ou toute autre partie de votre application.

Exemple pour envoyer l'email

```
use App\Mail\UserInfoQRMail;
use Illuminate\Support\Facades\Mail;
use App\Models\User;

// Récupérer l'utilisateur (exemple)
$user = User::find(1); // Remplacez par l'utilisateur réel

// Envoyer l'email
Mail::to($user->email)->send(new UserInfoQRMail($user));
```

Cela enverra l'email à l'utilisateur spécifié avec le QR code en pièce jointe.

Conclusion

Vous avez maintenant configuré avec succès Laravel pour envoyer un email avec un QR code en pièce jointe. Le processus a consisté à configurer les paramètres d'email, installer le package nécessaire pour la génération de QR codes et créer une classe Mailable pour envoyer l'email avec le QR code. Vous pouvez personnaliser cette fonctionnalité en ajoutant plus de contenu dynamique ou en modifiant la logique de génération du QR code selon vos besoins.