

Live_coding : Comprendre et implémenter un système de Jobs et Queues dans Laravel pour une gestion asynchrone efficace.


Étape 1 : Installation de Laravel

Si tu n'as pas encore un projet Laravel, crée-en un en exécutant cette commande :

```
composer create-project laravel/laravel live_coding
```

Ensuite, entre dans le dossier du projet et ouvre le serveur :

```
cd live_coding  
  
php artisan serve
```

 Vérifie que Laravel fonctionne en allant sur <http://127.0.0.1:8000>.

Tu devrais voir la page d'accueil de Laravel.

Étape 2 : Configuration de la Base de Données et des Queues

1 Modifier `.env` pour configurer la base de données

Dans le fichier `.env`, remplace ces lignes avec les bonnes informations :

```
DB_CONNECTION=pgsql
DB_HOST=127.0.0.1
DB_PORT=5432
DB_DATABASE=live_coding
DB_USERNAME=postgres
DB_PASSWORD= # Mets ton mot de passe ici
```

🔧 Vérifie que ta base de données existe dans phpMyAdmin ou avec MySQL Workbench. Si elle n'existe pas, crée-la :

```
CREATE DATABASE live_coding;
```

2 Configurer le driver de Queue

Dans `.env`, mets :

```
QUEUE_CONNECTION=database
```

📌 Explication : Laravel gère les files d'attente avec plusieurs drivers (sync, database, redis...), ici on utilise database.

3 Créer la table des Jobs

Laravel a une migration prête pour ça :

```
php artisan queue:table

php artisan migrate
```


Étape 3 : Configuration de l'envoi d'email avec Mailtrap

1 Configurer Mailtrap

Crée un compte sur [Mailtrap](#) et récupère tes identifiants SMTP.

Dans `.env`, remplace ces lignes :

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=ton_mailtrap_username
MAIL_PASSWORD=ton_mailtrap_password
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=hashim@gmail.com
MAIL_FROM_NAME="live_coding"
```

 Explication : Mailtrap permet de tester l'envoi d'emails sans envoyer de vrais emails.

Étape 4 : Créer un Mailable pour l'email de bienvenue

Un **Mailable** est une classe qui définit le contenu et l'envoi d'un email dans Laravel.

1 Générer le Mailable

```
php artisan make:mail WelcomeEmail
```

2 Modifier `app/Mail/WelcomeEmail.php`

```

<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;

class WelcomeEmail extends Mailable implements ShouldQueue
{
    use Queueable, SerializesModels;

    public $user;

    public function __construct($user)
    {
        $this->user = $user;
    }

    public function build()
    {
        return $this->subject('Bienvenue sur notre plateforme !')
            ->view('emails.welcome')
            ->with([
                'name' => $this->user->name,
            ]);
    }
}

```

3 Créer la vue de l'email

```
mkdir -p resources/views/emails
```

```
touch resources/views/emails/welcome.blade.php
```

Ajoute ce contenu dans `resources/views/emails/welcome.blade.php` :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bienvenue</title>
</head>
<body>
  <h1>Bienvenue, {{ $name }} !</h1>
  <p>Nous sommes ravis de vous compter parmi nous.</p>
</body>
</html>
```

Étape 5 : Créer un Job pour envoyer l'email

1 Générer le Job

```
php artisan make:job SendWelcomeEmail
```

2 Modifier `app/Jobs/SendWelcomeEmail.php`

```
<?php

namespace App\Jobs;

use App\Mail\WelcomeEmail;
```

```

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Queue\SerializesModels;
use Illuminate\Support\Facades\Mail;

class SendWelcomeEmail implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    protected $user;

    public function __construct($user)
    {
        $this->user = $user;
    }

    public function handle()
    {
        Mail::to($this->user->email)->send(new WelcomeEmail($this->user));
    }
}

```

Étape 6 : Créer une API pour déclencher le Job

1 Créer le contrôleur

```
php artisan make:controller UserController
```

2 Modifier `app/Http/Controllers/UserController.php`

```

<?php

namespace App\Http\Controllers;

use App\Jobs\SendWelcomeEmail;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function register(Request $request)
    {
        $request->validate([
            'name' => 'required|string',
            'email' => 'required|email|unique:users,email',
        ]);

        $user = (object) [
            'name' => $request->name,
            'email' => $request->email,
        ];

        // Déclencher le Job
        SendWelcomeEmail::dispatch($user);

        return response()->json(['message' => 'Utilisateur enregistré, email en
cours d'envoi.']);
    }
}

```

3 Ajouter la route dans `routes/api.php`

```

use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;

```

```
Route::post('/register', [UserController::class, 'register']);
```

Étape 7 : Exécuter les Workers et Tester avec Postman

1 Lancer le Worker

Dans un premier terminal, lance :

```
php artisan queue:work
```

2 Tester avec Postman

- **Méthode :** POST
- **URL :** `http://127.0.0.1:8000/api/register`
- **Body (JSON) :**

```
{
  "name": "hicham",
  "email": "hicham@gmail.com"
}
```

- **Résultat attendu :**

```
{
  "message": "Utilisateur enregistré, email en cours d'envoi."
}
```

- **Vérifie sur Mailtrap** si l'email a bien été envoyé. 🎉