LARAVEL MAILS

In Laravel, sending emails is made easy through the use of the built-in Mail library. Laravel provides a clean and expressive API for sending emails and includes drivers for SMTP, Mailgun, SparkPost, and others. Here's a basic overview of sending emails in Laravel:

1. Configuration:

Start by configuring your mail settings in the `.env` file. Set the `MAIL_DRIVER` and other related variables according to your email service provider.

```env
MAIL_DRIVER=smtp
MAIL_HOST=smtp.example.com
MAIL_PORT=587
MAIL_USERNAME=your_email@example.com
MAIL_PASSWORD=your_email_password
MAIL_ENCRYPTION=tls
```

2. Creating Mailables:

Mailables represent email messages in Laravel. You can create a new Mailable using the Artisan command:

```bash
php artisan make:mail MyMail
```

This generates a Mailable class in the `App\Mail` namespace. You can customize this class to define the email's subject, view, and data.

3.Configuring Mailables:

Set up your Mailable class to define the email subject, view, and data to be passed to the view.

```php
class MyMail extends Mailable
{
    public function build()
    {
        return $this->subject('My Subject')
                    ->view('emails.my_mail')
                    ->with(['data' => $someData]);
```

```
    }
}
```

4. Creating Email Views:

Create Blade views for your email messages in the `resources/views/emails` directory. For example, create `my_mail.blade.php`:

```blade
<p>Hello, {{ $data['name'] }}!</p>
```

5. Sending Emails:

Use the `Mail` facade to send emails in your controllers, services, or wherever you need.

```php
use App\Mail\MyMail;
use Illuminate\Support\Facades\Mail;

Mail::to('recipient@example.com')->send(new MyMail($data));
```

6. Queueing Emails:

To offload email sending to a queue, use the `queue` method. Ensure your queue worker is running.

```php
Mail::to('recipient@example.com')->queue(new MyMail($data));
```

7. Attachments:

You can attach files to your emails using the `attach` method.

```php
public function build()
{
    return $this->attach($pathToFile, [
        'as' => 'name.pdf',
        'mime' => 'application/pdf',
    ]);
}
```

8. Markdown Mailables (Optional):

Laravel supports Markdown for email templates. Create Markdown Mailables using the Artisan command:

```bash
php artisan make:mail MyMarkdownMail --markdown=emails.my_markdown_mail
```

Configure the Mailable class similar to regular Mailables.

9. Testing Emails:

Laravel provides a `MailFake` for testing. In your test, use the `Mail::fake` method to swap the Mail facade with the fake.

```php
use Illuminate\Support\Facades\Mail;

Mail::fake();

// Perform actions that trigger email sending

Mail::assertSent(MyMail::class, function ($mail) use ($data) {
```

```
    return $mail->hasTo('recipient@example.com') &&
           $mail->subject === 'My Subject' &&
           $mail->viewData['data'] === $data;
});
```

These are the basic steps for sending emails in Laravel. Utilizing Mailables and the Mail facade simplifies the process and makes email sending manageable and testable.