Публикация на тему

Отправка Email в Laravel

В публикации рассматривается использование Gmail SMTP для отправки email сообщений в Laravel

Анотация

-

-

Автор

Михалькевич Александр Викторович

Публикация

Наименование Отправка Email в Laravel

Автор А.В.Михалькевич

Специальность В публикации рассматривается использование Gmail SMTP для отправки email сообщений в Laravel,

Анотация

-

Anotation in English

-

Ключевые слова laravel, gmail, smtp, mailtrap, Mailtrap, PHP, mail

Количество символов 28024

Содержание

Введение

Если вы планируете использовать Laravel для отправки электронной почты через Gmail в своем текущем/будущем проекте Laravel, продолжайте читать, поскольку в этой статье мы шаг за шагом покажем вам, как это делается!

1 Отправка электронной почты с помощью Laravel

Laravel, известный своим выразительным и элегантным синтаксисом, является одним из наиболее часто используемых PHP-фреймворков. Laravel часто является идеальным выбором для создания полнофункциональных веб-приложений, в том числе и для отправки электронной почты.

Для отправки электронной почты можем использовать SMTP-сервера поставщика услуг электронной почты, например Gmail.

Подробнее о Laravel - http://erud.by/laravel

1 .1 Шаги, которые необходимо предпринять перед началом процесса отправки электронной почты в Laravel

Хотя отправка электронных писем в Laravel не является слишком сложной задачей, перед началом процесса важно ознакомиться со следующими основополагающими понятиями.

1 .2 Создать Laravel-приложение

Если у вас ещё нет приложения Laravel, вы можете использовать следующие шаги для его создания. Создать приложение можно с помощью терминала следующей командой:

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

Далее преходим в папку Laravel

```
cd laravel
```

и запускаем проект

```
php artisan serve
```

1 .3 Настройки

В рамках Laravel вы можете настроить службы электронной почты в `config/mail.php` файл. Этот файл содержит массив конфигурации почтовых программ с примерами записей конфигурации для различных почтовых драйверов/транспортов, поддерживаемых Laravel.

Значение конфигурации по умолчанию в этом файле определяет почтовую программу по умолчанию при отправке электронного письма из вашего приложения Laravel.

Благодаря множеству почтовых программ, которые можно настроить в `config/mail.php`, вы можете использовать разные службы отправки электронной почты для разных типов электронных писем.

1 .4 Конфигурация smtp

Первым шагом при отправке электронных писем с использованием Gmail SMTP является добавление конфигурации SMTP Gmail в файл `.env` вашего приложения.

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=465
MAIL_USERNAME=mygoogle@gmail.com
MAIL_PASSWORD=**********
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=mygoogle@gmail.com
MAIL_FROM_NAME="${APP_NAME}"
```

1 .5 Конфигурация почтового сервера

Чтобы указать отправителя — адрес электронной почты «from» и имя — у вас есть два варианта:

1. вы можете использовать объект «Envelope» сообщения или установить глобальный адрес «from».

```
use Illuminate\Mail\Mailables\Address;
use Illuminate\Mail\Mailables\Envelope;

/**
* Get the message envelope.
*
* @return \Illuminate\Mail\Mailables\Envelope
*/
public function envelope()
{
   return new Envelope(
       from: new Address('example@example.com', 'Test Sender'),
       subject: 'Test Email',
   );
```

```
}
```

2.Определить отправителя в файле `config/mail.php` в глобальной переменной:

```
'from' => ['address' => 'example@example.com', 'name' => 'App Name']
```

Примечание. Если вы хотите использовать один и тот же адрес «from» для всех писем, отправляемых вашим приложением, рекомендуется использовать глобальную переменную.

1 .6 Создание и использование Mail класса

После настройки smtp, можем создать Mail-класс

Далее создаем Mail-класс с помощью artisan-команды:

```
php artisan make:mail MyTestEmail
```

В папке `app/Mail` появится следующий файл:

```
namespace App\Mail;

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

class MyTestEmail extends Mailable
{
    use Queueable, SerializesModels;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct()
    {
        //
    }

    /**
     * Get the message envelope.
     *
     * @return \Illuminate\Mail\Mailables\Envelope
     */

    public function envelope()
    {
        return new Envelope(
            subject: 'My Test Email',
```

```
        );
    }
 /**
     * Get the message content definition.
     *
     * @return \Illuminate\Mail\Mailables\Content
     */

    public function content()
    {
        return new Content(
            view: 'view.name',
        );
    }

    /**
     * Get the attachments for the message.
     *
     * @return array
     */
    public function attachments()
    {
        return [];
    }
}
```

После создания почтового класса вы можете просмотреть его содержимое и настроить класс, используя следующие методы:

> Envelope – Возвращает экземпляр класса `Illuminate\Mail\Mailables\Envelope`, который позволяет вам определить тему и получателей электронного письма.
> Content – Возвращает экземпляр класса `Illuminate\Mail\Mailables\Content`, который позволяет вам определить шаблон Blade, используемый для создания содержимого сообщения электронной почты.
> Attachments – Возвращает массив вложений, которые можно добавить в электронное письмо.

Как видно из приведенного выше кода, метод content() вернет представление. Поэтому вам нужно перейти в каталог resources/views, создать новую папку и внутри нее файл Blade.php. Когда файл создан, вы можете добавить в него контент.

```
// resources/views/mail/test-email.blade.php
Привет! Это сообщение с сайта!
```

В данном представлении можно использовать html-код, но использование CSS и JavaScript - запрещено!

Для передачи переменной в шаблон письма, можно воспользоваться методом `with`:

```
namespace App\Mail;

use Illuminate\Bus\Queueable;
```

```
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
use Illuminate\Mail\Mailables\Content;
use Illuminate\Mail\Mailables\Envelope;
use Illuminate\Queue\SerializesModels;

class MyTestEmail extends Mailable
{
    use Queueable, SerializesModels;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct(private $name)
    {
        //
    }

    /**
     * Get the message envelope.
     *
     * @return \Illuminate\Mail\Mailables\Envelope
     */
    public function envelope()
    {
        return new Envelope(
            subject: 'My Test Email',
        );
    }

    /**
     * Get the message content definition.
     *
     * @return \Illuminate\Mail\Mailables\Content
     */
    public function content()
    {
        return new Content(
            view: 'mail.test-email',
            with: ['name' => $this->name],
        );
    }
}
```

Таким образом в шаблоне `test-email.blade.php` будет доступна переменная `$name`

```
// resources/views/mail/test-email.blade.php
Привет {{$name}}! Это сообщение с сайта!
```

Далее добавляем маршрут в файл `routers/web.php`, используя следующий код:

```
use Illuminate\Support\Facades\Route;
use App\Mail\MyTestEmail;
use Illuminate\Support\Facades\Mail;

Route::get('/testroute', function() {
    $name = "Funny Coder";

    // The email sending is done using the to method on the Mail facade
    Mail::to('testreceiver@gmail.com')->send(new MyTestEmail($name));
});
```

Примечание. Обычно, в реальных приложениях вызов класса Mail осуществляется в контроллере или прослушивателях.
Чтобы проверить функциональность, вы можете запустить команду:

```
php artisan serve
```

а затем в браузере вставьте URL-адрес созданного вами маршрута.

Как только вы это сделаете, если все работает правильно, электронное письмо должно быть отправлено на указанный вами адрес «кому».
Чтобы узнать, как отправить электронное письмо в формате HTML, электронное письмо с вложением или электронное письмо с несколькими получателями, прочтите документацию Laravel Laravel Mail.

2 Ограничения и возможные проблемы при использовании Gmail SMTP

Gmail, хотя и широко используется, имеет ограничения и проблемы, о которых не так часто говорят. Итак, чтобы помочь вам поддерживать проект Laravel на правильном пути и избежать любых потенциальных ловушек, мы сейчас рассмотрим эти ограничения и проблемы одно за другим.

Ограничение отправки. При использовании Gmail SMTP пользователи имеют дневной лимит отправки электронной почты, который зависит от используемой ими учетной записи Gmail. Превышение этого предела приведет к ошибкам или временной приостановке.

Ограничение скорости. Еще одно ограничение, с которым сталкиваются пользователи Gmail SMTP, — это ограничение на количество электронных писем, которые они могут отправлять в минуту или в час. Превышение этого лимита заблокирует или задержит отправку вашей электронной почты.

Безопасность. Для отправки электронных писем через Gmail SMTP потребуется безопасное соединение, например SSL или TLS, которое необходимо правильно настроить в настройках Gmail SMTP. В противном случае ваши электронные письма и их данные станут уязвимыми для перехвата и взлома.

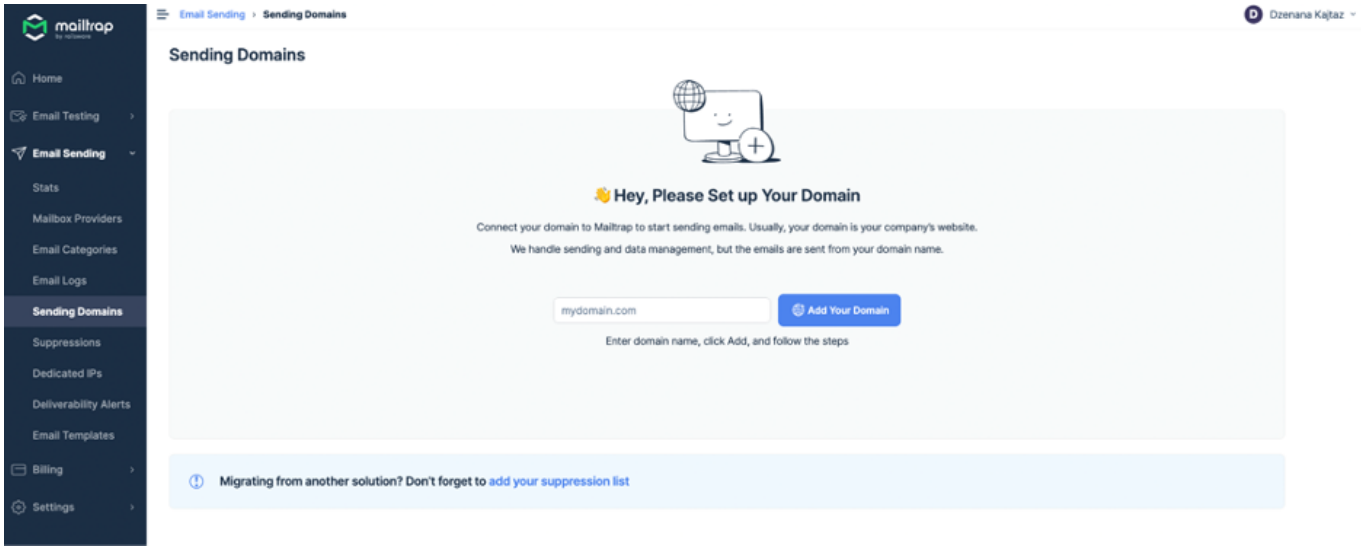
Настройки безопасности Google также требуют правильной настройки таких вещей, как двухфакторная аутентификация, чтобы избежать блокировки отправки электронной почты.

Персональный домен. Для отправки электронных писем из личного домена, например johndoe@yourcompany.com, с помощью Gmail SMTP, вам необходимо настроить правильные параметры DNS. В противном случае ваши электронные письма не пройдут проверку подлинности SPF, DKIM и DMARC и, таким образом, будут помечены как спам или просто отклонены.

3 Mailtrap

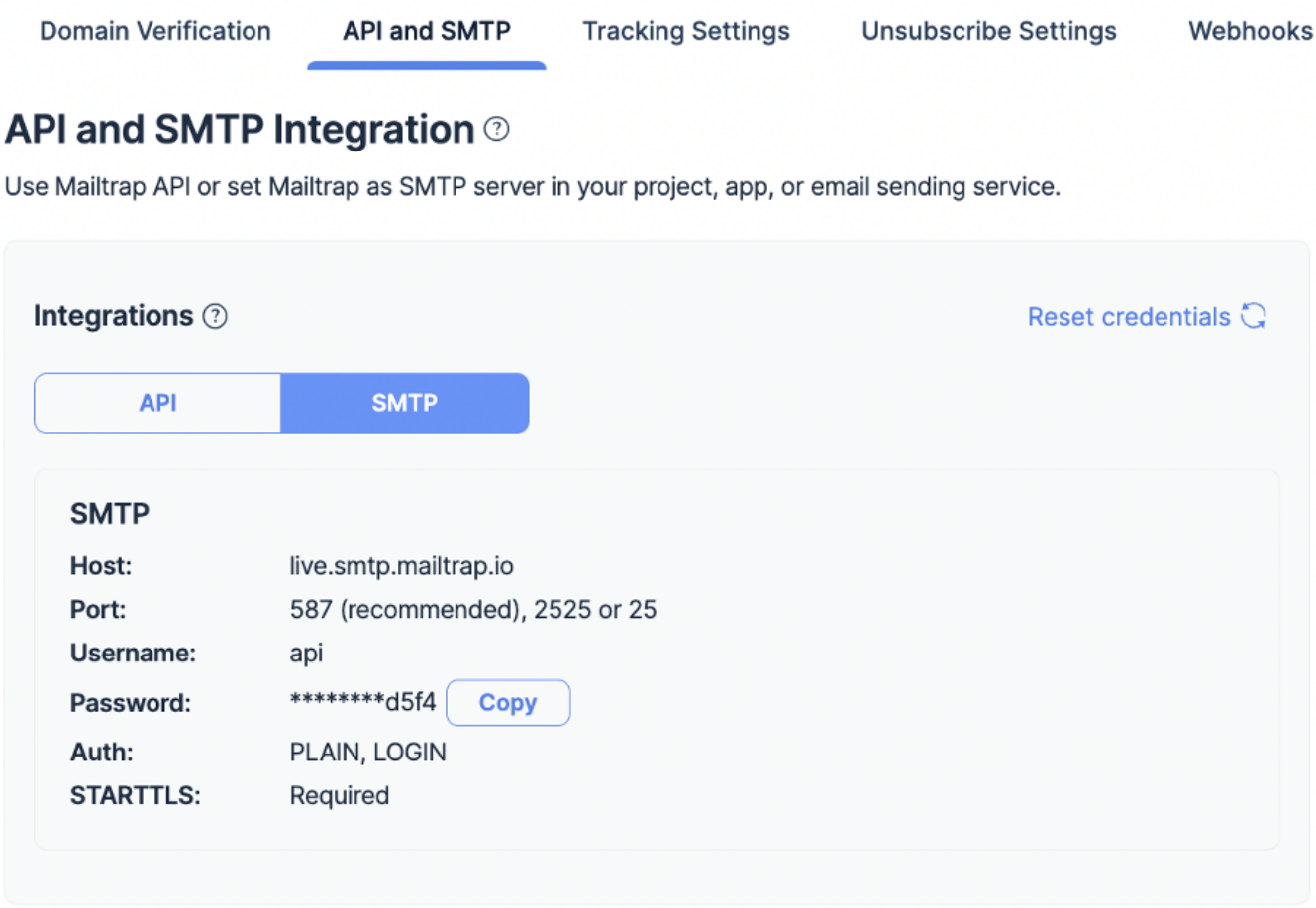
Наряду с распространенными, но в некоторой степени базовыми альтернативами, такими как Outlook SMTP, Yahoo Mail, Mail.RU, Yandex SMTP и т. д., в вашем распоряжении также есть SMTP-сервер отправки электронной почты **Mailtrap** — решение для отправки, которое предоставляет разработчикам инфраструктуру с высокой скоростью доставки.

Чтобы найти учетные данные SMTP для отправки электронной почты Mailtrap, сначала необходимо создать учетную запись Mailtrap и войти в нее. Затем в своей учетной записи в разделе «Отправка электронной почты» перейдите к разделу «Домены отправки», где вам нужно добавить и подтвердить свой домен.



Подробно процесс описан в видео ниже:

После проверки домена вы попадете на страницу интеграции API и SMTP, где сможете скопировать учетные данные SMTP, которые необходимо вставить в свое приложение/проект Laravel, чтобы начать использовать SMTP-сервер Mailtrap.



4 API электронной почты

В рамках отправки электронной почты Mailtrap также доступен API электронной почты.

Чтобы интегрировать этот API электронной почты в ваше приложение Laravel, вы можете использовать [Mailtrap PHP SDK](), который делает интеграцию проще и эффективнее, чем написание кода интеграции для вашего проекта вручную.

Вот шаги для использования SDK:

Установите PHP-клиент Mailtrap и зависимости с помощью Composer:

```
composer require railsware/mailtrap-php symfony/http-client nyholm/psr7
```

Добавьте транспорт Mailtrap в файл `config/mail.php`:

```
return [
    /*
    |--------------------------------------------------------------------------
    | Mailer Configurations
    |--------------------------------------------------------------------------
    */
    'mailers' => [
        // start mailtrap transport
```

```
        'mailtrap' => [
            'transport' => 'mailtrap'
        ],
        // end mailtrap transport
    ]
];
```

Добавьте настройки Mailtrap в Laravel в `.env` файл:

```
MAIL_MAILER="mailtrap"
MAILTRAP_HOST="send.api.mailtrap.io"
MAILTRAP_API_KEY="YOUR_API_KEY_HERE"
MAILTRAP_INBOX_ID=1000001
```

Создайте Mail класс для отправки электронной почты:

```
php artisan make:mail WelcomeMail
```

Появится файл `app/Mail/WelcomeMail.php`.

Вот примерное содержимое класса:

```
namespace App\Mail;

use Illuminate\Bus\Queueable;
use Illuminate\Mail\Attachment;
use Illuminate\Mail\Mailable;
use Illuminate\Mail\Mailables\Address;
use Illuminate\Mail\Mailables\Content;
use Illuminate\Mail\Mailables\Envelope;
use Illuminate\Mail\Mailables\Headers;
use Illuminate\Queue\SerializesModels;
use Mailtrap\EmailHeader\CategoryHeader;
use Mailtrap\EmailHeader\CustomVariableHeader;
use Symfony\Component\Mime\Email;
use Symfony\Component\Mime\Header\UnstructuredHeader;

class WelcomeMail extends Mailable
{
    use Queueable, SerializesModels;

    private string $name;

    /**
     * Create a new message instance.
     */
    public function __construct(string $name)
    {
        $this->name = $name;
    }

    /**
     * Get the message envelope.
```

```
     */
    public function envelope(): Envelope
    {
        return new Envelope(
            from: new Address('jeffrey@example.com', 'Jeffrey Way'),
            replyTo: [
                new Address('taylor@example.com', 'Taylor Otwell'),
            ],
            subject: 'Welcome Mail',
            using: [
                function (Email $email) {
                    // Headers
                    $email->getHeaders()
                        ->addTextHeader('X-Message-Source', 'example.com')
                        ->add(new UnstructuredHeader('X-Mailer', 'Mailtrap
PHP Client'));

                    // Custom Variables
                    $email->getHeaders()
                        ->add(new CustomVariableHeader('user_id', '45982'))
                        ->add(new CustomVariableHeader('batch_id',
'PSJ-12'));

                    // Category (should be only one)
                    $email->getHeaders()
                        ->add(new CategoryHeader('Integration Test'));
                },
            ]
        );
    }

    /**
     * Get the message content definition.
     */
    public function content(): Content
    {
        return new Content(
            view: 'mail.welcome-email',
            with: ['name' => $this->name],
        );
    }

    /**
     * Get the attachments for the message.
     *
     * @return array
     */
    public function attachments(): array
    {
        return [
Attachment::fromPath('https://mailtrap.io/wp-content/uploads/2021/04/mailtrap
```

```
-new-logo.svg')
                ->as('logo.svg')
                ->withMime('image/svg+xml'),
        ];
    }

    /**
     * Get the message headers.
     */
    public function headers(): Headers
    {
        return new Headers(
            'custom-message-id@example.com',
            ['previous-message@example.com'],
            [
                'X-Custom-Header' => 'Custom Value',
            ],
        );
    }
}
```

Далее создадим шаблон, файл `resources/views/mail/welcome-email.blade.php`

```
Привет, {{$name}} и добро пожаловать 

С уважением, администрация сайта
```

Теперь можем отправить письмо. Для разнообразия, сделаем это из консоли. Для этого в файле `app/routes/console.php` напишем следующий код:

```
use App\Mail\WelcomeMail;
use Illuminate\Support\Facades\Artisan;
use Illuminate\Support\Facades\Mail;

/*
|--------------------------------------------------------------------------
| Console Routes
|--------------------------------------------------------------------------
|
*/

Artisan::command('send-welcome-mail', function () {
    Mail::to('testreceiver@gmail.com')->send(new WelcomeMail("Jon"));
    // Also, you can use a specific mailer if your default mailer is not
"mailtrap", but you want to use it for welcome emails
    // Mail::mailer('mailtrap')->to('testreceiver@gmail.com')->send(new
WelcomeMail("Jon"));
})->purpose('Send welcome mail');
```

Теперь можете запустите команду CLI, которая отправит ваше электронное письмо:

```
php artisan send-welcome-mail
```

Вот и все! API отправки электронной почты Mailtrap теперь интегрирован с вашим приложением Laravel.

Start Sending With Mailtrap

5 Тестирование писем перед отправкой: зачем и как?

Для некоторых процесс настройки функции отправки электронной почты в приложении Laravel останавливается после написания кода отправки электронной почты. Для других, которые выбирают более осторожный путь, этот процесс также включает в себя добавление тестирования электронной почты в качестве важного шага перед отправкой.

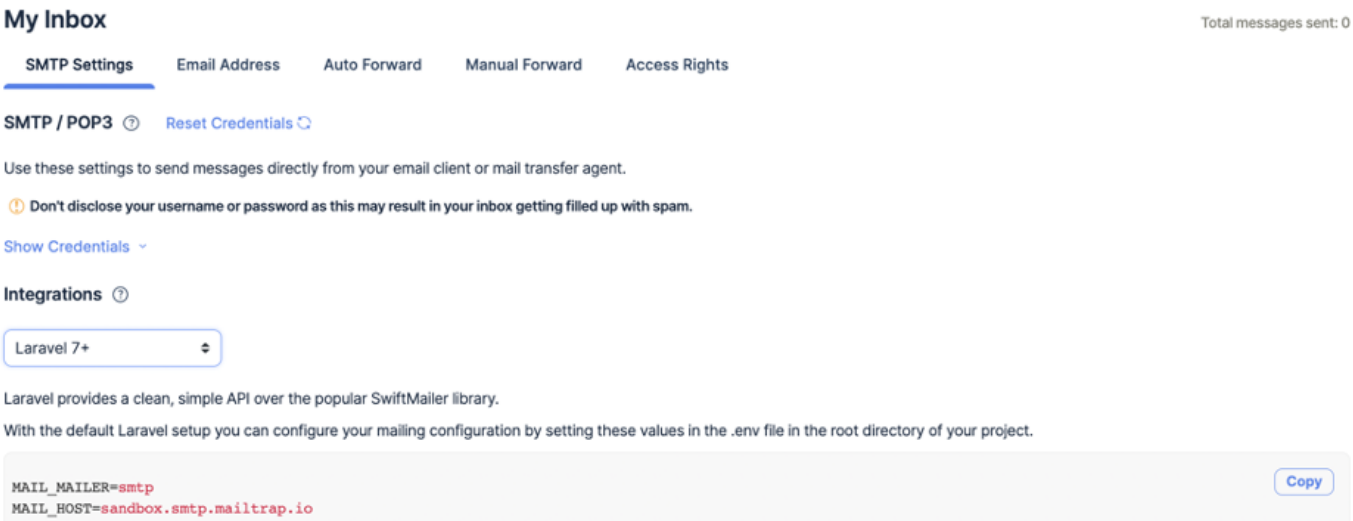
Посредством тестирования электронной почты вы можете проверить, как ваши электронные письма отображаются в веб-браузерах, а также насколько они отзывчивы. Вы также можете проверить количество спама, найти свой домен/IP-адрес в черных списках и сделать даже больше, если у вас есть подходящий инструмент для тестирования электронной почты.

Инструмент тестирования электронной почты, который включает в себя все упомянутые функции, а также другие, такие как создание виртуальных почтовых ящиков для различных проектов и этапов проекта, анализ подробной технической информации (заголовки электронных писем и данные транзакций SMTP) и т. д. — это Mailtrap. Тестирование электронной почты.

Mailtrap Email Testing — это решение, которое позволяет проверять и отлаживать электронные письма в промежуточных средах, средах разработки и контроля качества перед отправкой их получателям. Таким образом, создается безопасная среда для тестирования электронной почты, в которой отсутствует риск рассылки спама получателям.

Тестирование электронной в три этапа:

Создать аккаунт Mailtrap account .
В аккаунте переходим в настройки Testing – > Inboxes – > SMTP Settings.
Выбираем Laravel из списка интеграций.



Скопируйте сгенерированный код интеграции и вставьте его в скрипт отправки электронной почты.

Запустите сценарий и вскоре получите первое тестовое электронное письмо на свой виртуальный почтовый ящик, где вы сможете его протестировать и отладить.

Примечание. При тестировании вам не обязательно использовать реальные адреса электронной почты отправителя или получателя, поскольку тестовые электронные письма отправляются на виртуальные почтовые ящики, а проверка электронной почты не выполняется.

Mailtrap Email Testing также предлагает учетные данные SMTP для каждого из ваших виртуальных почтовых ящиков, доступ к которым можно получить, нажав «Показать учетные данные» на странице настроек SMTP.

Итак, если вы предпочитаете использовать учетные данные вместо фрагментов кода для интеграции этого решения для тестирования, просто скопируйте и вставьте их в свой сценарий отправки электронной почты, настройки MTA, настройки почтового клиента или любую другую систему, которая их поддерживает.

6 Отправка email используя Gmail SMTP и Mailtrap - видео

Заключение

Список использованных источников

Приложения