# Sistema de Autenticación API Rest con Laravel 5.6 — Parte 5

Claudio Vallejo
Feb 8, 2019 · 4 min read


## * Enviar notificaciones con espera en Redis *

Seguimos la parte 5 del Sistema de Autenticación API Rest con Laravel 5.6 con el sistema para **enviar notificaciones con queues en Redis.** Esta es una traducción al español, con algunas adiciones y complementaciones, más algunos pequeños ajustes del artículo escrito por Alfredo Barron (a quién recomiendo seguir) y no corresponde a un tutorial de mi autoría.

. . .

## Paso 1. Instalar Redis

Lo fundamental es entender que Redis es un estructura para almacenamiento de datos en memoria de código abierto (con licencia BSD), que se utiliza como base de datos,

caché y agente de mensajes.

> *“Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker.”*

Para su instalación o descarga recomiendo los siguientes tutoriales:

Ubuntu 16.04:

**How To Install and Configure Redis on Ubuntu 16.04 | DigitalOcean**

Redis is an in-memory, key-value store known for its flexibility, performance, and wide language support. In this...

www.digitalocean.com

Mac OSX:

**Install and config Redis on Mac OS X via Homebrew**

by Pete Houston

medium.com

Directamente desde el sitio web Redis.io:

**Redis**

Redis uses a standard practice for its versioning: major.minor.patchlevel. An even minor marks a stable release, like...

redis.io

# Paso 2. Configurar Ambiente en Laravel

En este paso debemos configurar nuestro archivo `.env` agregando `redis` dentro de la variable `QUEUE_DRIVER` . Finalmente es necesario utilizar un servicio de email para pruebas. Podemos utilizar mailtrap.io o mailgun, como lo requieran.

```
QUEUE_DRIVER=redis
```

```
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=user
MAIL_PASSWORD=password
MAIL_ENCRYPTION=null
```

## Paso 3. Instalar el controlador de Redis

Necesitamos instalar el controlador Redis en el proyecto para lo cual escribe en tu terminal siguiente comando:

```
composer require predis/predis
```

terminal

## Paso 4. Configurar el uso de Queue en las Notificaciones

Como ya habíamos trabajado anteriormente, existen algunas notificaciones que debemos ahora configurar para que puedan ser ejecutadas en cola de espera “queue”. Para ello abramos nuestro archivo `app/Notifications/SignupActivate.php` y agreguemos lo siguiente:

```
<?php

namespace App\Notifications;

use Illuminate\Bus\Queueable;
use Illuminate\Notifications\Messages\MailMessage;
use Illuminate\Notifications\Notification;
use Illuminate\Contracts\Queue\ShouldQueue;
```

```
class SignupActivate extends Notification implements ShouldQueue
{
...
```

## Paso 5. Ejecutar el trabajador de colas de espera "Queue worker"

Para poder probar las colas de espera "queue" en tu máquina local, hay que ejecutar el trabajador de dichas esperas "queue worker". Para ello hay que ejecutar el siguiente comando en la terminal:

```
php artisan queue:work
```

Ahora, cuando se cree un nuevo usuario (registro), podrás ver en consola lo siguiente:

Esto significa que el trabajador de la cola de espera está funcionando correctamente.

## Paso 6. (opcional para producción) Instalación y Configuración del Supervisor

> *Pasos utilizados en producción y servidores Linux*

Instalar el supervisor con el siguiente comando:

```
sudo apt-get install supervisor
```

Entrar a la carpeta `config.d` :

```
cd /etc/supervisor/conf.d
```

Crear el archivo de configuración

```
mkdir laravel-worker.conf
```

Editar el archivo de configuración (con su editor favorito. Puede ser vim, nano, etc…)

```
nano laravel-worker.conf
```

Dentro del archivo escribir las siguientes líneas de código:

```
[program:laravel-worker]
process_name=%(program_name)s_%(process_num)02d
command=php /var/www/project_name/artisan queue:work redis --sleep=3
--tries=3
autostart=true
autorestart=true
user=www-data
numprocs=8
redirect_stderr=true
stdout_logfile=/var/www/project_name/storage/logs/worker.log
```

Guardar el archivo.

## Inicializar el Supervisor

Una vez que el archivo de configuración ha sido creado, tu podrás editarlo e incializar los procesos con los siguientes comandos:

```
sudo supervisorctl reread

sudo supervisorctl update

sudo supervisorctl start laravel-worker:*
```

> *Desde este punto en adelante, tu aplicación está plenamente funcional y operativa. Solo necesitas correr tu sitio en el ambiente local que tengas (homestead, valet, single server o cualquier otro entorno). En el caso de un servidor simple o directo, basta con escribir el comando:*

```
php artisan serve
```

. . .

## Paso 7. Pruebas

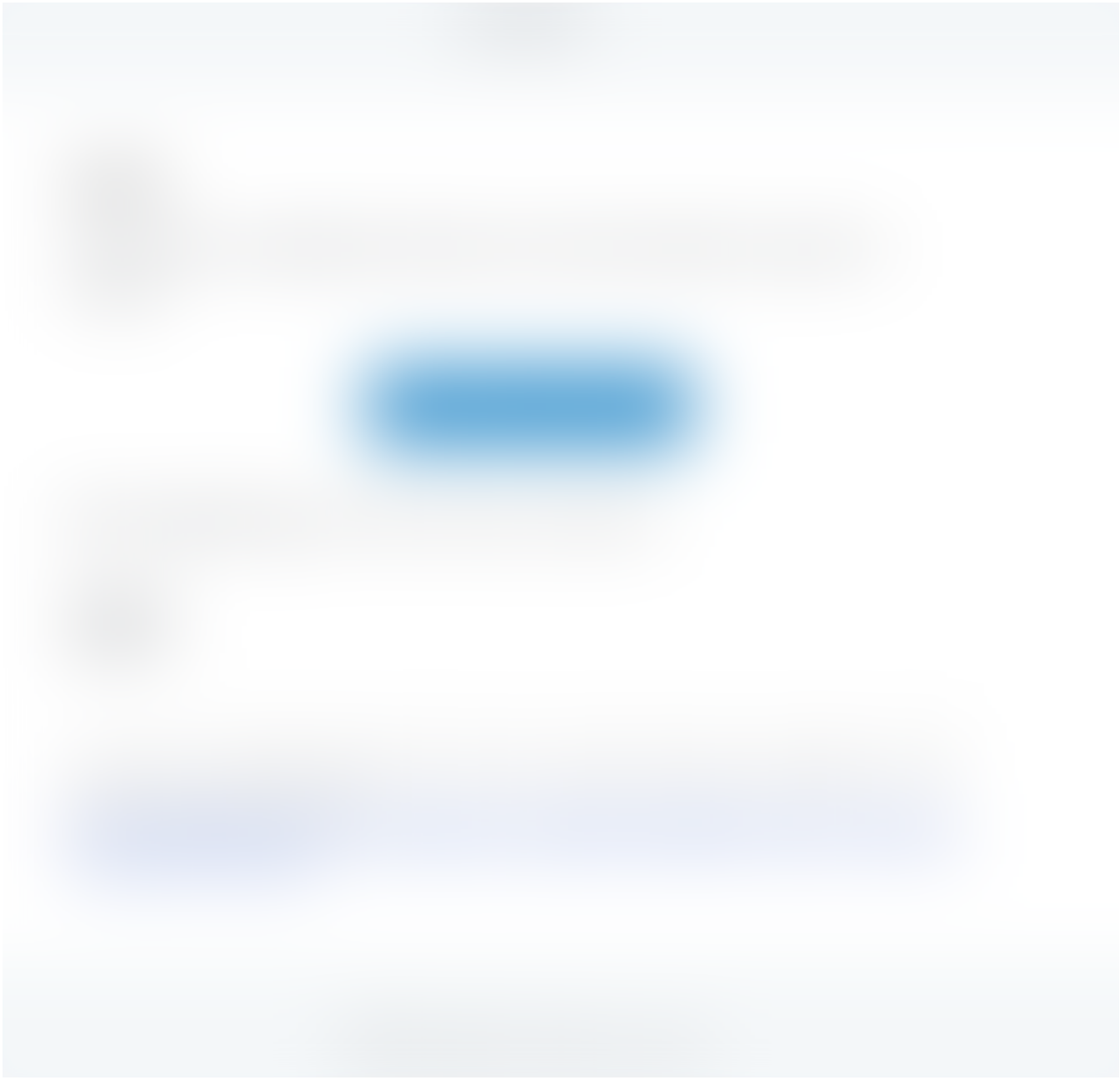
Para las pruebas, utilicé Postman (Postman)

Para la correcta utilización, hay que configurar las siguientes dos cabeceras:

```
Content-Type: application/json
X-Requested-With: XMLHttpRequest
```

### Registro de nuevo usuario

### Email de confirmación



> Con estas pruebas funcionales hemos terminado la quinta parte correspondiente a la utilización de Redis para enviar notificaciones en cola

. . .

## Serie: Sistema de Autenticación API Rest con Laravel 5.6

- Parte 1 : Instalación y configuración
- Parte 2 : Confirmación de cuenta y notificaciones
- Parte 3 : Generar un avatar
- Parte 4 : Restablecer contraseña

- Parte 5 : Enviar notificaciones con espera en Redis

. . .

Gracias por tu lectura.

Si te ha gustado, podrías darme un aplauso y seguirme :)



. . .

Puedes compartir esta publicación en tus redes sociales

**Claudio Vallejo (@cvallejo) | Twitter**

The latest Tweets from Claudio Vallejo (@cvallejo). Wine, programmer, diver & photographer lover. Agronomist Engineer...

twitter.com

Laravel API Español Redis Developer


Download on the
App Store


GET IT ON
Google Play