Sistema de Autenticación API Rest con Laravel 5.6 — Parte 2

Claudio Vallejo
Aug 5, 2018 · 5 min read

* Confirmación de cuenta y notificaciones *

Continuamos esta parte 2 del Sistema de Autenticación API Rest con Laravel 5.6 con la **generación de notificaciones y envío de confirmación de activación por mail.** Esta es una traducción al español, con algunos pequeños ajustes, del artículo escrito por Alfredo Barron (a quién recomiendo seguir) y no corresponde a un tutorial de mi autoría.

. . .

1. Agregar columnas en tabla users

Se deben agregar tres columnas, `active`, `activation_token` y el trait `softDeletes` en el archivo de migración `database/migrations/xxxx_create_users_table.php`.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->boolean('active')->default(false);
            $table->string('activation_token');
            $table->rememberToken();
            $table->timestamps();
            $table->softDeletes();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

Luego es necesario agregar el trait `SoftDeletes` y los atributos `fillable` y `hidden` en el modelo `App\User`.

```
<?php

namespace App;
```

```
use Illuminate\Database\Eloquent\SoftDeletes;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Passport\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, Notifiable, SoftDeletes;

    protected $dates = ['deleted_at'];

    protected $fillable = [
        'name', 'email', 'password', 'active', 'activation_token',
    ];

    protected $hidden = [
        'password', 'remember_token', 'activation_token',
    ];
}
```

Una vez realizado, hay que efectuar la migración desde 0 (comando refresh). Para ello, escribir en la terminal y escribir el siguiente comando:

```
php artisan migrate:refresh
```

. . .

2. Creación de la notificación de la creación de la cuenta

En la terminar hay que escribir el siguiente comando:

```
php artisan make:notification SignupActivate
```

Esto creará el archivo `app/Notifications/SignupActivate.php` , en éste se determinarán los canales de las notificaciones que serán utilizadas (en este caso usaremos `mail`)

```
public function via($notifiable)
{
    return ['mail'];
}
```

Luego crearemos las notificaciones por email.

```
public function toMail($notifiable)
{
    $url = url('/api/auth/signup/activate/'.$notifiable-
>activation_token);

    return (new MailMessage)
        ->subject('Confirma tu cuenta')
        ->line('Gracias por suscribirte! Antes de continuar, debes
configurar tu cuenta.')
        ->action('Confirmar tu cuenta', url($url))
        ->line('Muchas gracias por utilizar nuestra aplicación!');
}
```

2.1 Personalización del correo

Para poder personalizar el correo hay que exponer la configuración y exportar sus componentes. Para ello hay que ejecutar el siguiente comando:

```
php artisan vendor:publish --tag=laravel-mail
```

más información: Laravel:Customizing The Components

. . .

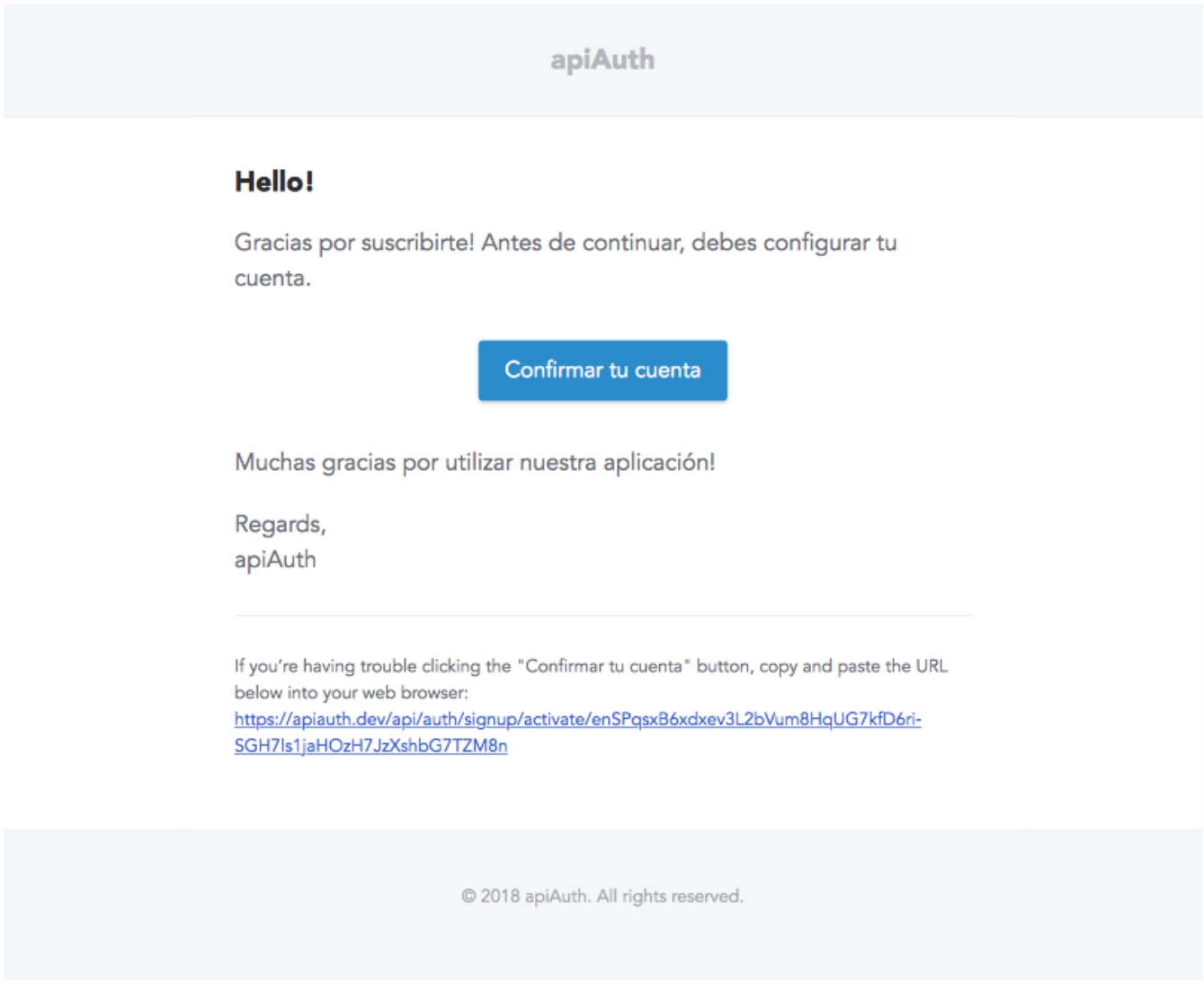
3. Crear y enviar token para confirmar la cuenta

Hay que actualizar el controlador `app/Http/Controllers/AuthController.php` y el método `signup` de la api. Para hacerlo hay que incluir el siguiente código:

```
<?php
...
use App\Notifications\SignupActivate;

class AuthController extends Controller
{
...
    public function signup(Request $request)
    {
        $request->validate([
            'name'      => 'required|string',
```

```
            'email'     => 'required|string|email|unique:users',
            'password'  => 'required|string|confirmed',
        ]);

        $user = new User([
            'name'              => $request->name,
            'email'             => $request->email,
            'password'          => bcrypt($request->password),
            'activation_token'  => str_random(60),
        ]);

        $user->save();

        $user->notify(new SignupActivate($user));

        return response()->json(['message' => 'Usuario creado
existosamente!'], 201);
    }
}
```

Cuando se cree una nueva cuenta ésta recibirá un email con el enlace para activar su cuenta. El próximo paso es crear la ruta y el método para activar la cuenta.



. . .

4. Agregar la ruta para la activación de la cuenta

Hay que agregar la nueva ruta `signup/activate/{token}` en el archivo `routes/api.php`.

```
<?php

Route::group(['prefix' => 'auth'], function () {
    Route::post('login', 'AuthController@login');
    Route::post('signup', 'AuthController@signup');
    Route::get('signup/activate/{token}',
'AuthController@signupActivate');

    Route::group(['middleware' => 'auth:api'], function () {
        Route::get('logout', 'AuthController@logout');
        Route::get('user', 'AuthController@user');
    });
});
```

. . .

5. Confirmar cuenta a usuarios activos

Hay que crear el método `signupActivate` en el controlador `app/Http/Controllers/AuthController.php` para activar la cuenta del usuario.

```
public function signupActivate($token)
{
    $user = User::where('activation_token', $token)->first();

    if (!$user) {
        return response()->json(['message' => 'El token de
activación es inválido'], 404);
    }

    $user->active = true;
    $user->activation_token = '';
    $user->save();

    return $user;
}
```

. . .

6. Validación de la cuenta

Para validar que la cuenta esté activa y no ha sido borrada, hay que actualizar el método `login` en el controlador `app/Http/Controllers/AuthController.php`.

```
public function login(Request $request)
{
    $request->validate([
        'email'       => 'required|string|email',
        'password'    => 'required|string',
        'remember_me' => 'boolean',
    ]);

    $credentials = request(['email', 'password']);
    $credentials['active'] = 1;
    $credentials['deleted_at'] = null;

    if (!Auth::attempt($credentials)) {
        return response()->json(['message' => 'No Autorizado'],
401);
    }

    $user = $request->user();
    $tokenResult = $user->createToken('Token Acceso Personal');
    $token = $tokenResult->token;

    if ($request->remember_me) {
        $token->expires_at = Carbon::now()->addWeeks(1);
    }

    $token->save();

    return response()->json([
        'access_token' => $tokenResult->accessToken,
        'token_type'   => 'Bearer',
        'expires_at'   => Carbon::parse($tokenResult->token-
>expires_at)->toDateTimeString(),
    ]);
}
```

> *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
```

. . .

7. Configuración archivo ".env"

Debes asegurarte de tener configurado las variables de entorno de tu proyecto para poder enviar los emails de validación.

Para ello puedes utilizar mailgun.com, mailtrap.io o el que más te sirva para este propósito.

. . .

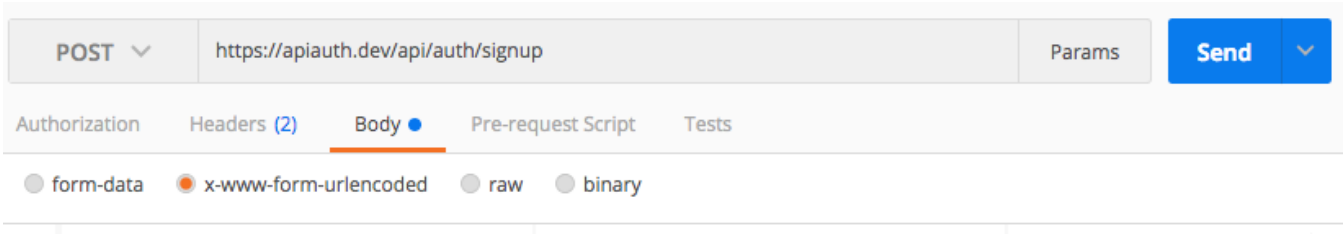
8. Pruebas

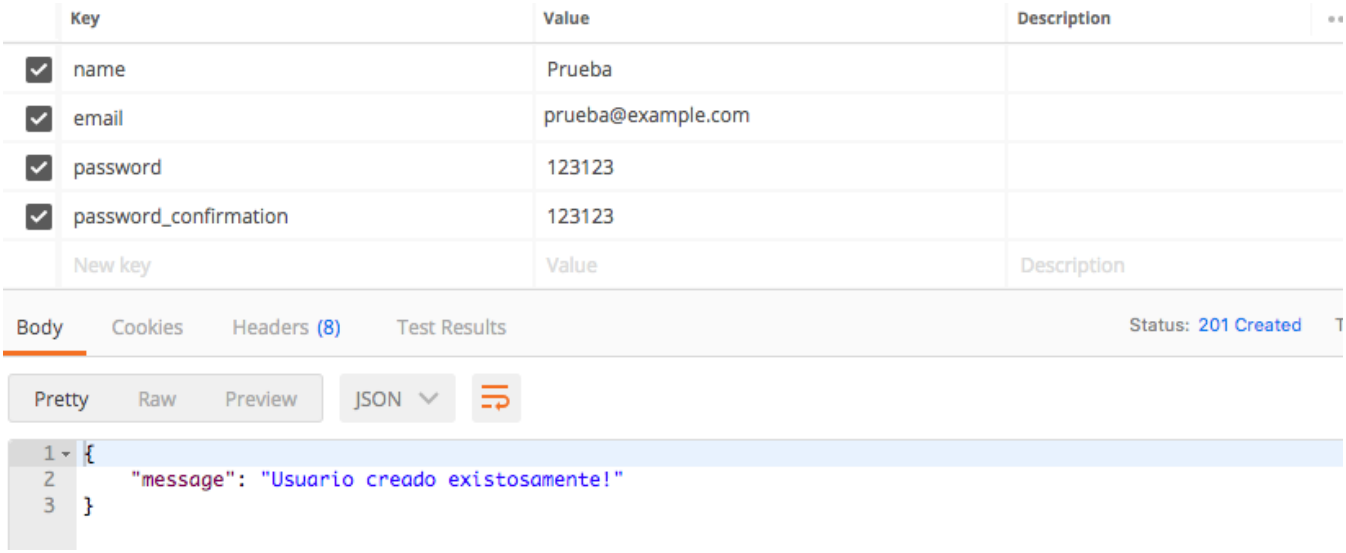
Para las pruebas, utilicé Postman (tiene opción para extensión en chrome o como app). Más info: Postman

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

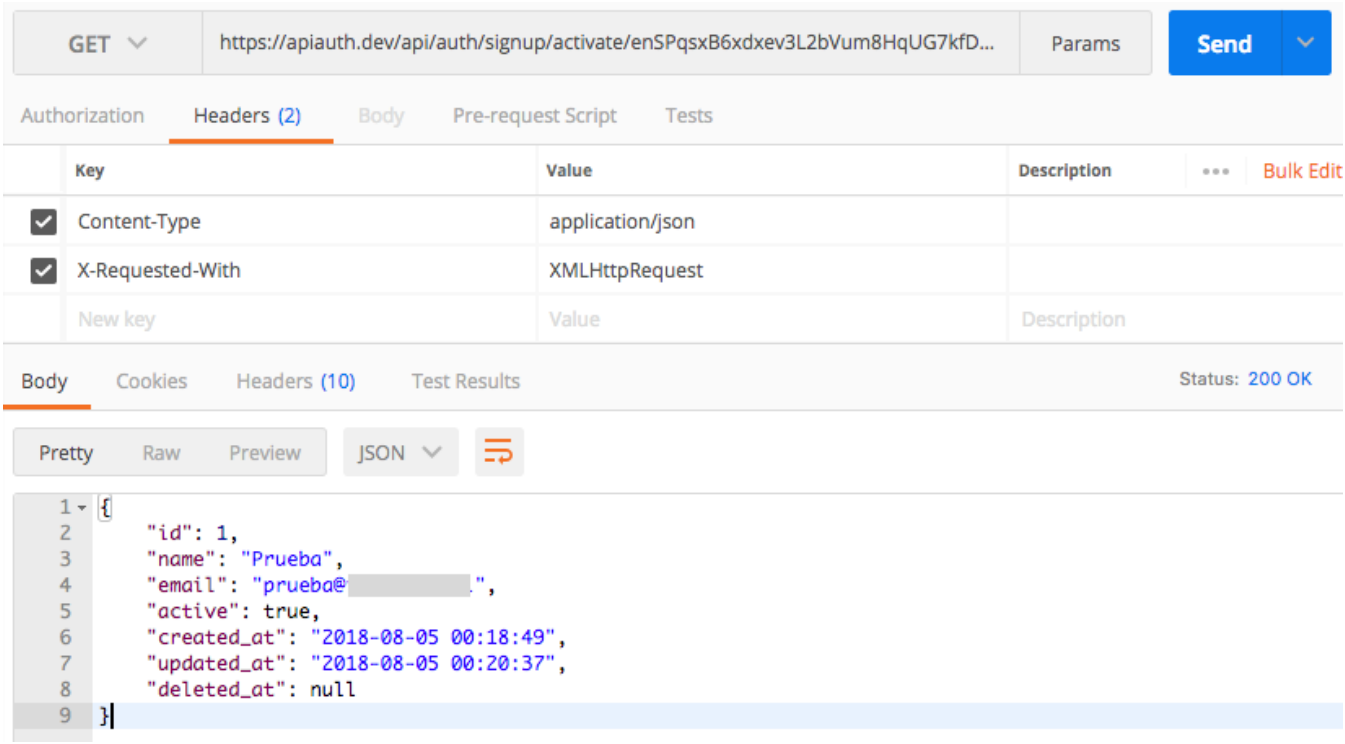
```
Content-Type: application/json
X-Requested-With: XMLHttpRequest
```

Creación de usuario / signup

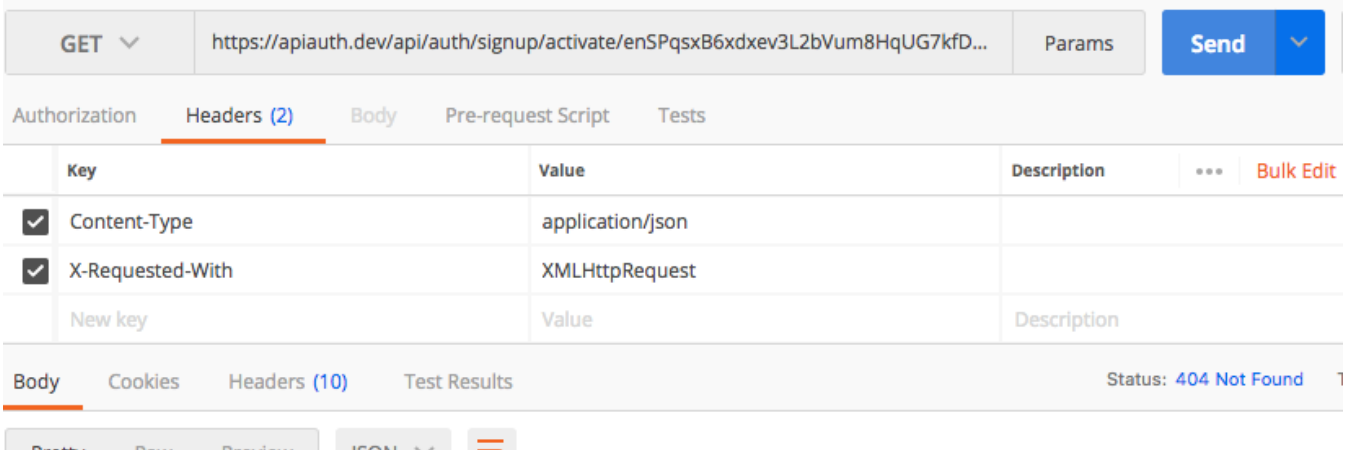




Activación de usuario por token



Error el re-enviar petición



Pretty Raw Preview JSON

```
{
    "message": "El token de activación es inválido"
}
```

> *Con estas pruebas funcionales hemos terminado la segunda parte correspondiente a la generación de notificaciones y autenticación de usuarios a través de token.*

. . .

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

Tweets with replies by Claudio Vallejo (@cvallejo) | Twitter

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

twitter.com

Laravel API Español Rest Api