Sistema de Autenticación API Rest con Laravel 5.6 — Parte 3

Claudio Vallejo
Sep 12, 2018 · 5 min read

* Generar un Avatar *

Continuamos esta parte 3 del Sistema de Autenticación API Rest con Laravel 5.6 con la **generación de avatares.** 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.

. . .

1. Instalación del paquete avatar

Lo primero que se necesita es la instalación del paquete de avatar de Laravot (https://github.com/laravolt/avatar). Realizar la instalación debes escribir el siguiente comando:

```
composer require laravolt/avatar
```

Opcionalmente puedes publicar el 'Service Provider' para verificar su configuración:

```
php artisan vendor:publish --
provider="Laravolt\Avatar\ServiceProvider"
```

```
$ php artisan vendor:publish --provider="Laravolt\Avatar\ServiceProvider"
Copied File [/vendor/laravolt/avatar/config/config.php] To [/config/laravolt/avatar.php]
Publishing complete.
```

Esto generará el archivo de configuración en la ruta `config/laravolt/avatar.php`

Adicionalmente una opción es registrar el alias en la carpeta `config\app.php`

```
'Auth'          => Illuminate\Support\Facades\Auth::class,
```

2. Agregar columna a la tabla usuarios

Deberemos agregar una columna 'avatar' a la tabla usuarios. Para aquello debemos modificar el archivo de la migración de dicha tabla

`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');
```

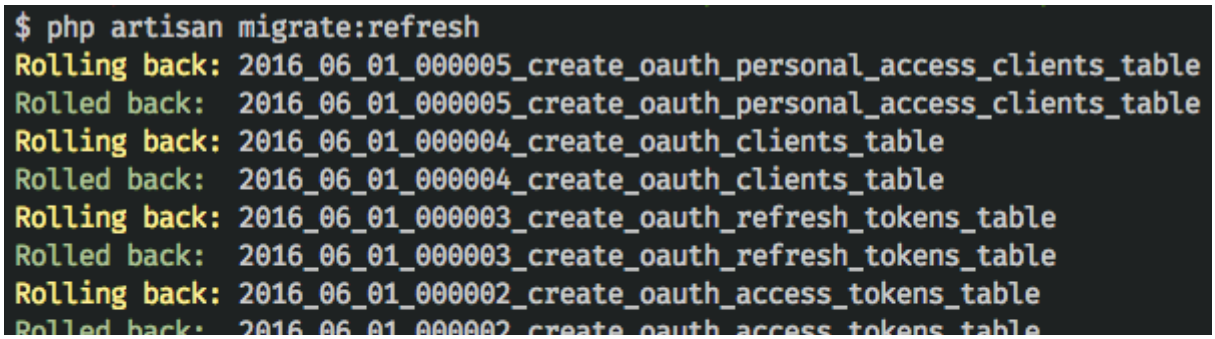
```
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->string('avatar')->default('avatar.png');
            $table->boolean('active')->default(false);
            $table->string('activation_token');
            $table->rememberToken();
            $table->timestamps();
            $table->softDeletes();
        });
    }
...
```

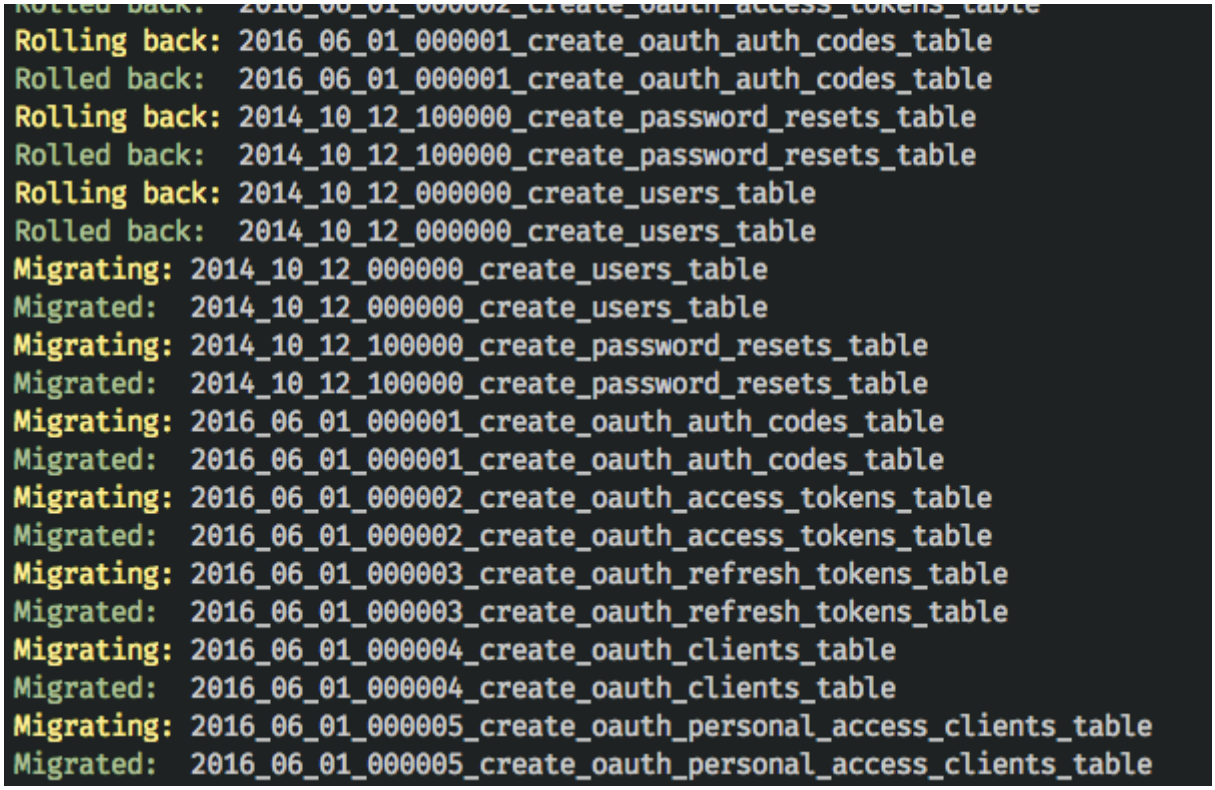
Luego deberemos agregar 'avatar' como atributo `$fillable` en el modelo `App\User` .

```
...

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

    protected $dates = ['deleted_at'];

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

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

Por último, deberemos correr el siguiente comando para generar una migración fresca:

```
php artisan migrate:refresh
```





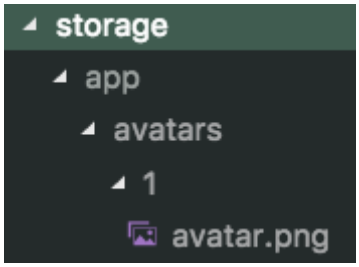
3. Crear avatar para cada cuenta de usuario

Deberemos actualizar el controlador `app/Http/Controllers/AuthController.php` en el método api `signup` con la inclusión del siguiente código:

```
<?php

namespace App\Http\Controllers;

use Avatar;
use Storage;
use App\User;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
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();

        $avatar = Avatar::create($user->name)->getImageObject()->encode('png');
        Storage::put('avatars/'.$user->id.'/avatar.png', (string) $avatar);

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

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

El avatar será creado en la carpeta `storage/avatars` .



4. Obtener el avatar de un usuario autenticado

Para finalizar es necesario anexar el atributo en el modelo `App\User` para obtener la url del avatar.

```
<?php

namespace App;

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

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

```
    protected $appends = ['avatar_url'];
    protected $dates = ['deleted_at'];

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

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

    public function getAvatarUrlAttribute()
    {
        return Storage::url('avatars/'.$this->id.'/'.$this->avatar);
    }
}
```

> *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
```

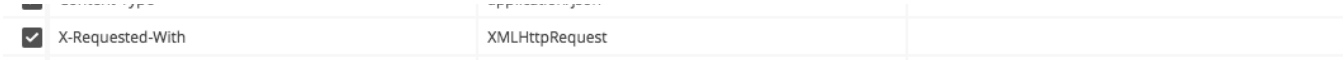
. . .

5. Pruebas

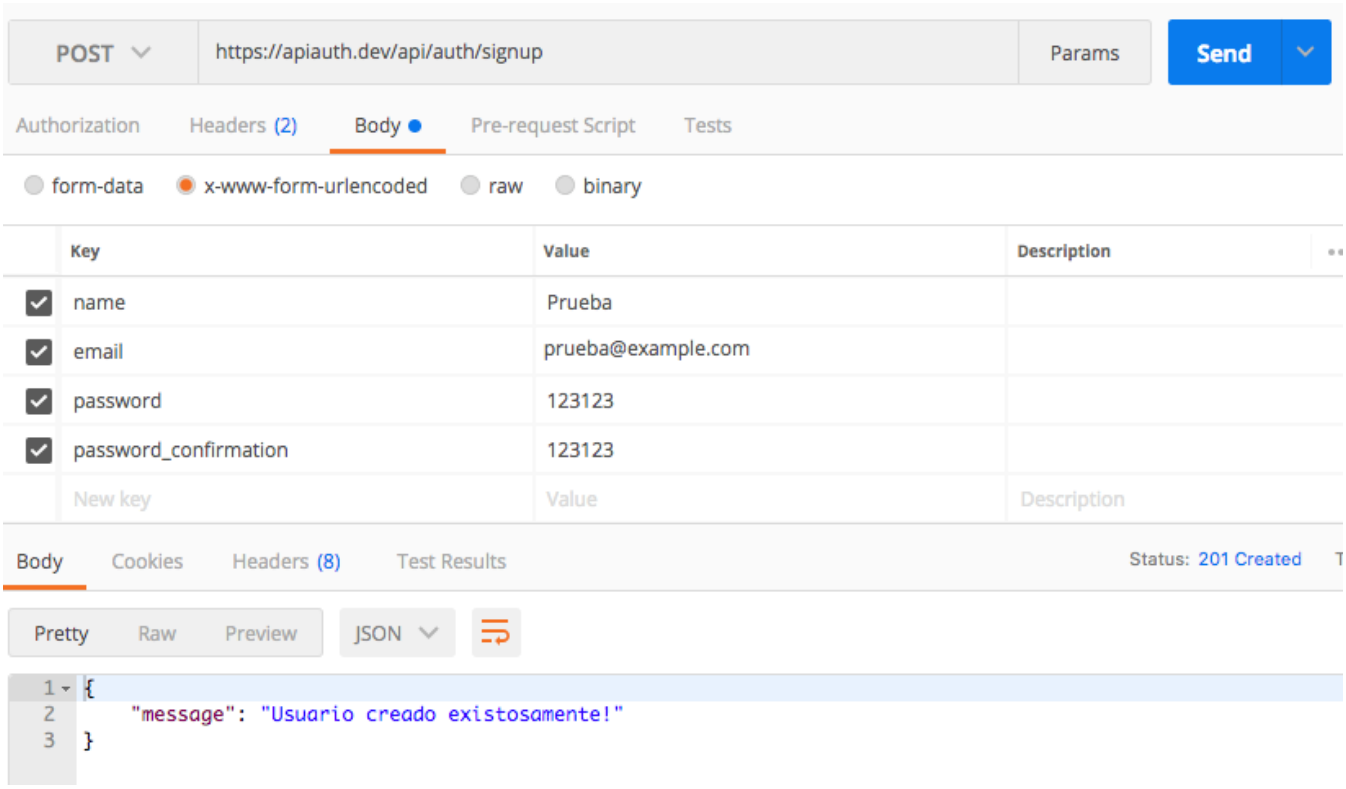
Para las pruebas, utilicé Postman (tiene opción para extensión en chrome o como app). Más info: Postman

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

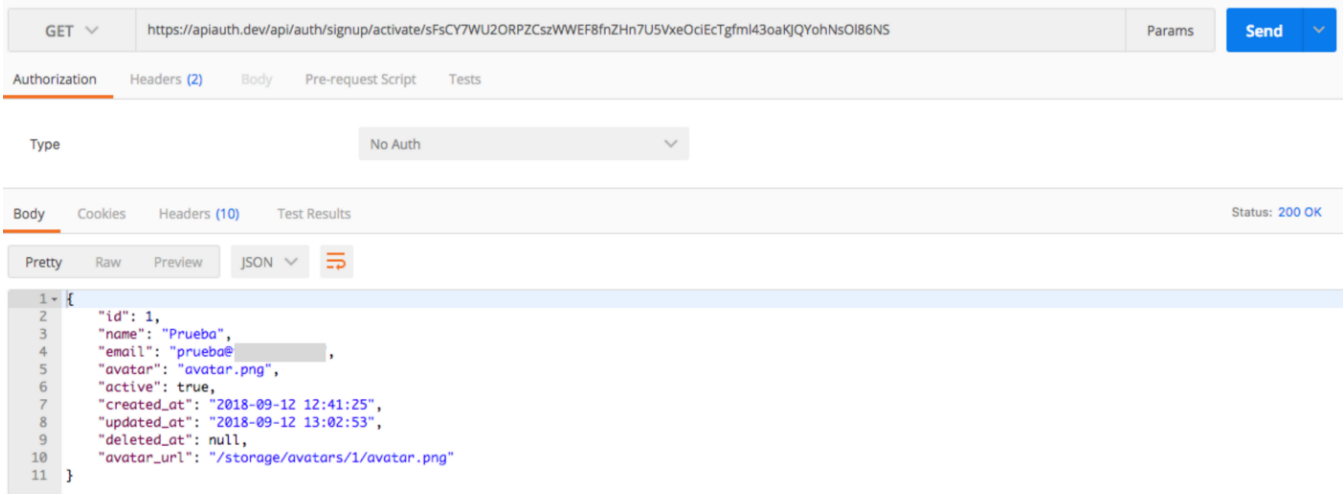
```
Content-Type: application/json
X-Requested-With: XMLHttpRequest
```



Creación de usuario / signup



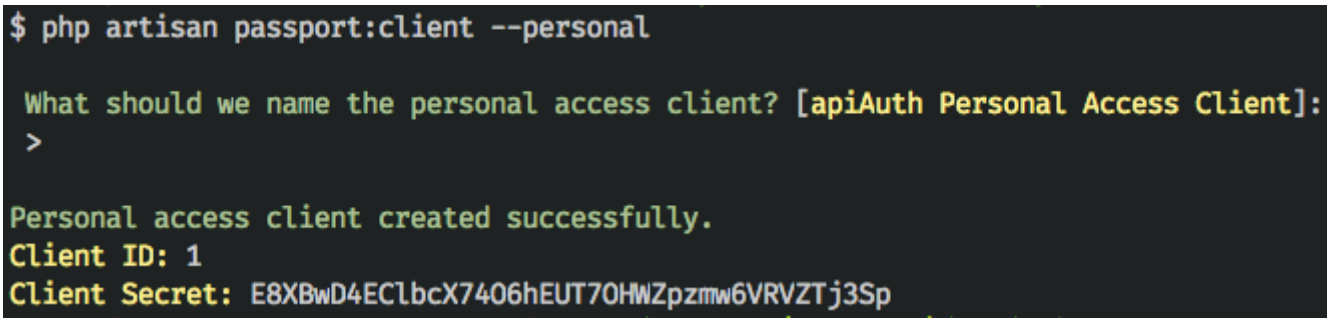
Activación de usuario por token



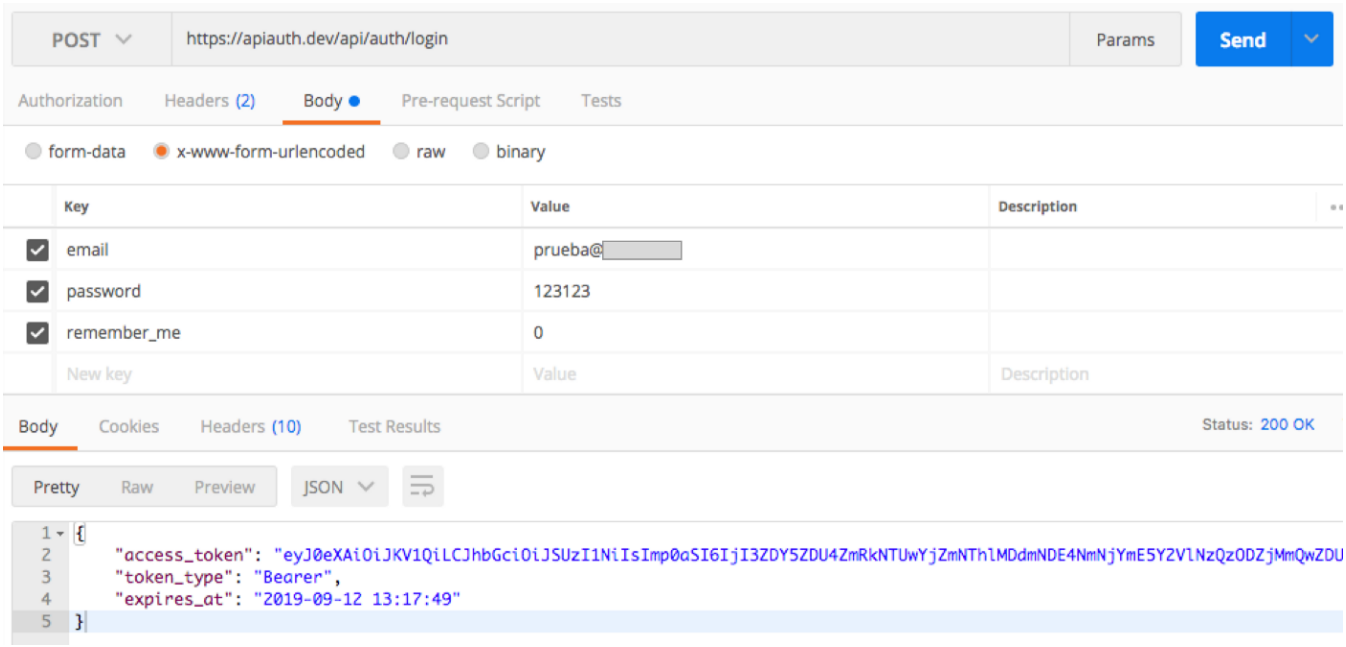
Login

Es importante que verifiques la creación del acceso personal del cliente (más info: https://laravel.com/docs/5.6/passport#client-credentials-grant-tokens) a través del comando :

```
php artisan passport:client --personal
```

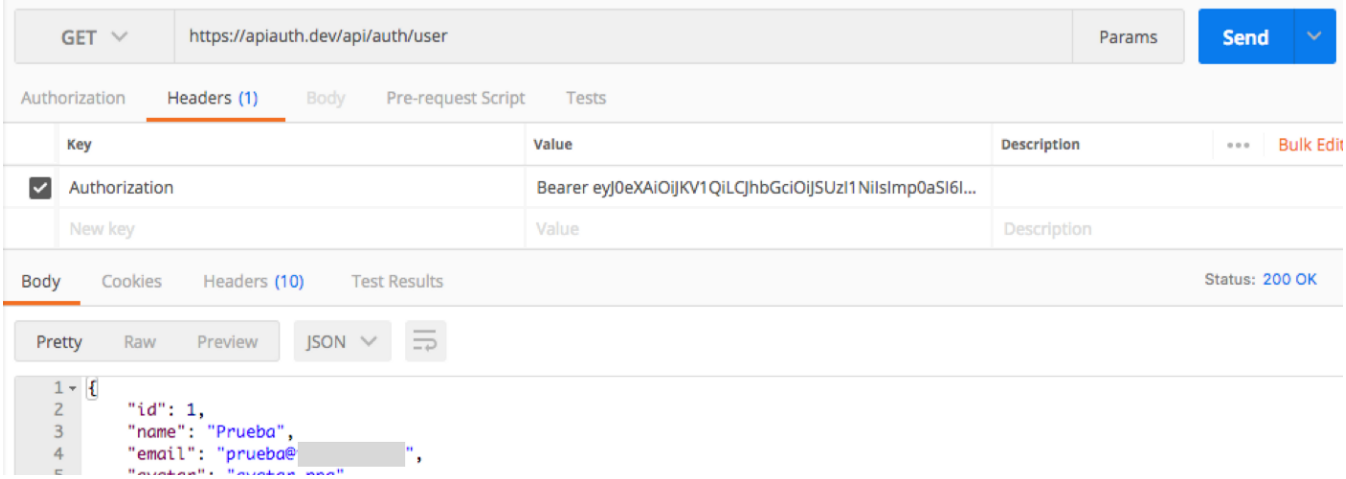


Posteriormente, puedes efectuar la prueba de login a través de Postman:



Usuario

Una vez que logueado, puedes ingresar a la ruta de 'user' y podrás ver la información de tu usuario.



```
    "avatar": "avatar.png",
    "active": 1,
    "created_at": "2018-09-12 12:41:25",
    "updated_at": "2018-09-12 13:02:53",
    "deleted_at": null,
    "avatar_url": "/storage/avatars/1/avatar.png"
}
```

> Con estas pruebas funcionales hemos terminado la tercera parte correspondiente a la generación de avatar para los usuarios.

. . .

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 Español API Avatar Rest Api