JWT token autenticación Lumen Laravel 8

Índice

1. Creando el Proyecto....2
2. Creando la base de Datos....2
3. Configurando la Conexión a la Base de Datos....2
4. Instalando Lumen Generator....2
5. Creando Archivo de Migración para tabla Products....3
6. Creando Modelo Product....3
7. Creando PostController....4
8. Instalando JWT....6
9. Pruebas de autenticación....12

1. Creando el Proyecto

composer create-project --prefer-dist laravel/lumen **apiJWT**

cd apiJWT

mysql -u root -p
clave de root mysql

2. Creando la base de Datos

create database testjwt;

grant all privileges on **testjwt**.* to '**pepe**'@'localhost' identified by '**12345**';

flush privileges;

quit

3. Configurando la Conexión a la Base de Datos

Editar la Conexion a la Base de Datos en el archivo **.env**

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=**testjwt**
DB_USERNAME=**pepe**
DB_PASSWORD=**12345**

4. Instalando Lumen Generator

Instalaremos **Lumen Generator** para generar desde la linea de comandos controladores, modelos etc.

composer require flipbox/lumen-generator

ve y edita el archivo **bootstrap/app.php**, busca la sección **Register Service Providers** y agrega las siguientes lineas, asegurando que solo se ejecutara en modo desarrollo y evitando que se utilice en modo producción:

```
// Flipbox Lumen Generator
if ($app->environment() !== 'production') {
    $app->register(Flipbox\LumenGenerator\LumenGeneratorServiceProvider::class);
}
```

5. Creando Archivo de Migración para tabla Products

Crearemos un archivo de migración para una tabla llamada products así:

desde la terminal ejecuta:

php artisan make:migration create_products_table

Ahora editaremos el archivo de migración en la ruta
database/migrations/2021_04_01_162458_**create_products_table.php**

verifica que dentro de la funcion up() tenga estos campos definidos la tabla products:

```
public function up()
   {
      Schema::create('products', function (Blueprint $table) {
         $table->id();
         $table->string('name');
         $table->integer('price');
         $table->longText('description');
         $table->timestamps();
      });
   }
```

Ejecutaremos la migración para que se cree la tabla **products** en nuestra base de datos así:

php artisan migrate

Si todo ha salido bien la tabla **products** fue creada y además encontraras una tabla llamada **migrations**, donde se almacenan los nombres de las migraciones ejecutadas, de lo contrario verifica las credenciales para conectarse a la base de datos en tu archivo **.env**

6. Creando Modelo Product

Procederemos a crear nuestro Modelo Product, para ello desde la terminal Ejecuta:

php artisan make:model Product

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Product extends Model
```

```
{

    //protected $table = 'products';

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = ['name', 'price', 'description'];

    /**
     * The attributes excluded from the model's JSON form.
     *
     * @var array
     */
    protected $hidden = ['created_at', 'updated_at'];
}
```

7. Creando PostController

Crearemos nuestro controlador PostController, con los métodos tipo crud para la api agregando --resource para tal fin:
php artisan make:controller PostController --resource

Ve al controlador en la ruta **app/Http/Controllers/PostController.php**, y edita el método index() debe quedar así:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

// añadido modelo product
use App\Models\Product;

class PostController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $products = Product::all();

        return response()->json($products);
```

```
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    //
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    //
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    //
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
```

```
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        //
    }
}
```

8. Instalando JWT

desde la terminal ejecuta:

composer require tymon/jwt-auth

edite el archivo **bootstrap/app.php**

// lineas descomentadas quitar las dobles plecas (//) que las anteceden a las directivas withFacades() y withEloquent(), deben quedar así:

$app->withFacades();
$app->withEloquent();

Vaya a la sección **Register Service Providers** del mismo archivo **bootstrap/app.php**, y descomente la linea (AuthServiceProvider) y agregue la linea de providers (Tymon\JWTAuth\Providers\) que se indica:

// **descomentado quitando las dobles plecas (//)para JWT**
$app->register(App\Providers\AuthServiceProvider::class);

// agregado JWT provider
$app->register(Tymon\JWTAuth\Providers\LumenServiceProvider::class);

Ahora busque en el mismo archivo **bootstrap/app.php,** la sección **Register Middleware**, y descomente, quitando las dobles plecas (//) la directiva (**$app->routeMiddleware**) siguiente:

```
$app->routeMiddleware([
    'auth' => App\Http\Middleware\Authenticate::class,
]);
```

Vamos a generar la llave secreta para nuestro proyecto, por lo que debes ejecutar desde la terminal:
php artisan jwt:secret

si vamos a nuestro archivo **.env**, debe haberse agregado de manera automática la directiva **JWT_SECRET**, que contiene nuestra llave generada

JWT_SECRET=IkrLvV4tQQJsI0ASEe2zpZYJpZKRXVdh6GCJZLPwimV9T6G1SGGquafbFiKPJgmS

Ahora editaremos nuestro Modelo User en la ruta **app/Models/User.php**, ya que debemos implementar **JWTSubject** interface, además agregaremos dos métodos **getJWTIdentifier()** y **getJWTCustomClaims()**.

```
<?php

namespace App\Models;

use Illuminate\Auth\Authenticatable;
use Illuminate\Contracts\Auth\Access\Authorizable as AuthorizableContract;
use Illuminate\Contracts\Auth\Authenticatable as AuthenticatableContract;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Laravel\Lumen\Auth\Authorizable;

// Añadido para JWT
use Tymon\JWTAuth\Contracts\JWTSubject;

class User extends Model implements AuthenticatableContract, AuthorizableContract, JWTSubject
{
    use Authenticatable, Authorizable, HasFactory;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email',
    ];
```

```
    /**
     * The attributes excluded from the model's JSON form.
     *
     * @var array
     */
    protected $hidden = [
        'password',
    ];



    // añadidos para JWT
    /**
     * Get the identifier that will be stored in the subject claim of the JWT.
     *
     * @return mixed
     */
    public function getJWTIdentifier()
    {
        return $this->getKey();
    }

    /**
     * Return a key value array, containing any custom claims to be added to the JWT.
     *
     * @return array
     */
    public function getJWTCustomClaims()
    {
        return [];
    }
}
```

Crear el directorio **config** en la raíz del proyecto y dentro de este el archivo **auth.php**

Editar el archivo **config/auth.php** y agregar lo siguiente:

```
<?php
// Añadido para utilizar JWT
return [
    'defaults' => [
        'guard' => 'api',
        'passwords' => 'users',
    ],

    'guards' => [
        'api' => [
```

```
            'driver' => 'jwt',
            'provider' => 'users',
        ],
    ],

    'providers' => [
        'users' => [
            'driver' => 'eloquent',
            'model' => App\Models\User::class
        ]
    ]
];
```

Edite el archivo **bootstrap/app.php** y en la sección **Register Config Files** agregue la linea siguiente $app->configure('auth'), con ello le indicamos a Lumen, que se lea nuestro archivo de configuración creado en la ruta **config/auth.php**.

```
// añadido para JWT
$app->configure('auth');
```

Creando las Rutas que serán sin autenticación (register y login) y protegiendo el resto de rutas definidas en el archivo **routes/web.php**, utilizando para ello middleware. Ve al archivo **routes/web.php** y agrega:

```
// rutas
$router->group(['prefix' => 'api'], function () use ($router) {

    // ruta para registrarse y loguearse sin protección
    $router->post('register', 'AuthController@register');
    $router->post('login', 'AuthController@login');

    // middleware para proteger las rutas
    $router->group(['middleware' => 'auth:api'], function () use ($router) {
        // rutas protegidas y que deben accederse unicamente con token
        $router->get('/post', 'PostController@index');
        $router->post('/post', 'PostController@store');
        $router->put('/post', 'PostController@update');
        $router->delete('/post', 'PostController@destroy');
        $router->post('logout', 'AuthController@logout');
        $router->post('refresh', 'AuthController@refresh');
    });
});
```

Finalmente crearemos nuestro controlador AuthController para manejar la autenticación de la api (app/Http/Controllers/**AuthController.php**):

php artisan make:controller AuthController

```
<?php

namespace App\Http\Controllers;

// añadido para JWT
use Illuminate\Support\Facades\Auth;

use Illuminate\Http\Request;

// validador
use Illuminate\Support\Facades\Validator;

// agregado para autenticarse JWT
use App\Models\User;

use App\Http\Controllers\Controller;

class AuthController extends Controller
{
    public function register(Request $request)
    {
        $this->validate($request, [
            'name'     => 'required',
            'email'    => 'required|email|unique:users',
            'password' => 'required|min:5',
        ]);

        $user = new User;
        $user->name     = $request->input('name');
        $user->email    = $request->input('email');
        $user->password = app('hash')->make($request->input('password'));
        $user->save();

        return response()->json(['message' => 'created'], 201);
    }
```

```
public function respondWithToken($token)
{
    // Return a token response to the user
    return response()->json([
        'token' => $token,
        'token_type' => 'bearer',
        'expires_in' => auth()->factory()->getTTL() * 60
    ], 200);
}


public function login(Request $request)
{
    $validator = Validator::make($request->all(), [
        'email' => 'required|string',
        'password' => 'required|string'
    ]);
    if ($validator->fails()) {
        return response()->json($validator->errors(), 400);
    }
    $credentials = $request->only(['email', 'password']);

    if (!$token = Auth::attempt($credentials)) {
        return response()->json(['message' => 'Unauthorized'], 401);
    }
    return $this->respondWithToken($token);
}

public function logout()
{
    // Add the token to a blacklist here
    auth()->logout();
    return response()->json(['message' => 'Logout Successfully !!!!'], 200);
}

/**
 * Refresh a token.
 *
 * @return \Illuminate\Http\JsonResponse
 */
public function refresh()
{
    return $this->respondWithToken(auth()->refresh());
}
```

```
    public function me()
    {
        return response()->json(auth()->user());
    }
}
```

Creando nuestra entidad users

desde la terminal ejecuta:

php artisan make:migration create_user_table --create=users

edita el archivo
database/migrations/2021_04_06_152928_**create_user_table.php**

Agrega las propiedades en tu entidad users en el método up() del archivo de migración así:

```
public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string("name");
             $table->string("email");
             $table->string("password");
            $table->timestamps();
        });
    }
```

Guarda el archivo y ejecuta la migración así:

php artisan migrate

9. Pruebas de autenticación

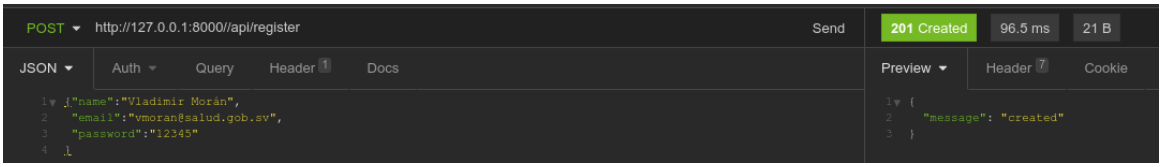
Abre la terminal y ejecuta:

php artisan serve

Utilizando Postman o Insomnia, crea los diferentes accesos para cada endpoint definido.

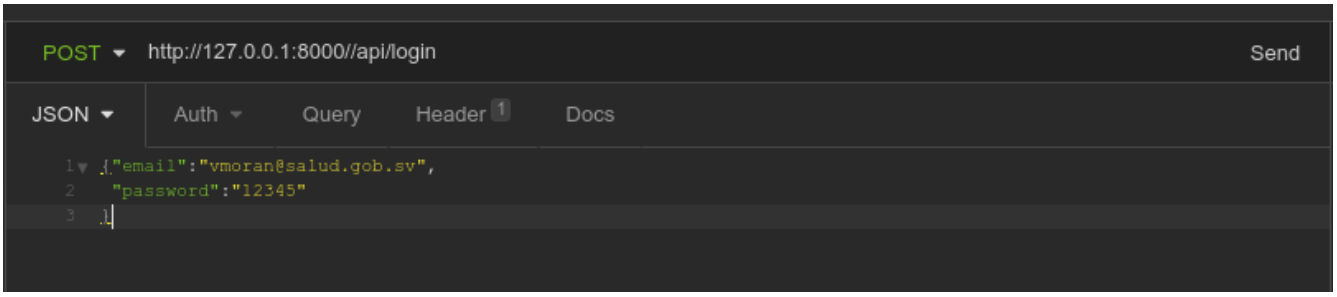
Register

url:http://127.0.0.1:8000/api/register
método: POST
type: JSON



Login

url: http://127.0.0.1:8000/api/login
metodo: POST
type: JSON



login te debe devolver el token a utilizar.

Listar productos

url:http://127.0.0.1:8000/api/post
metodo: GET
bearer token: poner el token obtenido en login xxxxxx

Refrescar Token

url:http://127.0.0.1:8000/api/refresh
metodo: post
Bearer: token obtenido en login xxxxxxxxxxxxxxxxxxx o en el refresh

Salir

url: http://127.0.0.1:8000/api/logout
metodo: POST
Bearer: token obtenido en login xxxxxxxxxxxxxxxxxxx o en el refresh

Referencias:
https://jwt-auth.readthedocs.io/en/develop/lumen-installation/

https://codekernel.co.uk/posts/lumen-jwt-authentication/