OBJETIVO:

Autenticar la API RESTful con un sistema de login con JWT

Contenido

Instalación de Tymeon/JWT-AUTH

Instalaremos "tymon/jwt-auth" en su ultima versión desarrollo, esta versión no es la release, pero esta preparada con distintas clases para funcionar en Lumen.

En nuestra terminal ingresaremos el siguiente comando, siempre en el directorio raíz de nuestro proyecto.

```
composer require tymon/jwt-auth:1.0.0-rc.2
```

Dicho comando instalará el **vendor** y se encargará de guardarlo en nuestro archivo ***composer.json***.

Crear el directorio config y el archivo auth.php

Si nuestro Lumen es virgen, deberemos crear el directorio config, luego en él crearemos un archivo llamado auth.php y lo rellenamos con el siguiente código:

```php
<?php
/**
 * User: riv4wi
 * Date: 29/04/18
 * Time: 17:07
 */
return [
    /*
    |--------------------------------------------------------------------------
    | Authentication Defaults
    |--------------------------------------------------------------------------
    |
    | This option controls the default authentication "guard" and password
    | reset options for your application. You may change these defaults
    | as required, but they're a perfect start for most applications.
    |
    */
    'defaults' => [
        'guard' => env('AUTH_GUARD', 'api'),
    ],
    /*
    |--------------------------------------------------------------------------
    | Authentication Guards
    |--------------------------------------------------------------------------
    |
    | Next, you may define every authentication guard for your application.
```

```
| Of course, a great default configuration has been defined for you
| here which uses session storage and the Eloquent user provider.
|
| All authentication drivers have a user provider. This defines how the
| users are actually retrieved out of your database or other storage
| mechanisms used by this application to persist your user's data.
|
| Supported: "session", "token"
|
*/
'guards' => [
    'api' => [
        'driver' => 'jwt',
        'provider' => 'users'
    ],
],
/*
|--------------------------------------------------------------------------
| User Providers
|--------------------------------------------------------------------------
|
| All authentication drivers have a user provider. This defines how the
| users are actually retrieved out of your database or other storage
| mechanisms used by this application to persist your user's data.
|
| If you have multiple user tables or models you may configure multiple
| sources which represent each model / table. These sources may then
| be assigned to any extra authentication guards you have defined.
|
| Supported: "database", "eloquent"
|
*/
'providers' => [
    'users' => [
        'driver' => 'eloquent',
        'model'  => \App\User::class,
    ],
],
/*
|--------------------------------------------------------------------------
| Resetting Passwords
|--------------------------------------------------------------------------
|
| Here you may set the options for resetting passwords including the view
| that is your password reset e-mail. You may also set the name of the
```

```
    | table that maintains all of the reset tokens for your application.
    |
    | You may specify multiple password reset configurations if you have more
    | than one user table or model in the application and you want to have
    | separate password reset settings based on the specific user types.
    |
    | The expire time is the number of minutes that the reset token should be
    | considered valid. This security feature keeps tokens short-lived so
    | they have less time to be guessed. You may change this as needed.
    |
    */
    'passwords' => [
        //
    ],
];
```

Habilitar routeMiddleware, registrar AuthServiceProvider, AppServiceProvider y LumenServiceProvider

Habilitar el middleware de autenticación, registrar el proveedor de autenticación y el de JWT en el archivo bootstrap/app.php

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


// Register AuthServiceProvider
$app->register(App\Providers\AuthServiceProvider::class);
// Register LumenServiceProvider
$app->register(Tymon\JWTAuth\Providers\LumenServiceProvider::class);
```

Crear Middleware para CORS domain

Ahora daremos un paso más en lineas de la comunicación bidireccional y montaremos un Middleware para el CORS, de este modo evitaremos tener problemas de cabeceras en caso

de usar dominios distintos (por ejemplo si quiero consultar desde http://manu.cloud a http://api.otrodominio.com.ar, con este Middleware no nos daría error de Cabeceras).

Crearemos un archivo php en el directorio /app/Http/Middleware llamado CorsMiddleware.php (/app/Http/Middleware/CorsMiddleware.php).

En su interior deben colocar el siguiente código:

```php
<?php
/**
 * User: riv4wi
 * Date: 29/04/18
 * Time: 22:22
 */
namespace App\Http\Middleware;
use Closure;
class CorsMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        $response = $next($request);
        $response->header('Access-Control-Allow-Origin','*');
        return $response;
    }
}
```

Agregar en el archivo de rutas las reglas para manejar las peticiones de tipo OPTIONS

Habilitar el middleware de autenticación, registrar el proveedor de autenticación y el de JWT en el archivo bootstrap/app.php

```
// Rules to fix messages of type OPTIONS, that usually emit in systems like Angular or others.
```

```
$router->options('/{any:.*}', function () {
    return response(['status' => 'success'])
        ->header('Access-Control-Allow-Methods','OPTIONS, GET, POST, PUT, DELETE')
        ->header('Access-Control-Allow-Headers', 'Authorization, Content-Type, Origin');
});
```

Registrar el CORS middleware

Habilitar el CORS middleware en el archivo bootstrap/app.php

```
// Middleware for apply rules to fix messages of type OPTIONS,

// that usually emit in systems like Angular or others.
$app->middleware([
    'CorsMiddleware' => App\Http\Middleware\CorsMiddleware::class
]);
```

Revisar que las fachadas y eloquent estén habilitados

Verificar en el archivo bootstrap/app.php

```
$app->withFacades();

$app->withEloquent();
```

Agregar contrato y métodos de JWT en el modelo de usuario

Ahora debemos declarar las funciones que JWT necesita en el modelo de usuario para poder generar el token y el login, por eso editaremos al archivo User.php que se encuentra en la carpeta App (app/User.php).

```
class User extends Model implements JWTSubject, AuthenticatableContract, AuthorizableContract

public function getJWTIdentifier()
```

```
{
    return $this->getKey();
}
public function getJWTCustomClaims()
{
    return [];
}
```

Adaptar método boot para validar que un email pertenece a un usuario

Revisar el archivo /var/www/laravel/apirf.lumen/app/Providers/AuthServiceProvider.php el método boot, este es el método que es responsable de autenticar al usuario.

```
$this->app['auth']->viaRequest('api', function ($request) {
    $api_token = $request->header('api-token'); // IMPORTANT: Name to header must be spinal-case
    if ($api_token && $api_token=='freedom'){
        return new User(); // This is didactic
    }
    return null;
});
```

Crear el controlador de autenticación

Ahora crearemos el archivo AuthController.php en el directorio app/Http/Controllers (app/Http/Controllers/AuthController.php) con el siguiente codigo:

```
<?php
namespace App\Http\Controllers;
/**
 * User: riv4wi
 * Date: 29/04/18
 * Time: 23:22
 */
use Illuminate\Http\Request;
use Tymon\JWTAuth\JWTAuth;
class AuthController extends Controller
{
```

```
    /**
     * @var \Tymon\JWTAuth\JWTAuth
     */
    protected $jwt;
    public function __construct(JWTAuth $jwt)
    {
        $this->jwt = $jwt;
    }
    public function postLogin(Request $request)
    {
        $this->validate($request, [
            'email'    => 'required|email|max:255',
            'password' => 'required',
        ]);
        try {
            if (! $token = $this->jwt->attempt($request->only('email', 'password'))) {
                return response()->json(['invalid_credentials'], 401);
            }
        } catch (\Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
            return response()->json(['token_expired'], 500);
        } catch (\Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {
            return response()->json(['token_invalid'], 500);
        } catch (\Tymon\JWTAuth\Exceptions\JWTException $e) {
            return response()->json(['token_absent' => $e->getMessage()], 500);
        }
        return response()->json(compact('token'));
    }
}
```

Crear la ruta de autenticación

Ahora debemos agregar una nueva ruta para autenticar a nuestro usuario, esto lo haremos en el archivo web.php de routes (routes/web.php)

```
$router->post('/auth/login', 'AuthController@postLogin');
$router->group(['prefix' => 'v1', 'middleware' => 'auth:api'], function () use ($router)
```

Referencias web adicionales

http://manu.cloud/framework/lumen/inicio-de-sesion-con-jwt-1-0-0-rc2-en-lumen-5-6/