Публикация на тему

Аутентифиация пользователей в Node.js с использованием JWT и Laravel

В этом руководстве мы аутентифицируем пользователя в Node, в то время как основа нашей платформы построена на Laravel.

Анотация

Аутентификация пользователей с помощью различных методов может быть утомительной и подверженной ошибкам работой. JWT предлагает унифицированный метод для упрощения этого процесса. В этом руководстве мы аутентифицируем пользователя в Node, в то время как основа нашей платформы построена на Laravel.

Authenticating users on different techniques can be tedious and error-prone work. JWT offers a uniform method to simplify this process. In this tutorial, we will authenticate a user in Node, while the basis of our platform is built upon Laravel.

Автор

Михалькевич Александр Викторович

Публикация

Наименование Аутентифиация пользователей в Node.js с использованием JWT и Laravel

Автор А.В.Михалькевич

Специальность В этом руководстве мы аутентифицируем пользователя в Node, в то время как основа нашей платформы построена на Laravel.,

Анотация

Аутентификация пользователей с помощью различных методов может быть утомительной и подверженной ошибкам работой. JWT предлагает унифицированный метод для упрощения этого процесса. В этом руководстве мы аутентифицируем пользователя в Node, в то время как основа нашей платформы построена на Laravel.

Anotation in English

Authenticating users on different techniques can be tedious and error-prone work. JWT offers a uniform method to simplify this process. In this tutorial, we will authenticate a user in Node, while the basis of our platform is built upon Laravel.

Ключевые слова Сокет, Socket, Socket.io, Laravel, Node, Node.js, аутентификация пользователя, JWT

Количество символов 15006

Содержание

Введение
1 Две части приложения
2 Возможные решения
3 Установка модуля JWT в Laravel
4 JWT-аутентификация в Laravel
5 Аутентифиция пользователей с помощью терминала
6 Реализация в Node.js
7 Запуск сервера Node.js, тестирование и необходимые пояснения
Заключение
Список использованных источников
Приложения

Введение

Данное решение будет полезно для реализации чата. Если основное приложение создано на Laravel, а сокетное соединение осуществляется с помощью Node.js

1 Две части приложения

Итак, у нас имеется приложение, состоящее из двух частей:

1. Административная часть на основе Laravel: клиенты могут зарегистрироваться и войти в свою учетную запись, а также настроить свой профиль.

2. Приложение чата, которое мы хотитим запускать исключительно с использованием Node.js / socket.io, но разрешить подключение необходимо только пользователелям, уже вошедшим в систему Laravel.

2 Возможные решения

Как в node.js получить авторизацию пользователя из Laravel?

Один из вариантов, который, который, возможно, является самым очевидным, - это взять идентификатор сеанса пользователя из файла cookie. По нему сделать запрос в базу данных, и тогда в node.js мы сможем в проанализировать ответ запроса и выяснить, аутентифицирован ли пользователь. И затем - получить пользовательские данные.

Другие, не очевидные варианты, - сохранение сеансов в Redis, сохранение токенов в cookie и т.д. Чтение сеансов или токенов позволит пользователю снова войти в систему с именем пользователя и паролем в приложении чата.

Но почему бы не использовать проверенный отраслевой стандарт: **веб-токены JSON (JWT)**?

Это позволит нам:

генерировать токены внутри Laravel;
представлять их браузеру через API;

позволять браузеру использовать этот токен для входа в Node.js/socket.io.
По сути мы также можем добавить информацию о пользователе.

Поскольку мы используем общий ключ в файле .env, нам не нужно хранить его в нашем коде. Вот решение, которого нам необходимо достичь:

Браузер запрашивает токен у Laravel.
Клиент подключается к Node.js / socket.io и аутентифицируется с помощью токена.
Node.js / socket.io проверяет токен и принимает дальнейшие сообщения.

3 Установка модуля JWT в Laravel

Для установки модуля JWT, сперва обновим composer

```
composer self-update
```

После чего, в своём Laravel-проекте выполним следующую команду

```
composer require tymon/jwt-auth
```

Добавим ServiceProvider в файле `config/app.php`

```
'providers' => [
    ...
    Tymon\JWTAuth\Providers\LaravelServiceProvider::class,
]
```

Для того, чтобы опубликовать файл конфигурации, выполним следующую команду:

```
php artisan vendor:publish --
provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
```

Теперь у нас появился файл `config/jwt.php`, который позволяет настраивать основы этого пакета.

Генерируем ключ:

```
php artisan jwt:secret
```

Это обновит файл `.env` чем-то вроде JWT_SECRET = foobar

4 JWT-аутентификация в Laravel

Обновляем модель User.

Для начала нужно имплементировать контракт `Tymon\JWTAuth\Contracts\JWTSubject`, который обязует реализовать два метода `getJWTIdentifier()` и `getJWTCustomClaims()`.

```
namespace App;

use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Notifications\Notifiable;
```

```
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable implements JWTSubject
{
    use Notifiable;

    // Rest omitted for brevity

    /**
     * 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 [];
    }
}
```

В файле `config/auth.php` переключаем драйвер guards api на `jwt`

```
'defaults' => [
    'guard' => 'web',
    'passwords' => 'users',
],

...

'guards' => [
    'api' => [
        'driver' => 'jwt',
        'provider' => 'users',
    ],
],
```

Создаем следующие маршруты:

```
 Route::get('auth/login', 'AuthController@login');
 Route::get('auth/logout', 'AuthController@logout');
```

```
Route::get('auth/token', 'AuthController@token');
Route::get('auth/me', 'AuthController@me');
Route::get('auth/tokenid', 'AuthController@tokenId');
```

Контроллер AuthController:

```
namespace App\Http\Controllers;

use Auth;
use App\Http\Controllers\Controller;

class AuthController extends Controller
{

    public function login()
    {
        $credentials = request(['email', 'password']);

        if (! $token = auth('web')->attempt($credentials)) {
            return response()->json(['error' => 'Unauthorized'], 401);
        }

        return $this->respondWithToken($token);
    }
    public function me()
    {
        return response()->json(auth('web')->user());
    }

    public function token()
    {
        return response()->json(auth('web')->user()->getRememberToken());
    }

    public function tokenId(){

        if (! $token = auth('api')->tokenById(Auth::user()->id)) {
            return response()->json(['error' => 'Unauthorized'], 401);
        }

        return $this->respondWithToken($token);
    }

    public function logout()
    {
        auth()->logout();
        return response()->json(['user'=>'logout']);
    }
```

```
    protected function respondWithToken($token)
    {
        return response()->json([
            'access_token' => $token,
            'token_type' => 'bearer',
            'expires_in' => auth('api')->factory()->getTTL() * 60
        ]);
    }
}
```

Прежде чем переходить к тестированию, иногда требуется обнулить кэш и маршруты:

```
php artisan config:cache
php artisan route:clear
php artisan cache:clear
```

5 Аутентификация пользователей с помощью терминала

Для аутентификации пользователей можем воспользоваться следующей командой:

```
curl -X GET -i
'http://localhost:8000/auth/login?email=mikhalkevich@ya.ru&password=11111111'
```

6 Реализация в Node.js

К этой части переходим, если у нас уже есть приложение на Laravel.

На стороне Node.js будем использовать следующие пакеты:

Express чтобы быстро вернуть html страницу, на которой будем выводить результат.
Socket.io для обработки сокетов.
Dotenv для чтения общего ключа из файла Laravel .env.
Socketio-jwt для обработки JWT аутентификации в socket.io
Mysql для подключения к базе данных Mysql

Переходим в папку node. И устанавливаем необходимые зависимости:

```
npm install express socket.io socketio-jwt dotenv mysql --save
```

Теперь внутри папки node, создаем файл auth.js со следующим содержимым:

```
// Load basics
var express = require('express');
var app = express();
var server = require('http').createServer(app);
var io = require('socket.io')(server);
var socketioJwt = require('socketio-jwt');
require('dotenv').config({path: '../laravel/.env'});
```

В этих строках кода мы подключили все необходимые компоненты, которые только что

установили. В строке `dotenv` загрузили файл .env из Laravel.

Далее подключаемся к базе данных:

```
var db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    port: '3307',
    database: 'world',
    password: ''
})
```

Настраиваем сокетное соединение:

```
io.on('connection', socketioJwt.authorize({
    secret: process.env.JWT_SECRET,
    timeout: 15000
}));// When authenticated, send back name + email over socket
io.on('authenticated', function (socket) {
    console.log(socket.decoded_token, socket.decoded_token.sub);
    socket.emit('id', socket.decoded_token.sub);
});
server.listen(3003);
```

На стороне Laravel пишем клиента:

```
        $(document).ready(function () {
            const socket = io.connect('http://localhost:3003');
            socket.on('connect', () => {
                socket
                    .emit('authenticate', { token:
'{{auth('api')->tokenById(auth::user()->id)}}' }) //send the jwt
                    .on('authenticated', () => {
                        socket.on('id', function (rows) {
                            var html = JSON.stringify(rows);
                            connectionUpdate(html);
                        });
                    })
                    .on('unauthorized', (msg) => {
                        console.log(`unauthorized:
${JSON.stringify(msg.data)}`);
                        throw new Error(msg.data.type);
                    })
            });
        });

        function connectionUpdate(str) {
            $('#connection').html(str);
            console.log(str);
        }
```

Осталось создать элемент с id="connection"

Node.js отправляет id аутентифицированного пользователя. Чтобы это работало, нам также нужно позволить Laravel создать токен и убедиться, что браузер действительно может получить этот токен.

7 Запуск сервера Node.js, тестирование и необходимые пояснения

Теперь запустим наш Node.js сервер:

```
cd ../node
node auth.js
```

Теперь снова вернемся в браузер, по адресу:

```
http://localhost:3000
```

Произойдет следующее:

> Node.js / Express реализует файл auth.html;
> auth.html отправляет запрос на адрес http://localhost/token и если пользователь авторизован, Laravel сгенерирует токен, который возвращается обратно в auth.html
> В этом же файле происходит соединение к Node.js / socket.io;
> Аутентификация браузера будет происходить с помощью сгенерированного токена;
> В случае успешного соединения, в консоли мы увидим данные аутентифицированного пользователя (в переменной $claims);
> Браузер покажет нам статус аутентифицированного пользователя, его имя и адрес электронной почты,

Терминал покажет примерно следующее:

```
{ name: 'Test',
  email: 'stefan@efectos.nl',
  sub: 1, // Laravel User ID
  iss: 'http://laravel.nodejs/token',
  iat: 1461934603,
  exp: 1461934603,
  nbf: 1461934603,
  jti: '88937e518256845b50a585de07ca4c0d' }
```

> Примечание: аутентификация и проверка токена выполняются только при подключении. Если браузер повторно подключится, он будет снова проверен, и если TTL истек, клиент больше не войдет в систему. Но если клиент никогда не отключается (а сервер продолжает работать), клиент входит в систему навсегда и никогда не выходит из системы. Это также означает, что пользователь чат-приложения все еще находится в системе, даже если сеанс Laravel завершен (истек или вышел из системы вручную).

Заключение

Надеюсь, что этот урок и пример помогут вам в работе над вашим потрясающим проектом!

Список использованных источников

1. [url] **Authenticate users in Node using JWT and Laravel** https://m.dotdev.co/authenticate-laravel-5-user-account-in-nodejs-socket-io-using-json-web-tokens-jwt-f74009d612f8
2. [url] **Socket.IO Server on Node.js** https://github.com/mikhalkevich/NodeSocket

Приложения