[**https://viblo.asia/p/laravel-xac-thuc-api-voi-laravel-56-passport-authentication-part1-Do754p8V5M6**](https://viblo.asia/p/laravel-xac-thuc-api-voi-laravel-56-passport-authentication-part1-Do754p8V5M6)

**[Laravel] Xác thực API với Laravel 5.6 Passport Authentication (Part1)**

This post has been more than **2 years** since it was last updated.

Chào mọi người, hôm nay mình lại quay lại đây, hôm nay mình sẽ hướng dẫn mọi người xác thực API bằng laravel passport

Trong các ứng dụng phần mềm hiện đại, các web API là không thể thiếu, có rất nhiều các mô hình ứng dụng sử dụng web API như mô hình server-to-server, hay mô hình SPA (Single Page Application). Trong quá trình phát triển các API, rất cần thiết phải bảo vệ dữ liệu khỏi những con mắt tò mò, điều này với các hệ thống truyền thống rất đơn giản còn với API thì sao? Laravel tạo ra một gói thư viện Passport giúp thực hiện xác thực trong API đơn giản đơn giản hơn, nó cung cấp đầy đủ OAuth2. Laravel Passport được xây dựng dựa trên League OAuth2 server được phát triển bởi Alex Bilbie Chúng ta bắt đầu luôn nhé

**Install Laravel**

Chúng ta tải laravel thông qua composer nhé

composer create-project --prefer-dist laravel/laravel auth

khi tải về mn nhớ tạo 1 file .env và chạy composer install và php artisan key:generate đây là những bước bình thường mình sẽ không nói sâu đến nữa

**Install Laravel Passport Package**

Chúng t a chỉ cần tải thư việc này về bằng composer nhé

composer require laravel/passport

Giống như hầu hết các package cho Laravel khác, chúng ta cần thêm service provider của Laravel Passport trong file config/app.php (việc thêm service provider cần được thực hiện sau khi package đã được tải về thông qua Composer)

Laravel\Passport\PassportServiceProvider::class,

Laravel Passport cung cấp sẵn một số migration class để tạo các bảng cần thiết để lưu trữ authorization codes, access tokens, refresh tokens, personal access tokens, thông tin về clients (danh sách các file migration có thể xem tại đây). Để cài đặt các bảng cần thiết cho Laravel Passport, chúng ta dùng lệnh sau:

php artisan migrate

**Generate Key**

Bước tiếp theo, chúng ta cần tạo encryption keys (được dùng khi tạo access tokens) và tạo các client liên quan đến Personal Access Grant và Password Grant

php artisan passport:install

sau khi chạy xong chúng ta thêm Laravel\Passport\HasApiTokens vào model 'App\user'. Trait này sẽ cung cấp 1 vài hepler cho phép kiểm tra token và phạm vi của người dùng

**Passport Config**

<?php

namespace App;

use Illuminate\Notifications\Notifiable;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Laravel\Passport\HasApiTokens;

class User extends Authenticatable

{

use Notifiable, HasApiTokens;

}

Tiếp theo là gọi phương thức Passport::routes trong AuthServiceProvider, Phương thức này dùng để đăng ký những routes cần thiết để cho những vấn đề liên quan đến access tokens and revoke access tokens, clients, and personal access tokens

<?php

namespace App\Providers;

use Laravel\Passport\Passport;

use Illuminate\Support\Facades\Gate;

use Illuminate\Foundation\Support\Providers\AuthServiceProvider as ServiceProvider;

class AuthServiceProvider extends ServiceProvider

{

/\*\*

\* The policy mappings for the application.

\*

\* @var array

\*/

protected $policies = [

'App\Model' => 'App\Policies\ModelPolicy',

];

/\*\*

\* Register any authentication / authorization services.

\*

\* @return void

\*/

public function boot()

{

$this->registerPolicies();

Passport::routes();

}

}

Cuối cùng là tại file config/auth.php, bạn nên chọn driver của api à passport, Điều này sẽ hướng ứng dụng của bạn sử dụng Passport's TokenGuard khi các thực yêu của của API

'guards' => [

'web' => [

'driver' => 'session',

'provider' => 'users',

],

'api' => [

'driver' => 'passport',

'provider' => 'users',

],

],

**Create Api Routes**

<?php

use Illuminate\Http\Request;

Route::group([

'prefix' => 'auth'

], function () {

Route::post('login', 'AuthController@login');

Route::post('signup', 'AuthController@signup');

Route::group([

'middleware' => 'auth:api'

], function() {

Route::get('logout', 'AuthController@logout');

Route::get('user', 'AuthController@user');

});

});

**Create Controller**

Chúng ta tạo 1 controller là AuthController nhé

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Carbon\Carbon;

use App\User;

class AuthController extends Controller

{

/\*\*

\* Create user

\*

\* @param [string] name

\* @param [string] email

\* @param [string] password

\* @param [string] password\_confirmation

\* @return [string] message

\*/

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)

]);

$user->save();

return response()->json([

'message' => 'Successfully created user!'

], 201);

}

/\*\*

\* Login user and create token

\*

\* @param [string] email

\* @param [string] password

\* @param [boolean] remember\_me

\* @return [string] access\_token

\* @return [string] token\_type

\* @return [string] expires\_at

\*/

public function login(Request $request)

{

$request->validate([

'email' => 'required|string|email',

'password' => 'required|string',

'remember\_me' => 'boolean'

]);

$credentials = request(['email', 'password']);

if(!Auth::attempt($credentials))

return response()->json([

'message' => 'Unauthorized'

], 401);

$user = $request->user();

$tokenResult = $user->createToken('Personal Access Token');

$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()

]);

}

/\*\*

\* Logout user (Revoke the token)

\*

\* @return [string] message

\*/

public function logout(Request $request)

{

$request->user()->token()->revoke();

return response()->json([

'message' => 'Successfully logged out'

]);

}

/\*\*

\* Get the authenticated User

\*

\* @return [json] user object

\*/

public function user(Request $request)

{

return response()->json($request->user());

}

}

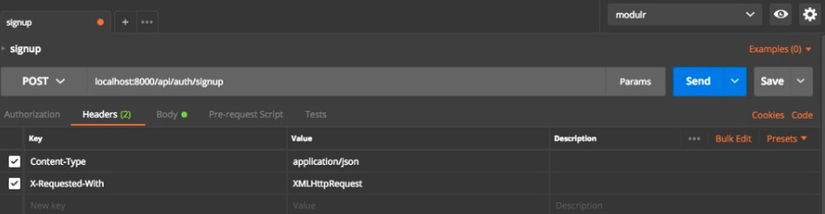
OK ngon rồi đấy, giờ chạy php artisan serve để test nhé

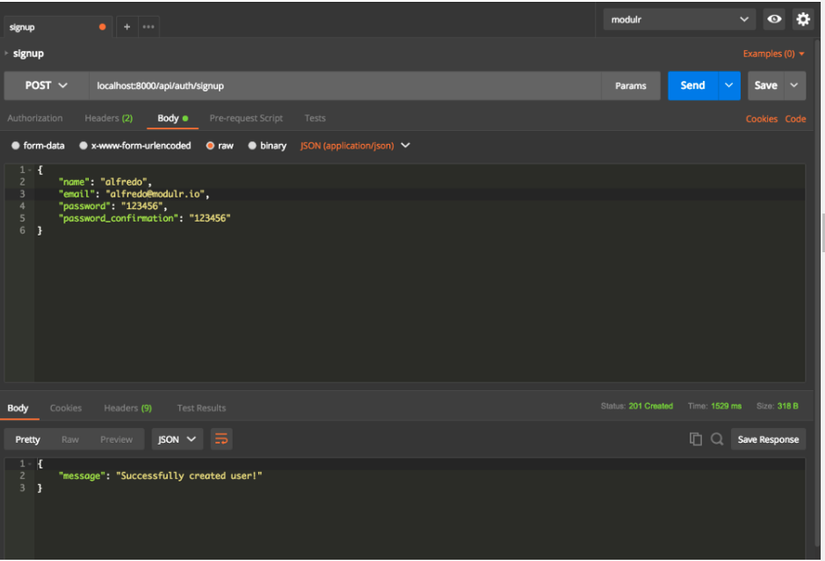
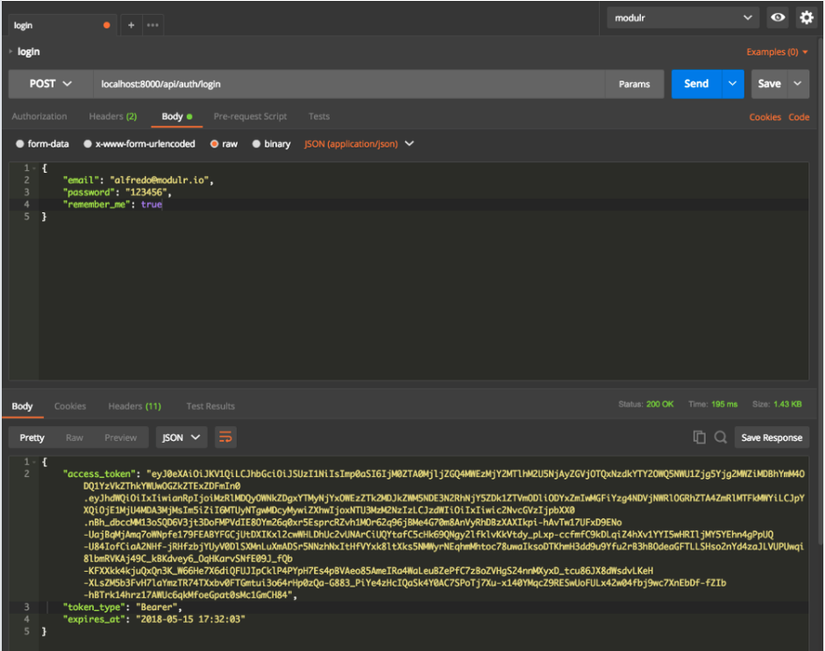
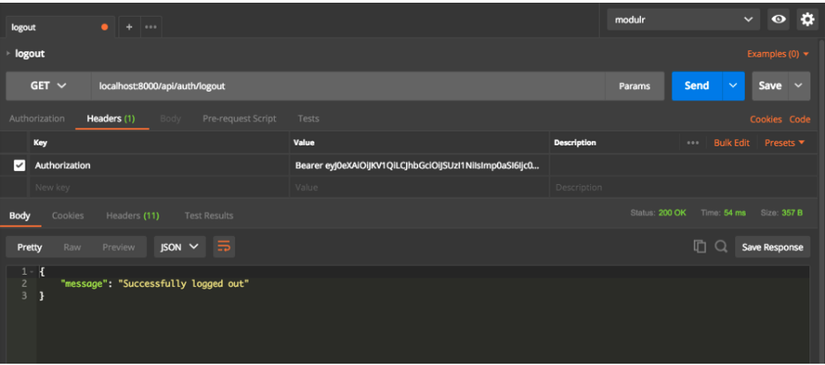
**Tests**

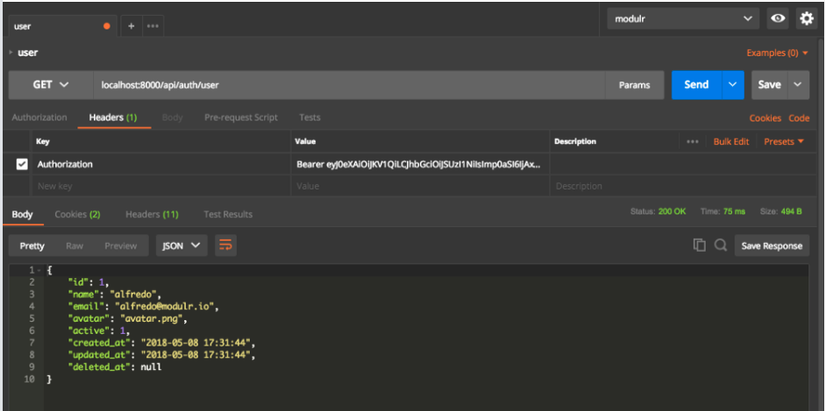
Chúng ta xử dụng Postman để test api nhé, trong phần Header mn nhớ có key này nhé

Content-Type: application/json

X-Requested-With: XMLHttpRequest



* Api Signup
* Api Login
* Api Logout
* Api User



**End**