1. Authenticate:

* Error: Nếu sử dụng code này, nó sẽ bug ra cái exception chứ không phải là cái response json

// somewhere in your controller

public function getAuthenticatedUser()

{

try {

if (! $user = JWTAuth::parseToken()->authenticate()) {

return response()->json(['user\_not\_found'], 404);

}

} catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {

return response()->json(['token\_expired'], $e->getStatusCode());

} catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {

return response()->json(['token\_invalid'], $e->getStatusCode());

} catch (Tymon\JWTAuth\Exceptions\JWTException $e) {

return response()->json(['token\_absent'], $e->getStatusCode());

}

// the token is valid and we have found the user via the sub claim

return response()->json(compact('user'));

}

* Fix cách 1: thêm **\** trước các namespace Exception

public function getAuthenticatedUser()

{

try {

if (! $user = JWTAuth::parseToken()->authenticate()) {

return response()->json(['user\_not\_found'], 404);

}

} catch (**\**Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {

return response()->json(['token\_expired'], $e->getStatusCode());

} catch (**\**Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {

return response()->json(['token\_invalid'], $e->getStatusCode());

} catch (**\**Tymon\JWTAuth\Exceptions\JWTException $e) {

return response()->json(['token\_absent'], $e->getStatusCode());

}

// the token is valid and we have found the user via the sub claim

return response()->json(compact('user'));

}

* Fix cách 2: nên use mấy cái exception đó trước khi dùng

public function getAuthenticatedUser()

{

try {

if (! $user = JWTAuth::parseToken()->authenticate()) {

return response()->json(['user\_not\_found'], 404);

}

} catch (TokenExpiredException $e) {

return response()->json(['token\_expired'], $e->getStatusCode());

} catch (TokenInvalidException $e) {

return response()->json(['token\_invalid'], $e->getStatusCode());

} catch (JWTException $e) {

return response()->json(['token\_absent'], $e->getStatusCode());

}

// the token is valid and we have found the user via the sub claim

return response()->json(compact('user'));

}

1. Auto refresh token

* Nên tạo 1 middleware để check, nếu nó hết hạn thì create lại token mới và trả về
* Example: file APIAuth middleware

public function handle($request, Closure $next)

{

try

{

if (! $user = JWTAuth::parseToken()->authenticate() )

{

return response()->json([

'code' => 101 // means auth error in the api,

'response' => null // nothing to show

]);

}

}

catch (TokenExpiredException $e)

{

// If the token is expired, then it will be refreshed and added to the headers

try

{

$refreshed = JWTAuth::refresh(JWTAuth::getToken());

$user = JWTAuth::setToken($refreshed)->toUser();

header('Authorization: Bearer ' . $refreshed);

}

catch (JWTException $e)

{

return response()->json([

'code' => 103 // means not refreshable

'response' => null // nothing to show

]);

}

}

catch (JWTException $e)

{

return response()->json([

'code' => 101 // means auth error in the api,

'response' => null // nothing to show

]);

}

// Login the user instance for global usage

Auth::login($user, false);

return $next($request);

}

* Sau đó đăng ký middleware trong file Kernel.php