**TỔNG QUAN VỀ JSON WEB TOKEN**

# Giới thiệu về thư viện JWT

* JWT(JSON Web token) là một tiêu chuẩn mở (RFC 7519) định nghĩa một cách nhỏ gọn và an toàn để truyền tải thông tin giữa các bên một cách an toàn dưới dạng 1 đối tượng JSON . Các thông tin này được xác thực và có độ tin cậy cao vì nó có chứa chữ ký số.
* Hiểu một cách đơn giản thì đây là thư viện giúp *encode()* và *decode()* một mảng các thông tin ta đưa vào thành một chuỗi json theo một template mà nó sẽ đảm bảo được bảo mật.
* Đối vơi PHP, thư viện hỗ trợ sử dụng JWT là **PHP – JWT** <https://github.com/firebase/php-jwt>
* ứng dụng phổ biến nhất của JWT chính là sử dụng lưu trữ token cho việc đăng nhập của người dùng.
* Đối với Laravel và đăng nhập, PHP thường sử dụng API Authentication sử dụng JWT trong Laravel

# Khi nào nên sử dụng JWT

JWT thường được sử dụng với Authentication (Authentication là quá trình xác định và xác nhận danh tính của một khách truy cập vào một ứng dụng). Khi người dùng đã đăng nhập vào hệ thống, thì những request tiếp theo của người dùng gửi lên sẽ phải bao gồm JWT token. Nếu JWT token đó có quyền thì người dùng mới được truy cập các dịch vụ, tài nguyên và tương tác với cơ sở dữ liệu. JSON Web Tokens là cách tốt nhất để trao đổi thông tin giữa các bên một cách an toàn. **JWT cho phép tất cả các đặc tính này áp dụng cho API Authentication và thường được đặt trong HTTP Authorization headers**.

Sử dụng JWT là cách tốt để áp dụng cơ chế bảo mật đối với các dịch vụ API RESTFUL mà có thể được sử dụng để truy cập vào cơ sở dữ liệu của bạn.

# **JSON Web Tokens hoạt động như thế nào?**

* Trong quá trình xác thực, người dùng đăng nhập thành công bằng cách sử dụng các thông tin của họ (email or username, password), JSON Web Token sẽ được trả lại và phải được lưu lại dưới local (thường là trong local storage, nhưng có lúc cookie cũng có thể được sử dụng) thay vì cách truyền thống là tạo ra một session trên server và trả lại cookie.
* Bất cứ khi nào người dùng muốn truy cập vào route hoặc tài nguyên cần có quyền, họ phải gửi JWT trong **Authorization** header sử dụng Bearer schema như sau: Authorization: Bearer <token> **?**
* Để bảo mật, trong chuỗi JSON không được chứa field password phòng người khác bắt được chuỗi token và giải mã chuỗi token.

# Cài đặt và sử dụng PHP-JWT

## Cài đặt:

* Sử dụng cmd để run lệnh: composer require firebase/php-jwt.
* Sau khi run thành công sẽ xuất hiện các file cần thiết gồm: thư mục vendor, composer.json, composer.lock.

## Sử dụng:

* Sau khi cài đặt xong, bất cứ file nào muốn sử dụng thư viện, ta cần import thư viện như sau:

<?php

// initializing variables

use \Firebase\JWT\JWT;

require \_\_DIR\_\_ . '/vendor/autoload.php';

* Dưới đây là một ví dụ về sử dụng PHP-JWT

<?php

use \Firebase\JWT\JWT;

$key = "example\_key";

$token = array(

"iss" => "http://example.org",

"aud" => "http://example.com",

"iat" => 1356999524,

"nbf" => 1357000000

);

/\*\*

\* IMPORTANT:

\* You must specify supported algorithms for your application. See

\* https://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms-40

\* for a list of spec-compliant algorithms.

\*/

$jwt = JWT::encode($token, $key);

$decoded = JWT::decode($jwt, $key, array('HS256'));

print\_r($decoded);

/\*

NOTE: This will now be an object instead of an associative array. To get

an associative array, you will need to cast it as such:

\*/

$decoded\_array = (array) $decoded;

/\*\*

\* You can add a leeway to account for when there is a clock skew times between

\* the signing and verifying servers. It is recommended that this leeway should

\* not be bigger than a few minutes.

\*

\* Source: http://self-issued.info/docs/draft-ietf-oauth-json-web-token.html#nbfDef

\*/

JWT::$leeway = 60; // $leeway in seconds

$decoded = JWT::decode($jwt, $key, array('HS256'));

?>

Trong đó:

$decoded = JWT::decode($token, $key, array('HS256'));

* + $token là một mảng các thông tin cần được mã hóa
  + $key là tên của token ta cần lưu để phân biệt giữa các token (do developer đặt)
  + array(‘HS256’) là một kiểu mã hóa mà thư viện hỗ trợ sẵn. Đây là kiểu mã hóa phổ biến hay sử dụng nhất. Ngoài kiểu mã hóa **HS256** thư viện còn hỗ trợ các kiểu sau:

public static $supported\_algs = array(

'HS256' => array('hash\_hmac', 'SHA256'),

'HS512' => array('hash\_hmac', 'SHA512'),

'HS384' => array('hash\_hmac', 'SHA384'),

'RS256' => array('openssl', 'SHA256'),

'RS384' => array('openssl', 'SHA384'),

'RS512' => array('openssl', 'SHA512'),

);

* + các kiểu mã hóa trên được khai báo trong file: \vendor\firebase\php-jwt\src\JWT.php
  + Ngoài ra, developer cũng có thể định nghĩa riêng kiểu mã hóa cho mình.