

# Topic #7 – Laravel Sail Docker - Passport

Education Team



LAMPART

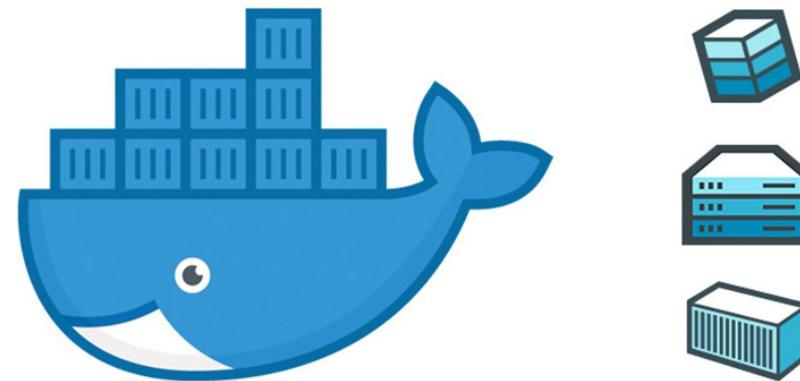
Laravel version : 8x



## 1. Docker

### a) Docker là gì

- Docker là một **open platform**
- Sử dụng ý tưởng **cô lập các tài nguyên** để **đóng gói ứng dụng** cùng **tất cả các thành phần phụ** thuộc để có thể **chạy ở bất kì đâu.**

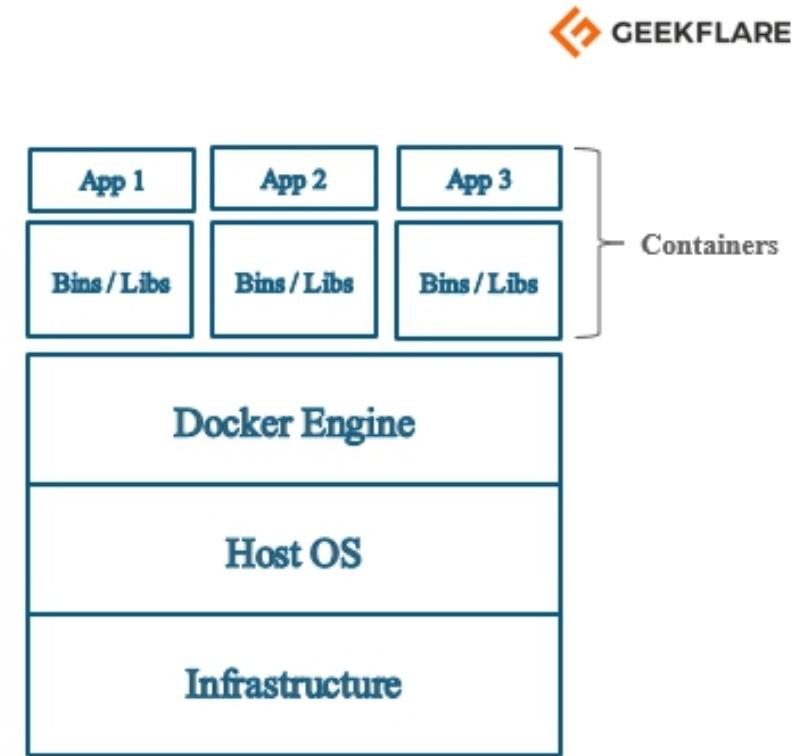
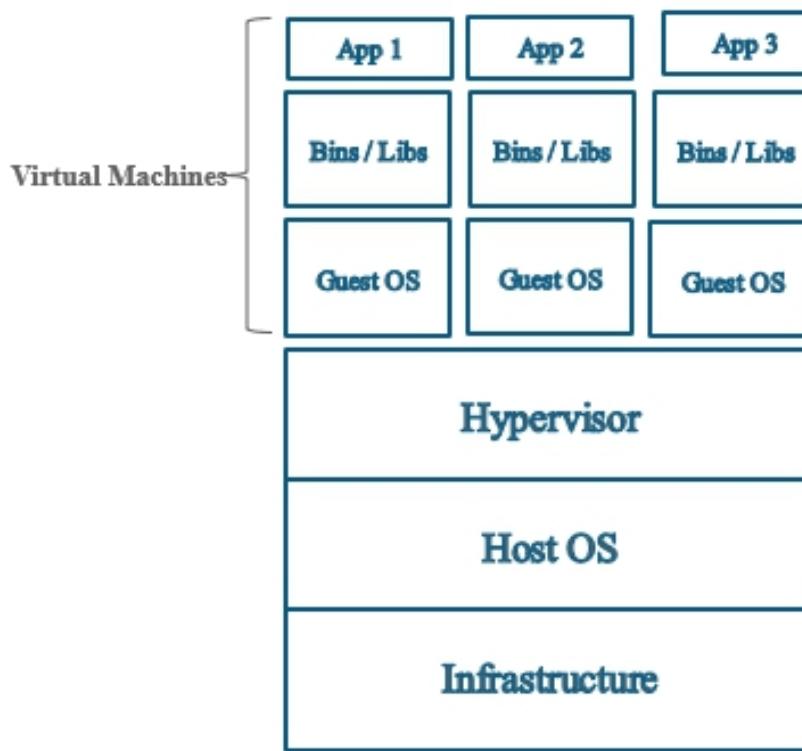




# 1. Docker

## b) Docker vs VM

- Docker sử dụng hệ thống tập tin phân lớp (AuFS)





## 1. Docker

---

### b) Docker vs VM

- Phần Hệ điều hành dùng chung trong Docker gọi là **image**.
- Phần dùng riêng trong Docker gọi là **Container**.



# 1. Docker

## b) Docker vs VM

Virtual Machine	Docker
cô lập quá trình cấp phần cứng	cô lập quá trình cấp hệ điều hành
Mỗi VM cần một OS riêng biệt, hoàn chỉnh	Mỗi container có thể chia sẻ OS
VM có thể di chuyển qua máy chủ khác dễ dàng	Container chỉ có thể xoá đi và cài lại chứ không thể di chuyển
Tạo ra VM mất nhiều thời gian	Container có thể tạo ra trong vài giây
Sử dụng nhiều tài nguyên hơn	Sử dụng ít tài nguyên hơn
VM được làm sẵn khó tìm	Dễ dàng có được các container tích hợp sẵn
VM chiếm vài GB bộ nhớ	Các container chỉ chiếm vài KB/MB



## 1. Docker

### c) Docker không phải là “viên đạn bạc”

- Docker không tự khắc phục những lỗ hổng bảo mật của ứng dụng.
- Docker không biến những ứng dụng microservices trở nên kì diệu hơn.
- Docker không thay thế cho máy ảo.



## 1. Docker

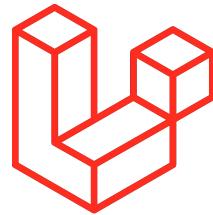
### d) Cài đặt Docker



<https://docs.docker.com/desktop/>

## 2. Laravel Sail

---



<https://laravel.com/docs/8.x/sail>

## 2. Laravel Sail

### a) Laravel sail

là một **cli** hỗ trợ môi trường phát triển **Docker** của laravel.

- Cài đặt sail package: **composer require laravel/sail --dev**
- public sail docker-compose file:

```
php artisan sail:install
```

## 2. Laravel Sail

### b) Docker compose file



docker-compose.yml

```
# For more information: https://laravel.com/docs/sail
version: '3'
services:
    laravel.test:
        build:
            context: ./vendor/laravel/sail/runtimes/8.0
            dockerfile: Dockerfile
            args:
                WWWGROUP: '${WWWGROUP}'
        image: sail-8.0/app
        ports:
            - '${APP_PORT:-80}:80'
        environment:
            WWWUSER: '${WWWUSER}'
            LARAVEL_SAIL: 1
        volumes:
            - '.:/var/www/html'
        networks:
            - sail
depends_on:
    - mysql
```



## 2. Laravel Sail

b) Docker compose file



docker-compose.yml

```
mysql:  
  image: 'mysql:8.0'  
  ports:  
    - '${FORWARD_DB_PORT:-3306}:3306'
```

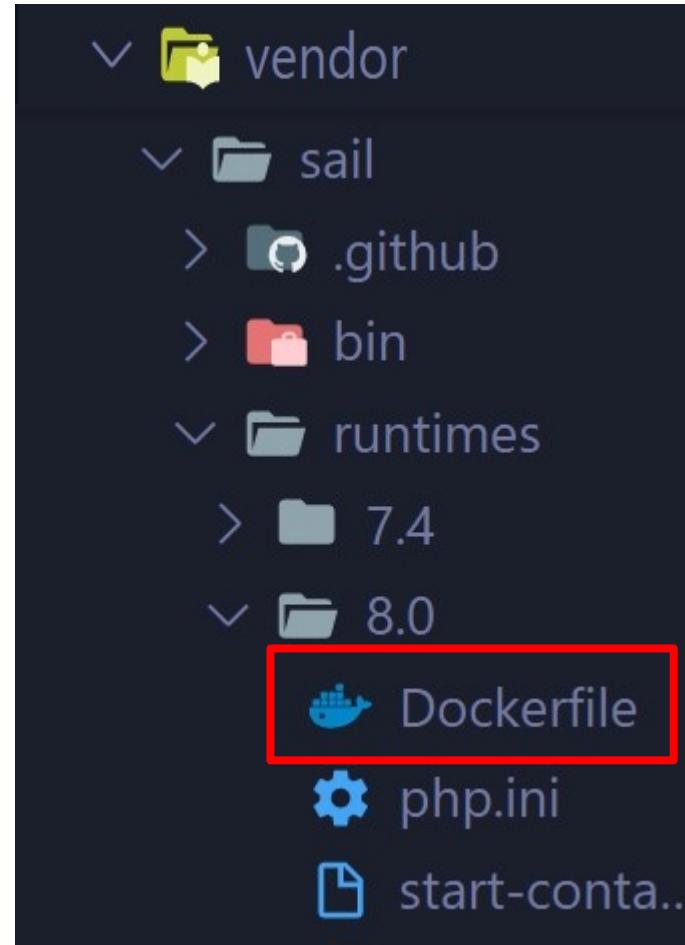
## 2. Laravel Sail

### b) Docker compose file

```
laravel.test:  
  build:  
    context: ./vendor/laravel/sail/runtimes/8.0  
    dockerfile: Dockerfile  
  image: sail-8.0/app  
  ports:  
    - '${APP_PORT:-80}:80'  
  volumes:  
    - '.:/var/www/html'  
  depends_on:  
    - mysql
```

## 2. Laravel Sail

### b) Docker compose file



## 2. Laravel Sail

### b) Executing commands

Chạy toàn bộ container trong docker compose file

**./vendor/bin/sail up**

**docker-compose up**

```
λ ./vendor/bin/sail up
Unsupported operating system [MINGW64_NT-10.0-18363]. Lar
Creating network "testdocker_1_sail" with driver "bridge"
Creating testdocker_1_mysql_1 ... done
Creating testdocker_1_laravel.test_1 ... done
Attaching to testdocker_1_mysql_1, testdocker_1_laravel.t
mysql_1           | 2020-12-23 01:54:15+00:00 [Note] [Entry]
mysql_1           | 2020-12-23 01:54:24+00:00 [Note] [Entry]
mysql_1           | 2020-12-23 01:54:25+00:00 [Note] [Entry]
mysql_1           | 2020-12-23T01:54:26.357567Z 0 [System]
mysql_1           |
mysql_1           | 2020-12-23 01:54:34,033 CRIT Supervisor
specified in the config file. If you intend to run as root
laravel.test_1    | 2020-12-23 01:54:34,033 INFO Included ex
mysql_1           | 2020-12-23T01:54:34.057575Z 0 [Warning]
'./var/run/mysqld' in the path is accessible to all OS user
laravel.test_1    | 2020-12-23 01:54:34,130 INFO RPC interface
```



## 2. Laravel Sail

### b) Executing commands

down toàn bộ container trong docker compose file

**./vendor/bin/sail down**

**docker-compose down**

```
λ ./vendor/bin/sail down
Unsupported operating system [MINGW64_NT-10.0-18
Stopping testdocker_1_laravel.test_1 ... done
Stopping testdocker_1_mysql_1 ... done
Removing testdocker_1_laravel.test_1 ... done
Removing testdocker_1_mysql_1 ... done
Removing network testdocker_1_sail
```



## 2. Laravel Sail

### b) Executing commands

Để thuận tiện cho lần sau sử dụng ta alias:

```
alias sail='bash vendor/bin/sail'
```

## 2. Laravel Sail

### b) Executing commands

Ta có thể sử dụng các lệnh **PHP** trong container

```
sail php --version
```

```
docker-compose exec laravel.test php --version
```

```
λ sail php --version
Unsupported operating system [MINGW64_NT-10.0-18363]. Laravel Sail
PHP 8.0.0 (cli) (built: Nov 27 2020 12:26:22) ( NTS )
Copyright (c) The PHP Group
Zend Engine v4.0.0-dev, Copyright (c) Zend Technologies
with Zend OPcache v8.0.0, Copyright (c), by Zend Technologies
```

## 2. Laravel Sail

## b) Executing commands

Ta có thể sử dụng các lệnh **Composer** trong container

# **sail composer require laravel/jetstream**

```
docker-compose exec laravel.test composer require laravel/jetstream
```

```
λ sail composer require laravel/jetstream
Unsupported operating system [MINGW64_NT-10.0-18363]. Laravel Sa
Using version ^1.6 for laravel/jetstream
./composer.json has been updated
Running composer update laravel/jetstream
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
```



## 2. Laravel Sail

### b) Executing commands

Ta có thể sử dụng các lệnh **artisan** trong container

**sail artisan migrate**

**docker-compose exec laravel.test php artisan migrate**

```
λ sail artisan migrate
Unsupported operating system [MINGW64_NT-10.0-18363]. Laravel Sail supports macOS
Migration table created successfully.
Migrating: 2014_10_12_00000_create_users_table
Migrated: 2014_10_12_00000_create_users_table (1,878.88ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (1,630.64ms)
Migrating: 2014_10_12_200000_add_two_factor_columns_to_users_table
Migrated: 2014_10_12_200000_add_two_factor_columns_to_users_table (1,887.30ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (1,448.98ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (2,547.86ms)
Migrating: 2020_12_16_041902_create_sessions_table
Migrated: 2020_12_16_041902_create_sessions_table (3,904.64ms)
```



## 2. Laravel Sail

### b) Executing commands

Ta có thể sử dụng các lệnh **node** và **npm** trong container

```
sail node --version
```

```
sail npm run dev
```

```
λ sail node --version
Unsupported operating system
v15.4.0
```



## 2. Laravel Sail

### b) Executing commands

Ta có thể truy cập vào container bằng **shell**

**sail shell**

```
λ sail shell
Unsupported operating system [MINGW64_NT-10.0-
sail@6fb354fa388e:/var/www/html$ ls
README.md  composer.json  docker-compose.yml
app        composer.lock   node_modules
artisan    config          package-lock.json
bootstrap  database        package.json
sail@6fb354fa388e:/var/www/html$ |
```



# Nội dung

## 1. Json web token

- a) Cấu tạo
- b) Cách xác thực
- c) Ứng dụng vào Laravel Passport

## 2. Laravel passport

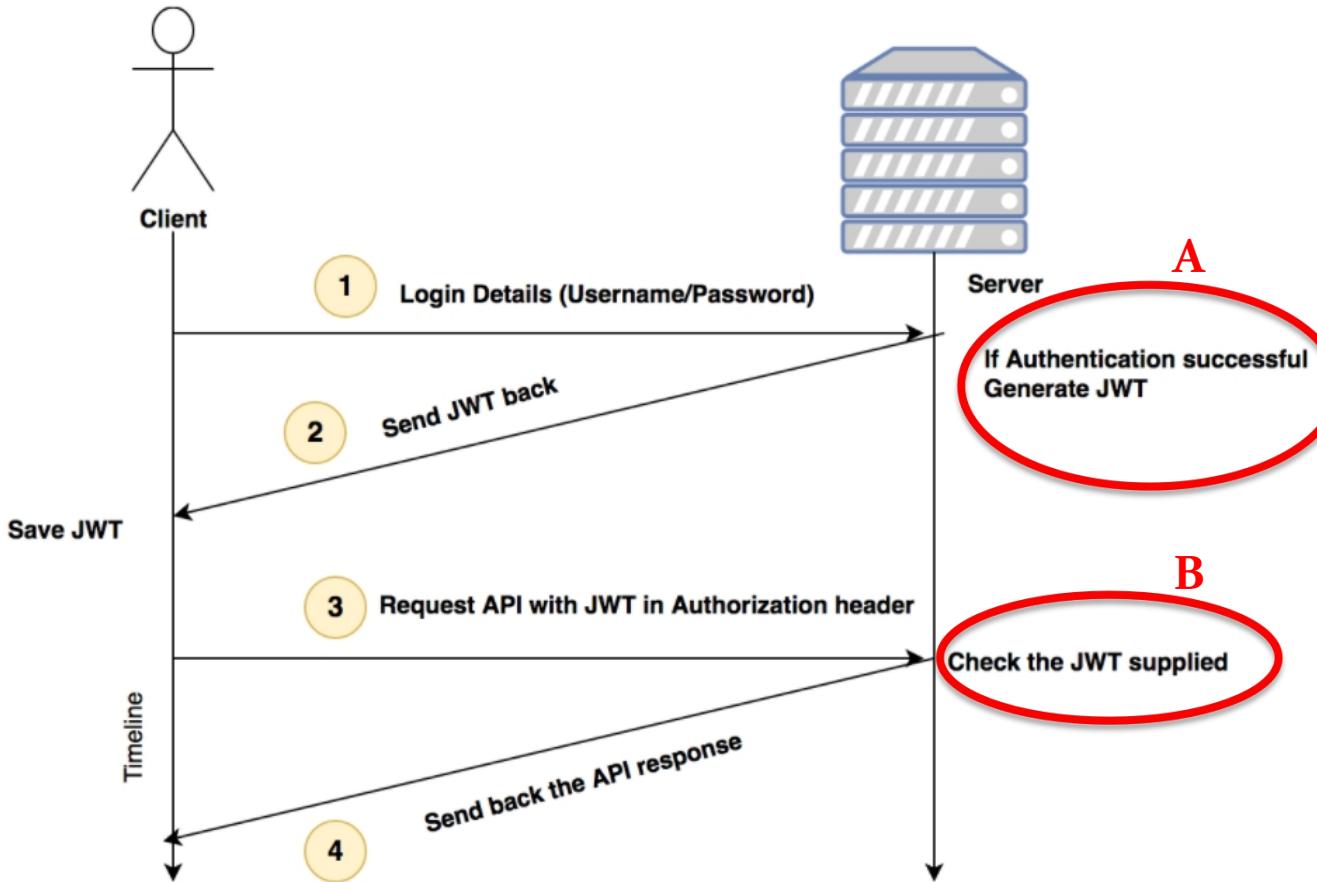
- a) Tổng quát
- b) Mô hình cơ sở dữ liệu
- c) Authorization code Grant
- d) Authorization code Grant with PKCE
- e) Password grant token
- f) Personal Access Token
- g) Refresh Token

## Mục tiêu

1. Ứng dụng được Laravel passport vào dự án
2. Hiểu rõ cơ chế hoạt động của Laravel passport
3. Ghi nhận thêm các đóng góp từ ACE Công ty
4. Chia sẻ, trao đổi, thảo luận trên tinh thần tích cực nhất



# 1. Json Web Token



Các Câu hỏi đặt ra:

- token được tạo ra như thế nào ??
- Nó có phải 1 chuỗi random không ??
- Làm sao check được token này ??
- Và token là của ai ??

Trong khi database không lưu lại chuỗi này



# 1. Json Web Token

## a) Cấu Tạo

eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJzdWliOilxMjM0NTY3ODkwliwibmFtZSI6IkpxvaG4gRG9IliwiaWF0ljoxNTE2MjM5MDlyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrz0ogtVhfEd2o

1

Header

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

2

Payload

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

3

Signature

```
HMACSHA256(  
  BASE64URL(header)  
  .  
  BASE64URL(payload),  
  secret)
```

JWT bao gồm 3 phần gồm header, payload, signature và được ngăn cách bằng dấu (.):



# 1. Json Web Token

## Header

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}
```

“typ” (type) chỉ ra rằng đối tượng là một JWT  
“alg” (algorithm) xác định thuật toán mã hóa cho chuỗi là HS256 ( Trong laravel passport dùng RS256)

## Payload

```
{  
  "user_name": "admin",  
  "user_id": "1513717410",  
  "authorities": "ADMIN_USER",  
  "jti": "474cb37f-2c9c-44e4-8f5c-1ea5e4cc4d18"  
}
```

Phần payload sẽ chứa các thông tin mình muốn đặt trong chuỗi Token như username , userId , token\_id ( trong laravel passport thì payload sẽ chứa dữ liệu bảng oauth\_access\_token )

## Signature

```
data = base64urlEncode( header ) + "." + base64urlEncode( payload )  
  
signature = Hash( data, secret );
```

- base64UrlEncoder : thuật toán mã hóa header và payload
- Sau đó dùng thuật toán HS256 ( trong laravel passport dùng RS256) kèm theo khoá bí mật ( secret) và chuỗi data ở trên ta được signature
- Cuối cùng kết hợp 3 chuỗi header + payload + signature ta được chuỗi JWT

## 1. Json Web Token

### b) Cách xác thực

Token thuộc người dùng nào ??

Server sẽ đọc dữ liệu trong payload để biết token này thuộc về ai

Vậy ai đó sửa đổi dữ liệu trong phần payload thì phải làm sao ?? ví dụ như user\_id=1 sửa thành user\_id=3 sửa như vậy thì có sao không ??

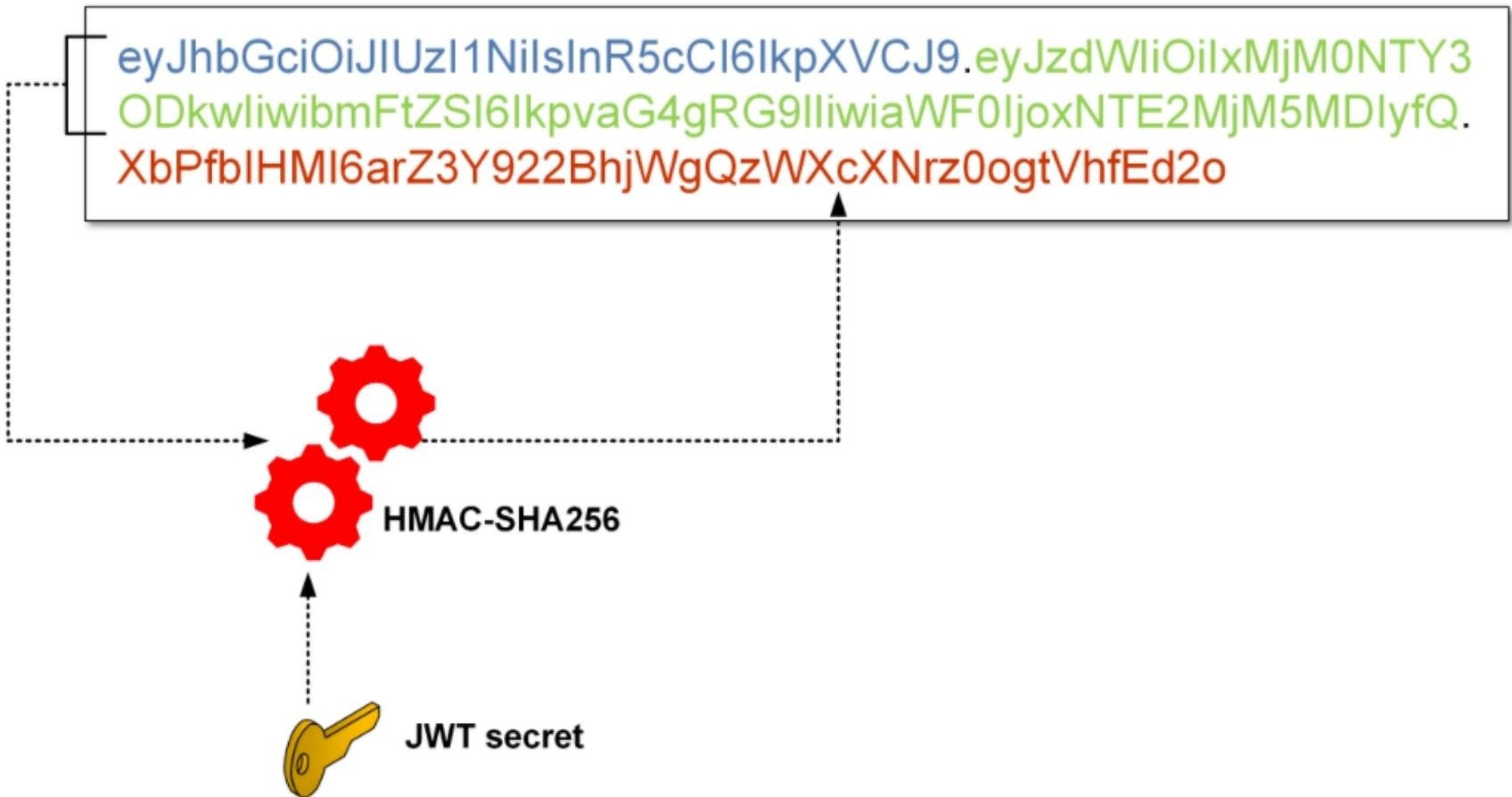
Server dùng mã secret được lưu trên server cùng với header và payload được lưu trong token người dùng gửi lên để hash tạo ra signature và kiểm tra signature người vừa gửi lên có trùng với signature server vừa tạo hay không

Nếu không trùng thì token đã bị ai chính sửa dữ liệu vì chỉ cần chỉnh dù chỉ 1 chút thì mã signature sinh ra đã khác so với nguyên bản

Tóm lại người dùng hoặc ai đó có thể có mã token và đọc dữ liệu trong phần payload nhưng không thể chỉnh sửa được vì nếu chỉnh sửa sẽ bị phát hiện



# 1. Json Web Token



## 1. Json Web Token

### c) Ứng dụng vào Laravel Passport

Laravel passport dùng 1 cặp khoá public key và private key ( dùng thuật toán RS256 để tạo token ). Private key chính là secret ( trong HS256 ).

Signature= hash (data , private key)

Còn public key dùng để xác minh token ( chỉ xác minh được **true** hoặc **false** chứ **không có chức năng tạo được token** )



# 1. Exception

```
oauth-private.key x oauth-public.key x
1 -----BEGIN RSA PRIVATE KEY-----
2 MIIJKAIKAAKCgEAvQePdvxLSLA+UQQixX0byF0WjoMhCDci6XjLRqs/eZKOL+mg
3 TmEWCGuOeFuAVjzzisjpGRufsvG/WtJPPMj9rl/9pXv9llI0QUE6Dl0DwRqkePVR
4 8kTjLMXz9CBpIqvATHzZEHgdHE/v4J0v4FX6kQkP5u45XhCihnsbaanpL7mAxt0
5 HatdSyNAaITiJW6tJG81mrGlmxsg1UXqAB1v004PsYFREIcaPfu3bAriy4NT+cwl
6 /406/2f63dKPkV8m1PHbYVuYR4ouoWVN18/b+MmzBAdwMaQNXm9YGZpvUx4Ymkh6
7 K50JZBWLCflkWfGyFdHB88Glg34NbVJWilaCMW6px0rQ5wemSqe3vHmZMmXFkQc
8 7wZ4z7ZNozj0ZTdaEADxHdsokwLSr2Aiip5ctT+tE+0QkMI0xEf30YzJ3sdTdIys
9 D38D72b1oq4IZx9CW4mBpnv7a8Y3cDEKY8Je1k60DuR7uKRwCcQEU+k15oRQggw+
10 omXw21f+AtXPzRrvYAJDyFrPLu3L4l8dr9mK5t/uKWxtm9pY9DoM9jVbwqySqXk
11 D+H78X5pwLCN3SFH1eDIoJ+20jLQB54rMSE4U45GwfzDFUwEfhsr8DSPTMi3pafm
12 x3EGnfKmh91x1MR0TNepnCF91l0M/Ypt4UD0w0mgYHKRrVIo05RpXKzyQpUCAwEA
13 AQKCAGaEx09d0WTwv7YV3PWIbvWLn1Sf6cwEGW4u08NNg263bdyHAqRefsJm6lDN
14 WiDAafZBv8rskgnjKie9MwdB33wm22xieJjm0//ia5a71xBIGRsDUdu19e+Dh8vc
15 ETbtQiIiSk5vwaY5b4MwvJfz6KyW8XNGrT1i57z13zi4le0wDwFGuj6i4hczt86x
16 9gkYinx1F8KDqK9NRQRW+vpuDuH5Zvl5Lpqze7wTIZxRFEitzEXVi5+ARK05eIXJ
17 fpJv2i1pe0P+qzpVU70aBf0x09/F+B1JHzpKEwg90JCEPpvb04U0P7jf4dca4Gj
18 znpSPXKJhmSDafspja/oq79BkSpzS0hL31bgjm/kckuEKKdcftTfTAFv2N7d2Jx
19 zXmE1nwg08VlneYHtT7I6oBH9YSB8uf1EsR+TsitrkVwU90TW8oggK5wTLA6hNnB
20 npk7wEgHe4vIZg7NvnR/0480+4XF3FWnQ8oxYGYW0wTF8nw5Q02mrVg6eDID0XJM
21 i5YDqgfEON08WjYX9Yafu/eIjZVJNdsCr1B2xNTUGxIM19KCu0uBs4ScccTavnVr
22 tNDryxtCl2zWMn4q3EMiVLx8By+2rMwHdHaSM4hU08D9b/orJLc22gC6ERgR5KpP
23 UmF06jX/pdqKV5zZ16Jfn4JmLA87MIX/iTLWFp0iz/Di/4UiIQKAQEA+V3mQRbh
24 zsauDa0y1lCb5KbR82pATmx7v2YkM9Q+I+3zH5ly4c6rz2Piksm83ao1IckSJ0U3Y
25 +ydGB/lzi9aMA0DAmprDuh/3P9d7gS1YgtEA4e8oMnuuinQg63pgtQyvK+wEs7
26 pBe6n7bnMckJAWY/8yLD8ftHuy0nppCM8X9Y97qhN07eST5L5CvH9/9Cvo4gERI
27 WoyGC70BNhni9nQuTQyYM4kM6DAvHGtDXhJTjg8WYqo0DLoeTBKX4ZyW9IuJIrv
28 BDvpCmCE7sPzdMYR7wmM110VRgtXJcokFH2i0CoSh2N20Ycnx5UcdHhPQqExwZ4
29 o3ep0DZXJC9kqQKCAQEawg7JDkvkTe6cl6tLHepu85gLp7kjL1v/79y1T0dDjAQ
30 hVS3d8NTDrssdScCaxSfwF1MXusEh+cre7pPcl2jvk2ZyJrq72viYzN08ReJCDpbN
31 vVC8TXQ8r+E5F+zWf0oetB33JRCn+/e2LrmWKPHZLaCVG6qWScsUkoBD4Mhn1k++
32 I7Lvzakw14M8dGA3S40esTPvAhYUWdIGMimHLi96kiNdnr1Ra/h7el5FEXLaB9Xo
```

```
oauth-private.key x oauth-public.key x
1 -----BEGIN PUBLIC KEY-----
2 MIICijANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvQePdvxLSLA+UQQixX0b
3 yF0WjoMhCDci6XjLRqs/eZKOL+mgTmEWCGuOeFuAVjzzisjpGRufsvG/WtJPPMj9
4 rl/9pXv9llI0QUE6Dl0DwRqkePVR8kTjLMXz9CBpIqvATHzZEHgdHE/v4J0v4FX
5 6kQkP5u45XhCihnsbaanpL7mAxt0HatdSyNAaITiJW6tJG81mrGlmxsg1UXqAB1v
6 004PsYFREIcaPfu3bAriy4NT+cwl/406/2f63dKPkV8m1PHbYVuYR4ouoWVN18/b
7 +MmzBAdwMaQNXm9YGZpvUx4Ymkh6K50JZBWLCflkWfGyFdHB88Glg34NbVJWila
8 CMW6px0rQ5wemSqe3vHmZMmXFkQc7wZ4z7ZNozj0ZTdaEADxHdsokwLSr2Aiip5c
9 tT+tE+0QkMI0xEf30YzJ3sdTdIysD38D72b1oq4IZx9CW4mBpnv7a8Y3cDEKY8je
10 1k60DuR7uKRwCcQEU+k15oRQggw+omXw21f+AtXPzRrvYAJDyFrPLu3L4l8dr9m
11 K5t/uKWxtm9pY9DoM9jVbwqySqXkD+H78X5pwLCN3SFH1eDIoJ+20jLQB54rMSE4
12 U45GwfzDFUwEfhsr8DSPTMi3pafmx3EGnfKmh91x1MR0TNepnCF91l0M/Ypt4UD0
13 w0mgYHKRrVIo05RpXKzyQpUCAwEAAQ==

-----END PUBLIC KEY-----
```

1 cặp public key, private key được lưu tại thư mục storage trong project laravel



# 1. Json Web Token

```
{  
  "token_type": "Bearer",  
  "expires_in": 31536000,  
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  
eyJhdWQ1OixIiwiianRpIjoiNDVmMzMzMGZmOTk3NmE4NzYwMGQ3M2EwYWMyMDF1ODk2MWUxMtg0MWJkOT10MWQzMjI1NmU20GE5OTIxMTNmODh1MzIxMTYzNjZjMWFiYTylCJpYXQ10jE2MDY5MjM3MTksIm5iZiI6MTYwNjk  
vMzcxOSwiZXhwIjoxNjM4NDU5NzE5LCJzdWIiOiIxIiwic2NvcGVzIjpBX0.  
d5MfaGFbA1dBEJ6Izlgjt4cYM5y3iSSUtqtjaGdd8eXm1phs5VTqSMwoKIpRSwV1Z3_1y8RetpGdkoYtPD130Py-Qm7Xr_c3MFI3L1h88SaYJciJ7dDA9W5vway5rPh3uq2nhajBc2UkpZyoByzjhNvqyWBCxRNFVpSG26AyGB  
VD1I-yuVW0jxfCd6TH3Se7uYoXYyTc7T_CgT7e2M1-oNfTG3S2Dgy1bUDKXApAqanCG4fZLgcJLCxGBEKK5JAckvabV9FG6dNx3cd1MKRwpGbweXu_ui4yfW9rt8tMDv4B96H6cZFHUR0jbE1EbPucp9c9NOAEAJ6PydLDooN7  
EXAsfeLE4Rj1X6L9rQ0JDxtL8-4aY1T1i1jUjk92SuGNCSw9wDr0IPrm07rsN0pbP62ec1zgYQjpP7vmQ0YERhht1lz_3gCSj8rdWhJbskCyBKS17bMwFv8a72KhL0ss8EDAumsmn7hzBU0NwEd0sns0YtY9uGGTGMz1PXJBsY  
OifmWGk_oXymtIRCwfPOi2VtU7w0uMq5DrbBcc4gFwEKRaSho3eH1Ud4X9JPuX0J0gJw9x9JRxFy4RM-b-A3XixK1WbvnKV1ME43tsp6xih0n9bJcLM7JgQQnewNByytozz4bf9uYNajxeoEOurqbjyyHkPbTkt0dtQY02P8"  
}  
,  
  "refresh_token":  
    "def5020052778f980178e910e27be1225c31f3c48bc6a603b9d72e0a33c94e043b320202f11b2f9a6ad35a8291d29800d87670abaa376efa3f9fc6550af65a2e232d982a56e24f89169a917228241942123d3f3f4c  
89049f01de3a630d01b6cff71b10c5e29ef60b1330597792b500067a0890a7ba41a87beb12eea5c60a9c6285924031cfaa23e9d17184e88a97199f3c18cfcae9da1b35d25690de20355bc741aab03c331d3cae0a7  
69656d84057336168af9c2d9603afcc6944789e412dae1662a03674775cc7e566b724ae829b1678a7e765e8ca5914276db03e016d81401b28a2603ffa2cea91164dd796ff1016ab0767e96f18e526369d48bf7e87b1  
fde746ac86f69085fec3d4f9ad351df027834ba1ed93f82d85fae2eaa173ba14e55f654d2ce822178ea6eb785b44f278248565a9cbc6479c6e148ebe9c9b6c8be915a9d0f67edb02ed574da9e3708079d672095ed  
857884562fdd95cf7606210"
```

Ví dụ kết quả trả token về thực tế của passport



# 1. Json Web Token

## Tạo 1 Token trên trang web jwt.io

The screenshot shows the jwt.io token creation interface. A red box highlights the token string on the left, which is annotated with "Token Nhận được" (Received Token). Another red box highlights the "PAYLOAD: DATA" section in the middle, annotated with "Nhập dữ liệu payload". A third red box highlights the "VERIFY SIGNATURE" section at the bottom, annotated with "Nhập private key".

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIxIiwianRpIjoiNDVmMzMzMGMZmO... ey  
JhdWQiOiIxIiwianRpIjoiNDVmMzMzMGMZmO...  
mE4NzYwMGQ3M2EwYWMyMDFlODk2MWUxMTg0MWJk  
OTI0MWQzMjI1NmU20GE5OTIxMTNmODh1MzIxMTY  
zNjZjMWFiYTUiLCJpYXQiOjE2MDY5MjM3MTksIm  
5iZiI6MTYwNjkyMzcxOSwiZXhwIjoxNjM4NDU5N  
zE5LCJzdWIiOiIxIiwic2NvcGVzIjpbXX0.d5Mf  
aGFbA1dBEJ6Izltjt4cYM5y3iSSUtqjtAoGdd8e  
Xm1phs5VTqSMwoKIpRSwV1Z3\_ly8RetpGdkoYtP  
D130Py-  
Qm7Xr\_c3tFI3l1h88SaYJciJ7DA9W5vway5rPh  
3uq2nhaJbc2Ukpzy0ByZjIINVqyJBCXrNFVpSG26  
AyGBVD1I  
yuVWOjxfCaSTH3Se7uYoXYvYtC7T\_CgT7e2M1-  
oNFTG3S2Dgy1bUDKXApAqanCG4fZLgcJLCxGBEK  
K5JAckvabV9FG6dNx3cd1MKRwpGbweXu\_ui4yfW  
9rt8tMdV4B96H6cZFHZUR0jbE1EbPucp9c9NOAEA  
J6PydLDooN7EXAsfeLE4Rj1X6L9rQ0JDxtL8-  
4aY1t1jUjk92SuGNCsw9wDr0IPrm07rsN0pbP6  
2ec1zgYQjpP7vmQ0YERhht1lz\_3gCsj8rdWhJbs  
kCyBKS17bMwFvF8a72KhL0ss8EDAumsmn7hzBU0  
NwEd0sns0YtY9uGGTGmZ1PXJBsY0ifmWGk\_oXy  
mtIRCwfP0i2VtU7wOuMq5DrbBcc4gFwEKRaSho3  
eH1Ud4X9JPuXOJ0gJw9x9JRxyu4RM-b-  
A3XixKlWVbnvKV1ME43tsp6xih0n9bJcLM7JgQQ  
wewNbbytozz4bf9UuYNajxeoOurqbjyyHkPbTk  
t0dtQY02P8

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "RS256"
}
```

Nhập dữ liệu header

PAYLOAD: DATA

```
{
  "aud": "1",
  "jti": "45f330ff9976a87600d73a0ac201e8961e11841bd9241d32256e68a992113f88e32116366c1aba6",
  "iat": 1606923719,
  "nbf": 1606923719,
  "exp": 1638459719,
  "sub": "1",
  "scopes": []
}
```

Nhập dữ liệu payload

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key or Certificate. Enter it in plain text only if you want to verify a token
)
```

Nhập private key

Thử xem token vừa tạo và token của passport ở trang trước có giống nhau hay không ??



# 1. Json Web Token

## Kiểm tra 1 token trên trang web jwt.io

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIxIiwianRpIjoiNDVmMzMzMGZmOTk3NmE4NzYwMGQ3M2EwYWMyMDF1ODk2MWUxMTg0MWJkOTI0MWQzMjI1NmU2OGE5OTIxMTNmODh1MzIxMTYzNjZjMWFiYTQyMjI1NjI0M3MTksIm5iZiI6MTYwNjkyMzcxOSwiZXhwIjoxNjM4NDU5NzE5LCJzdWIiOiIxIiwic2NvcGVzIjpbXX0.d5MfaGFbA1dBEJ6Izltgtj4cYM5y3iSStUqtjAoGdd8eXm1phs5VTqSMwoKIpRSwV1Z3\_1y8RtpGdkoYtPD130Py-Qm7Xr\_c3MFI3L1h88SaYJciJ7dDA9W5vwayrPh3uq2nhajBc2UkpZyoByzjhNvqyWBCXrNFVpSG26AyGBVD1I-yuVW0jxfCd6TH3Se7uYoXYyTc7T\_CgT7e2M1-oNfTG3S2Dgy1bUDKXApAqanCG4fZLgcJLCxGBEKK5JAckvabV9FG6dNx3cd1MKRwpGbweXu-ui4yFW9rt8tMDv4B96H6cZFHZUR0jbE1EbPucp9c9NOAEAJ6PydLDo0N7EXAsfeLE4Rj1X6L9rQ0JDxtL8-4aY1Ti1jUjk92SuGNCsw9wDr0IPrm07rsN0pbP62ec1zgYQjpP7vmQ0YERhht1lz\_3gCsj8rdWhJbskCyBKS17bMwFvF8a72KhL0ss8EDaumsmn7hzBU0NwEd0sns0YtY9uGGTGMz1PXJBsY0ifmWGk\_oXymtIRCwfP0i2VtU7w0uMq5DrbBcc4gFwEKRaSho3eH1Ud4X9JPuXOJ0gJw9x9JRxYu4RM-b-A3XixKLWbnvKV1ME43tsp6xih0n9bJcLM7JgQQwewNByytozz4bf9UuYNajxeoEOurqbjyyHkPbTk t0dtQY02P8

Subject (from the token refers to)

Nhập token

Signature Verified

Kết quả nhận được

SHARE JWT

HEADER: ALGORITHM & TOKEN TYPE

```
{ "typ": "JWT", "alg": "RS256" }
```

PAYOUT: DATA

```
{ "aud": "1", "jti": "45f3330ff9976a87600d73a0ac201e8961e11841bd9241d32256e68a992113f88e32116366c1aba6", "iat": 1606923719, "nbf": 1606923719, "exp": 1638459719, "sub": "", "scopes": [] }
```

VERIFY SIGNATURE

```
RSASHA256( base64UrlEncode(header) + "." + base64UrlEncode(payload))
```

0M/Ypt4Ude  
w0mgYHKrVt05RpXKzyQpUCAwE  
AAQ==  
-----END PUBLIC KEY-----

nhập public key

Private Key. Enter it in plain text only if you want to generate a new token. The key never leaves your browser.

)



# 1. Json Web Token

```
Route::get('uri: 'CheckToken', function(\Psr\Http\Message\ServerRequestInterface $request)
{
    // lấy access token được gửi kèm từ request
    $header = $request->getHeader(name: 'authorization');
    $jwt = \trim((string) \preg_replace(pattern: '/^(?:\s+)?Bearer\s/', replacement: '', $header[0]));
    //tách dữ liệu được nganh cách bằng(.) thành mảng,
    //ở đây mình sẽ thu được mảng 3 phần từ lần lượt header, payload và signature
    $jwt = explode(delimiter: '.', $jwt);
    //đọc khóa public key trong file
    $pubkeyid = openssl_pkey_get_public(certIFICATE: "file://C:/xampp/htdocs/server/storage/oauth-public.key");
    //data = header + payload
    $data=$jwt[0] . '.' . $jwt[1];
    $signature=$jwt[2];
    $signature = strtr($signature, '-_', '+/');
    // Decode Base64 string
    $signature= base64_decode($signature);
    //xác minh token trong đó OPENSSL_ALGO_SHA256 là thuật toán mình muốn đưa vào
    $ok = openssl_verify($data, $signature, $pubkeyid, signature_alg: OPENSSL_ALGO_SHA256);
    if($ok==true)
    {
        //lấy dữ liệu payload
        $payload =base64_decode($jwt[1]);
        $payload=json_decode($payload, assoc: true);
        //tra về user
        return User::find($payload['sub']);
    }
    else
        return false;
});
```

Tạo hàm xác thực token đơn giản

## 1. Json Web Token

Câu hỏi :

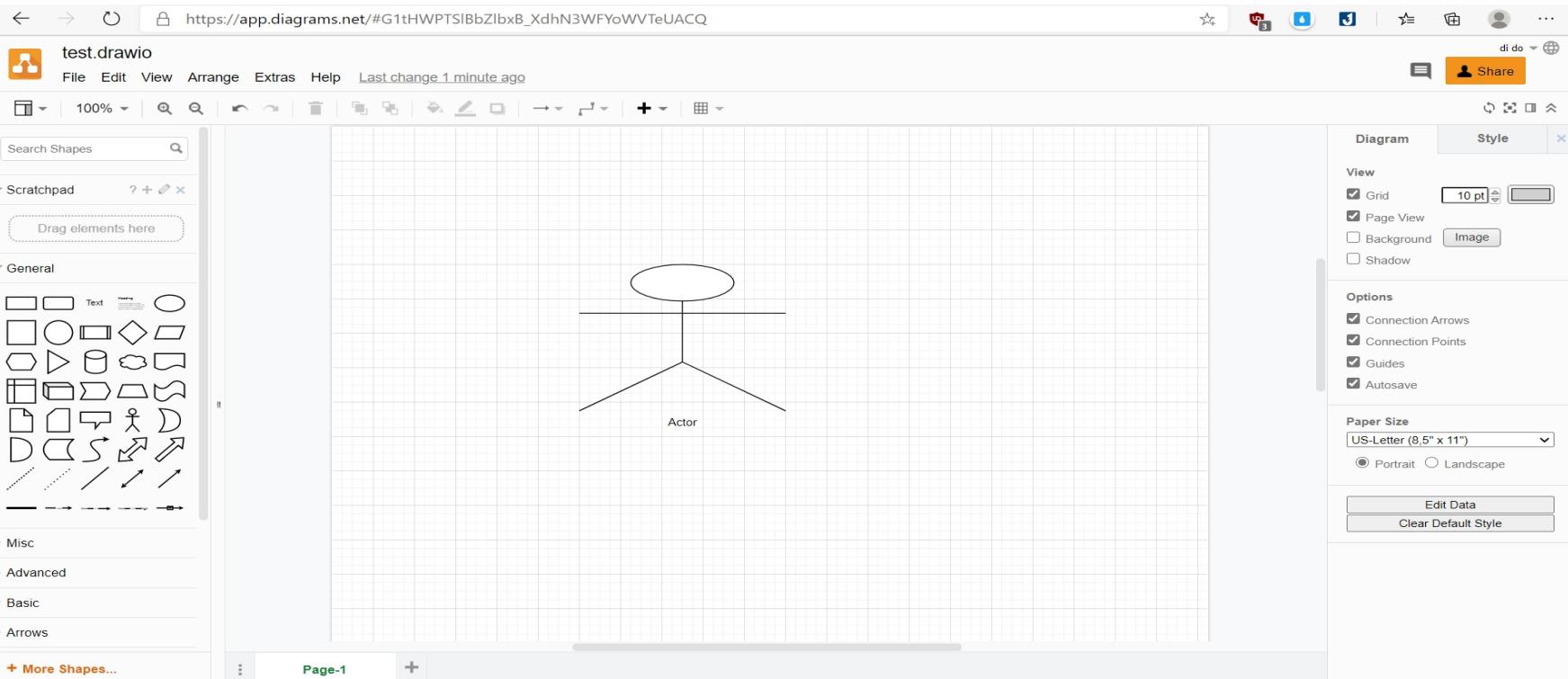
Tại sao Laravel passport lại dùng thuật toán mã hoá RS256 mà không dùng HS256 ?

Tại sao lại phải dùng 1 cặp khoá private key và public key mà không dùng 1 mã secret ?

Đáp án sẽ trả lời trong phần trình bày Laravel passport

# 1. Json Web Token

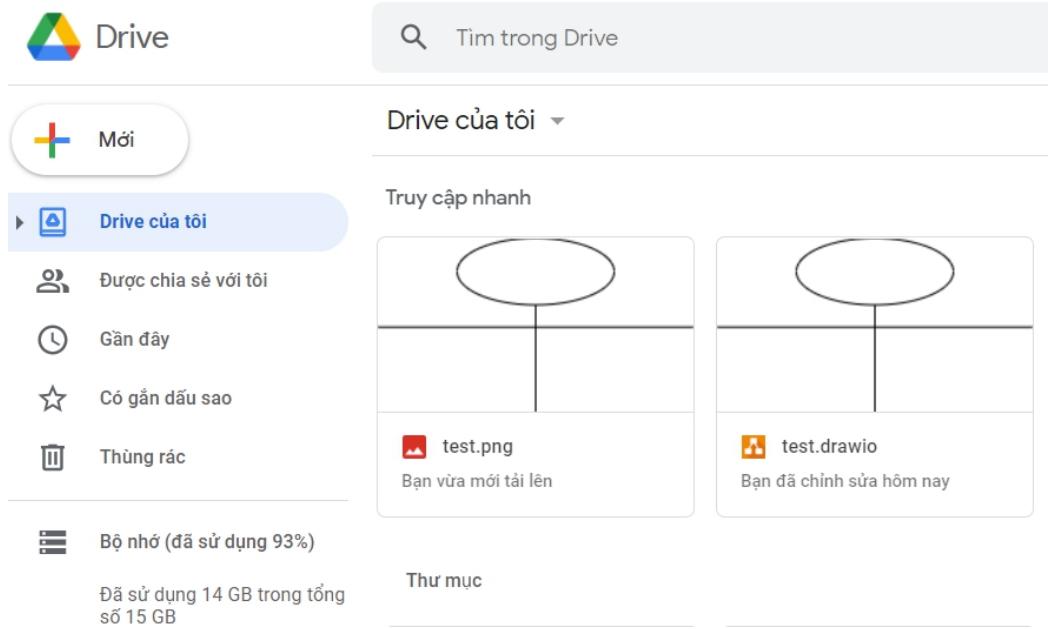
## a) Tổng quát



Ai cũng biết đây là trang web draw.io để vẽ lược đồ. Chú ý quan sát ở góc trên bên phải màn hình thì thấy tên của mình

Khi mình bấm nút save vào google drive thì lược đồ này sẽ được lưu vào google drive của mình

# 1. Json Web Token



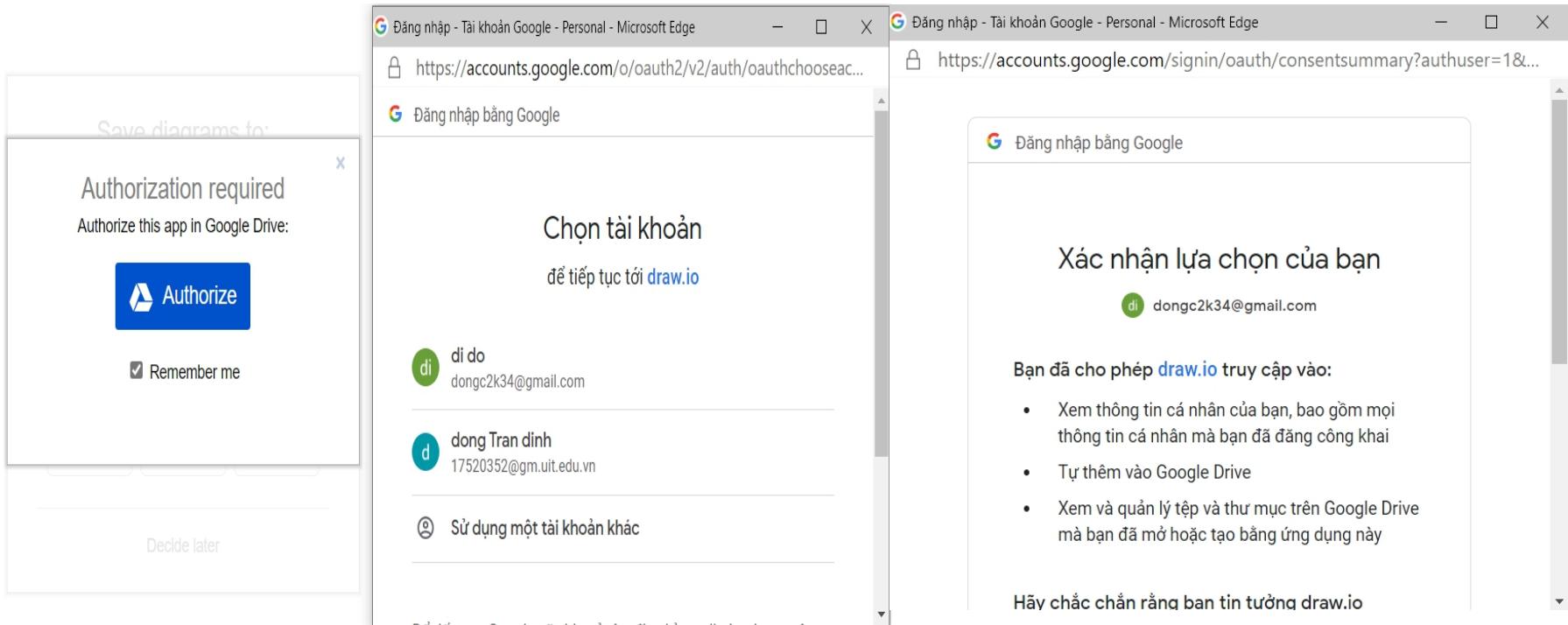
Vào google drive của mình thì thấy ảnh đã được lưu

Tại sao trang web này có thể truy cập vào google drive của để lưu ảnh ?

Tại sao nó thể biết tên mình, có thể truy cập thông tin của mình

Hãy nhớ lại lúc bắt đầu truy cập vào trang web

## 2 Laravel passport



Trước đó mình có bấm nút Authorize và đăng nhập bằng tài khoản google rồi bấm xác nhận cho phép draw.io truy cập vào dữ liệu của mình

## 2 Laravel Passport

Cứ tưởng **Laravel passport** là **google**, nó cung cấp 1 cơ chế cho bên thứ 3 ( như trang web draw.io, 1 trang web bất kỳ, 1 cái app di động nào đó, ứng dụng desktop hoặc các thiết bị trong nhà và đã được uỷ quyền của người dùng) truy cập đến tài nguyên của người dùng

**Laravel passport** được viết dựa trên **oauth2.0** ( giao thức tiêu chuẩn để uỷ quyền )

4 đối tượng quan trọng cần phải nắm rõ :

**Resource owner** : Chính là người dùng ( **user** )

**Client** : Trang web, 1 app di động, ứng dụng desktop bên thứ 3

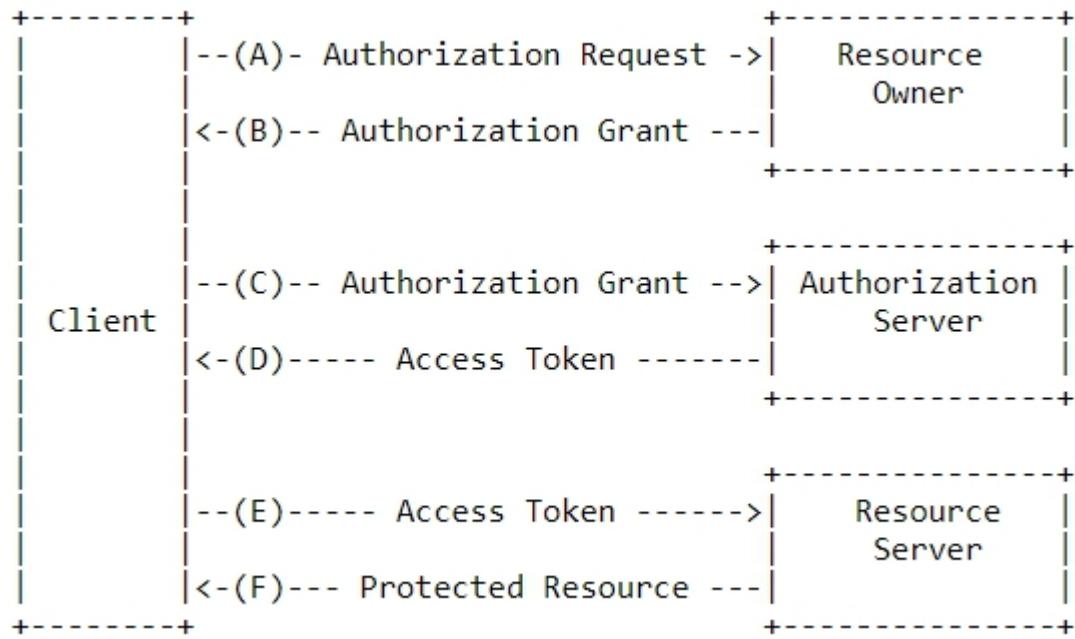
**Resource server** : nơi lưu tài nguyên, có chức năng tiếp nhận và xử lý các yêu cầu truy cập access token ( để dễ hiểu thì cứ tượng nó là **web api**, gửi token lên thì nó trả về dữ liệu )

**Authorizaton server** : Có chức năng **uỷ quyền** , tạo **access token**, tạo **refresh token** và **xác thực** refresh token ( **không có chức năng xác thực token** )



## 2 Laravel passport

### Cách thức hoạt động chung

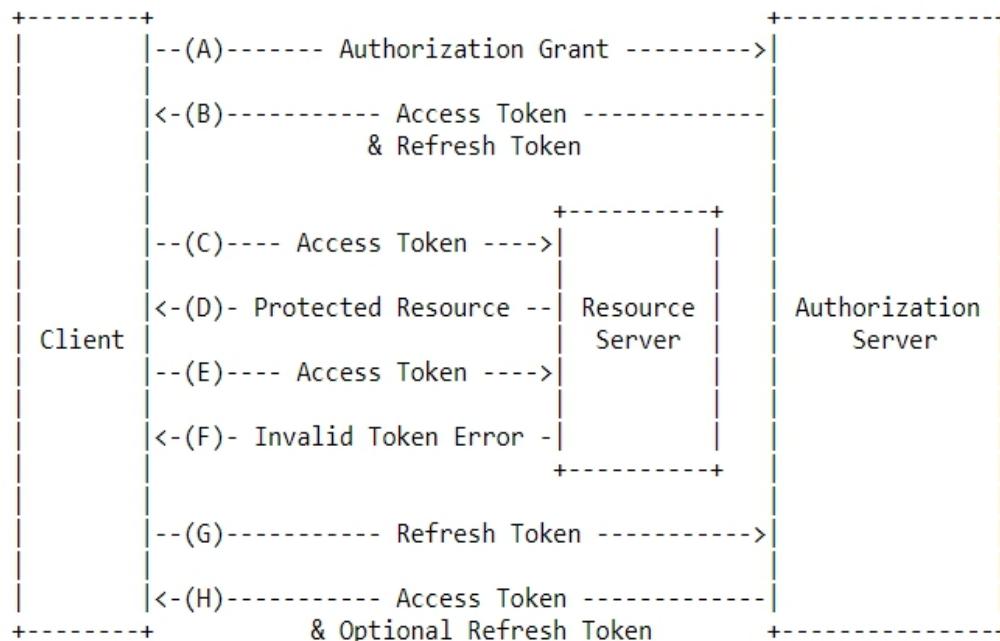


- (A) Client yêu cầu uỷ quyền từ người dùng (resource owner)
- (B) Người dùng cấp quyền cho client
- (C) Từ quyền người vừa cấp, client gửi lên authorization server yêu cầu cấp token
- (D) authorization server trả về token cho client (**có thể kèm refresh token**)
- (E) Gửi access token tới resource server(web api) yêu cầu tài nguyên
- (F) Trả về tài nguyên( dữ liệu )



## 2. Laravel Passport

### Refresh token

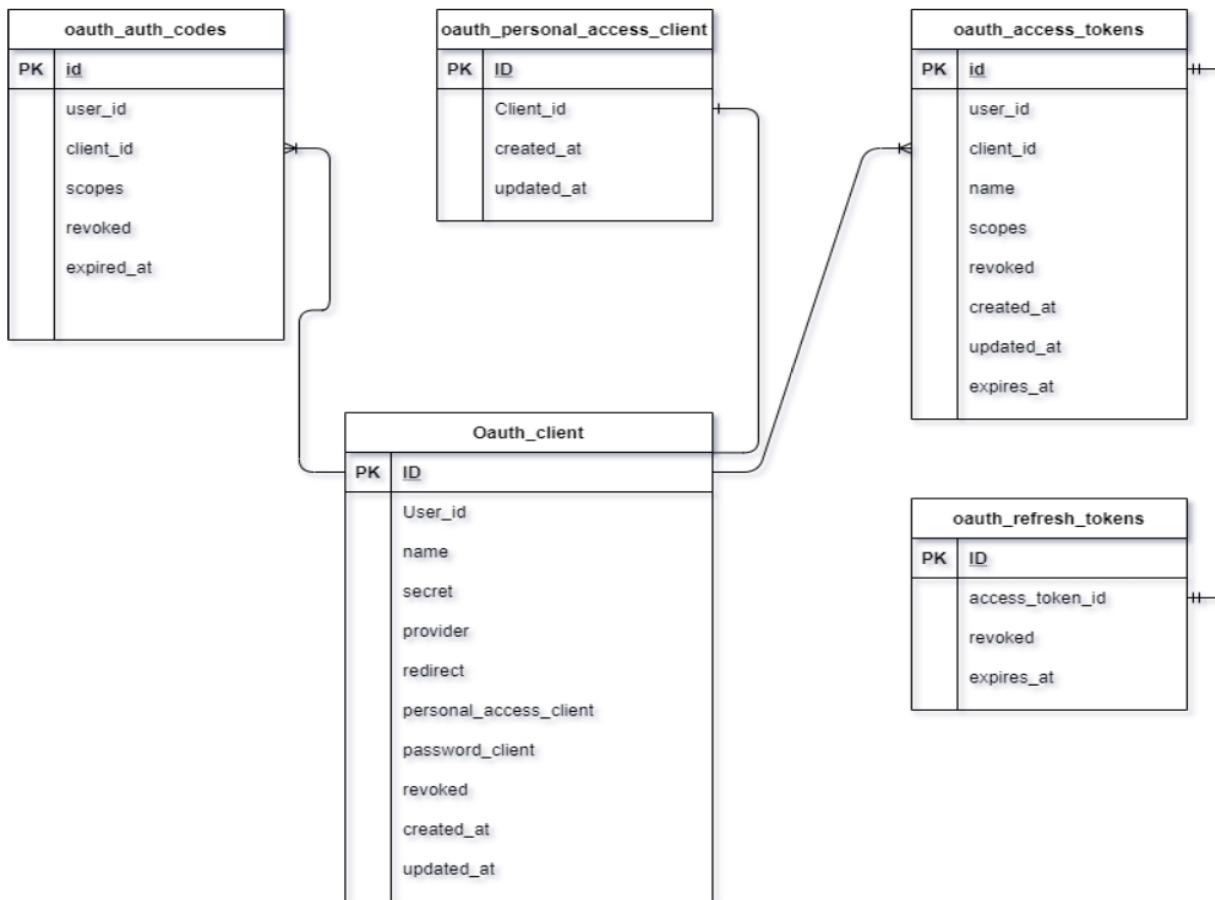


- (A) Từ quyền người cấp, client gửi lên authorization server yêu cầu cấp token
- (B) authorization server trả về access token + refresh token
- (C) Client gửi token lên yêu cầu dữ liệu
- (D) Resource owner( web api ) trả về dữ liệu
- (E) Giống bước C
- (F) Nếu token hết hạn, Resource owner(web api) thông báo lỗi
- (G) Client gửi refresh token lên yêu cầu cấp lại access token và refresh token



## 2. Laravel passport

### b) Mô hình cơ sở dữ liệu



Oauth\_client : lưu thông tin client

Oauth\_auth\_codes : lưu thông tin mã uỷ quyền

Oauth\_access\_tokens : lưu thông tin token

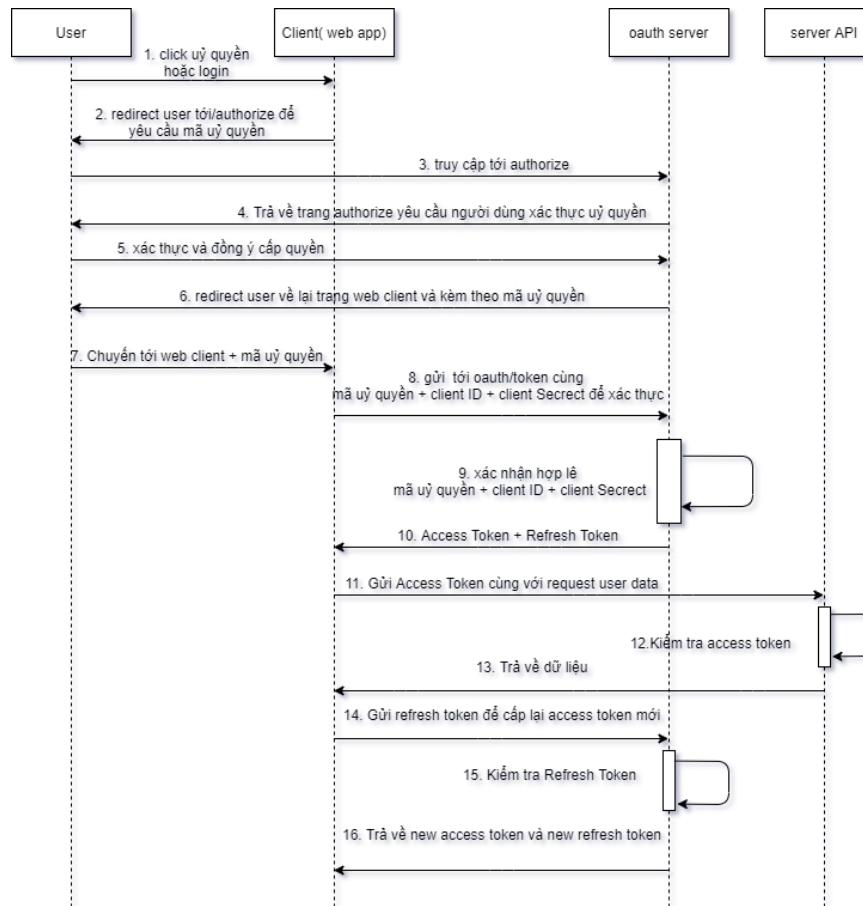
Oauth\_refresh\_token : lưu thông tin refresh token

Oauth\_personal\_access\_client :  
lưu thông tin personal\_client



## 2. Laravel Passport

### c) Authorization code grant



Laravel passport đã viết giúp mình tất cả thao tác trên oauth server và server api.

Ở bước 8 client phải gửi client id và **client secret** để xác thực client

Tokens, Refresh token (AES 256 CBC), Mã uỷ quyền đều đã được mã hoá



## 2. Laravel Passport

**Code demo ( chỉ code demo trên client, không demo trên server chưa passport vì passport viết sẵn cho mình hết rồi )**

```
public function redirect()
{
    $queries = http_build_query([
        'client_id' => 1,
        'redirect_uri' => 'http://localhost/client/public/oauth/callback',
        'response_type' => 'code',
        'scope' => '*'
    ]);
    return redirect( to: 'http://127.0.0.1:8000/oauth/authorize?' . $queries);
}
```

```
Route::get('/redirect', function (Request $request) {
    $request->session()->put('state', $state = Str::random(40));

    $query = http_build_query([
        'client_id' => 'client-id',
        'redirect_uri' => 'http://example.com/callback',
        'response_type' => 'code',
        'scope' => '',
        'state' => $state,
    ]);

    return redirect('http://your-app.com/oauth/authorize?'.$query);
});
```

Tạo 1 hàm xử lý ở client khi người dùng bấm vào login hoặc ủy quyền



## 2. Laravel passport

```
public function callback(Request $request)
{
    $response = Http::post( url: 'http://127.0.0.1:8000/oauth/token', [
        'grant_type' => 'authorization_code',
        'client_id' => 4,
        'client_secret' => 'Fj6QtajfjNFssMmodR40BpLpp2PN3QhKpp8DAuJW',
        'redirect_uri' => 'http://localhost/client/public/oauth/callback',
        'code' => $request->code
    ]);
    $response = $response->json();

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

    $request->user()->token()->create([
        'access_token' => $response['access_token'],
        'expires_in' => $response['expires_in'],
        'refresh_token' => $response['refresh_token']
    ]);
    return redirect( to: '/home');
}

Route::get('/callback', function (Request $request) {
    $state = $request->session()->pull('state');

    throw_unless(
        strlen($state) > 0 && $state === $request->state,
        InvalidArgumentException::class
    );

    $http = new GuzzleHttp\Client;

    $response = $http->post('http://your-app.com/oauth/token', [
        'form_params' => [
            'grant_type' => 'authorization_code',
            'client_id' => 'client-id',
            'client_secret' => 'client-secret',
            'redirect_uri' => 'http://example.com/callback',
            'code' => $request->code,
        ],
    ]);

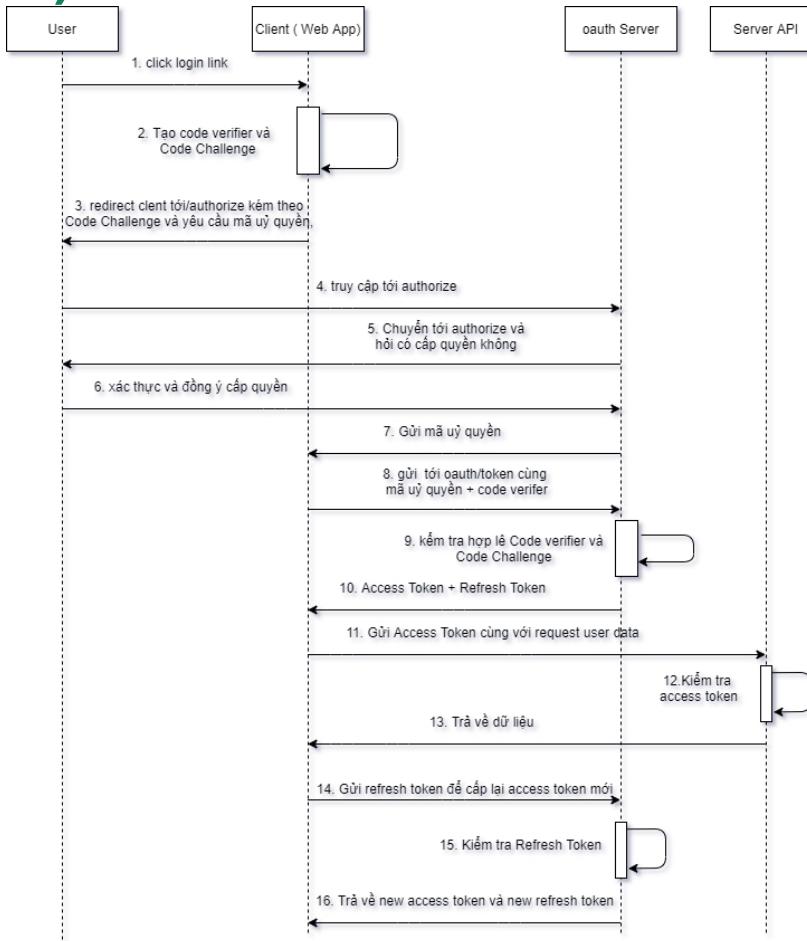
    return json_decode((string) $response->getBody(), true);
});
```

Hàm callback nơi client nhận mã uỷ quyền thông qua request  
Sau đó gọi lên oauth server để nhận token và refresh token  
Bây giờ đã có token thì chỉ cần gửi token lên web api và nhận về dữ liệu



## 2. Laravel Passport

### d) Authorization Code Grant with PKCE



Loại này dành cho mobile app hoặc web app (nơi không thể lưu trữ mã secret an toàn, còn gọi cách khác là public client)

Điểm khác so với lược đồ authorization code grant là ở bước 2,3,8,9

Bước 2 tạo code verifier random sau đó dùng thuật toán HSA256 để hash code verifier và sẽ nhận được code challenge

Bước 3 sẽ gửi code challenge vừa tạo đi và tên thuật toán

Bước 8 sau khi nhận được mã ủy quyền, gửi code verifier + mã ủy quyền cho oauth server

Bước 9 oauth server sẽ lấy code verified hash với thuật toán đã gửi lên trước đó (HSA256) để tạo ra code challenge. Nếu trùng thì xác thực thành công



## 2. Laravel Passport

Code demo ( chỉ code demo trên client, không demo trên server chưa passport vì passport viết sẵn cho mình hết rồi )

```
public function redirect2(Request $request)
{
    $request->session()->put('state', $state = Str::random( length: 40));

    $request->session()->put('code_verifier', $code_verifier = Str::random( length: 128));

    $codeChallenge = strtr(rtrim(
        base64_encode(hash( algo: 'sha256', $code_verifier, raw_output: true))
        , charlist: '='), '+/!', '-_');

    $query = http_build_query([
        'client_id' => '3',
        'redirect_uri' => 'http://localhost/client/public/oauth/callback2',
        'response_type' => 'code',
        'scope' => '*',
        'state' => $state,
        'code_challenge' => $codeChallenge,
        'code_challenge_method' => 'S256',
    ]);
    return redirect( to: 'http://127.0.0.1:8000/oauth/authorize?' . $query);
}
```

```
Route::get('/redirect', function (Request $request) {
    $request->session()->put('state', $state = Str::random(40));

    $request->session()->put('code_verifier', $code_verifier = Str::random(128));

    $codeChallenge = strtr(rtrim(
        base64_encode(hash('sha256', $code_verifier, true))
        , '='), '+/!', '-_');

    $query = http_build_query([
        'client_id' => 'client-id',
        'redirect_uri' => 'http://example.com/callback',
        'response_type' => 'code',
        'scope' => '',
        'state' => $state,
        'code_challenge' => $codeChallenge,
        'code_challenge_method' => 'S256',
    ]);

    return redirect('http://your-app.com/oauth/authorize?'. $query);
});
```

Tạo 1 hàm xử lý ở client khi người dùng bấm vào login hoặc uỷ quyền



## 2. Laravel Passport

```
public function callback2(Request $request)
{
    $state = $request->session()->pull('state');

    $codeVerifier = $request->session()->pull('code_verifier');
    throw_unless(
        condition: strlen($state) > 0 && $state === $request->state,
        exception: InvalidArgumentException::class
    );

    $response = (new \GuzzleHttp\Client())->post( uri: 'http://127.0.0.1:8000/oauth/token', [
        'form_params' => [
            'grant_type' => 'authorization_code',
            'client_id' => '3',
            'redirect_uri' => 'http://localhost/client/public/oauth/callback2',
            'code_verifier' => $codeVerifier,
            'code' => $request->code,
        ],
    ]);

    $response = json_decode((string) $response->getBody(), assoc: true);
    $request->user()->token()->delete();

    $request->user()->token()->create([
        'access_token' => $response['access_token'],
        'expires_in' => $response['expires_in'],
        'refresh_token' => $response['refresh_token']
    ]);
    return redirect( to: '/home');
}
```

```
Route::get('/callback', function (Request $request) {
    $state = $request->session()->pull('state');

    $codeVerifier = $request->session()->pull('code_verifier');

    throw_unless(
        condition: strlen($state) > 0 && $state === $request->state,
        exception: InvalidArgumentException::class
    );

    $response = (new \GuzzleHttp\Client)->post('http://your-app.com/oauth/token',
        'form_params' => [
            'grant_type' => 'authorization_code',
            'client_id' => 'client-id',
            'redirect_uri' => 'http://example.com/callback',
            'code_verifier' => $codeVerifier,
            'code' => $request->code,
        ],
    ]);

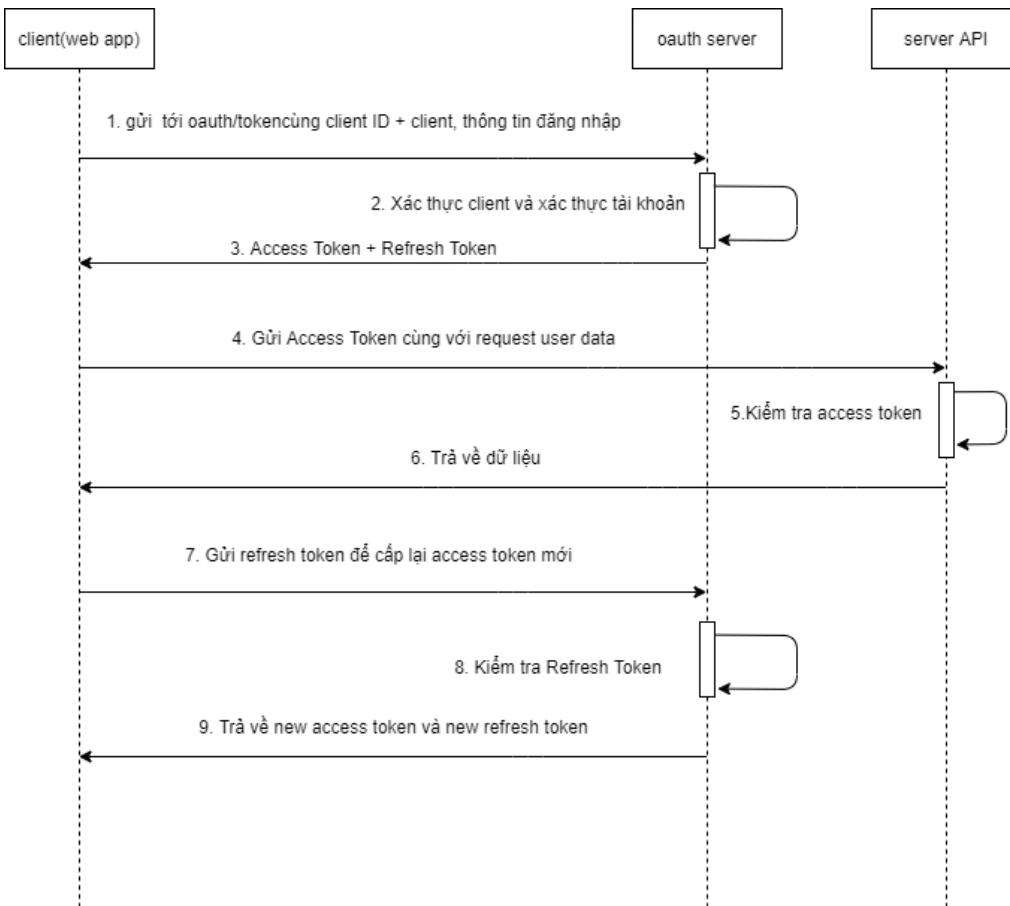
    return json_decode((string) $response->getBody(), true);
});
```

Hàm callback nơi client nhận mã uỷ quyền thông qua request  
Sau đó gọi lên oauth server để nhận token và refresh token  
Bây giờ đã có token thì chỉ cần gửi token lên web api và nhận về dữ liệu



## 2. Laravel Passport

### e) Password Grant Client



Ở loại này thì user  
cho client tài khoản  
và mật khẩu của  
user

Client sẽ gửi tài  
khoản, mật khẩu  
của user lên nhận lại  
token và refresh  
token



## 2. Laravel Passport

### Demo trên postman

The screenshot shows a POST request to `http://127.0.0.1:8000/oauth/token`. The 'Body' tab is selected, showing form-data parameters:

KEY	VALUE	DESCRIPTION
grant_type	password	
client_id	2	
client_secret	ULu59nftLb7F8eCQdLsRXshRSR0rWtEvvmoj8mmn	
username	dongc2k34@gmail.com	
password	dinhdong	
scope	*	
Key	Value	Description

The response body is displayed in JSON format:

```
1 {  
2   "token_type": "Bearer",  
3   "expires_in": 31535999,  
4   "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIyIiwianRpIjoiNmQ2Y2Z1Y2R1ZGVnN2R1ZDBjYjE2Yzc2YWMyNDA0ZTFhOTFhMjJmOGI0ODYyMmV6refresh_token": "def50200f2919811f63153482d3176414c0e1320354346d0d13a0a3e3ead64a98406a16e41d3b849d6dd5dfbe804095823772638806fab75a887ec581cb5"  
6 }
```



## 2. Laravel Passport

### Cách demo bằng code

```
$http = new GuzzleHttp\Client;

$response = $http->post('http://your-app.com/oauth/token', [
    'form_params' => [
        'grant_type' => 'password',
        'client_id' => 'client-id',
        'client_secret' => 'client-secret',
        'username' => 'taylor@laravel.com',
        'password' => 'my-password',
        'scope' => '',
    ],
]);

return json_decode((string) $response->getBody(), true);
```



## 2. Laravel Passport

### f) Personal Access Tokens

Ở 3 loại phía thì thấy chỉ có truy cập cho bên thứ 3 ( không phải user )

Laravel passport cung cấp 1 chế tạo ra personal token cho phép người có thể truy cập bằng access token với chỉ 3 câu lệnh

```
$user = App\Models\User::find(1);

// Creating a token without scopes...
$token = $user->createToken('Token Name')->accessToken;

// Creating a token with scopes...
$token = $user->createToken('My Token', ['place-orders'])->accessToken;
```



## 2. Laravel Passport

Tạo 1 chức năng đăng nhập, nếu thành công trả về access token

```
public function testLogin(Request $request)
{
    $request->validate([
        'email' => ['required', 'email'],
        'password' => ['required']
    ]);
    $user = User::where('email', $request->email)->first();
    if (!$user || !Hash::check($request->password, $user->password)) {
        throw ValidationException::withMessages([
            'email' => ['The provided credentials are incorrect']
        ]);
    }

    return $user->createToken('Auth Token')->accessToken;
}
```

## 2. Laravel Passport

### g) Refresh token

Khi token hết hạn hoặc mất thì việc tạo lại token mới là rất cần thiết

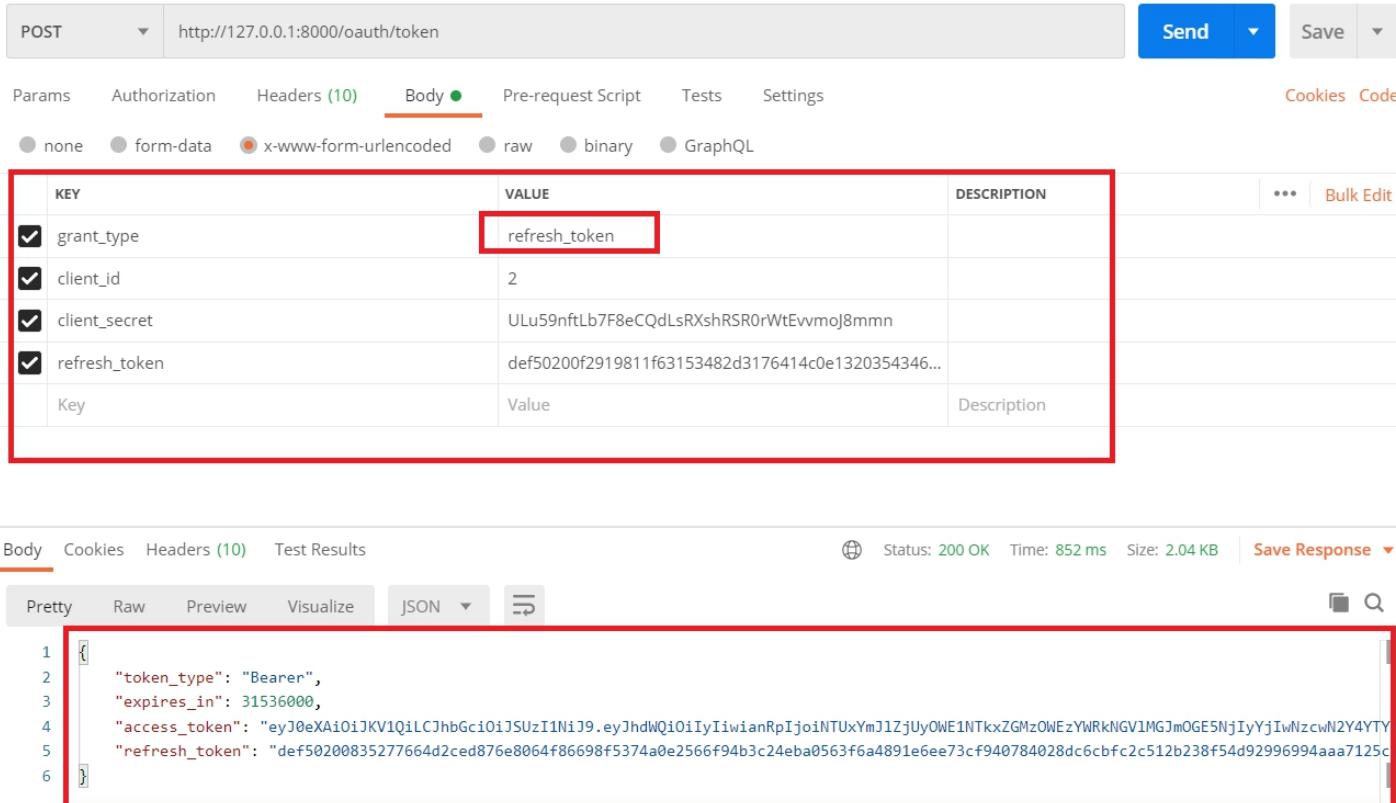
Laravel Passport tạo lại token và refresh token bằng cách gửi refresh token cũ lên

```
$http = new GuzzleHttp\Client;

$response = $http->post('http://your-app.com/oauth/token', [
    'form_params' => [
        'grant_type' => 'refresh_token',
        'refresh_token' => 'the-refresh-token',
        'client_id' => 'client-id',
        'client_secret' => 'client-secret',
        'scope' => '',
    ],
]);
return json_decode((string) $response->getBody(), true);
```

## 2. Laravel Passport

### Demo trên postman



The screenshot shows a POST request to `http://127.0.0.1:8000/oauth/token`. The 'Body' tab is selected, showing form-data parameters:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> grant_type	refresh_token	
<input checked="" type="checkbox"/> client_id	2	
<input checked="" type="checkbox"/> client_secret	ULu59nftLb7F8eCQdLsRXshRSR0rWtEvvmoj8mmn	
<input checked="" type="checkbox"/> refresh_token	def50200f2919811f63153482d3176414c0e1320354346...	
Key	Value	Description

The response body is displayed in JSON format:

```

1 {
2   "token_type": "Bearer",
3   "expires_in": 31536000,
4   "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIyIiwianRpIjoiNTUxYmJ1ZjUyOWE1NTkxZGMzOWEzYWRkNGV1MGJmOGE5NjIyYjIwNzcwN2Y4YTYz",
5   "refresh_token": "def50200835277664d2ced876e8064f86698f5374a0e2566f94b3c24eba0563f6a4891e6ee73cf940784028dc6cbfc2c512b238f54d92996994aaa7125c"
6 }

```