MỤC LỤC

[LỜI GIỚI THIỆU 2](#_Toc514369472)

[CHƯƠNG 1: RESTful API trong Nodejs 3](#_Toc514369473)

[CHƯƠNG 2: SWAGGER 4](#_Toc514369474)

[1. Swagger là gì? 4](#_Toc514369475)

[2. Hướng dẫn cài đặt swagger trong Nodejs 4](#_Toc514369476)

[CHƯƠNG 3: ĐẶC TẢ YÊU CẦU 10](#_Toc514369477)

[1. Mô tả tính năng 10](#_Toc514369478)

[2. Giải thuật thu thập (crawl) 10](#_Toc514369479)

[3. Các API đã viết 11](#_Toc514369480)

[4. Script thu thập 12](#_Toc514369481)

[5. Ưu nhược điểm của giải thuật 14](#_Toc514369482)

[**CHƯƠNG 4: VAGRANT** 15](#_Toc514369483)

[CHƯƠNG 5: BÁO CÁO CÔNG VIỆC 25](#_Toc514369484)

[GÓP Ý MÔN HỌC 31](#_Toc514369485)

[ acbxyz 31](#_Toc514369486)

# LỜI GIỚI THIỆU

Như ở Chuyên đề Web 1 đã giới thiệu về Nodejs, thì sang Chuyên đề Web 2 nhóm sẽ dựa trên nền tảng Nodejs để viết API bằng Framework Swagger .

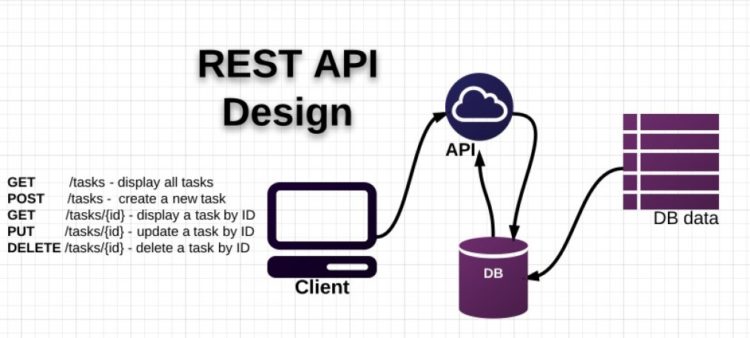
Hiện nay có rất nhiều ngôn ngữ lập trình hoặc framework hỗ trợ cho việc thiết kế API cho ứng dụng web nhưng theo việc sử dụng và hỗ trợ mạnh cho việc thiết kế API thì framework Swagger gần như là sự lựa chọn của đa số các lập trình viên.

Chúng em xin chân thành cảm ơn thầy Phan Thanh Nhuần đã hướng dẫn chúng em hoàn thành cả hai chuyên đề phát triển web trong kì này.

# CHƯƠNG 1: RESTful API trong Nodejs

REST là viết tắt của Representational State Tranfer. REST là một chuẩn web dựa vào các kiến trúc cơ bản sử dụng giao thức HTTP. Nó xử lí tài nguyên, nơi mà mỗi thành phần là một tài nguyên và nguồn tài nguyên này có thể được truy cập qua các giao diện chung bởi sử dụng các phương thức HTTP chuẩn. REST lần đầu tiên được giới thiệu bởi Roy Fielding năm 2000.

Về cơ bản, một REST Server cung cấp các chế độ truy cập đến nguồn tài nguyên và REST Client truy cập và sửa đổi các nguồn tài nguyên này bởi sử dụng phương thức HTTP. Ở đây mỗi nguồn tài nguyên được xác định bởi một URI. REST sử dụng các cách biểu diễn khác nhau để biểu diễn các nguồn tài nguyên như text, JSON, XML nhưng phổ biến nhất vẫn là JSON.



**Phương thức HTTP được sử dụng trong REST**

Dưới đây là các phương thức HTTP được sử dụng rộng rãi trong kiến trúc REST

* **GET** - Được sử dụng chỉ để đọc các nguồn tài nguyên.
* **PUT** - Được sử đụng để tạo mới các nguồn tài nguyên.
* **DELETE** - Được sử dụng để xóa các nguồn tài nguyên.
* **POST** - Được sử dụng để cập nhật các bản ghi hiện tại và tạo mới nguồn tài nguyên.

# CHƯƠNG 2: SWAGGER

## Swagger là gì?

[Swagger](http://swagger.io/) là một công cụ open source khá đơn giản nhưng lại vô cùng mạnh mẽ. Swagger cung cấp 3 tools chính cho các developers :

* [Swagger-Editor](http://swagger.io/swagger-editor/) : dùng để design lên các APIs hoàn toàn mới hoặc edit lại các APIs có sẵn thông qua 1 file config (\*.yaml)
* [Swagger-Codegen](https://github.com/swagger-api/swagger-codegen) : dùng để generate ra code từ các file config có sẵn
* [Swagger-UI](https://github.com/swagger-api/swagger-ui) : dùng để generate ra file html,css,… từ 1 file config

Do cung cấp các tool khá đa dạng nên việc viết document bằng swagger cũng có 2 cách tiếp cận:

* Top-down approach: Nghĩa là chúng ta sẽ design lên các api trước khi viết code
* Bottom-up approach: Nghĩa là từ các api có sẵn sẽ generate ra 1 file config
  1. Hướng dẫn cài đặt swagger trong Nodejs

**Step 1:**

Để sử dụng được swagger, trước tiên ta cần cài đặt modun Swagger vào. Trong Command line/Terminal gõ dòng lệnh sau:

***npm install -g swagger***

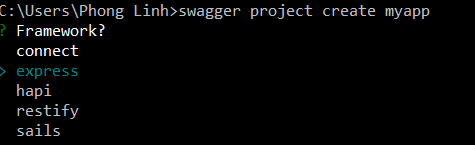
**Step 2:**

Sau khi đã cài thành công swagger vào máy, ta bắt đầu tạo project cho dự án với dòng lệnh sau:

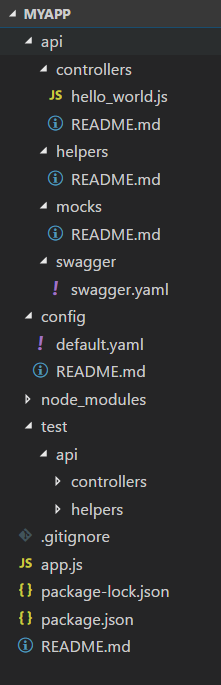
***swagger project create myapp***

Với: “myapp” là tên project

Sau khi chạy dòng lệnh trên, bên dưới sẽ hiển thị tên framework cần sử dụng, hãy chọn framework express bên dưới đây:



Sau khi tạo thành công, chúng ta hãy nhìn vào thư mục dự án:



**Step 3:**

* Mở hộp thoại command line, chỉ đường dẫn tới thư mục **myapp** được tạo bước trên. Ví dụ như mình lưu project theo đường dẫn: **E:\NodeJS\myapp**
* Bây giờ, để chạy dự án myapp, hãy mở command line hoặc terminal bên trong project **myapp,**  chạy dòng lệnh bên dưới:

***swagger project start***

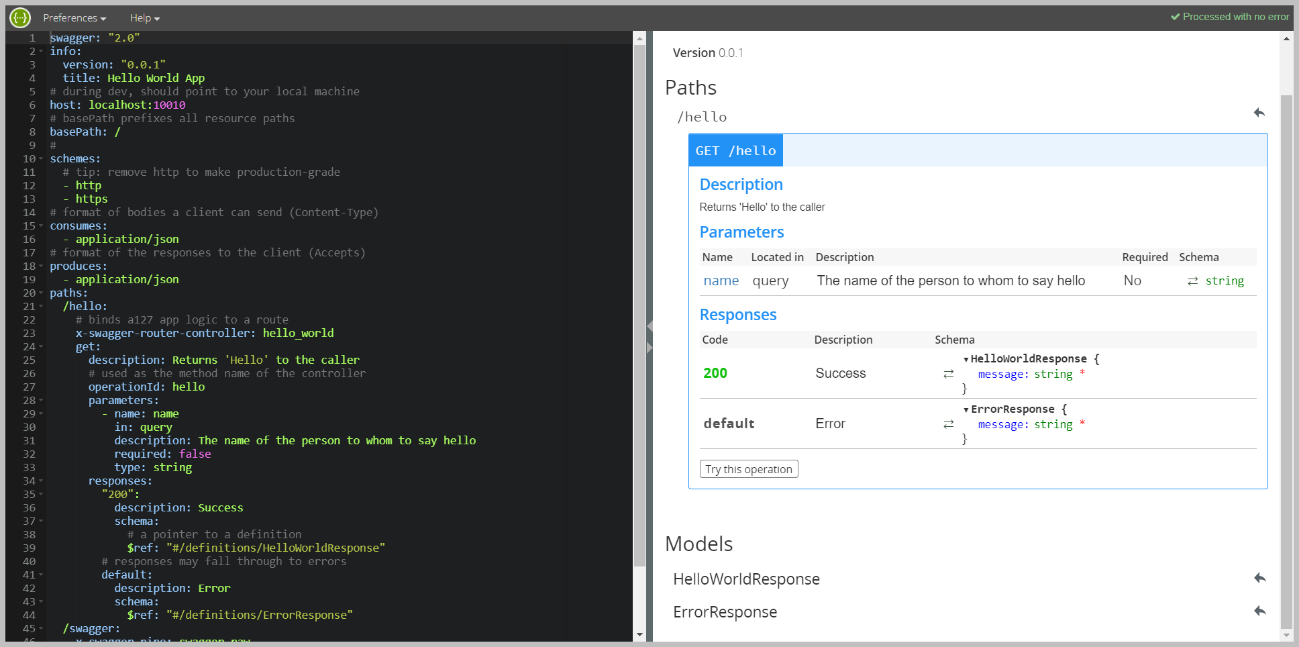
* Với swagger, nếu chúng ta thay đổi bất kì một file nào trong hệ thống thì nó sẽ tự động khởi động lại máy chủ mà không cần dùng lệnh start.

**Step 4:**

* Sau khi khởi động thì mở một cửa sổ command line thứ hai trong project **myapp** với dòng lệnh sau:

***swagger project edit***

* Nếu như dòng lệnh chạy thành công thì trên trình duyệt sẽ mở một tab mới với nội dung như trong hình bên dưới:

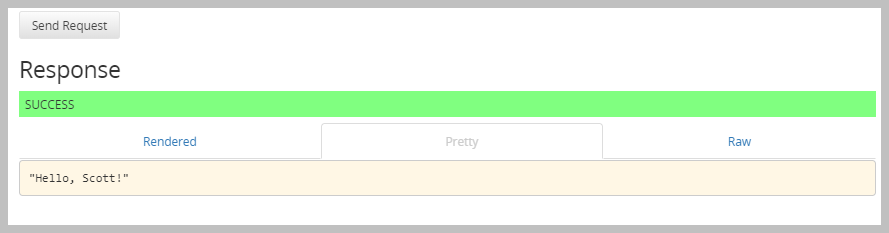


Bây giờ chúng tả thử với một ví dụ để hiểu thêm về swagger. Ở phía bên phải của màn hình, bạn sẽ thấy đường dẫn ví dụ cho một yêu cầu Get/hello, hãy thử nhấn vào “Try this operation” để kiểm tra.

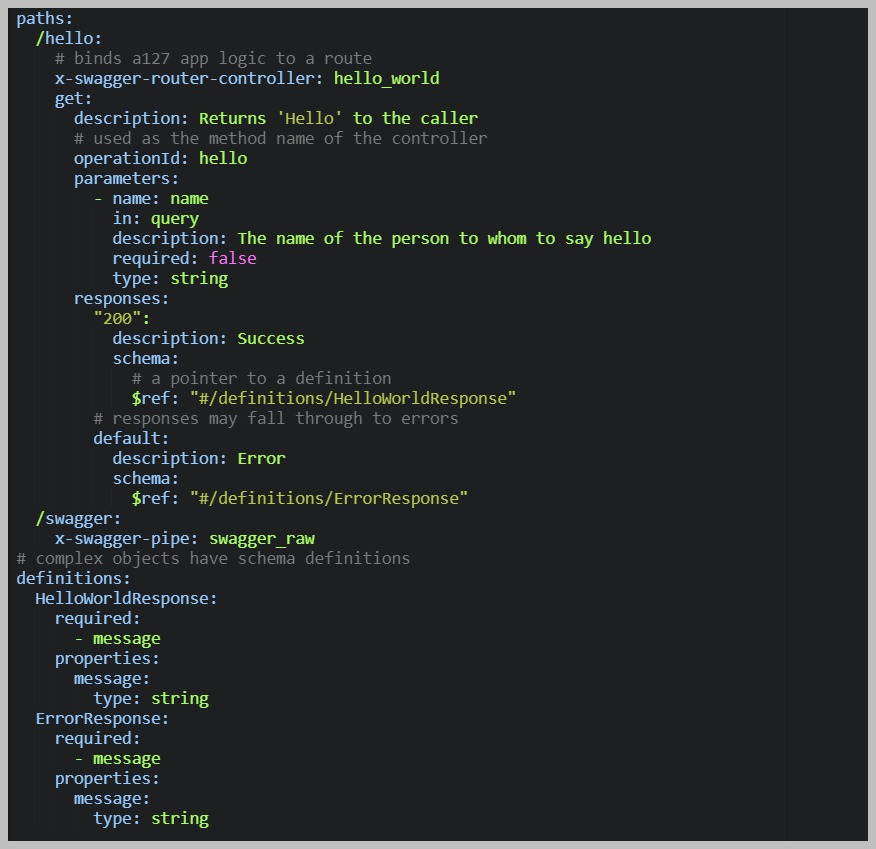
Tại mục tham số “name” yêu cầu phải nhập tên để hệ thống kiểm tra. Hãy nhập một tên bất kì.



Giả sử mình nhập tên là Scott, sau khi nhấn Send request thì response sẽ trả về success và kết quả sẽ như hình bên dưới.



Bây giờ chúng ta sẽ tìm hiểu về cách viết file yaml, hãy nhìn vào màn hình bên trái trình duyệt, để ý kĩ phần **paths**



Trong ví dụ này mình sẽ lấy dữ liệu nên sẽ dùng tới phương thức **get,** ngoài ra chúng ta còn các phương thức khác là: **POST, PUT, DELETE, PATCH, HEAD, OPTION.**

* **x-swagger-router-controller**: Đây là bộ điều khiển, với file nằm trong /api/controllers/hello\_world.js.
* **description**: Mô tả phương thức đang thực thi.
* **operationId:** Đây là hàm nằm trong controller, những operationId này phải là duy nhất trong số tất cả các hoạt động được mô tả trong API.
* **Paramaters**: Trong swagger, các tham số được định nghĩa trong phần thông số của một thao tác hoặc đường dẫn. Để mô tả một tham số, bạn chỉ định tên (**name**), vị trí (**in**), loại dữ liệu (**type**) (được định nghĩa bởi schema(lược đồ) hoặc content (nội dung)) và các thuộc tính khác, ví dụ như mô tả (description) hoặc required.

**Lưu ý:** rằng các tham số là một mảng, do đó, trong YAML, mỗi tham số phải được liệt kê với dấu gạch ngang (-) phía trước nó.

* **Schema:** nó trỏ tới phần định nghĩa chúng ta đã định nghĩa bên dưới phần **definition.**

# CHƯƠNG 3: ĐẶC TẢ YÊU CẦU

* 1. Mô tả tính năng
* ­­­Thu thập dữ liệu từ các website bất động sản
* Cho phép truy xuất dư liệu đã được lấy về thông qua API
* Có thể tùy chỉnh các pattern để thu thập dữ liệu thông qua API
* Tính năng bảo mật API (sử dụng Token để xác thực cho mỗi API)
* Thêm một site bất kì vào để thu thập thông qua API
  1. Giải thuật thu thập (crawl)

Lists url of site

Get all site

List pattern

Lists url of post

Data necessary

Insert data into database

Collect details information such as: price, area, description, title,... by url of post

Collect url of post from each site with pattern

Get all pattern by site

* 1. Các API đã viết

|  |  |  |
| --- | --- | --- |
| **Method** | **Path** | **Description** |
| GET | /Pattern/{Token} | Lấy ra toàn bộ pattern |
| POST | /Pattern/{Token} | Thêm pattern mới |
| GET | /Pattern/FindBy/{Token} | Tìm pattern theo từng tiêu chí. |
| GET | /Pattern/{Token}/{patternID} | Lấy ra pattern theo patternID |
| PUT | /Pattern/{Token}/{patternID} | Lấy ra pattern theo patternID |
| DELETE | /Pattern/{Token}/{patternID} | Xóa pattern theo patternID |
| GET | /Pattern/CheckPattern/ | Check pattern theo từng tiêu chí. |
| GET | /Site/{Token} | Lấy tất cả sites có trong database |
| POST | /Site/{Token} | Thêm mới 1 site |
| PUT | /Site/{Token}/{siteID} | Cập nhật site theo siteID |
| DELETE | /Site/{Token}/{siteID} | Xóa site theo siteID |
| GET | /Crawler/CollectDataDetailBySiteID | Lấy thông tin chi tiết của tin tức theo siteID |
| GET | /Crawler/CollectAllURLBySiteID | Lấy URL bài viết của tất cả các link có trong siteID |
| GET | /User/{Token}: | Lấy hết tất cả user hiện có trong database |
| POST | /User/{Token} | Thêm mới 1 user vào database |
| GET | /User/{Token}/{userID} | Lấy ra user theo userID |
| PUT | /User/{Token}/{userID} | Cập nhật user theo userID |
| DELETE | /User/{Token}/{userID} | Xóa user theo userID |
| POST | /User/GetToken/ | Lấy ra token của user theo userID |
| GET | /PattCategory/{Token} | Lấy tất cả pattern\_categories có trong database |
| POST | /PattCategory/{Token} | Thêm mới 1 pattern\_categories |
| GET | /PattCategory/{Token}/{pattCategoryID} | Lấy ra patternCategory theo patternCategoryID |
| PUT | /PattCategory/{Token}/{pattCategoryID} | Cập nhật patternCategory theo patternCategoryID |
| DELETE | /PattCategory/{Token}/{pattCategoryID} | Xóa Pattern Category theo patternCategoryID |
| GET | /Data/{Token} | Lấy toàn bộ dữ liệu từ data |
| GET | /Data/FindBy/{Token} | Tìm dữ liệu theo yêu cầu đưa ra |

* 1. Script thu thập

getAllSite().then(function (data) {

sites = data;

});

getAllPattern().then(function (data) {

patterns = data;

}).then(function () {

for (var i = 0; i < sites.length; i++) { // loop for each site

var siteID = sites[i].site\_id; // Get siteID

console.log("Collect data from site: " + siteID.toString());

var jsonSiteURL = JSON.parse(sites[i].site\_url); // parse string to JSON

var siteCategories = jsonSiteURL.site\_url\_categories; // Array Site categories

var patternsOfSite = getPatternBySiteID(patterns, siteID); // Get all patterns by SiteID

var patternURL = patternsOfSite[0].pattern\_regex;

var typePage = jsonSiteURL.type\_page\_url;

for (var j = 0; j < siteCategories.length; j++) {

var urlSite = siteCategories[j].category\_url;

console.log(urlSite);

// console.log();

// Define jsOptions variable with json type to contains the options that is a parameter of a [collect] function

var pageLimit = Math.round(totalDataNeedToGet / sites.length / siteCategories.length / 20);

if (pageLimit < 1) pageLimit = 1;

var jsOptions = {

LinkPage: urlSite, // The url of page

TypePage: typePage, // The type of url. You can see it in [sites] table

PatternURL: patternURL, // Pattern to get a URL of each page

SiteID: siteID,

PageLimit: parseInt(pageLimit),

IsInsert: false

}

// console.log(jsOptions);

// execute collect function

crawlerModel.collect(jsOptions, function (data\_crawler) {

// console.log(data\_crawler);

if (data\_crawler.status == "OK" && data\_crawler.data.length > 0) {

var listUrls = data\_crawler.data;

var countExisted = 0, countFinishedAllURL = 0;

var values = [];

for (var k = 0; k < listUrls.length; k++) {

checkURLExisted(listUrls[k], function (isExisted, urlRequest) {

if (isExisted == false) {

getHTML(urlRequest, function (data\_details\_HTML, urlSiteDetail) {

// console.log("Request to: " + urlSiteDetail);

if (data\_details\_HTML != "") {

// Push all data to array

var dataInput = pushDataToArray(siteID, urlSiteDetail, data\_details\_HTML, patternsOfSite);

values.push(dataInput);

}

countFinishedAllURL++;

// check countFinished

if (values.length >= insertEveryURL || countFinishedAllURL >= listUrls.length) {

insertDataToDB(values);

values = [];

}

});

} else {

countExisted++;

countFinishedAllURL++;

}

if (countExisted >= listUrls.length) {

console.log("All of site are existed !");

countExisted = 0;

}

if (countFinishedAllURL >= listUrls.length) {

insertDataToDB(values);

console.log("Finished site: " + urlRequest);

}

});

}

}

});

}

}

});

* 1. Ưu nhược điểm của giải thuật
* Ưu điểm: Thời gian crawler nhanh vì bản chất Nodejs bất đồng bộ nên không cần chờ xử lý request một cách tuần tự.
* Nhược điểm: Chính vì tính chất bất đồng bộ, dẫn đến việc code khó kiểm soát và debug.

**CHƯƠNG 4: VAGRANT**

**Step 1.** Cài đặt virtuabox:

Vào đường dẫn bên dưới để tải về phiên bản virtuabox thích hợp và cài đặt:

*https://www.virtualbox.org/wiki/Downloads*

**Step 2.** Cài đặt Git Bash:

Vào đường dẫn bên dưới để tải về phiên bản Git Bash thích hợp và cài đặt:

*https://git-scm.com/download/*

**Step 3.** Cài đặt vagrant:

Vào đường dẫn bên dưới để tải về phiên bản vagrant thích hợp và cài đặt:

*https://www.vagrantup.com/downloads.html*

**Step 4.** Add box vào vagrant:

* CentOS 7:

*vagrant box add centos/7*

* Ubuntu 14.04LTS:

*vagrant box add ubuntu/trusty64*

**Step 5.** Cài đặt centos/7 vào vagrant:

* Tạo thư mục chứa dữ liệu sau này thao tác với máy ảo vagrant:

*mkdir folder\_nam*e

*cd folder\_name*

* Gõ lệnh: "vagrant box list" để xem danh sách các box OS được add vào.

* NOTE: Tên các box trong list sẽ là tên để tạo máy ảo. Nếu dùng sai tên sẽ không tạo được. Ví dụ: trong box list có các tên "centos/7", "ubuntu/14".. thì khi tạo máy ảo sẽ dùng những tên này.

* Tạo máy ảo:

vagrant init centos/7

* Lúc này vagrant sẽ tạo cho ta 1 file vagrantfile trong thư mục folder\_name. Mở file vagrantfile và tìm đến những dòng sau để bỏ những dấu # đầu dòng:

*# config.vm.network "forwarded\_port", guest: 80, host: 8080*

*# config.vm.network "private\_network", ip: "192.168.33.10"*

*# config.vm.network "public\_network"*

Sửa thành:

*config.vm.network "forwarded\_port", guest: 80, host: 8080*

*config.vm.network "private\_network", ip: "192.168.33.10"*

*config.vm.network "public\_network"*

* Start máy ảo:

*vagrant up*

* Login máy ảo:

*vagrant ssh*

* Sau khi sửa xong và khởi động lại máy ảo hoàn tất. Vào địa chỉ 127.0.0.1:8080 hoặc 192.168.33.10 để vào webserver.

**Step 6.** Cài đặt httpd (Apache2):

* Centos/7:

*sudo yum -y install httpd*

* Start service httpd apache:

*sudo service httpd start*

* Bật tự động Start service httpd apache khi khởi động máy ảo:

*sudo chkconfig httpd on*

**Step 7.** Đồng bộ hoá dữ liệu giữa máy ảo và máy thật:

* Mở file vagrantfile và tìm đến dòng sau:

# config.vm.synced\_folder "../data", "/vagrant\_data"

Sữa lại thành:

config.vm.synced\_folder "data", "/home/vagrant/vagrant\_data"

* NOTE: "data" là tên thư mục chứa dữ liệu cần đồng bộ trên máy thật được đặt trong thư mực "folder\_name" chứ máy ảo. "vagrant\_data" là thư mục được tạo trên máy ảo để đồng bộ hoá dữ liệu từ máy ảo vào thư mục "data" trên máy thật.

**Step 8.** Cài đặt nodejs:

* Centos/7:

*curl --silent --location https://rpm.nodesource.com/setup\_8.x | sudo bash -*

*sudo yum -y install nodejs*

* Ubuntu:

*curl -sL https://deb.nodesource.com/setup\_8.x | sudo -E bash -*

*sudo apt-get install -y nodejs*

**Step 9.** Cài đặt Swagger:

*sudo npm install -g swagger*

**Step 10.** Fix can't sycn\_folder:

* Mở git bash và chạy 2 dòng lệnh sau:

*vagrant plugin install vagrant-vbguest*

*vagrant vbguest*

* Sau khi chạy xong 2 dòng lệnh trên thì có thể tạo lại máy ảo và synced\_folder bình thường.

**Step 11.** Link box add:

*http://www.vagrantbox.es*

* Add box cũng có thể dùng lệnh sau:

*vagrant box add {title} {url}*

Với: "title" tên box, "url" link tải box. Ví dụ: vagrant box add ubuntu1404 https://github.com/kraksoft/vagrant-box-ubuntu/releases/download/14.04/ubuntu-14.04-amd64.box

**Step 12.** Cài đặt MySQL:

*sudo yum -y install mariadb-server mariadb*

* Sau đó sẽ tạo một liên kết khởi động cho MySQL (cho phép MySQL khởi động cùng hệ thống ) và khởi động MySQL server.

*sudo systemctl start mariadb.service //*khởi động services mysql

*sudo systemctl enable mariadb.service //*server tự động bật service mysql

* Đặt password cho tài khoản root MySQL:

*sudo mysql\_secure\_installation*

----------------------------------------------------------------------------------------------

*[root@server /]# mysql\_secure\_installation*

*/usr/bin/mysql\_secure\_installation: line 379: find\_mysql\_client: command not found*

*NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB*

*SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!*

*In order to log into MariaDB to secure it, we'll need the current*

*password for the root user. If you've just installed MariaDB, and*

*you haven't set the root password yet, the password will be blank,*

*so you should just press enter here.*

*Enter current password for root (enter for none): <--ENTER*

*OK, successfully used password, moving on...*

*Setting the root password ensures that nobody can log into the MariaDB*

*root user without the proper authorisation.*

*Set root password? [Y/n] y*

*New password: <--password mariadb của bạn*

*Re-enter new password:*

*Password updated successfully!*

*Reloading privilege tables..*

*... Success!*

*By default, a MariaDB installation has an anonymous user, allowing anyone*

*to log into MariaDB without having to have a user account created for*

*them. This is intended only for testing, and to make the installation*

*go a bit smoother. You should remove them before moving into a*

*production environment.*

*Remove anonymous users? [Y/n] <--ENTER*

*... Success!*

*Normally, root should only be allowed to connect from 'localhost'. This*

*ensures that someone cannot guess at the root password from the network.*

*Disallow root login remotely? [Y/n] <--ENTER*

*... Success!*

*By default, MariaDB comes with a database named 'test' that anyone can*

*access. This is also intended only for testing, and should be removed*

*before moving into a production environment.*

*Remove test database and access to it? [Y/n] <--ENTER*

*- Dropping test database...*

*... Success!*

*- Removing privileges on test database...*

*... Success!*

*Reloading the privilege tables will ensure that all changes made so far*

*will take effect immediately.*

*Reload privilege tables now? [Y/n] <--ENTER*

*... Success!*

*Cleaning up...*

*All done! If you've completed all of the above steps, your MariaDB*

*installation should now be secure.*

*Thanks for using MariaDB!*

----------------------------------------------------------------------------------------------

**Step 13.** Cài đặt PHP:

*sudo yum -y install php*

* Sau khi cài đặt hoàn tất khởi động lại Apache:

*systemctl restart httpd.service*

* Để MySQL hỗ trợ trên PHP ta có thể cài gói php-mysql , ta cũng có thể cài một số module PHP5 khác cần thiết , có thể tìm các module PHP5 khác như sau

*yum search php*

* Chọn 1 cái bạn cần và cài đặt nó:

*yum -y install php-mysql*

* Khởi động lại Apache2

*systemctl restart httpd.service*

**Step 14**. Cài đặt phpMyAdmin để quản lý database trên web dễ hơn:

* Add the EPEL Repository

*rpm -iUvh* [*http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm*](http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm)

* Install phpMyAdmin:

*yum -y update*

*yum -y install phpMyAdmin*

* Tiếp theo sẽ cấu hình phpMyAdmin . Chỉnh sửa Apahe để cho phép các kết nối đến phpMyAdmin không chỉ từ localhost:

*vi /etc/httpd/conf.d/phpMyAdmin.conf*

----------------------------------------------------------------------------------------------

*Alias /phpMyAdmin /usr/share/phpMyAdmin*

*Alias /phpmyadmin /usr/share/phpMyAdmin*

*#<Directory /usr/share/phpMyAdmin/>*

*# <IfModule mod\_authz\_core.c>*

*# # Apache 2.4*

*# <RequireAny>*

*# Require ip 127.0.0.1*

*# Require ip ::1*

*# </RequireAny>*

*# </IfModule>*

*# <IfModule !mod\_authz\_core.c>*

*# # Apache 2.2*

*# Order Deny,Allow*

*# Deny from All*

*# Allow from 127.0.0.1*

*# Allow from ::1*

*# </IfModule>*

*#</Directory>*

*<Directory /usr/share/phpMyAdmin/>*

*Options none*

*AllowOverride Limit*

*Require all granted*

*</Directory>*

---------------------------------------------------------------------------------------------

* Tiếp theo sẽ khởi động lại Apache2:

*systemctl restart httpd.service*

* Sau khi khởi động lại, có thể kết nối đến phpMyAdmin bằng cách truy cập đến http://yourip/phpMyAdmin (ví dụ http://192.168.33.10/phpMyAdmin)

# CHƯƠNG 5: BÁO CÁO CÔNG VIỆC

1. **Nguyễn Văn Khoa**

|  |  |
| --- | --- |
| **Ngày** | **Công việc** |
| 06/03/2018 | Tạo mô hình mvc  Kết nối database |
| 07/03/2018 | Kết nối, truy xuất database |
| 12/03/2018 | Lấy dữ liệu từ trang bất động sản |
| 13/03/2018 | Sửa database theo đúng mô hình |
| 14/03/2018 | x |
| 19/03/2018 | Viết giao diện cho các chức năng có sẵn |
| 20/03/2018 | x |
| 02/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs |
| 03/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs |
| 04/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs, viết database |
| 05/04/2018 | Viết API (file yaml) + model + controller của table sites  Viết các chức năng cơ bản như add, find, findAll, update, delete |
| 09/04/2018 | Viết API (file yaml) + model + controller của table sites  Viết các chức năng cơ bản như add, find, findAll, update, delete |
| 10/04/2018 | Viết API (file yaml) + model + controller của table sites  Viết các chức năng cơ bản như add, find, findAll, update, delete |
| 12/04/2018 | x |
| 16/04/2018 | Viết API crawler detail |
| 17/04/2018 | Check pattern cho từng site |
| 18/04/2018 | Test pattern |
| 19/04/2018 | Kiểm tra dữ liệu lấy về |
| 23/04/2018 | Viết API check pattern |
| 24/04/2018 | Check + Update status pattern table when crawler |
| 02/05/2018 | x |

1. **Đỗ Thanh Phong**

|  |  |
| --- | --- |
| **Ngày** | **Công việc** |
| 06/03/2018 | Tạo mô hình mvc  Kết nối database |
| 07/03/2018 | Kết nối, truy xuất database, hiển thị ra view |
| 12/03/2018 | Lấy dữ liệu từ trang bất động sản |
| 13/03/2018 | Sửa code theo mô hình OOP |
| 14/03/2018 | x |
| 19/03/2018 | Viết giao diện cho các chức năng có sẵn |
| 20/03/2018 | x |
| 02/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs |
| 03/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs |
| 04/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs, viết database |
| 05/04/2018 | Viết API crawler, API cho table data |
| 09/04/2018 | Viết API crawler, API cho table data |
| 10/04/2018 | Viết API crawler, API cho table data |
| 12/04/2018 | x |
| 16/04/2018 | Viết API crawler detail |
| 17/04/2018 | Viết API crawler dữ liệu từ trang timmuanhadat, lấy dữ liệu chi tiết. |
| 18/04/2018 | x |
| 19/04/2018 | x |
| 23/04/2018 | Viết API check pattern |
| 24/04/2018 | Check + Update status pattern table when crawler |
| 02/05/2018 | x |

1. **Phạm Vũ Hoàng Phượng**

|  |  |
| --- | --- |
| **Ngày** | **Công việc** |
| 06/03/2018 | Tạo mô hình mvc  Kết nối database |
| 07/03/2018 | Kết nối, truy xuất database |
| 12/03/2018 | Lấy dữ liệu từ trang bất động sản |
| 13/03/2018 | x |
| 14/03/2018 | x |
| 19/03/2018 | Lấy dữ liệu |
| 20/03/2018 | x |
| 02/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs |
| 03/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs |
| 04/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs, viết database |
| 05/04/2018 | Viết API (file yaml) + model + controller của table User  Viết các chức năng cơ bản như add, find, findAll, update, delete |
| 09/04/2018 | Viết API (file yaml) + model + controller của table User  Viết các chức năng cơ bản như add, find, findAll, update, delete |
| 10/04/2018 | Viết API (file yaml) + model + controller của table User  Viết các chức năng cơ bản như add, find, findAll, update, delete |
| 12/04/2018 | x |
| 16/04/2018 | Truy xuất data crawler |
| 17/04/2018 | Viết API data detail và document swagger |
| 18/04/2018 | Viết API data detail và document swagger |
| 19/04/2018 | Kiểm tra dữ liệu lấy về |
| 23/04/2018 | Viết API check pattern |
| 24/04/2018 | Check + Update status pattern table when crawler |
| 02/05/2018 | x |

1. **Trần Trí Tín**

|  |  |
| --- | --- |
| **Ngày** | **Công việc** |
| 06/03/2018 | x |
| 07/03/2018 | Tạo mô hình mvc, kết nối database |
| 12/03/2018 | Lấy dữ liệu từ trang bất động sản |
| 13/03/2018 | Webpack Reactjs |
| 14/03/2018 | x |
| 19/03/2018 | Reactjs |
| 20/03/2018 | x |
| 02/04/2018 | Sử dụng CSS tự viết react - webpack |
| 03/04/2018 | Sử dụng CSS tự viết react - webpack |
| 04/04/2018 | Sử dụng CSS tự viết react - webpack |
| 05/04/2018 | Sử dụng CSS tự viết react - webpack |
| 09/04/2018 | Viết API crawler, API cho table data |
| 10/04/2018 | Viết API crawler, API cho table data |
| 12/04/2018 | x |
| 16/04/2018 | x |
| 17/04/2018 | x |
| 18/04/2018 | x |
| 19/04/2018 | Kiểm tra dữ liệu lấy về |
| 23/04/2018 | Viết API check pattern |
| 24/04/2018 | Check + Update status pattern table when crawler |
| 02/05/2018 | x |

1. **Lê Cẩm Tú**

|  |  |
| --- | --- |
| **Ngày** | **Công việc** |
| 06/03/2018 | Sử dụng framework Express  Tạo mô hình mvc dựa vào install express |
| 07/03/2018 | Kết nối, truy xuất database |
| 12/03/2018 | Lấy dữ liệu từ trang bất động sản |
| 13/03/2018 | Sửa database theo đúng mô hình |
| 14/03/2018 | x |
| 19/03/2018 | Lấy dữ liệu |
| 20/03/2018 | x |
| 02/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs |
| 03/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs |
| 04/04/2018 | Xây dựng API demo dựa trên Swagger - Express Nodejs, viết database |
| 05/04/2018 | Viết API (file yaml) + model + controller của table Pattern\_categories  Viết các chức năng cơ bản như add, find, findAll, update, delete |
| 09/04/2018 | Viết API (file yaml) + model + controller của table Pattern\_categories  Viết các chức năng cơ bản như add, find, findAll, update, delete |
| 10/04/2018 | Viết API (file yaml) + model + controller của table Pattern\_categories  Viết các chức năng cơ bản như add, find, findAll, update, delete |
| 12/04/2018 | x |
| 16/04/2018 | Truy xuất data crawler |
| 17/04/2018 | Add pattern vào database |
| 18/04/2018 | Add pattern đã test vào database |
| 19/04/2018 | Kiểm tra dữ liệu lấy về |
| 23/04/2018 | Viết API check pattern |
| 24/04/2018 | Check + Update status pattern table when crawler |
| 02/05/2018 | x |

# GÓP Ý MÔN HỌC

# acbxyz