http://niralar.com/setting-up-django-development-environment-using-anaconda/

manage.py giúp cho bạn những câu lệnh để quản lý cái project của bạn (ví dụ chạy website của bạn hoặc làm những chức

năng liên quan đến database của bạn)

wsgi.py dùng 1 file hỗ trợ cho bạn khi quá trình bạn đẩy website lên serve (gọi quá trình deployment)

urls.py chứa đường dẫn website của bạn

settings.py một số các biến toàn cục cấu hình cho toàn bộ website của bạn

\_init\_.py hiện một module của python thôi nó rỗng cũng được

Cài đặt django

Câu lệnh: pip install Django

Nguyên cứu cấu trúc của project của dự án

+ Phần View (views.py)

+ Phần url (urls.py)

+ Phần Model (models.py)

+ Phần cấu hình (settings.py)

Tạo ra một cái app

Câu lệnh: python manage.py startapp <tên app>

\*\*\*\*\* ví dụ chạy 1 trang web google.com thì nó sẽ chạy vào đâu trước, làm sao định tuyến được đường dẫn đấy tức là phần url , link tới cái views nào, views đó code gì trong đó, phương thức của html đó là get và post \*\*\*\*\*\*\*\*

url -> view -> xử lý logic -> template

Phần xác thực người dùng

Đăng nhập hệ thống của mình , xác thực, phân quyền , Django rest framework (viet api, trong api có 2 phần quan trọng đó là json web token) (link <https://www.django-rest-framework.org/>)

Session, Cookie

Chạy lệnh dir để xem cấu trúc của Django

Chạy project thử file manage.py

Lệnh terminal: python manage.py runserver

Để tạo ra 1 cái application dùng câu lệnh: python manage.py startapp <tên app>

-> khi tạo nó có migrations (nó lưu lại lịch sử của bạn tạo ra database và thay đổi database như thế nào)

-> \_init\_ báo hiệu đây là 1 module của python

-> models là ánh xạ giữa cơ sở dữ liệu và python

Chạy lệnh lệnh python manage.py để xem các gợi ý nếu chạy sai để copy -> chạy lệnh python manage.py makemigrations (để lưu lại lịch sử database)

Để tạo ra database

Lệnh: python manage.py migrate

Tạo tài khoản đăng nhập

Lệnh: python manage.py createsuperuser

Link chạy <http://127.0.0.1:8000/admin>

* Chú ý cái template nào thay đổi thì cho nó vào {% block content %} thì bên extend vẫn block content (còn nếu {% block content123 %} thì bên extend vẫn block content123)
* <!-- {% if dsquest %} neu dsquest ton tai roi moi duyet for-->  
  <!-- item đang là 1 object nên phải in ra từng thuộc tính của nó -->  
    
   Nếu muốn lấy ra tất cả dữ liệu thì chỉ cần Question.objects.all()  
    
   Nếu muốn lấy 1 số thành phần thì sao thì có hỗ trợ hàm get\_object\_or\_404

python manage.py shell (chạy vào shell để test dữ liệu )

from polls.models import Question

a = Question.objects.get(pk = 1) (pk là get theo khóa chính)

a.id

a.question\_test

a.chooice\_set.all() (trả về chooice trong admin mình them)

b = a.chooice\_set.all()

b[0].chooice\_test

exit() để thoát từ shell sang open terimal

cấu trúc file Django gồm có \_init\_.py, asgi.py, settings.py, urls.py, wsgi.py

manage.py

path('detail/<int:question\_id>', views.detailView, name="detail"),

là truyền vào cái question\_id kiểu int

Luồng xử lý: url -> view -> xử lý logic ->templates

**Session** là một phiên làm việc là một khái niệm phổ biến được dùng trong lập trình web có kết nối với database. Đặc biệt các chức năng như đăng nhập, đăng xuất người dùng sẽ khó có thể thực hiện được nếu không sử dụng **session**.

Cách sử dụng Session

Một session bắt đầu khi client gửi request đến server, nó tồn tại xuyên suốt từ trang này đến trang khác trong ứng dụng web và chỉ kết thúc khi hết thời gian timeout hoặc khi bạn đóng ứng dụng. Giá trị của session sẽ được lưu trong một file trên server.

Ví dụ khi bạn đăng nhập vào một trang web và đăng nhập với tài khoản đã đăng ký trước đó. Server sau khi xác thực được thông tin bạn cung cấp là đúng thì nó sẽ sinh ra một tập tin chứa dữ liệu cần lưu trữ của người dùng.

Với mỗi session sẽ được cấp phát một định danh duy nhất SessionID. Khi kết thúc một phiên làm việc và bắt đầu một phiên mới, dĩ nhiên bạn sẽ được cấp một SessionID khác với trước đó. Bạn có thể tuỳ ý quyết định xem nên lưu trữ những thông tin nào vào Session. Nhưng thông thường chúng ta chỉ nên lưu những thông tin tạm thời trong **session.**

## Cookie là gì?

Giống như **session**, **cookie** cũng được dùng để lưu những thông tin tạm thời. Nhưng tập tin cookie sẽ được truyền từ server tới browser và được lưu trữ trên máy tính của bạn khi bạn truy cập vào ứng dụng.

## Cách sử dụng Cookie

**Cookie** thường được tạo ra khi người dùng truy cập một website, **cookie**sẽ ghi nhớ những thông tin như tên đăng nhập, mật khẩu, các tuỳ chọn do người dùng lựa chọn đi kèm. Các thông tin này được lưu trong máy tính để nhận biết người dùng khi truy cập vào một trang web.

Khi người dùng truy cập đến một trang web có sử dụng **cookie**, web server của trang đó sẽ tự động gửi **cookie** đến máy tính của người dùng. Khi truy cập đến các trang web sử dụng được **cookie** đã lưu, những **cookie** này tự động gửi thông tin của người dùng về cho chủ của nó (người tạo ra **cookie**). Tuy nhiên những thông tin do **cookie** ghi nhận không được tiết lộ rộng rãi, chỉ có website chứa **cookie** mới có thể xem được những thông tin này.

Mỗi cookie thường có khoảng thời gian timeout nhất định do lập trình viên xác định trước. Những thông tin được lưu vào cookie ví dụ như thông tin đăng nhập, thao tác người dùng, tần xuất ghé thăm website, thời gian truy cập… Tất cả chúng đều là những thông tin mang tính tạm thời và được lưu trong 1 khoảng thời gian.

**Cookie** được xem là một thành phần không thể thiếu được với những website có khối lượng dữ liệu lớn, có số lượng người dùng đông, và có những chức năng đi kèm với thành viên đăng ký.

# **So sánh giữa Cookie và Session**

| **Cookie** | **Session** |
| --- | --- |
| **Cookie** được lưu trữ trên trình duyệt của người dùng. | **Session** không được lưu trữ trên trình duyệt. |
| Dữ liệu **cookie** được lưu trữ ở phía client. | Dữ liệu **session** được lưu trữ ở phía server. |
| Dữ liệu **cookie** dễ dàng sửa đổi hoặc đánh cắp khi chúng được lưu trữ ở phía client. | Dữ liệu **session** không dễ dàng sửa đổi vì chúng được lưu trữ ở phía máy chủ. |
| Dữ liệu **cookie** có sẵn trong trình duyệt đến khi expired. | Sau khi đóng trình duyệt sẽ hết phiên làm việc (session) |

### Cache

Cache là bộ nhớ đệm, vùng lưu trữ tạm thời trong máy tính. Nó khác với cookie ở chỗ thông tin lưu trữ ở đây là các tài liệu web, các hình ảnh, các video, HTML, .... Một phát kiến khá sáng suốt của con người trong thời kỳ công nghệ thông tin này để giảm tải băng thông, tăng tốc độ load, truy cập web. Bộ nhớ cache bao gồm bản sao các bit của trang Web được lưu trữ trên ổ đĩa cứng. Trình duyệt tải các bit khi bạn truy cập một trang Web nào đó, tốc độ truy cập sẽ nhanh hơn và tiết kiệm được băng thông khi "download". Bộ nhớ cache sẽ được lưu trữ cho đến khi bạn tự tay xóa nó đi. Do đó đừng dại mà xóa cache khi không cần thiết nhé (mình gỡ bỏ CCleaner trên máy tính từ lâu rồi (yaoming) )

Bộ nhớ cache chính là nơi trình duyệt của bạn lưu trữ những file coppy để bạn không phải tải lại lần nữa khi duyệt web. VD. Lần đầu khi bạn truy cập vào một trang web có rất nhiều hình ảnh, Bạn mất khoảng 1s để load xong trang (chắc do mạng cùi bắp), Nhưng nhờ có bộ nhớ cache, thời gian để bạn load trang có thể gần như là ngay lập tức luôn cũng được đó.

**\*\* Xử lý hành động submit form**

<a href="{% url 'detail' %}"><p>{{item.question\_test}}</p></a>

**Do mình đã khai báo nó bên file urls** path('detail/<int:question\_id>', views.detailView, name="detail"),

Để xử lý đoạn này liên kết với url

 {% csrf\_token %} để tránh một lỗi bảo mật vì python nó rất bảo mật

JavaScript khởi đầu là một ngôn ngữ thông dịch nhưng sau một chặng đường dài phát triển và cải tiến, JavaScript mà hiện giờ có thể được coi là một ngôn ngữ lai, vừa là thông dịch vừa là biên dịch.

Những khái niệm trên trong quá trình lập trình thường không được coi trọng lắm nhưng sẽ hay hơn nếu những người lập trình hiểu rõ hơn một phần nào đó những việc phía dưới máy tính thực hiện sau khi chúng ta viết một chương trình.

## Ngôn ngữ thông dịch

Ngôn ngữ được coi là thông dịch khi chương trình được viết ra khi chạy sẽ được trực tiếp thành mã máy (ngôn ngữ mà máy tính có thể hiểu được) để máy tính thực thi chúng. Khi chương trình chạy đến dòng lệnh nào sẽ chuyển thành mã máy đến đó để máy tính có thể thực thi.

Bộ thông dịch thực hiện quá trình thông dịch gọi là interpreter.

**Ưu điểm**

* Interpreter dễ hiện thực hơn do bỏ qua việc kiểm tra lỗi và tối ưu code thường được thực hiện trong quá trình compiled.
* Hỗ trợ đa nền tảng.
* Kích thước chương trình thực thi nhỏ hơn.

**Nhược điểm**

* Chương trình có độ tin cậy thấp hơn do bỏ qua bước kiểm tra loại bỏ một số lỗi thường thực hiện trong quá trình compiled.
* Source code dễ dàng bị dịch ngược.
* Tốc độ thực thi chậm hơn.
* Tiềm tàng nguy cơ có lỗi do thiếu

## Ngôn ngữ biên dịch

Cách hoạt động của trình biên dịch khác so với thông dịch. Thay vì chạy trực tiếp thành mã máy, trình biên dịch sẽ phải chuyển đổi ngôn ngữ lập trình thành mã máy rồi chứa kết quả vào ổ đĩa cứng để có thể thực thi ở lần chạy sau.

Bộ biên dịch thực hiện quá trình biên dịch được gọi là compiler.

**Ưu điểm**

* Chương trình sau đó được thực thi nhanh hơn.
* Độ tin cậy cao
* Khó bị dịch ngược mã nguồn.

**Nhược điểm**

* Khó xây dựng một compiler có tính chính xác cao để chuyển toàn bộ chương trình thành mã máy.
* Mã máy của mỗi nền tảng là khác nhau, khó thực hiện đa nền tảng.

Python thường được gọi là ngôn ngữ lập trình kiểu thông dịch.

**Lệnh pass trong Python** giống như tên của nó, được sử dụng khi một lệnh là cần thiết theo cú pháp nhưng bạn không muốn bất cứ lệnh hoặc khối lệnh nào được thực thi.

Lệnh pass là một hoạt động null và không có gì xảy ra khi nó thực thi.

Hiểu một cách đơn giản pass có nghĩa là bỏ qua. Ví dụ sau sẽ in ra màn hình số chẵn từ 1 đến 20, để làm được điều này chúng ta sẽ lặp i từ 1 đến 20, kiểm tra điều kiện nếu i chia hết cho 2 thì là số chẵn (bỏ qua), ngược lại i là số lẻ:

pass

Chúng ta thường sử dụng lệnh này như một placeholder. Giả sử, bạn có một vòng lặp, hoặc một hàm, nhưng chưa biết nên xây dựng nó như thế nào, chưa biết nên code sao cho tối ưu và muốn để lại làm sau. Nhưng hàm, lệnh đó không thể có một khối lệnh rỗng, trình biên dịch sẽ báo lỗi, vì thế, chỉ cần sử dụng lệnh pass để xây dựng một khối lệnh rỗng, lúc này trình biên dịch sẽ hiểu và không "phàn nàn" gì bạn nữa.

////////////////////////

**Bài học URL namespace và static trong django**

url namespace va static trong django do nếu trong url có nhiều app có views tên giống nhau thì thay vì mình sẽ  
app\_name = 'polls' (là tên app) truyền vào action="{% url 'polls:vote' qs.id %}"

dùng hàm {% load static % } để load file static vào xem code

<link href=”stylesheet” href=”{% static <ten thu muc> -> file css %}”

Example: <link rel="stylesheet" href="{% static 'polls/style.css' %}">

**Bài học Model form trong Django**

**Cài đặt Django câu lênh: pip install django**

**Tạo project: django-admin startproject demoform**

**Start app: python manage.py startapp news (tao app)**

Liên kết app news này vào ứng dụng chính trong file settings.py (trong INSTALLED\_APPS) -> ten news (vi du tao app ten news)

Tạo đường dẫn url trong file urls.py

from django.urls import path, include

from . import views

(dấu . là thư mực hiện tại)

<https://stackoverflow.com/questions/47563013/unable-to-import-path-from-django-urls>

cách gọi path hay url ví dụ như này để chạy trình duyệt xem Django version bao nhieu de

gọi cho đúng kiểm tra bằng cách python -m Django –version

(url('', views.index, name='index'),

from django.conf.urls import url, include

**Khi tạo database xong nhớ chạy câu lênh sau:**

Câu lênh: python manage.py makemigrations

Và python manage.py migrate

Sau đó run lai server voi cau lenh python manage.py runserver

**Tạo user, pass để login trang admin**

python manage.py createsuperuser

**Có 2 kiểu form model đó là: form dựa trên model (ModelForm ) và form tự định nghĩa**

**Ví dụ form tự định nghĩa trong forms.py class SendEmail có widget tự định nghĩa**

**Thì thêm thuộc tính attrs - {‘class’: ‘sonnguyen’} đối với form tự định nghĩa**

**Ví dụ:** content = forms.CharField(widget=forms.Textarea(attrs={'class':'sonnguyen','id': 'noidung'}))

**Còn nếu form dựa trên model (ModelForm) thì khai báo ra như sau: ví dụ trong file forms.py class PostForm**

widgets = {  
 'title' : forms.TextInput(attrs={'class':'tieude'})  
}

**(nghĩa là nó tên của trường nào) ví dụ nó là title: thì khai báo ở trên**

**Các thuộc tính khác cũng làm như vậy chẳng hạn như value…**

**Và cũng có nhiều kiểu khác nhau mà các bạn dựa trên tên tiếng anh**

**……………. Bài học Class Base View ------------------------------**

* **Lợi ích sử dụng class base view:**

**Class base view là chúng ta kế thừa lại những gì mà của các lớp**

**Ví dụ chúng ta viết ra lớp cơ sở chẳng hạn (có những đặc điểm chung, viết những thuộc tính chung đó vào class đó, sau đó chúng ta kế thừa và sử dụng lại)**

urlpatterns = [  
 # cach viet nay goi function ben view xem trong file views.py co chu thich  
 # path('', views.index, name="index" ),  
 #  
   
 # Cach goi nay cua class view xem trong file views.py co chu thich  
 path('', views.IndexClass.as\_view(), name="index"),  
 #  
 path('add/', views.add\_post, name="add"),  
 path('save/', views.save\_news, name="save"),  
 path('email/', views.email\_view, name="email"),  
 path('process/', views.process, name="pro"),  
   
   
]

* **khi mà chúng ta sử dụng hàm as\_view() này nó sẽ convert class này thành view cho chúng ta**

**Các method trong python: get, post, put, patch, delete, head, options, trace**

**Ctrl + shift + T để trở lại cái vừa mở trên trình duyệt**

def \_\_str\_\_(self):  
 return self.title

**để lưu trong trang admin hiển thị ra tên title -> thay vì nó lưu tên object (1)….**

**……………. Bài học User Anthenticate Trong Django -------------------------**

Add thư viện: from Django.contrib.auth import authenticate

* Đây là hàm giúp ta xác thực nội dung

Trước khi chạy phải copy cái tên name trong file app.py khai báo trong file settings.py

Nếu đụng đến database thì trong file settings.py có chỗ cấu hình

DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),  
 }  
}

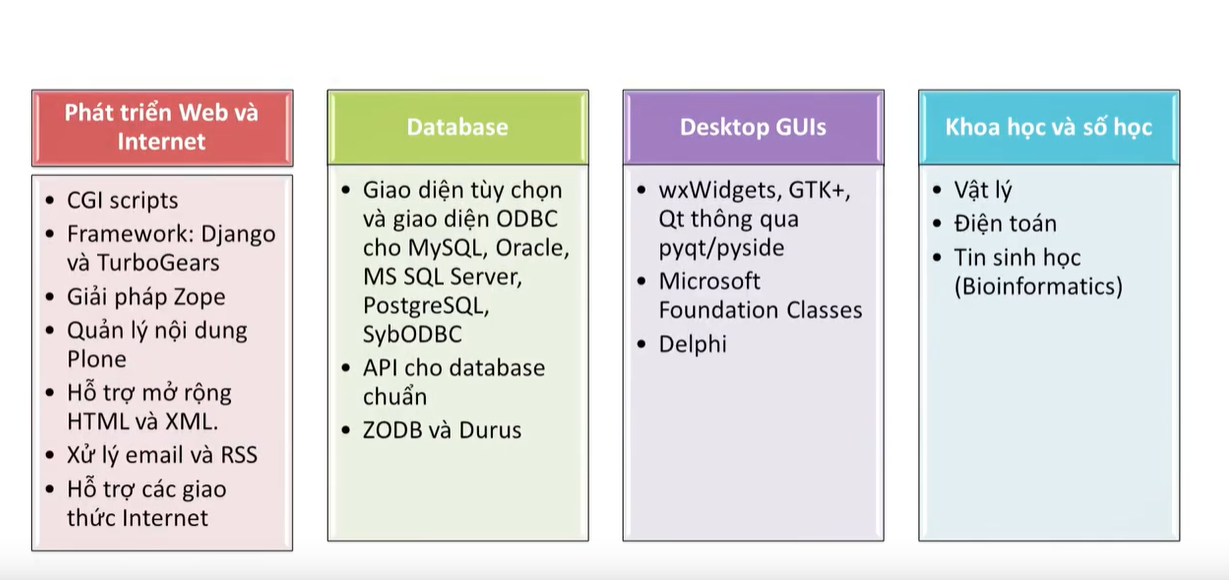
Nếu dùng mysql or cái nào đó thì thay tên vào (chỗ sqlite3)…..

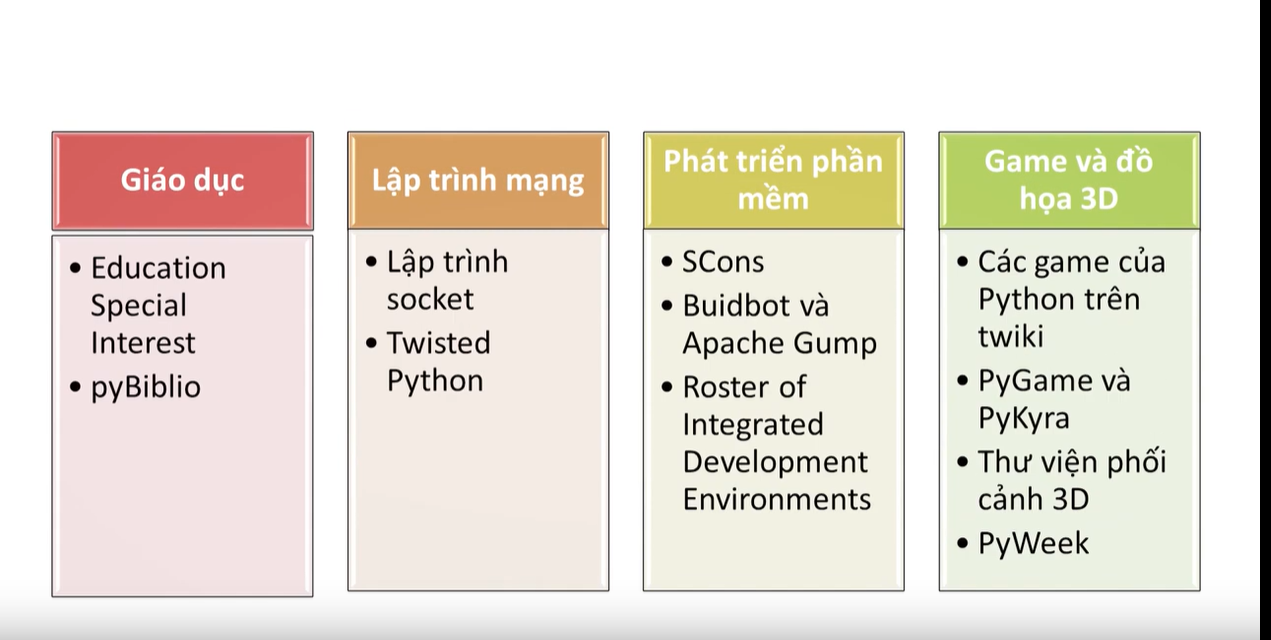
**……………. Bài học Tìm hiểu về user model trong Django -------------------------**

Python phát triển trên các nền tảng



ứng dụng của python





Nó viết code dễ dàng, ngắn hơn

Tạo ra tên là Authen và trong authen có app ten là login

django-admin startproject Authen

python manage.py startapp Login

python manage.py migrate (**nó chạy bên phía admin)**

python manage.py createsuperuser (câu lệnh này chạy gõ username và password) để đăng nhập trên admin

**……………. Bài học Phân quyền view decorator trong Django -------------------------**

**Cach 1:**

from django.contrib.auth import decorators

Bắt nó đăng nhập login\_url = ‘/login/’ là đường dẫn login khai báo path trong file urls.py

@decorators.login\_required(login\_url = '/login/')

Cach 2:

Mixins kết nối các đối tượng trong django

from django.contrib.auth.mixins import LoginRequiredMixin

sửa đoạn này coi thêm cho hiểu

class ViewUser(View, LoginRequiredMixin):  
 login\_url = '/newsform/login/'  
 def get(self, request):  
 # if not request.user.is\_authenticated:  
 # return HttpResponse('ban vui long dang nhap')  
 # else:  
 # return HttpResponse('<h1>day la view user</h1>' )  
   
 return HttpResponse('<h1>day la view user</h1>')  
   
 def post(self, request):  
 pass

**……………. Bài học Phân quyền user trong Django -------------------------**

File models.py

class PostLogin(models.Model):  
 title = models.CharField(max\_length=100)  
 content = models.TextField(max\_length=255)

file admin.py

from .models import Post, PostLogin

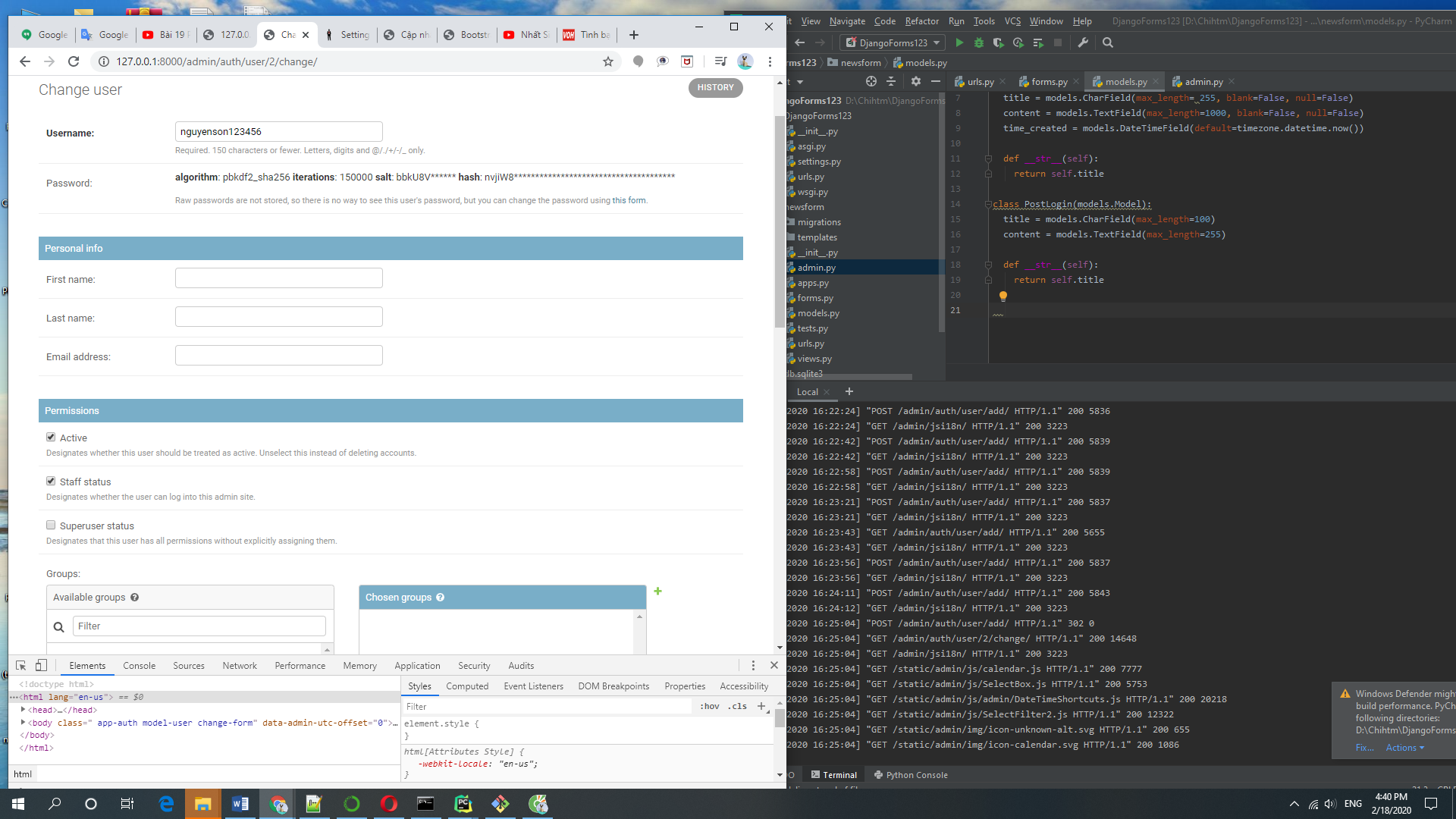
admin.site.register(PostLogin)

nhớ khi tạo database trong models thì chạy câu lệnh để nó tạo ra:

python manage.py makemigrations

python manage.py migrate

Sau khi tạo tài khoản xong chưa đăng nhập được có báo đỏ thì click vào chọn tick chỗ -> staff status



**……………. Bài học check user permissions (quyền) trong Django -------------------------**

Nếu lỗi nhớ thêm csrf\_token vào trang html bất kỳ đâu cũng được

{% csrf\_token %}

def post(self, request):  
 f = PostForm(request.POST)  
 if not f.is\_valid():  
 return HttpResponse('Ban nhap sai du lieu roi')  
 f.save()  
   
 return HttpResponse('Oke')

f.save (nó sẽ lưu xuống database)

Nếu muốn chạy login trước thì thêm

Login\_url = ‘/newsform/login/,

Và nhớ import thư viện LoginRequiredMixin vào

from django.contrib.auth.mixins import LoginRequiredMixin

class AddPost(LoginRequiredMixin, View):  
 login\_url = '/newsform/login/'  
 def get(self, request):  
 f = PostFormLogin()  
 context = {'fm': f}  
 return render(request, 'news/add\_post.html', context)  
   
 def post(self, request):  
 f = PostFormLogin(request.POST)  
 if not f.is\_valid():  
 return HttpResponse('Ban nhap sai du lieu roi')  
 f.save()  
 return HttpResponse('Oke')

def post(self, request):  
 f = PostFormLogin(request.POST)  
 if not f.is\_valid():  
 return HttpResponse('Ban nhap sai du lieu roi')  
 if request.user.has\_perm('newsform.add\_post'): # viet thuong ten model  
 f.save()  
 else:  
 return HttpResponse('may khong co quyen')  
 return HttpResponse('Oke')

kiểm tra nếu nó có quyền không (có quyền thì vào trang admin để chọn set nó).

Với câu lệnh

if request.user.has\_perm('newsform.add\_post'): # viet thuong ten model

(‘newsform là tên app trong file app.py và .add\_post là tên model )

print(request.user.get\_all\_permissions())

Lệnh này muốn biết user nó có quyền gì nó in ra để biết

**……………. Bài học custom user model trong Django -------------------------**

from django.contrib.auth.models import AbstractUser

có nghĩa là AbstractUser để thêm 1 số thuộc tính cho user

address = models.CharField(default='', max\_length=255)  
 class MyUser(AbstractUser):  
 sex\_choice = ((0, "Nu"), (1, "Nam"), (2, "Gioi tinh khong xac dinh"))  
 age = models.IntegerField(default = 0)  
 sex = models.IntegerField(choices=sex\_choice, default=0) # default = 0 la chon nu  
 address = models.CharField(default='', max\_length=255)

trong models.py

và muốn thêm vào chúng ta phải cấu hình trong file setting.py

AUTH\_USER\_MODEL = 'newsform.MyUser'

* Newsform la ten của cái app chúng ta trong file apps.py, MyUser là tên class trong file models.py cần được cấu hình
* Rồi sau đó vào file admin.py thêm nó vào chỗ hệ quản trị đó để nó render ra
* from .models import Post, PostLogin, MyUser
* admin.site.register(MyUser)

xong chạy câu lệnh: python manage.py makemigrations

và câu lệnh: python manage.py migrate

(còn nếu ko dc thì chạy python manage.py migrate newsform (newsform là tên app))

**……………. Bài học tùy chỉnh giao diện admin custom admin site trong Django -------------------------**

STATIC\_ROOT = 'static'

Static\_root này khi các bạn dùng một câu lệnh trong manage thì nó lấy tất cả các static của các app trong project chúng ta và nhét vào thư mục có tên là static

Tạo ra thư mục có tên static (giống như mình định nghĩa nó trong static\_root trong file settings.py) cùng cấp với file manage.py

Chạy câu lệnh python manage.py để coi

Rồi sau đó chạy lại câu lệnh python manage.py collectstatic

STATICFILES\_DIRS = [  
 os.path.join(BASE\_DIR, "static"),  
   
]

Với static là cái tên static\_root cấu hình và đã tạo ra thư mục static

TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [os.path.join(BASE\_DIR, templates)],  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]

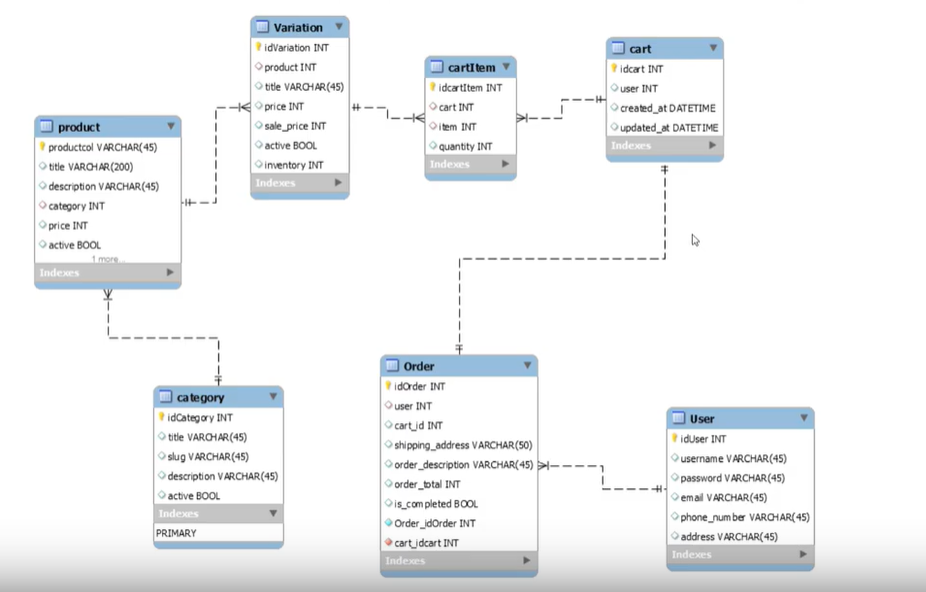
Nếu muốn thêm templates vào trong trang admin thì thêm vào DIRS , với templates là tên thư mục mình tạo muốn đặt tên gì cũng được

* sau đó mình tạo templates rồi tạo thư mục admin trong templates

------------ Link source Django xem mẫu trên mạng----------------------

<https://github.com/django/django/blob/master/django/contrib/admin/templates/admin/login.html>

**……………. Bài học xây dựng website bán hàng trong Django -------------------------**



Database coi để kham khảo

Nhớ phải cài Django trước khi khởi tạo app

Khởi tạo app: django-admin startproject shop

Ví dụ tạo từng thư mục để dễ quản lý (theo như database thôi)

django-admin startapp product

django-admin startapp cart

……

Nếu muốn nhìn chung vào một hệ thông shop (bán hàng) thì quan trọng có như sau:

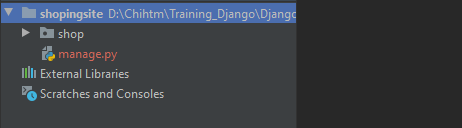
Product, cart, order, user

Khi tạo app xong thì khai báo tên app vào settings.py để chạy ví dụ tạo 4 app nên đã được gọi và tô màu vàng

MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
 'cart',  
 'product',  
 'order',  
 'user',  
]

django-admin startproject shop .

là tạo ta manage.py luôn khoải phải thêm thư mục nữa như câu lệnh Django-admin startproject shop

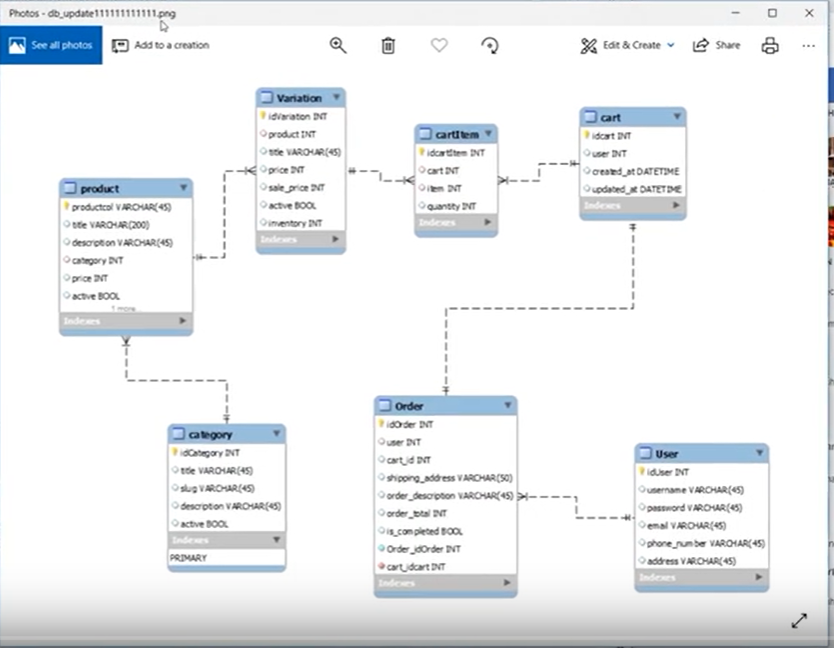


Xong chạy tiếp câu lệnh: python manage.py migrate

python manage.py createsuperuser (tạo ra username và password) để login trong trang admin

ten login la shop123, pass shop123

**……………. Bài học xây dựng model website bán hàng trong Django -------------------------**



Khai bao trong models cua product

Tương tự như các app hãy nhìn database để làm

# Danh muc  
class Category(models.Model):  
 # cai id no tu sinh khong can phai khai bao  
 title = models.CharField(default='', max\_length='')  
 slug = models.CharField(max\_length=100, default='')  
 description = models.TextField(default='')  
 active = models.BooleanField(default=True)  
   
  
class Product(models.Model):  
 title = models.CharField(max\_length=255, default='')  
 description = models.TextField(default='')  
 # do category lien ket voi bang Category nen no la khoa ngoai nen khai bao de lien ket  
 category = models.ForeignKey(Category, on\_delete=models.CASCADE)  
 price = models.IntegerField(default=0)  
 active = models.BooleanField(default= True)

class Variation(models.Model):  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 title = models.CharField(max\_length=255, default='')  
 price = models.IntegerField(default=0)  
 sale\_price = models.IntegerField(default=0)  
 inventory = models.IntegerField(default=0)  
 active = models.BooleanField(default=True)

models cua cart

class Cart(models.Model):  
 create\_at = models.DateTimeField(auto\_now\_add=True)  
 update\_at = models.DateTimeField(auto\_now=True)

có 2 cái trường auto\_now và auto\_now\_app khác nhau ở chỗ nào:

auto\_now ví dụ update cuối cùng thì nó lấy ngày hiện tại của hệ thống

auto\_now\_app thì cái trường nó chỉ tạo ra khi nào thôi

CustomerUser khi khách hàng đăng kí chẳng hạn, nên kế thừa từ abstractUser (abstractUser có user và password …)

From Django.contrib.auth.models import AbstractUser

Và thêm AUTH\_USER\_MODEL = ‘user.CustomerUser’ trong fike settings.py

**Khi tạo database trong models xong thì chạy câu lệnh:**

**Câu lênh: python manage.py makemigrations**

Và **python manage.py migrate**

**Từ khóa để tìm templates boostrap bán hàng**

**ecommerce template boostrap free**

<https://uicookies.com/free-bootstrap-ecommerce-website-template/>

Tạo thư mục static, templates thì khai báo vào file setting.py đã tô màu vàng để coi

STATICFILES\_DIRS = [  
 os.path.join(BASE\_DIR, 'static'),  
]

TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [os.path.join(BASE\_DIR, 'templates')],  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]

Khi load static thì thêm {% load static %} vào trong đầu file html

Và sau đó nếu gọi file css hay js gì đó thì href = “{% static ‘tenthumuc/tenduongdanfile’ %}”

Ví dụ:

{% load static %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<title>OneTech</title>  
<meta charset="utf-8">  
<meta http-equiv="X-UA-Compatible" content="IE=edge">  
<meta name="description" content="OneTech shop project">  
<meta name="viewport" content="width=device-width, initial-scale=1">  
<link rel="stylesheet" type="text/css" href="{% static 'homepage/styles/bootstrap4/bootstrap.min.css' %}">  
<link href="plugins/fontawesome-free-5.0.1/css/fontawesome-all.css" rel="stylesheet" type="text/css">  
<link rel="stylesheet" type="text/css" href="plugins/OwlCarousel2-2.2.1/owl.carousel.css">  
<link rel="stylesheet" type="text/css" href="plugins/OwlCarousel2-2.2.1/owl.theme.default.css">  
<link rel="stylesheet" type="text/css" href="plugins/OwlCarousel2-2.2.1/animate.css">  
<link rel="stylesheet" type="text/css" href="plugins/slick-1.8.0/slick.css">  
<link rel="stylesheet" type="text/css" href="styles/main\_styles.css">  
<link rel="stylesheet" type="text/css" href="styles/responsive.css">  
  
</head>

Block này là css

{% block cssblock %}   
{% endblock%}

**……………. Bài học Thêm một trường mới vào database trong Django -------------------------**

Ví dụ thêm trường

class Product(models.Model):  
 title = models.CharField(max\_length=255, default='')  
 description = models.TextField(default='')  
 # do category lien ket voi bang Category nen no la khoa ngoai nen khai bao de lien ket  
 category = models.ForeignKey(Category, on\_delete=models.CASCADE)  
 product\_img = models.CharField(max\_length=255, default='')  
 price = models.IntegerField(default=0)  
 active = models.BooleanField(default=True)

trong migrations nó lưu lại lịch sử tạo models

chạy câu lênh: **python manage.py makemigrations**

**và python manage.py migrate**

tạo tài khoản admin: python manage.py createsuperuser

**……………. Bài học một số lỗi sẽ gặp với migration khi làm việc với model trong Django -------------------------**

* + Khi tồn tại database rồi mà khi thêm vào một trường nào đó thì nên xóa db.sqlite3 để tạo lại (vì trong database đã tồn tại rồi)

**……………. Bài học Cài đặt Django Rest Framework trong Django -------------------------**

Link:

<https://www.django-rest-framework.org/>

pip install djangorestframework

Add 'rest\_framework' to your INSTALLED\_APPS setting.

**……………. Bài học get json từ API Django Rest Framework trong Django -------------------------**

* + Tạo ra app moi

**Câu lệnh: python manage.py startapp course**

* + Cài đặt tên app course này vào **file settings.py** trong demoapi
  + Tạo database class trong model.py và chạy câu lệnh: **python manage.py makemigrations và python manage.py migrate**
  + Tạo tài khoản admin câu lệnh**: python manage.py createsuperuser**
  + Tạo ra file(phần) serializers.py có nhiệm vụ chuyển đổi dữ liệu từ model thành dạng json trả về cho các hệ thống khác, client hoặc cho front end sau đó import serializers vào trong file serializers.py

from rest\_framework import serializers

serializers (giúp lấy dữ liệu từ model chuyển thành dạng json và trả về cho client và cũng có thể nhận dữ liệu từ client dạng json chuyển đổi convert nó thành dạng object của model

* + Sau đó viết nó vào view
* from rest\_framework.views import APIView
* from rest\_framework.response import Response
* from rest\_framework import status

from django.shortcuts import render  
from rest\_framework.views import APIView  
from rest\_framework.response import Response  
from rest\_framework import status  
from .models import Course  
  
  
# Create your views here.  
  
class GetAllCourse(APIView):  
 def get(self, request):  
 list\_course = Course.objects.all()  
 return Response(list\_course, status=status.HTTP\_200\_OK)

list\_course = Course.objects.all()

**lấy về tất cả các khóa học**

Tất cả dữ liệu:

Models.py

from django.db import models  
  
class Course(models.Model):  
 title = models.CharField(max\_length=255)  
 price = models.IntegerField(default=0)  
 content = models.CharField(max\_length=255)

serializers.py

from rest\_framework import serializers  
from .models import Course  
  
class GetAllCourseSerializer(serializers.ModelSerializer):  
 class Meta:  
 models = Course  
 fields = ('id', 'title')

viviews.py

from django.shortcuts import render  
from rest\_framework.views import APIView  
from rest\_framework.response import Response  
from rest\_framework import status  
from .models import Course  
from .serializers import GetAllCourseSerializer  
  
  
# Create your views here.  
  
class GetAllCourseAPI(APIView):  
 def get(self, request):  
 list\_course = Course.objects.all()  
 mydata = GetAllCourseSerializer(list\_course, many=True)  
 return Response(data=mydata.data, status=status.HTTP\_200\_OK)

urls.py

from django.contrib import admin  
from django.urls import path  
from course.views import GetAllCourseAPI  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('course/', GetAllCourseAPI.as\_view()),  
]

postman ở đây đại diện cho một cái điện thoại, đại diện cho một trình duyệt web

trình duyệt của người dùng ở phía client gửi dữ liệu lên server của chúng ta

**\*\*\*\*\*\*\*\*\* Cách test viết api chạy trong post man \*\*\*\*\*\*\*\*\*\*\*\***

client gửi json lên server -> server bắt được dữ liệu và làm gì đó sau đó nó insert trong database