

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**🕯✡🕮🕮✡🕯**

**A logo with hands holding a book and flames

Description automatically generated**

**MÔN HỌC:**

**THỰC TẬP LẬP TRÌNH VỚI PYTHON**

**Đề tài: Sử dụng Django trong ngôn ngữ lập trình**

**Python để thiết kế website**

**GVBM: Huỳnh Hoàng Hà**

**SVTH: MSSV**

**Ngô Trần Quốc Bảo 22139004**

**Phan Nhị Hào 22139019**

**Lớp thứ 3 – Tiết 4-6**

**Mã môn học: PRPL218164\_23\_2\_02**

**Tp. Hồ Chí Minh, tháng 4 năm 2023**

**MỤC LỤC**

[Phần 1: Giới thiệu tổng quan về Django và lí do nên sử dụng Django trong thiết kế web 3](#_Toc167698681)

[1.1 Giới thiệu tổng quan về Django 3](#_Toc167698682)

[1.2 Lí do nên sử dụng Django trong thiết kế web 3](#_Toc167698683)

[Phần 2: Mô hình MVT (Model-View-Template) trong Django và cách hoạt động 5](#_Toc167698684)

[2.1. Sơ lược về mô hình MVT (Model-View-Template) 5](#_Toc167698685)

[**2.1.1. Model (M):** 5](#_Toc167698686)

[**2.1.2. View (V):** 5](#_Toc167698687)

[**2.1.3. Template (T):** 5](#_Toc167698688)

[2.2. Cách Mô hình MVT hoạt động trong Django 5](#_Toc167698689)

[Phần 3: Khởi tạo project Django và sơ qua về framwork 8](#_Toc167698690)

[3.1 Cài đặt Django 8](#_Toc167698691)

[3.2. Tạo project mới với Django: 9](#_Toc167698692)

[3.3. Tạo app và làm việc với app bên trong project của mình 11](#_Toc167698693)

[Phần 4: Tạo và quản lí URL patterns trong Django và điều hướng truy cập trang web trong Django 14](#_Toc167698694)

[4.1. Tạo và quản lí URL Patterns: 14](#_Toc167698695)

[4.2. Điều hướng truy cập trang web trong Django 15](#_Toc167698696)

[Phần 5: Xây dựng giao diện người dùng với Template và View trong Django 16](#_Toc167698697)

[5.1. Template: 16](#_Toc167698698)

[5.2. View: 17](#_Toc167698699)

[5.3. Xây dựng giao diện người dùng với Template và View trong Django 17](#_Toc167698700)

[Phần 6: Thiết kế cơ sở dữ liệu trong Django và liên kết với ứng dụng web 20](#_Toc167698701)

[6.1. Thiết kế cơ sở dữ liệu trong Django 20](#_Toc167698702)

[**6.1.1. Định nghĩa Models** 22](#_Toc167698703)

[**6.1.2. Migrations:** 23](#_Toc167698704)

[**6.1.3. Admin Interface:** 23](#_Toc167698705)

[6.2. Liên kết với ứng dụng web 24](#_Toc167698706)

[Phần 7: Demo tạo một trang web đơn giản bằng Django 25](#_Toc167698707)

# **Phần 1: Giới thiệu tổng quan về Django và lí do nên sử dụng Django trong thiết kế web**

Như chúng ta đã biết, Python là một ngôn ngữ rất mạnh trong lĩnh vực trí tuệ nhân tạo (Artificial Intelligence – AI) cũng như các ứng dụng liên quan. Nhưng có bao giờ các bạn tự hỏi liệu Python có được dùng trong phát triển web hay không? Câu trả lời là có. Python tuy không được sử dụng rộng rãi như PHP nhưng cũng có những framework hỗ trợ mạnh mẽ dùng để thiết kế web như Django, Flask, Pyramid… Trong tài liệu này chúng ta sẽ cùng nhau tìm hiểu về framework Django cũng như xây dựng một mô hình web cơ bản bằng Django.

**1.1** **Giới thiệu tổng quan về Django**

Django là một framework Python cấp cao khá nổi tiếng với đầy đủ các module, thư viện hỗ trợ mạnh mẽ được xây dựng bởi các nhà phát triển có kinh nghiệm. Mặc dù cung cấp mã nguồn mở và miễn phí nhưng nó vẫn có thể xử lí nhiều rắc rối của việc phát triển web, vì vậy bạn có thể tập trung vào viết ứng dụng của mình mà không cần phải tốn công sức để phát triển ứng dụng của mình lại từ những bước nền đầu tiên

Django cho phép chúng ta có thể tái sử dụng lại và có thể tự chạy các component, từ đó giúp việc phát triển web trở nên nhanh chóng và gọn gàng. Ưu điểm của nó là hỗ trợ việc xử lý dữ liệu phía Backend (BE) và cả hệ thống template phía Frontend (FE).

**1.2 Lí do nên sử dụng Django trong thiết kế web**

Với django bạn có thể lấy các ứng dụng web từ ý tưởng để khởi chạy trong vài phút. Và để làm được điều này thì django có một vài tính năng vượt trội như sau:

* Nhanh: Django được thiết kế để giúp các nhà phát triển đưa các ứng dụng từ ý tưởng đến hoàn thành càng nhanh càng tốt. Django giảm thời gian phát triển bằng cách cung cấp các thành phần sẵn có và một quy trình phát triển rõ ràng, nhờ đó tăng năng suất và hiệu quả làm việc.
* Có đầy đủ các thư viện/module cần thiết: Django bao gồm hàng tá các tính năng bổ sung mà bạn có thể sử dụng để xử lý các tác vụ phát triển Web phổ biến. Django chăm sóc xác thực người dùng, quản trị nội dung, bản đồ trang web, nguồn cấp dữ liệu RSS và nhiều tác vụ khác - ngay lập tức. Ngoài ra Django còn cung cấp một hệ thống ORM (Object-Relational Mapping) mạnh mẽ, giúp tương tác với cơ sở dữ liệu một cách dễ dàng và an toàn.
* Đảm bảo về tính bảo mật: Django rất coi trọng vấn đề bảo mật, bằng các tính năng bảo mật tích hợp sẵn nó giúp các nhà phát triển tránh được nhiều lỗi bảo mật phổ biến, chẳng hạn như SQL SQL, kịch bản chéo trang, giả mạo yêu cầu chéo trang và nhấp chuột. Hệ thống xác thực người dùng của nó cung cấp một cách an toàn để quản lý tài khoản và mật khẩu người dùng.
* Khả năng mở rộng tốt: Một số địa điểm bận rộn nhất trên hành tinh sử dụng khả năng có thể mở rộng nhanh chóng và linh hoạt của django để đáp ứng nhu cầu giao thông nặng nhất.
* Tính linh hoạt: Có một cộng đồng lớn của các nhà phát triển sử dụng Django, cung cấp sự hỗ trợ và tài liệu đa dạng, phong phú. Các công ty, tổ chức và chính phủ đã sử dụng Django để xây dựng tất cả mọi thứ - từ hệ thống quản lý nội dung đến mạng xã hội đến nền tảng điện toán khoa học.

# **Phần 2: Mô hình MVT (Model-View-Template) trong Django và cách hoạt động**

## **2.1. Sơ lược về mô hình MVT (Model-View-Template)**

Mô hình **Model-View-Template** (**MVT)** là một biến thể của mô hình MVC (Model-View-Controller) được sử dụng trong Django framework. Trong mô hình này, các thành phần chính bao gồm:

### **2.1.1. Model (M):**

Đại diện cho dữ liệu và logic xử lý dữ liệu. **Models** định nghĩa cấu trúc và quan hệ của dữ liệu trong cơ sở dữ liệu. Nhiệm vụ của nó là thực hiện quản lý dữ liệu. Nghĩa là **Model** sẽ kiểm soát mọi sự thay đổi của dữ liệu cũng như việc giao tiếp CRUD giữa BE và Database.

### **2.1.2. View (V):**

Đại diện cho phần giao diện người dùng và xử lý logic hiển thị. **Views** là các hàm hoặc phương thức trong Django xử lý logic của ứng dụng thông qua các url được định nghĩa và trả về các phản hồi cho người dùng. Khi có một request từ client, **View** sẽ là trung tâm để thực hiện các yêu cầu của request đó và sẽ trả về (response) dữ liệu theo yêu cầu.

### **2.1.3. Template (T):**

Đại diện cho phần hiển thị của ứng dụng thông qua việc kết nối với **View**. Đây chính là phần UI

**Templates** chứa mã HTML với các biến và logic điều kiện để render dữ liệu từ views và hiển thị cho người dùng.

Mô hình MVT tách biệt rõ ràng giữa logic xử lý dữ liệu (**Model**), logic hiển thị (**View**), và phần giao diện (**Template**), giúp tăng tính mở rộng, bảo trì và tái sử dụng của ứng dụng web.

Ưu điểm của nó là giúp mình dễ dàng nắm được tổng thể cách xây dựng một website cũng như giúp mình đáp ứng tiến độ công việc được nhanh hơn.

## **2.2. Cách Mô hình MVT hoạt động trong Django**

A diagram of a computer

Description automatically generated

Trong quá trình hoạt động, khi một request được gửi từ client đến server, Django xác định các hàm trong **View** tương ứng và gọi nó. View xử lý request, tương tác với **Models** để lấy dữ liệu, sau đó trả về dữ liệu đã được render qua **template** để hiển thị cho người dùng.

VD: **Model** *Post* đại diện cho bài đăng trong blog. **View** *home* lấy tất cả các bài đăng từ database và truyền chúng vào **template** để hiển thị. **Template** sử dụng vòng lặp để hiển thị mỗi bài đăng.

# Models

from django.db import models

class Post(models.Model):

title = models.CharField(max\_length=100)

content = models.TextField()

date\_posted = models.DateTimeField(auto\_now\_add=True)

# Views

from django.shortcuts import render

from .models import Post

def home(request):

posts = Post.objects.all()

return render(request, 'blog/home.html', {'posts': posts})

# Templates (‘blog/home.html’)

{% for post in posts %}

<h2>{{ post.title }}</h2>

<p>{{ post.content }}</p>

<p>Posted on {{ post.date\_posted }}</p>

{% endfor %}

# **Phần 3: Khởi tạo project Django và sơ qua về framwork**

## **3.1 Cài đặt Django**

Đầu tiên chúng ta cần cài đặt framework trên máy tính của mình. Ở đây chúng ta sẽ cài đặt thông qua pip install trong python (Lưu ý là bạn sẽ phải cài đặt Python trước)

Bước 1: Mở Command Prompt trên máy tính bằng cách: Nhấn phím **Window** và nhập vào ô tìm kiếm **Command Prompt** hoặc **cmd** sau đó nhấn **Enter** mở

Bước 2: Cài đặt Django bằng cách gõ lệnh pip install django vào trong cmd

A screenshot of a computer

Description automatically generated

Khi cài đặt thành công máy sẽ hiển thị giao diện như thế này:

A screenshot of a computer program

Description automatically generated

Trường hợp máy đã cài đặt Django rồi thì máy sẽ hiển thị như thế này:

A computer screen with white text

Description automatically generated

## **3.2. Tạo project mới với Django:**

Để render ra 1 template với full cấu trúc cho djanngo project thì chúng ta cần chạy lệnh **django-admin startproject <project\_name>** với phần *project\_name* là tên project của mình (Lưu ý là không đặt tên project là django hoặc test)

Ở đây mình sẽ đặt là tên project là python\_web, và sau khi chạy lệnh trên mình sẽ có được 1 folder với cấu trúc sau:

A screenshot of a computer program

Description automatically generated

Những file trong này đều có 1 chức năng riêng và cụ thể nó sẽ như sau:

* ***\_\_init\_\_.py:*** Tập tin này thường được để trống hoặc chứa một số mã Python cơ bản. Khi một gói Python được import, tập tin \_init\_.py sẽ được tự động thực thi, và các đối tượng mà nó định nghĩa sẽ được gắn tên trong không gian tên của gói init\_\_.py.
  + Gói Python (Python package): Một gói Python là một thư mục chứa một tập hợp các module Python liên quan đến nhau. Để Python hiểu rằng một thư mục là một gói Python, bạn cần tạo một tập tin \_init\_.py trong thư mục đó.
* ***asgi.py*** được tạo ra để định nghĩa và cấu hình một ASGI (Asynchronous Server Gateway Interface) application. ASGI là một giao diện được sử dụng để xử lý các yêu cầu HTTP và WebSockets một cách không đồng bộ, phù hợp với các ứng dụng web yêu cầu hiệu suất cao và khả năng mở rộng. ASGI là một giao diện tiêu chuẩn cho các máy chủ web Python để tương tác với các ứng dụng web bất đồng bộ.
* ***settings.py*** chứa tất cả các cấu hình của dự án Django. Đây là một tập tin Python module với các biến cấp module.. Trong file này chứa các setting cơ bản như DEBUG, ALLOWED\_HOSTS, INSTALLED\_APP, DATABASES, ...
* ***urls.py*** là file khai báo các URL của project (kiểu như routing, với địa chỉ nào thì sẽ thực thi hàm nào).
* ***wsgi.py*** là file dùng để deploy project lên server. WSGI (Web Server Gateway Interface) là một giao diện tiêu chuẩn giữa ứng dụng web và máy chủ web trong Python
* ***manage.py*** là file để tạo app, migrate,..., được sử dụng để thực thi các câu lệnh quản lý dự án.

Để làm việc với thư mục này thì bạn cần phải chuyển vào thư mục, chúng ta cần nhập lệnh **cd <tên thư mục>** để trỏ vào thư mục

Bây giờ các bạn có thể chạy lệnh **python manage.py migrate** để áp dụng các thay đổi trong các file migration vào cơ sở dữ liệu, Django sẽ tự động cập nhật dữ liệu của mình sau khi chạy lệnh **migrate**

Sau đó chạy lệnh **python manage.py runserver** và vào link http://127.0.0.1:8000/ để thấy được trang web mình vừa mới tạo.

A screenshot of a computer

Description automatically generated

## **3.3. Tạo app và làm việc với app bên trong project của mình**

Thông thường chúng ta sẽ không làm việc với phần thư mục mặc định của django mà mình sẽ tạo những trang nội dung trong project (thường được gọi là app) và làm việc với nó. Để tạo áp thì mình sẽ sử dụng lệnh **django-admin startapp <tên app>** với tên app là tên app cần tạo. Một trang web có thể có nhiều app nên chúng ta cần đặt tên để dễ phân biệt

Ở đây thì mình sẽ tạo 1 app tên là app1 => câu lệnh của mình sẽ là: **python manage.py startapp app1**. Sau khi thực hiện lệnh thì trong project của mình sẽ được tự động thêm vào 1 folder tên là app1 và cấu trúc folder app1 mới tạo của mình sẽ như sau:

A screenshot of a computer

Description automatically generated

Những file trong này đều có 1 chức năng riêng và cụ thể nó sẽ như sau:

* ***\_\_init\_\_.py:*** Tập tin này thường được để trống hoặc chứa một số mã Python cơ bản. Khi một gói Python được import, tập tin \_init\_.py sẽ được tự động thực thi, và các đối tượng mà nó định nghĩa sẽ được gắn tên trong không gian tên của gói init\_\_.py.
* ***admin.py*** được sử dụng để đăng ký các model để hiển thị trong trang quản trị (admin site). Bằng cách đăng ký model trong file admin.py, bạn có thể tùy chỉnh cách các đối tượng của model được hiển thị và sửa đổi trong trang quản trị.
* ***apps.py*** dùng để cung cấp cấu hình cho app. Trong file này chúng ta có thể định nghĩa các cấu hình cụ thể cho app của mình chẳng hạn như tên ứng dụng, tên hiển thị, hoặc thậm chí cài đặt 1 AppConfig tùy chình
* ***models.py*** là file được dùng để định nghĩa các model, là các lớp Python đại diện cho các bảng trong cơ sở dữ liệu. Các model này xác định cấu trúc dữ liệu được lưu trữ trong cơ sở dữ liệu và định nghĩa các thuộc tính và phương thức để tương tác với dữ liệu. Khi chúng ta tạo hoặc thay đổi các model trong file ***models.py*** thì sẽ cần phải chạy các lệnh **makemigrations** (cú pháp: **python manage.py** **makemigrations**) và **migrate (**cú pháp: **python manage.py migrate)** để áp dụng các thay đổi này vào cơ sở dữ liệu. Điều này giúp Django tự động tạo hoặc cập nhật các bảng trong cơ sở dữ liệu tương ứng với cấu trúc dữ liệu được định nghĩa trong các model.
* ***tests.py*** được sử dụng để viết các unit tests cho ứng dụng đó. Các unit tests này được sử dụng để đảm bảo rằng mã nguồn của ứng dụng hoạt động đúng như mong đợi và không gây ra lỗi khi triển khai. Bằng cách viết các tests trong file ***tests.py***, bạn có thể kiểm tra các phần khác nhau của ứng dụng, bao gồm các model, view, forms và các phần khác, để đảm bảo tính chính xác và độ tin cậy của mã nguồn.
* ***views.py*** được sử dụng để định nghĩa các hàm xử lý yêu cầu từ phía người dùng và trả về các phản hồi tương ứng. Cụ thể, file này chứa các hàm view, là các hàm Python có nhiệm vụ xử lý logic của ứng dụng web, từ việc truy xuất dữ liệu từ cơ sở dữ liệu, xử lý dữ liệu, đến việc render các template và trả về các phản hồi HTTP.

Sau đó bạn cần config lại URL bằng cách tạo thêm file ***urls.py*** trong thư mục app1.

Trong này sẽ chứa những config URL dành cho app1.

A screenshot of a computer

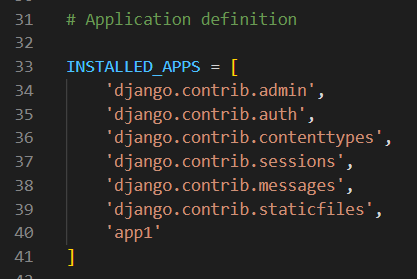
Description automatically generated

Tiếp theo bạn import thêm include từ thư viện **django.urls** thêm đoạn code **path('app1/', include('mypage.urls'))**, sau vào file ***urls.py*** của thư mục python\_web để thực hiện việc trỏ rootURL đến module *app1.urls.*

A screen shot of a computer code

Description automatically generated

Cuối cùng để app1 có thể hoạt động thì chúng ta cần khai báo nó trong **INSTALLED\_APPS** của **settings.py**

****

# **Phần 4: Tạo và quản lí URL patterns trong Django và điều hướng truy cập trang web trong Django**

Trong Django, bạn tạo và quản lí URL patterns thông qua việc định nghĩa các đường dẫn trong file ***urls.py*** của mỗi ứng dụng. Mỗi URL pattern xác định một đường dẫn và liên kết nó với một view cụ thể.

## **4.1. Tạo và quản lí URL Patterns:**

Đầu tiên chúng ta cần phải tạo 1 file urls.py trong ứng dụng của mình. Để tạo file urls.py thì chúng ta nhấp chuột phải vào thư mục app của mình và chọn New File sau đó để tên là urls.py

VD: Ở đây chúng ta sẽ tạo urls.py trong thư mục app1:

A screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Kế đến chúng ta sử dụng module urlpatterns trong file ***urls.py*** để quản lí các URL

* Trước tiên chúng ta cần import các module cần thiết: 2 module mặc định là **from django.contrib import admin** và **from django.urls import path** cùng với **views** trong cùng thư mục với cú pháp **from . import views**
* Mỗi URL pattern được định nghĩa bằng phương thức **path()** hoặc **re\_path()**.

A screen shot of a computer program

Description automatically generated

## **4.2. Điều hướng truy cập trang web trong Django**

Mỗi URL pattern được liên kết với một hàm bên trong **views.py**, nơi tương tác với **Models** để xử lí logic và trả về response.

Django sử dụng các **view** để xác định hành động khi người dùng truy cập các đường dẫn cụ thể.

A screen shot of a computer program

Description automatically generated

Lưu ý: Thực tế khi chạy thì web sẽ có lỗi vì chúng ta chưa có file HTML nào trong thư mục**Templates**, vì vậy cần tạo 2 file HTML có tên là ***home.html*** và ***gioithieu.html*** trong thư mục **Templates** để web có thể hiển thị

# **Phần 5:** **Xây dựng giao diện người dùng với Template và View trong Django**

Trong Django, bạn sử dụng **Template** và **View** để xây dựng giao diện người dùng:

## **5.1. Template:**

**Templates** là các tệp HTML chứa mã để hiển thị dữ liệu cho người dùng.

Django sử dụng hệ thống **template** để tạo các trang web dựa trên dữ liệu được cung cấp bởi **View**.

Bạn có thể sử dụng các mã HTML trong **template**, vòng lặp, và điều kiện để tạo giao diện động.

Để cài đặt **Templates** chúng ta click chuột phải vào thư mục ứng dụng (app) và chọn tạo thư mục mới với tên ***Templates*** hoặc ***templates***

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Trong thư mục Templates, chúng ta tiếp tục tạo một thư mục mới có cùng tên với thư mục ứng dụng (app) của chúng ta:

*Lưu ý là thư mục này phải cùng tên với app chính bên ngoài của chúng ta*:

A screenshot of a computer

Description automatically generated

Trong thư mục này sẽ là nơi chưa các file HTML mà chúng ta sẽ dùng để xây dựng tgiao diện người dùng

A screenshot of a computer

Description automatically generated

## **5.2. View:**

**Views** là các hàm hoặc phương thức lớp xử lý yêu cầu từ người dùng và trả về các trang web được render.

**Views** tương tác với **Model** để lấy dữ liệu và truyền nó vào **Template** để render cho người dùng.

## **5.3. Xây dựng giao diện người dùng với Template và View trong Django**

Tạo Templates:

* Tạo các tập tin HTML để định dạng và hiển thị giao diện người dùng.
* Trong các tập tin template, bạn có thể sử dụng các biến context để hiển thị dữ liệu động từ views.
* Lưu các tập tin template trong thư mục templates trong ứng dụng Django của bạn.

Tạo Views:

* Xây dựng các views để xử lý logic của ứng dụng và trả về dữ liệu cho các templates.
* Các views thường là các hàm Python được định nghĩa trong file views.py của ứng dụng Django.
* Trong các views, bạn có thể truy vấn cơ sở dữ liệu, thực hiện tính toán và chuẩn bị dữ liệu để hiển thị trong templates.

Kết nối Views và Templates:

* Trong views, sử dụng hàm render() để render một template và truyền vào request, tên template và một context dictionary chứa dữ liệu.
* Trong templates, sử dụng các biến context để hiển thị dữ liệu được trả về từ views.
* Đảm bảo rằng các URL patterns trong urls.py đã được liên kết với các views tương ứng.

Hiển thị Giao diện:

* Khi người dùng truy cập một URL được liên kết với view của bạn, Django sẽ gọi view tương ứng để xử lý yêu cầu.
* View sẽ render template và trả về kết quả cho trình duyệt của người dùng, hiển thị giao diện người dùng được tạo ra từ template.

VD:

Trong file ***views.py*** của ứng dụng Django, xác định các views để xử lý các yêu cầu từ người dùng:

A screen shot of a computer screen

Description automatically generated

Tạo các templates HTML trong thư mục **templates/app** của ứng dụng Django:

A screen shot of a computer program

Description automatically generated

Trong file urls.py của ứng dụng Django, liên kết các URLs với các views tương ứng:

A screen shot of a computer program

Description automatically generated

Kiểm tra kết quả:

A white background with black dots

Description automatically generated

# **Phần 6: Thiết kế cơ sở dữ liệu trong Django và liên kết với ứng dụng web**

Trong Django, bạn thiết kế cơ sở dữ liệu bằng cách định nghĩa các **Class** bên trong **Models** (***models.py***) trong ứng dụng của mình. Mỗi **Model** tương ứng với một bảng trong cơ sở dữ liệu và mô tả cấu trúc dữ liệu và quan hệ giữa chúng.

## **6.1. Thiết kế cơ sở dữ liệu trong Django**

Để thiết kế cơ sở dữ liệu trong Django thì trước hết chúng ta cần phải tạo một một tài khoản quản trị cho cơ sở dữ liệu:

Đầu tiên chúng ta chạy lệnh **python manage.py createsuperuser** trong cmd để tạo **superuser**

A black background with yellow and white text

Description automatically generated

Ở dòng Username chúng ta sẽ đặt tên cho tài khoản admin của chúng ta, ở đây sẽ dùng mặc định là admin. Kế đến là nhập email vào dòng tiếp theo

A black background with white text

Description automatically generated

Kế đến chúng ta sẽ tạo password với 2 bước nhập passwrd và nhập lại password (Lưu ý: password khi nhập sẽ không hiển thị nhằm mục đích bảo mật)

A screen shot of a computer program

Description automatically generated

Trường hợp password quá đơn giản sẽ hiện thị ra cảnh báo, nếu muốn sử dụng password này thì gõ “y”, nếu không thì gõ “n” để nhập lại password

A screen shot of a computer

Description automatically generated

Như vậy là chúng ta đã hoàn thành việc tạo tài khoản admin để quản lí cơ sở dữ liệu. Kế đến chúng ta có thể runserver để chạy thử web (dùng lệnh **python manage.py runserver**) và vào đường dẫn <http://127.0.0.1:8000/admin> để truy cập trang đăng nhập vào tài khoản admin của mình

A screenshot of a computer

Description automatically generated

Nhập đúng tài khoản admin và click login để vào trang cơ sở dữ liệu của Django

A screenshot of a computer

Description automatically generated

Tiếp theo sẽ là xây dựng cơ sở dữ liệu trong Django

### **6.1.1. Định nghĩa Models**

Sử dụng các **Class** trong Python để định nghĩa các **Models**.

Mỗi thuộc tính của **Class** đại diện cho một trường trong bảng cơ sở dữ liệu.

Chúng ta có thể sử dụng các trường (fields) như CharField, IntegerField, ForeignKey, và nhiều loại trường khác để xác định các trường trong bảng cơ sở dữ liệu.

**Models** cũng xác định các quan hệ giữa các bảng, bao gồm quan hệ một nhiều, nhiều nhiều và một một.

Lưu ý: Để các Class đã được định nghĩa trong file ***models.py*** có thể hoạt động trong cơ sở dữ liệu thì ta cần khai báo nó vào file ***admin.py*** của app

VD: có 2 class Customer và Product đại diện cho 2 trường cơ sở dữ liệu trong database của ứng dụng

A screen shot of a computer code

Description automatically generated

Để 2 Class trên có thể hoạt động được thì cần đăng kí trong ***admin.py***

A black screen with white text

Description automatically generated

### **6.1.2. Migrations:**

Django sử dụng migrations để đồng bộ hóa cấu trúc cơ sở dữ liệu với các thay đổi trong **model**.

Bạn tạo migrations bằng lệnh **python manage.py makemigrations** và áp dụng chúng bằng lệnh **python manage.py migrate**

### **6.1.3. Admin Interface:**

Django cung cấp một giao diện quản trị tự động dựa trên cấu trúc model bạn định nghĩa.

Chúng ta có thể thêm, sửa đổi và xóa dữ liệu trực tiếp thông qua giao diện này.

VD: Bên dưới là giao diện của 2 trường Customer và Product của ví dụ phía trên A screenshot of a computer

Description automatically generated

## **6.2. Liên kết với ứng dụng web**

**Models** được sử dụng để tương tác với cơ sở dữ liệu trong ứng dụng Django.

**Views** sử dụng **Models** để lấy dữ liệu từ cơ sở dữ liệu và truyền nó vào **Templates** để hiển thị cho người dùng.

# **Phần 7: Demo tạo một trang web đơn giản bằng Django**

<https://drive.google.com/file/d/1dgYYvQH3srfIqyBdZq5MLqzgf_92XyRk/view>

Video tham khảo tạo một trang web bán hàng đơn giản bằng Django:

<https://www.youtube.com/playlist?list=PLUocOGc7RDEIChluBEd5bxuU8Z86-0PyO>