### Crear projecto django

Insertar Código:

django-admin startproject estaticos

### Arquitectura MTV (Model-Template-View) en Django

El patrón **MTV** (Model-Template-View) es la arquitectura que utiliza el framework **Django** para separar la lógica de la aplicación en tres componentes principales, con el fin de facilitar el desarrollo y mantenimiento de aplicaciones web. A continuación se describe cada uno de estos componentes:

#### 1. ****Model (Modelo)****

El **Model** (Modelo) representa la estructura de los datos de la aplicación y su relación con la base de datos. En Django, los modelos son clases que definen los campos y comportamientos de los datos que se almacenarán en la base de datos. Cada modelo se corresponde generalmente con una tabla en la base de datos. Además, los modelos gestionan la lógica de negocio y la validación de los datos.

Ejemplo:

python

Copiar código

from django.db import models

class Articulo(models.Model):

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

contenido = models.TextField()

fecha\_publicacion = models.DateTimeField(auto\_now\_add=True)

#### 2. ****Template (Plantilla)****

El **Template** (Plantilla) es la capa encargada de presentar la información al usuario. Se trata de archivos HTML que definen cómo se estructuran y se visualizan los datos. Los templates en Django pueden incluir etiquetas y filtros de Django que permiten la interacción dinámica con los datos enviados desde el **View** (Vista).

En Django, las plantillas se encargan de definir la interfaz de usuario (UI), utilizando el lenguaje de plantillas de Django, que permite incrustar expresiones de Python dentro de los archivos HTML.

Ejemplo:

html

Copiar código

<!-- archivo: articulo\_detail.html --><h1>{{ articulo.titulo }}</h1><p>{{ articulo.contenido }}</p><p>Publicado el: {{ articulo.fecha\_publicacion }}</p>

#### 3. ****View (Vista)****

La **View** (Vista) es la capa encargada de manejar la lógica de la aplicación, procesando las solicitudes del usuario y devolviendo la respuesta adecuada. En Django, las vistas son funciones o clases que reciben solicitudes HTTP, interactúan con el modelo para obtener o modificar datos y, finalmente, devuelven una respuesta, generalmente en forma de un template renderizado.

Las vistas procesan la lógica del negocio, como la validación de datos, la autenticación de usuarios y la manipulación de información antes de presentarla al usuario.

Ejemplo:

python

Copiar código

from django.shortcuts import renderfrom .models import Articulo

def articulo\_detalle(request, id):

articulo = Articulo.objects.get(id=id)

return render(request, 'articulo\_detail.html', {'articulo': articulo})

### Flujo de trabajo en la arquitectura MTV

1. **El usuario realiza una solicitud** (por ejemplo, accede a una página específica de la aplicación).
2. **La vista (View)** recibe esta solicitud, recupera los datos necesarios del modelo (Model) o realiza cualquier lógica de negocio.
3. **El modelo (Model)** proporciona los datos solicitados, realizando consultas a la base de datos si es necesario.
4. **La vista (View)** utiliza el template correspondiente para generar la respuesta HTML.
5. **El template (Template)** recibe los datos proporcionados por la vista y los presenta al usuario en un formato legible.

### Diferencias con MVC

Aunque la arquitectura MTV de Django se basa en principios similares al patrón MVC (Modelo-Vista-Controlador), existe una diferencia clave en el nombre de las capas. En **Django**, la capa **"View"** es la que controla el flujo de datos y la lógica de la aplicación, mientras que en **MVC**, la **"Controller"** se encarga de esta tarea. De esta manera, Django usa una terminología ligeramente diferente, pero la separación de responsabilidades sigue siendo similar.

### Crear migración

Copiar código

py manage.py migrate

Crear vista nombre del archivo “views.py”

Insetar código :

from django.http import HttpResponse

def saludo(request):

return HttpResponse("hello")

def despedida(request):

    return HttpResponse("good bye")

Configuración de rutas

Insetar Código:

from django.contrib import admin

from django.urls import path

from . import views

urlpatterns = [

    path("admin/", admin.site.urls),

    path("saludo/", views.saludo, name="saludo"),

    path("despedida/", views.despedida, name="despedida"),

]

Configuración con int en las rutas

Insertar Código en django\my\_firt\_project\my\_firt\_project\views.py

from django.http import HttpResponse

def saludo(request):

    return HttpResponse("hello")

def despedida(request):

    return HttpResponse("good bye")

def adulto(request, edad):

    if edad >= 18:

        return HttpResponse("Eres mayor de edad")

    else:

        return HttpResponse("No eres mayor de edad")

Insertar Código en django\my\_firt\_project\my\_firt\_project\urls.py

from django.contrib import admin

from django.urls import path

from . import views

urlpatterns = [

    path("admin/", admin.site.urls),

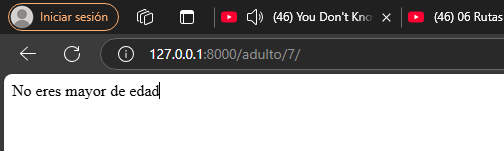
    path("saludo/", views.saludo, name="saludo"),

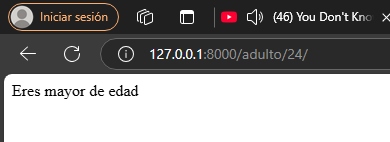
    path("despedida/", views.despedida, name="despedida"),

    path("adulto/<int:edad>/", views.adulto, name="adulto"),

]

Testear resultado





# Projecto template

Crear proyecto:

Inserta código en terminal:

django-admin startproject templates

Crear archivo django\templates\templates\views.py e insertar código:

from django.shortcuts import render

def simple(request):

    return render(request, "home.html", {})

Configurar archivo django\templates\templates\settings.py:

TEMPLATES = [

    {

        "BACKEND": "django.template.backends.django.DjangoTemplates",

        "DIRS": ["theme"],

        "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",

            ],

        },

    },

]

Crear carpeta theme y archvio home.html django\templates\theme\home.html e insertar código:

Insertar código:

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>Pagina web</h1>

</body>

</html>

Y por último configurar django\templates\templates\urls.py para gestionar lo que ingresa el usuario en la url del navegador.

Insertar código:

from django.contrib import admin

from django.urls import path

from . import views

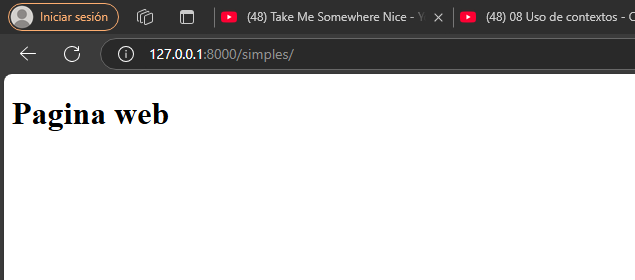
urlpatterns = [

    path("admin/", admin.site.urls),

    path("simples/", views.simple, name="simple"),

]

Resultado



# Uso de contexto.

El uso de contexto es para pasar uno o mas datos en array.

Insertar código en “templates\templates\urls.py”

Código:

from django.contrib import admin

from django.urls import path

from . import views

urlpatterns = [

    path("admin/", admin.site.urls),

    path("simples/", views.simple, name="simple"),

*# uso de srting para pasar contexto.*

    path("dinamico/<str:name>", views.dinamico, name="dinamico"),

]

Crerar archivo de plantilla en templates\theme\dinamico.html

Insertar Código:

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>Hola {{name}}</h1>

    <h3>Categorias</h3>

    <p>{{categories}}</p>

</body>

</html>

Ahoira configurar la función dinamico en templates\templates\views.py

Insertar Código:

from django.shortcuts import render

def simple(request):

    return render(request, "home.html", {})

def dinamico(request, name):

    context = {"name": name}

    return render(request, "dinamico.html", context)

# Bucles y condicionales en plantillas

En la plantilla se muestrar los datos dinamicamente al usuario en el archivo templates\theme\dinamico.html

**Insetar Código:**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>Hola {{name}}</h1>

    <h2>Categorías disponibles:</h2>

    <ul>

        {% for category in categories %}

            <li>{{ category }}</li>

        {% endfor %}

    </ul>

</body>

</html>

Para agregar condición de si es ‘code’ marca con negrita el dato en el archivo templates\theme\dinamico.html

Insertar Código:

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>Hola {{name}}</h1>

    <h2>Categorías disponibles:</h2>

    <ul>

        {% for category in categories %}

            {% if category == 'code' %}

                <li><b>{{category}}</b></li>

            {% else %}

                <li>{{ category }}</li>

            {% endif %}

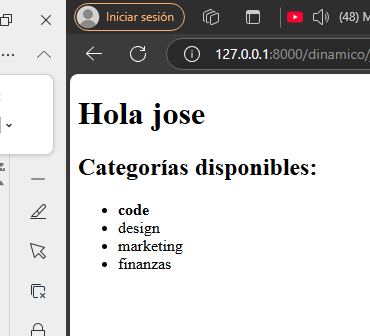
        {% endfor %}

    </ul>

</body>

</html>

Resultado:



# Comentarios y filtros

Puedes comentar de la siguiente manera en django

Insertar Código:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>Hola {{name}}</h1>

    <h2>Categorías disponibles:</h2>

    <ul>

*{#Esto es un comentario de una sola linea#}*

*{% comment %}*

*Esto es un comentario en una plantilla de Django.*

*No se mostrará en la salida renderizada.*

*{% endcomment %}*

{% for category in categories %}

            {% if category == 'code' %}

                <li><b>{{category}}</b></li>

            {% else %}

                <li>{{ category }}</li>

            {% endif %}

        {% endfor %}

    </ul>

</body>

</html>

Para el manejo de filtros, pueder mostar el total de categorias o toda una frase en mayusculas usando

length y upper

Insertar Código:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>Hola {{name}}</h1>

    <h2>Categorías disponibles:</h2>

    <ul>

*{#Esto es un comentario de una sola linea#}*

*{% comment %}*

*Esto es un comentario en una plantilla de Django.*

*No se mostrará en la salida renderizada.*

*{% endcomment %}*

{% for category in categories %}

            {% if category == 'code' %}

                <li><b>{{category|upper}}</b></li>

            {% else %}

                <li>{{ category }}</li>

            {% endif %}

        {% endfor %}

        <h5>Categorías totales: {{ categories|length }}</h5>

    </ul>

</body>

</html>