**Conceptos Fundamentales de Django**

1. **Modelos**: Django utiliza un ORM (Object-Relational Mapper) para interactuar con la base de datos. Los modelos son clases de Python que representan las tablas de la base de datos y sus relaciones.
2. **Vistas**: Las vistas son funciones o clases de Python que procesan las solicitudes HTTP y devuelven respuestas. Pueden interactuar con los modelos para recuperar o almacenar datos en la base de datos.
3. **Plantillas**: Las plantillas son archivos HTML que contienen marcadores de Django (etiquetas, filtros y variables) que se utilizan para renderizar el contenido dinámico.
4. **Rutas (URLs)**: Las rutas mapean las URLs de la aplicación a las vistas correspondientes. Se definen en el archivo urls.py de cada aplicación.
5. **Admin**: Django proporciona una interfaz de administración preconstruida que permite a los administradores gestionar los datos del sitio web a través de una interfaz gráfica de usuario.
6. **Formularios**: Los formularios en Django se utilizan para recopilar datos del usuario. Pueden ser generados automáticamente a partir de modelos o definidos manualmente.

**Ejemplos de Django**

**Creación de un Modelo**

python

from django.db import models

class Producto(models.Model):

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

precio = models.DecimalField(max\_digits=10, decimal\_places=2)

**Creación de una Vista**

python

from django.shortcuts import render

from .models import Producto

def lista\_productos(request):

productos = Producto.objects.all()

return render(request, 'productos/lista\_productos.html', {'productos': productos})

**Creación de una Plantilla**

html

<!-- templates/productos/lista\_productos.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Lista de Productos</title>

</head>

<body>

<h1>Lista de Productos</h1>

<ul>

{% for producto in productos %}

<li>{{ producto.nombre }} - {{ producto.precio }}</li>

{% endfor %}

</ul>

</body>

</html>

**Aplicaciones de Django**

1. **Sitios Web Dinámicos**: Django es ideal para construir sitios web dinámicos con contenido generado dinámicamente a partir de una base de datos.
2. **Plataformas de Comercio Electrónico**: Puedes utilizar Django para construir plataformas de comercio electrónico completas con catálogos de productos, carritos de compras y procesamiento de pagos.
3. **Aplicaciones Web Empresariales**: Django es una excelente opción para construir aplicaciones web empresariales que gestionan datos complejos y requieren un alto nivel de seguridad y escalabilidad.
4. **APIs Web**: Django puede ser utilizado para construir APIs web para servir datos a aplicaciones móviles, aplicaciones de escritorio o aplicaciones web de una sola página (SPA).
5. **Blogs y CMS**: Django es adecuado para construir blogs y sistemas de gestión de contenido (CMS) personalizados con características como gestión de usuarios, publicación de contenido y comentarios.

### Paso 1: Instalar Django

Primero, asegúrate de tener Python instalado en tu sistema. Luego, puedes instalar Django utilizando pip, el gestor de paquetes de Python:

pip install django

### Paso 2: Crear un Proyecto Django

Abre tu terminal o símbolo del sistema y navega hasta el directorio donde quieres crear tu proyecto Django. Luego, ejecuta el siguiente comando para crear un nuevo proyecto Django:

django-admin startproject mi\_proyecto

Esto creará un directorio llamado mi\_proyecto con la estructura básica de archivos para tu proyecto Django.

### Paso 3: Crear una Aplicación Django

Dentro del directorio mi\_proyecto, ejecuta el siguiente comando para crear una nueva aplicación Django:

python manage.py startapp mi\_aplicacion

Esto creará un directorio llamado mi\_aplicacion dentro de tu proyecto Django con la estructura básica de archivos para tu aplicación.

### Paso 4: Definir Modelos

Abre el archivo models.py dentro del directorio de tu aplicación (mi\_aplicacion) y define los modelos que necesitas para tu aplicación. Por ejemplo:

python

from django.db import models

class Producto(models.Model):

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

precio = models.DecimalField(max\_digits=10, decimal\_places=2)

def \_\_str\_\_(self):

return self.nombre

### Paso 5: Crear Migraciones y Aplicarlas a la Base de Datos

Django utiliza migraciones para gestionar los cambios en la estructura de la base de datos. Ejecuta los siguientes comandos para crear las migraciones y aplicarlas a la base de datos:

python manage.py makemigrations

python manage.py migrate

### Paso 6: Crear Vistas

Abre el archivo views.py dentro del directorio de tu aplicación (mi\_aplicacion) y define las vistas que necesitas para tu aplicación. Por ejemplo:

python

from django.shortcuts import render

from .models import Producto

def lista\_productos(request):

productos = Producto.objects.all()

return render(request, 'mi\_aplicacion/lista\_productos.html', {'productos': productos})

### Paso 7: Crear Plantillas HTML

Crea un directorio llamado templates dentro del directorio de tu aplicación (mi\_aplicacion). Dentro de este directorio, crea archivos HTML para tus plantillas.

### Paso 8: Definir URLs

Crea un archivo llamado urls.py dentro del directorio de tu aplicación (mi\_aplicacion) y define las URLs de tu aplicación.

### Paso 9: Ejecutar el Servidor de Desarrollo

Ejecuta el siguiente comando para iniciar el servidor de desarrollo de Django:

python manage.py runserver

Esto iniciará el servidor de desarrollo en tu máquina local. Puedes acceder a tu sitio web visitando la dirección http://127.0.0.1:8000 en tu navegador web.