

# Trabajando con Django

Django es un framework de desarrollo web de código abierto y gratuito, escrito en el lenguaje de programación Python. Te permite crear aplicaciones web dinámicas y complejas de manera rápida y eficiente.

## ¿Cómo funciona Django?

Django funciona siguiendo un patrón de arquitectura llamado Modelo-Vista-Plantilla (MVT), el cual divide la aplicación en tres partes bien diferenciadas:

### 1. Modelo (Model):

Representa la estructura de datos de la aplicación, definiendo los elementos con los que se trabajará, como usuarios, productos, pedidos, etc.

Se encarga de interactuar con la base de datos, gestionando la creación, lectura, actualización y eliminación de datos (CRUD).

Utiliza un Mapeador Objeto-Relacional (ORM) para simplificar la interacción con la base de datos, permitiendo trabajar con objetos en lenguaje Python en lugar de consultas SQL directas.

### 2. Vista (View):

Controla la lógica de la aplicación y actúa como intermediario entre el modelo y la plantilla.

Recibe las solicitudes del usuario (por ejemplo, a través de una URL) y consulta al modelo para obtener los datos necesarios.

Procesa y prepara los datos recuperados del modelo para que sean presentados de manera adecuada en la plantilla.

No se encarga de la presentación en sí, sino que delega esa tarea a la plantilla.

### 3. Plantilla (Template):

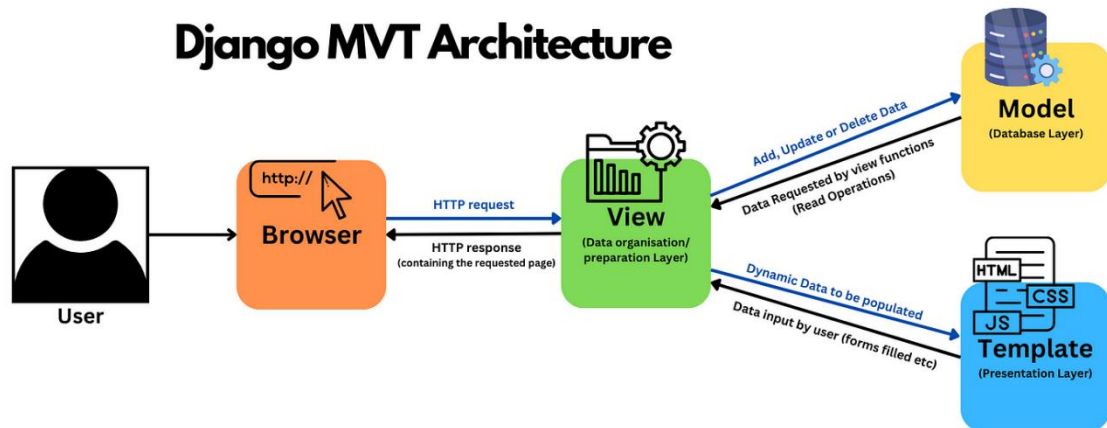
Define la presentación de la información al usuario, utilizando HTML, CSS y el lenguaje de plantillas de Django.

Recibe los datos preparados por la vista y los inserta en la estructura HTML de la plantilla.

Genera la respuesta HTML final que se envía al navegador del usuario.

Permite crear interfaces de usuario dinámicas y personalizadas, utilizando variables, lógica condicional y bucles para mostrar los datos de manera flexible.

## ¿Cómo interactúan los componentes MVT en Django?



- ✓ **Solicitud del usuario:** El usuario realiza una acción, como visitar una URL o enviar un formulario.
- ✓ **Enrutamiento a la vista:** Django identifica la URL solicitada y la asocia con una vista específica.
- ✓ **Ejecución de la vista:** La vista correspondiente se ejecuta, recuperando datos del modelo mediante el ORM.
- ✓ **Preparación de datos:** La vista procesa y formatea los datos recuperados del modelo para su presentación.
- ✓ **Renderizado de la plantilla:** La vista selecciona la plantilla adecuada y le pasa los datos preparados.
- ✓ **Generación de la respuesta:** La plantilla genera la respuesta HTML final utilizando los datos proporcionados.
- ✓ **Envío de la respuesta:** Django envía la respuesta HTML generada al navegador del usuario.

### Beneficios de la arquitectura MVT:

- ✓ **Separación de preocupaciones:** Cada componente (modelo, vista, plantilla) se encarga de una tarea específica, lo que facilita la comprensión, el mantenimiento y la reutilización del código.
- ✓ **Flexibilidad y escalabilidad:** La arquitectura MVT permite crear aplicaciones web complejas y escalables, adaptándose a diferentes necesidades y flujos de trabajo.
- ✓ **Desarrollo rápido:** Django ofrece herramientas y componentes que agilizan el desarrollo de aplicaciones web, reduciendo el tiempo y esfuerzo necesario.

Django proporciona una serie de herramientas y componentes que facilitan el desarrollo de aplicaciones web, incluyendo:

- ✓ **Un sistema de administración de contenidos (CMS)** integrado, que te permite crear y administrar fácilmente el contenido de tu sitio web.
- ✓ **Un sistema de autenticación y autorización**, que te ayuda a proteger tu sitio web de accesos no autorizados.
- ✓ **Un sistema de plantillas**, que te permite crear interfaces de usuario dinámicas y personalizadas.
- ✓ **Una amplia biblioteca de funciones**, que te ayudan a realizar tareas comunes como enviar correos electrónicos, procesar formularios y administrar archivos.

#### ¿Qué ventajas tiene usar Django?

- ✓ **Desarrollo rápido:** Django te permite crear aplicaciones web de manera rápida y eficiente, gracias a su arquitectura modular y a su amplia gama de herramientas y componentes.
- ✓ **Escalabilidad:** Django es lo suficientemente escalable como para manejar desde pequeños sitios web hasta grandes aplicaciones empresariales.
- ✓ **Seguridad:** Django te ayuda a proteger tu sitio web de vulnerabilidades de seguridad comunes, como la inyección SQL y los ataques de tipo cross-site scripting (XSS).
- ✓ **Flexibilidad:** Django es un framework muy flexible que te permite crear una amplia variedad de aplicaciones web.
- ✓ **Comunidad grande y activa:** Django cuenta con una comunidad grande y activa de desarrolladores que pueden ayudarte a resolver problemas y a aprender a usar el framework.

#### ¿Qué tipo de aplicaciones web puedo crear con Django?

Django se puede usar para crear una amplia variedad de aplicaciones web, como:

- Sitios web de noticias
- Tiendas en línea
- Redes sociales
- Aplicaciones de gestión de proyectos
- Foros en línea
- Blogs
- Y mucho más

En resumen, Django es una herramienta poderosa y versátil que te permite crear aplicaciones web dinámicas y complejas de manera rápida y eficiente. Si estás buscando un framework de desarrollo web para tu próximo proyecto, Django es una excelente opción.

#### Diferencias entre mvc y mvt



MVC y MVT son dos patrones de arquitectura para desarrollo web, con similitudes y diferencias clave:

#### Similitudes:

**Separación de responsabilidades:** Dividen la aplicación en 3 capas:

- ✓ **Modelo:** Datos de la aplicación.
- ✓ **Vista:** Lógica de la aplicación e interacción con el usuario.
- ✓ **Controlador/Plantilla:** Presentación de los datos (Controlador en MVC, Plantilla en MVT).

**Objetivo:** Crear aplicaciones organizadas, mantenibles y escalables.

#### Diferencias:

##### 1. Rol del Controlador/Plantilla:

##### MVC:

- ✓ El controlador orquesta la interacción entre modelo y vista.

- ✓ La vista se enfoca en la presentación (plantillas, etc.).
- MVT:**
- ✓ La plantilla tiene un rol más protagonista en la presentación.
  - ✓ La vista es más simple, la lógica de presentación se divide entre vista y plantilla.
  - ✓ No se utiliza un controlador explícito.

## 2. Enfoque en la presentación:

- **MVC:** La vista se centra en la presentación, la lógica compleja se maneja en el controlador o capas de lógica de negocios.
- **MVT:** La plantilla tiene un papel más activo en la presentación, asumiendo algunas responsabilidades del controlador.

## 3. Implementación en frameworks:

- **MVC:** Frameworks como Ruby on Rails o Spring MVC implementan un controlador explícito.
- **MVT:** Frameworks como Django o Django Girls no utilizan un controlador explícito.

## En resumen:

- **MVC:** Mayor separación de responsabilidades, útil para proyectos complejos con mucha lógica de presentación.
- **MVT:** Código más conciso, ideal para proyectos que priorizan la simplicidad y la rapidez de desarrollo.

**¿Cuál elegir?** Depende de las preferencias del desarrollador y las características del proyecto. En el caso de Django, se le considera un framework MVT.

**Nota:** Esta distinción no afecta el uso práctico del framework, ya que los principios fundamentales de la separación de responsabilidades y la organización del código siguen presentes.

## Instalación de Django

Primeramente, debemos tener instalado Python, verificamos digitando Python en nuestro terminal, si esta instalado nos mostrara la versión, luego instalamos django “**pip3 install Django==5.0.6**”

```

PS C:\xampp\htdocs\Trabajando con Django> pip3 install Django==5.0.6
Defaulting to user installation because normal site-packages is not writeable
Collecting Django==5.0.6
  Downloading Django-5.0.6-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref<4,>=3.7.0 (from Django==5.0.6)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from Django==5.0.6)
  Downloading sqlparse-0.5.0-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata (from Django==5.0.6)
  Downloading tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading Django-5.0.6-py3-none-any.whl (8.2 MB)
   8.2/8.2 MB 3.7 MB/s eta 0:00:00
Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
Downloading sqlparse-0.5.0-py3-none-any.whl (43 kB)
   44.0/44.0 kB 1.1 MB/s eta 0:00:00
Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)
   345.4/345.4 kB 4.2 MB/s eta 0:00:00
Installing collected packages: tzdata, sqlparse, asgiref, Django
WARNING: The script sqlformat.exe is installed in 'C:\Users\EDMAR\AppData\Local\Pack
ages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Py
thon312\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning,
  use --no-warn-script-location.
WARNING: The script django-admin.exe is installed in 'C:\Users\EDMAR\AppData\Local\
Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages
\Python312\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning,
  use --no-warn-script-location.
Successfully installed Django-5.0.6 asgiref-3.8.1 sqlparse-0.5.0 tzdata-2024.1
PS C:\xampp\htdocs\Trabajando con Django>
  
```

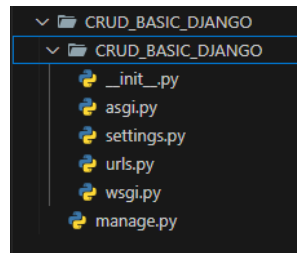
## Creamos un Proyecto

Con el comando “python -m django startproject NOMBRE\_PROYECTO”

```

PS C:\xampp\htdocs\Trabajando con Django> python -m django startproject CRUD_BASIC_DJ
ANGO
PS C:\xampp\htdocs\Trabajando con Django>
  
```

Lo que automáticamente crea el proyecto



Ahora correremos un servidor para mostrar nuestro proyecto, con el comando **"python3 manage.py runserver"**

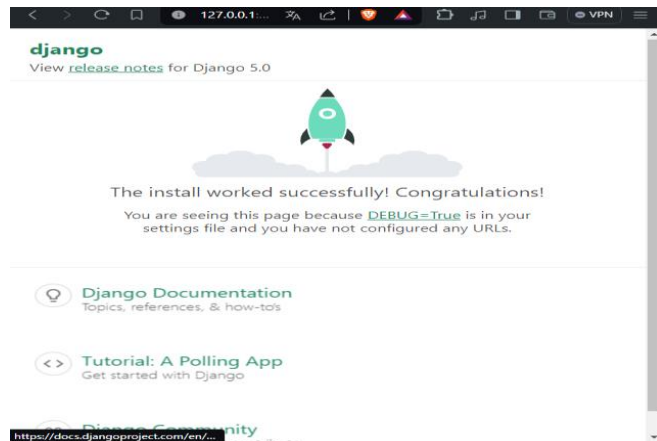
```
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO> python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the
migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 18, 2024 - 10:13:49
Django version 5.0.6, using settings 'CRUD_BASIC_DJANGO.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[18/May/2024 10:14:02] "GET / HTTP/1.1" 200 10629
Not Found: /favicon.ico
[18/May/2024 10:14:02] "GET /favicon.ico HTTP/1.1" 404 2121
```

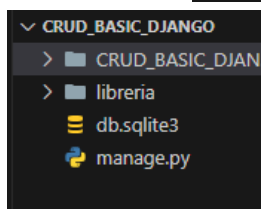
Nos dará nuestra ruta de url en este caso <http://127.0.0.1:8000> el cual nos mostrará nuestro proyecto



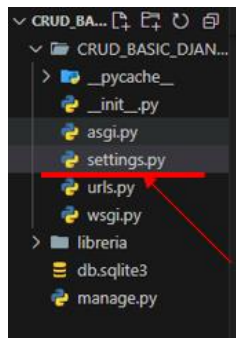
## Creación de una App

En la consola digitamos el siguiente comando **"python3 manage.py startapp NOMBREDELA\_APP"**

```
TERMINAL ... powershell + v [ ] [ ] ...
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO>
python3 manage.py startapp libreria
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO>
```

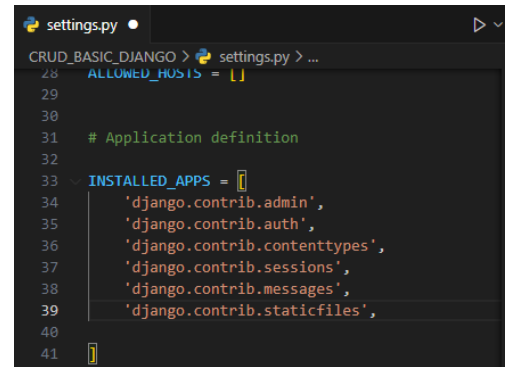


se nos crea automáticamente una carpeta con el nombre de la app que le pusimos anteriormente.

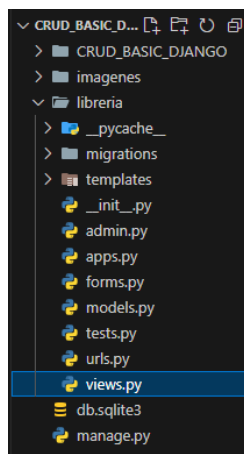


- Después de haber creado la app debemos registrarlo en nuestro proyecto para esto debemos ingresar a `settings.py` que se encuentra dentro del proyecto.

dentro de este archivo en la línea 33 encontraremos todas las apps que están por default en Django. aquí registraremos nuestra app y todas las apps que creamos.

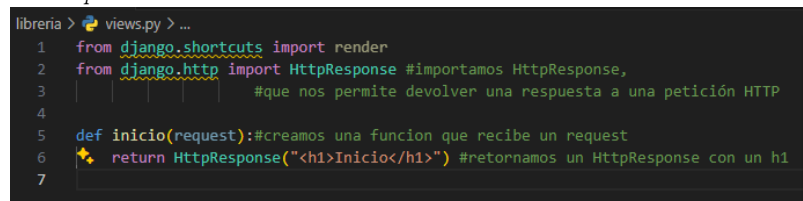


## Creamos una Vista

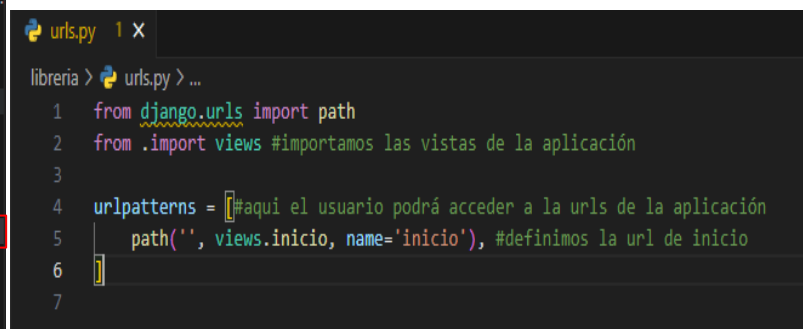
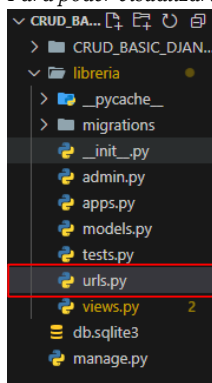


Dentro de nuestra app buscamos el archivo `views.py` el que se va a encargar de todas las vistas.

--haremos una pequeña prueba para ver como se comporta la pagina con la vista , para esto creamos una función en el cual imprimiremos un texto dentro de un `h1`.



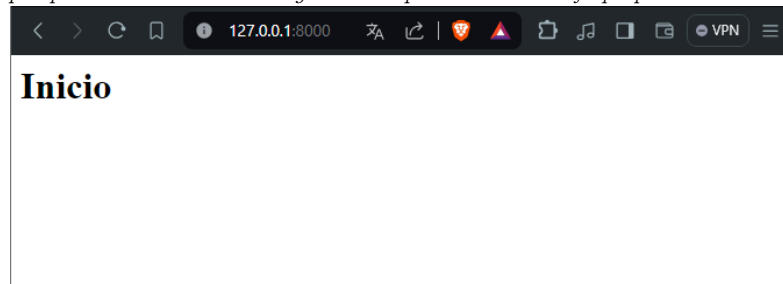
Para poder visualizarlo en la web, lo que realizaremos es crear un archivo `urls.py` dentro de nuestra app



Ahora agregamos las urls de nuestra app al proyecto para esto en el archivo `urls.py` del proyecto agregamos:

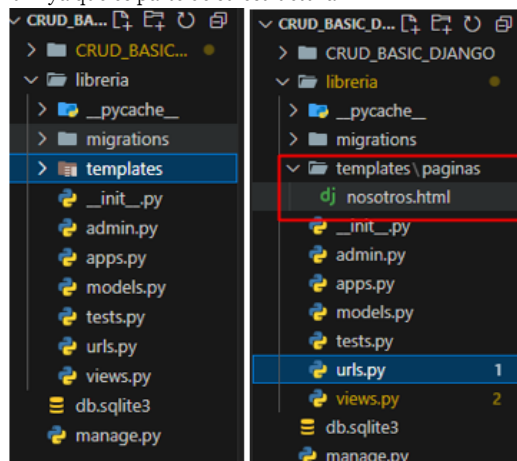
```
CRUD_BASIC_DJANGO > urls.py > ...
9      2. Add a URL to urlpatterns: path('', views.home, name='home')
10     Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13     Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16     """
17     from django.contrib import admin
18     from django.urls import path
19     from django.urls import include #importamos include,
20     #que nos permite incluir urls de otras aplicaciones
21
22     urlpatterns = [
23         path('admin/', admin.site.urls),
24         path('', include('libreria.urls')), #incluimos las urls de la aplicación
25     ]
```

Con estos simples pasos al actualizar el navegador nos aparecerá el mensaje que pusimos en la vista de la app



### Ahora crearemos las vistas en html

Crearemos una carpeta llamada templates, django sabe que dentro de esta carpeta se encuentran todas las vistas html ya que es parte de su estructura



dentro de la carpeta templates creamos una carpeta llamada paginas en el cual creamos un archivo html que lo mostraremos mas adelante.

Modificaremos el archivo `nosotros.html`

```
libreria > templates > paginas > dj_nosotros.html
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-s
6      <title>Nosotros</title>
7  </head>
8  <body>
9      <h1>Nosotros</h1>
10     <p>Somos una empresa dedicada a la venta de libros.</p>
11 </body>
12 </html>
```



Ahora en el archivos `views.py` creamos una funcion para mostrar las paginas que creamos.

```
libreria > views.py > ...
1 from django.shortcuts import render
2 from django.http import HttpResponse #importamos HttpResponse,
3 #que nos permite devolver una respuesta a un
4
5 def inicio(request):#creamos una funcion que recibe un request
6     return HttpResponse("<h1>Inicio</h1>") #retornamos un HttpRe
7
8 def nosotros(request):
9     return render(request, "paginas/nosotros.html") #retornamos
```

Ahora en `urls.py`

```
libreria > urls.py > ...
1 from django.urls import path
2 from . import views #importamos las vistas de la aplicación
3
4 urlpatterns = [#aqui el usuario podrá acceder a la urls de la ap
5     path('', views.inicio, name='inicio'), #definimos la url de
6     path('nosotros/', views.nosotros, name='nosotros'), #definim
7
8
```

Lo que nos hace referencia que cada ves que pongamos en la barra de navegacion **`nosotros/`** nos mostrara la pagina `nosotros.html`

Creamos una Plantilla base, con la finalidad que se esta plantilla se muestre en todas las rutas que realicemos

```
libreria > templates > dj base.html
1 {% load static %} <!-- load static es para cargar archivos estat
2 <!doctype html>
3 <html lang="en">
4 <head>
5     <title>{% block titulo %} {% endblock titulo %}</title><
6     <!-- Required meta tags -->
7     <meta charset="utf-8" />
8     <meta
9         name="viewport"
10         content="width=device-width, initial-scale=1, shrink
11     />
12
13     <!-- Bootstrap CSS v5.2.1 -->
14     <link
15         href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/c
16         rel="stylesheet"
17         integrity="sha384-T3c6CoIi6uLrA9TneEoa7RxnatzjcDSCm
18         crossorigin="anonymous"
19     />
20 </head>
```

En el cual agregaremos al inicio `{% load static %}` lo que dira que la plantilla sera estatica, Luego en titulo `{% block titulo %}{% endblock titulo %}` aqui se agregara el tittulo de cada pagina y para el contenido `{%block contenido %}{%endblock contenido%}` al igual en este bloque se agregara el contenido de la pagina

Ahora como mostraremos los datos?

Modificaremos el archivo **`nosotros.html`** con la finalidad de que pueda interactuar con la plantilla base.

```
libreria > templates > paginas > dj nosotros.html
1 {% extends 'base.html' %} <!--sirve para extender de la plantill
2 {% block titulo%} Nosotros {% endblock titulo %} <!--sirve para
3 {% block contenido %} <!--sirve para sobrescribir el contenido
4 <h1>Nosotros</h1>
5 <p>Somos una empresa dedicada a la venta de libros.</p>
6 {% endblock contenido %} <!--sirve para sobrescribir el conteni
7
```

`{%extends 'base.html'%}` lo que hace es llamar a la plantilla base

Y en cada block enviaremos los datos los cuales seran remplazados en la plantilla base.

Dando como resultado:



## Inicio

Esta es la pagina de inicio.

Ahora agregaremos una linea de codigo, que cuando seleccionemos cualquier enlace de la pagina como inicio, libros o nosotros se active para esto hacemos una validacion

```
<div class="collapse navbar-collapse" id="collapsibleNavId">
  <ul class="navbar-nav me-auto mt-2 mt-lg-0">
    <li class="nav-item">
      <a class="nav-link {% if request.path == '/' %}active{% endif %}" href="{% url 'inicio' %}" aria-current="page">Inicio
      <span class="visually-hidden"{{current}}</span></a>
      <!-- el siguiente codigo {% if request.path == '/' %}active{% endif %} lo que hace es que si la url es
           igual a la que se esta comparando se le agrega la clase active -->
    </li>
    <li class="nav-item">
      <a class="nav-link {% if request.path == '/libros/' %}active{% endif %}" href="{% url 'libros' %}">libros</a>
    </li>
    <li class="nav-item">
      <a class="nav-link {% if request.path == '/nosotros/' %}active{% endif %}" href="{% url 'nosotros' %}">Nosotros</a>
    </li>
  </ul>
</div>
```

Ahora empezaremos a crear las vistas para el crud de los libros

**Pagina para listar los datos:**



Solo mantenemos datos estaticos.

```
libreria > templates > paginas > libros > dj index.html
1  {% extends 'base.html' %} <!--sirve para extender de la plantilla base.html-->
2  {% block titulo%} Libros {% endblock titulo %} <!--sirve para sobrescribir el contenido de la etiqueta title-->
3  {% block contenido %} <!--sirve para sobrescribir el contenido de la etiqueta body-->
4  <div class="container">
5    <div class="row justify-content-center align-items-center g-2">
6      <h1>Lista de los libros </h1>
7      <div class="col-4">
8        <a name="" id="" class="btn btn-primary" href="{% url 'crear' %}" role="button" >Nuevo Libro</a>
9      </div>
10     <div class="col-8"></div>
11     <div class="table-responsive">
12       <table class="table table-primary" >
13         <thead>
14           <tr>
15             <th scope="col">#</th>
16             <th scope="col">Titulo</th>
17             <th scope="col">Portada</th>
18             <th scope="col">Autor</th>
19             <th scope="col">Editorial</th>
20             <th scope="col">Fecha de publicacion</th>
21             <th scope="col">Accion</th>
22           </tr>
23         </thead>
24         <tbody>
25           <tr>
26             <th scope="row">1</th>
27             <td>El principito</td>
28             <td></td>
29             <td>Antoine de Saint-Exupéry</td>
30             <td>Reynal & Hitchcock</td>
31             <td>6 de abril de 1943</td>
```

Ahora en el archivo views.py creamos una funcion para crear, en la cual le definiremos que archivo debemos mostrar.

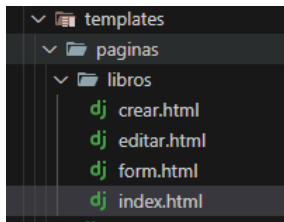
```
13 def crear(request):
14     return render(request, "paginas/libros/crear.html") #retornamos un render con la plantilla crear.html
```

Y en urls.py definimos el url de la pagina

```
path('libros/crear/', views.crear, name='crear'), #definimos la url de crear
```

Crearemos un formulario el cual se utilizara para crear y para actualizar los datos, en el archivo form.html





dentro de la carpeta libros creamos el archivo form.html y escribimos el siguiente código

```

1 <form enctype="multipart/form-data" method="post">
2 <!--django utiliza un token con la finalidad de evitar ataques csrf-->
3 {% csrf_token %}
4 {% for campo in formulario %} <!--recorremos el formulario enviado desde la vista-->
5 <div class="mb-3">
6 <label for="" class="form-label">{{campo.label}}</label><br>
7 <div class="form-control">
8 <input type="text" class="form-control" name="{{campo.name}}" id="{{campo.id}}"/>
9 </div>
10 </div>
11 <div class="col-12 help-text">{{campo.errors}}</div>
12 </div>
13 <div class="d-grid gap-2">
14 <input type="submit" value="Enviar Información"/>
15 </div>
16 </form>

```

Modificamos el archivo settings.py del proyecto para configurar nuestra base de datos. En la línea 77 por default django trabaja con sql lite

```

77 DATABASES = {
78     'default': {
79         'ENGINE': 'django.db.backends.sqlite3',
80         'NAME': BASE_DIR / 'db.sqlite3',
81     }
82 }
83

```

Lo modificaremos para trabajar con mysql.

```

77 DATABASES = {
78     'default': {
79         'ENGINE': 'django.db.backends.mysql',
80         'NAME': 'CRUD_BASIC_DJANGO',
81         'USER': 'root',
82         'PASSWORD': '',
83         'HOST': 'localhost',
84         'PORT': '3306'
85     }
86 }

```

Pero para que funcione necesitamos instalar un paquete de mysql, lo realizaremos con el siguiente comando “pip3 install pymysql”, “pip3 install mysql” y “pip3 install pillow”

```

PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO> pip3 install pymysql
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pymysql in c:\users\edwar\appdata\local\packages\pythonsoftwarefoundation.python.3.12_qb
z5n2kfra8p0\localcache\local-packages\python312\site-packages (1.1.0)
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO>

```

```

PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO> pip3 install mysql
Defaulting to user installation because normal site-packages is not writeable
Collecting mysql
  Downloading mysql-0.0.3-py3-none-any.whl.metadata (746 bytes)
Collecting mysqlclient (from mysql)
  Downloading mysqlclient-2.2.4-cp312-cp312-win_amd64.whl.metadata (4.6 kB)
Downloading mysql-0.0.3-py3-none-any.whl (1.2 kB)
Downloading mysqlclient-2.2.4-cp312-cp312-win_amd64.whl (203 kB)
203.3/203.3 kB 588.6 kB/s eta 0:00:00
Installing collected packages: mysqlclient, mysql
Successfully installed mysql-0.0.3 mysqlclient-2.2.4
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO>

```

```
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO> pip3 install pillow
Defaulting to user installation because normal site-packages is not writeable
Collecting pillow
  Downloading pillow-10.3.0-cp312-cp312-win_amd64.whl.metadata (9.4 kB)
  Downloading pillow-10.3.0-cp312-cp312-win_amd64.whl (2.5 MB)
    2.5/2.5 MB 3.8 MB/s eta 0:00:00
Installing collected packages: pillow
Successfully installed pillow-10.3.0
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO>
```

Ahora configuramos el `__init__.py` para poder interactuar con la base de datos

```
CRUD_BASIC_DJANGO > __init__.py
1 import pymysql
2 pymysql.install_as_MySQLdb()
```

Ahora si corremos el servicio `"python3 manage.py runserver"` nos saldra un error porque no hemos corrido ninguna migración.

```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS


System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin
, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 18, 2024 - 13:10:49
Django version 5.0.6, using settings 'CRUD_BASIC_DJANGO.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

### Modelo y migraciones

```
models.py 2 x
libreria > models.py > Libro
1 from django.db import models
2
3 # nos sirve para capturar toda la estructura de la tabla
4 class Libro(models.Model):
5     idlibro = models.AutoField(primary_key=True)
6     titulo = models.CharField(max_length=100, verbose_name='Titulo del libro')
7     portada = models.ImageField(upload_to='imagenes/', verbose_name='Portada del libro')
8     autor = models.CharField(max_length=100, verbose_name='Autor del libro')
9     editorial = models.CharField(max_length=100, verbose_name='Editorial del libro')
10    fecha_publicacion = models.DateField(verbose_name='Fecha de publicacion del libro')
11
12
```

Nuestra base de datos :

 crud\_basic\_django

esta vacia no tenemos ninguna tabla,

Crearemos una migracion para crear la tabla libro, con el comando `"python3 manage.py makemigrations"`

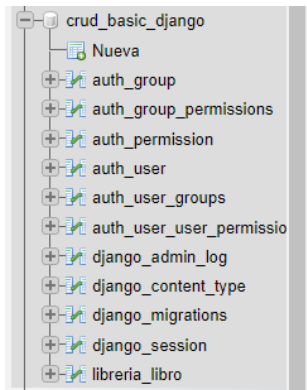
```
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO> python3 manage.py makemigrations
Migrations for 'libreria':
  libreria\migrations\0001_initial.py
    - Create model Libro
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO>
```

Ahora correremos la migración `"python3 manage.py migrate"`

```
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO> python3 manage.py migrate
System check identified some issues:

WARNINGS:
?: (mysql.W002) MariaDB Strict Mode is not set for database connection 'default'
   HINT: MariaDB's Strict Mode fixes many data integrity problems in MariaDB, such as data truncation upon inserti
on, by escalating warnings into errors. It is strongly recommended you activate it. See: https://docs.djangoproject.com
/en/5.0/ref/databases/#mysql-sql-mode
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, libreria, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying libreria.0001_initial... OK
  Applying sessions.0001_initial... OK
```

Y ahora nuestra base de datos:



ya contamos con nuestra tabla, lo que toma primero el nombre de la app y el nombre del modelo.

Ahora en el archivo `admin.py` de la app

```
libreria > admin.py
1  from django.contrib import admin
2  from .models import Libro
3  # Register your models here.
4  admin.site.register(Libro)
5
```

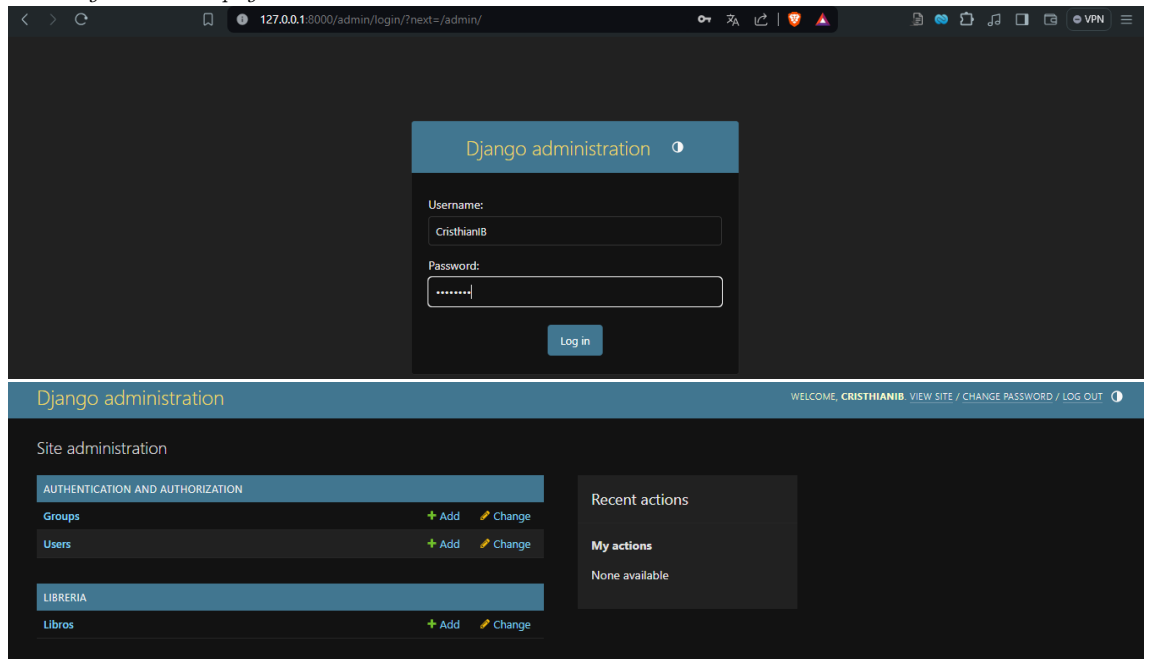
Crearemos un usuario administrativo con las líneas de comando `"python3 manage.py createsuperuser"`

Usuario: `CristhianIB`

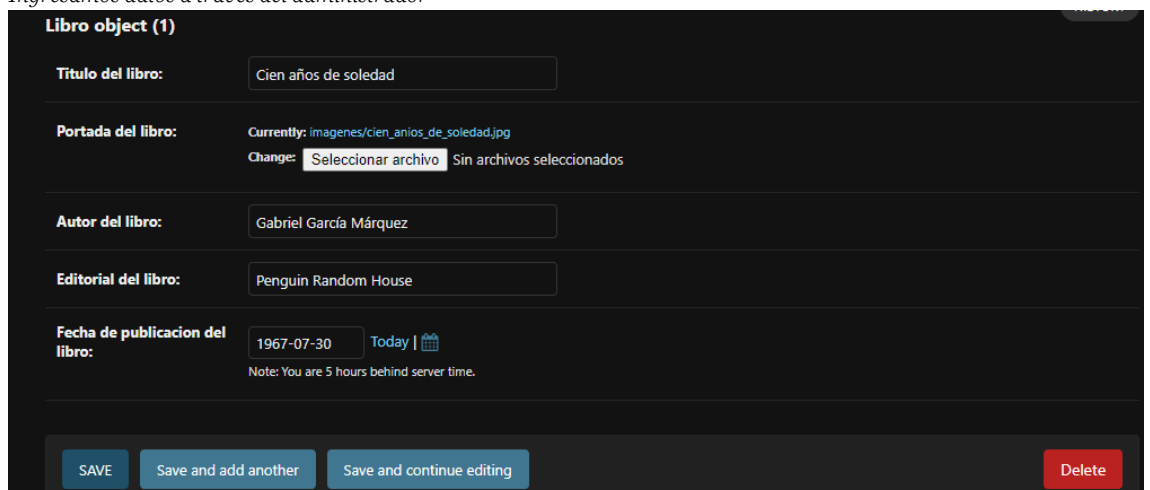
Password: `Admin123`

```
PROBLEMAS 4 SALIDA TERMINAL PUERTOS
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO> python3 manage.py createsuperuser
Username (leave blank to use 'edwar'): CristhianIB
Email address: cristhian.infante117@gmail.com
Password:
Password (again):
Error: Your passwords didn't match.
Password:
Password (again):
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
PS C:\xampp\htdocs\Trabajando con Django\CRUD_BASIC_DJANGO>
```

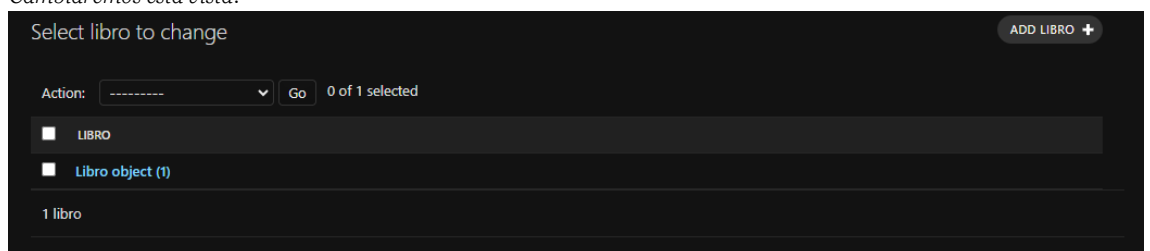
Ahora ingresamos a la pagina del administrador



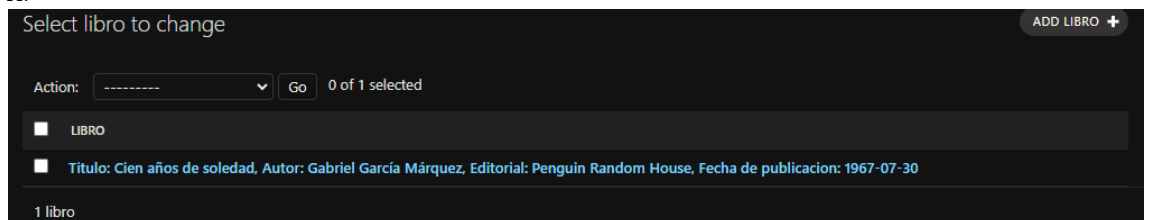
Ingresamos datos a traves del administrador



Cambiaremos esta vista:



A:



Con las siguientes lineas de codigo dentro del modelo :

```

12 def __str__(self): #nos sirve para mostrar los datos de la tabla
13     fila = "Titulo: " + self.titulo + " -" + " Autor: " + self.autor + " -" + " Editorial: " + self.editorial + " -" + " Fecha de publicacion: " + self.fecha_publicacion
14     return fila
15
16
libreria > models.py > ...
1 from django.db import models
2
3 nos sirve para capturar toda la estructura de la tabla
4 class Libro(models.Model):
5     idlibro = models.AutoField(primary_key=True)
6     titulo = models.CharField(max_length=100,verbose_name='Titulo del libro')
7     portada = models.ImageField(upload_to='imagenes/',verbose_name='Portada del libro')
8     autor = models.CharField(max_length=100,verbose_name='Autor del libro')
9     editorial = models.CharField(max_length=100,verbose_name='Editorial del libro')
10    fecha_publicacion = models.DateField(verbose_name='Fecha de publicacion del libro')
11
12 def __str__(self): #nos sirve para mostrar los datos de la tabla
13     fila = "Titulo: " + self.titulo + " -" + " Autor: " + self.autor + " -" + " Editorial: " + self.editorial + " -" + " Fecha de publicacion: " + self.fecha_publicacion
14     return fila
15

```

Un inconveniente que tenemos es que cuando eliminamos los datos las imágenes quedan en el sistema lo que haremos es eliminar las imágenes cuando eliminemos un dato.

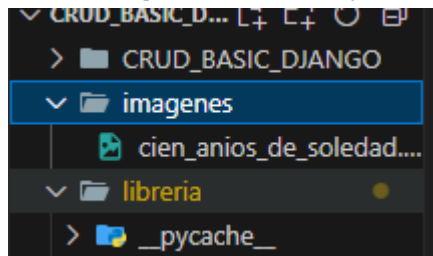
```

16 def delete(self,using=None,keep_parents=False):#nos sirve para eliminar la imagen de la carpeta imagenes
17     self.portada.storage.delete(self.portada.name)#elimina la imagen de la carpeta imagenes
18     super().delete()#elimina el registro de la tabla
19

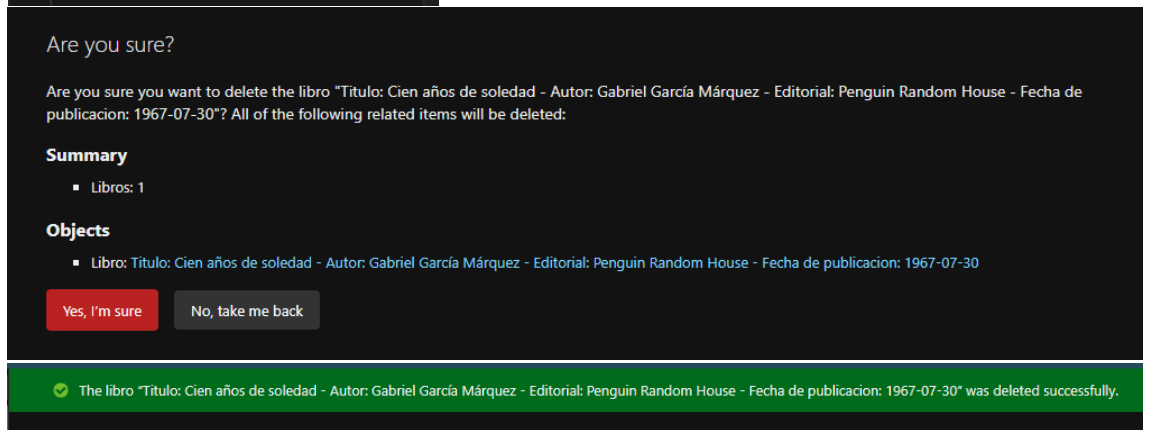
```

Hacemos la prueba.

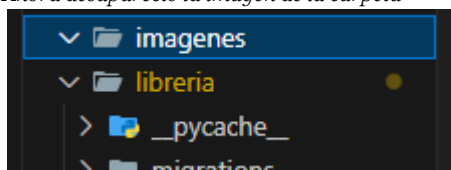
En nuestra carpeta tenemos una imagen



desde el administrador eliminaremos el registro y debería eliminar la imagen.



Ahora desaparecio la imagen de la carpeta



**Mostrando los datos en tabla:**

Primero modificamos el archivo views.py

```

def libros(request):
    libros=Libro.objects.all() #obtenemos todos los libros de la base de datos
    return render(request, "paginas/libros/index.html",{'libros':libros}) #retornamos un render con la plantilla lib

```

Libros=libro.objects.all() almacenara todos los datos de la base de datos los enviaremos a la vista a traves de {'libros':libros}

En el archivo index.html de la carpeta libros

Modificamos el tbody de la tabla para mostrar los datos:

```

</thead>
<tbody>
    {% for libro in libros %}
    <tr>
        <th scope="row">{{libro.idlibro}}</th>
        <td>{{libro.titulo}}</td>
        <td></td>
        <td>{{libro.autor}}</td>
        <td>{{libro.editorial}}</td>
        <td>{{libro.fecha_publicacion}}</td>
        <td>
            <a href="{% url 'editar' %}" class="btn btn-primary">Editar</a>
            <a href="#" class="btn btn-danger">Eliminar</a>
        </td>
    </tr>
    {% endfor %}
</tbody>

```

Dandonos como resultado

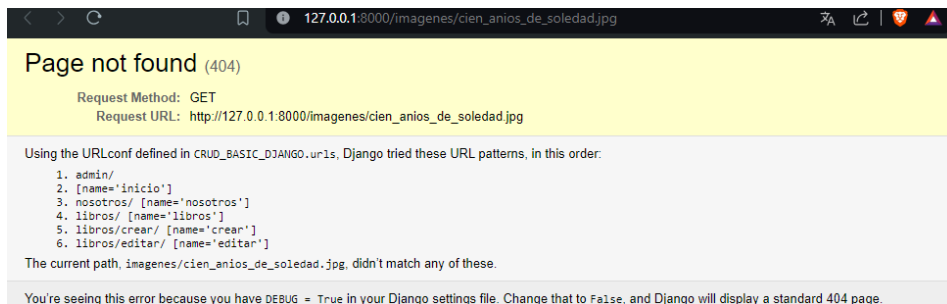
CRUD\_DJANGO
Inicio
libros
Nosotros

## Lista de los libros

Nuevo Libro

#	Titulo	Portada	Autor	Editorial	Fecha de publicacion	Accion
2	Cien años de soledad		Gabriel García Márquez	Penguin Random House	May 16, 2024	<div>Editar</div> <div>Eliminar</div>
3	El Señor de los Anillos		J.R.R. Tolkien	HarperCollins	May 16, 2024	<div>Editar</div> <div>Eliminar</div>

Pero como podemos apreciar no aparecen las imágenes ya que django protege las rutas y no deja pasar informacion.



Para poder visualizar las imágenes debemos realizar lo siguiente:

1º En el archivo settings.py del proyecto, agregamos las siguientes lineas

```

14 import os #importamos la libreria os, que nos permite interactuar con el sistema operativo
15
129 MEDIA_ROOT= os.path.join(BASE_DIR, '')#creamos una carpeta media en la raiz del proyecto
130 MEDIA_URL='/imagenes/'#creamos una url para la carpeta imagenes en la raiz del proyecto
131

```

2º En urls.py de la app

```

from django.conf import settings #importamos settings de django, que nos permite acceder a las configuraciones del proyecto
from django.contrib.staticfiles.urls import static #importamos static de django.contrib.staticfiles.urls, que nos permite acceder a los archivos

]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT) #accedemos a los archivos estaticos

libreria > urls.py > ...
1 from django.urls import path
2 from . import views #importamos las vistas de la aplicación
3 from django.conf import settings #importamos settings de django, que nos permite acceder a las configuraciones del proyecto
4 from django.contrib.staticfiles.urls import static #importamos static de django.contrib.staticfiles.urls, que nos permite acceder a los archivos
5
6
7 urlpatterns = [#aquí el usuario podrá acceder a la urls de la aplicación
8     path('', views.inicio, name='inicio'), #definimos la url de inicio
9     path('nosotros/', views.nosotros, name='nosotros'), #definimos la url de nosotros
10    path('libros/', views.libros, name='libros'), #definimos la url de libros
11    path('libros/crear/', views.crear, name='crear'), #definimos la url de crear
12    path('libros/editar/', views.editar, name='editar'), #definimos la url de editar
13 ]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT) #accedemos a los archivos estaticos
14

```

Dandonos como resultado la visualizacion de las imágenes:



# Lista de los libros

Nuevo Libro

#	Titulo	Portada	Autor	Editorial	Fecha de publicacion	Accion	
2	Cien años de soledad		Gabriel García Márquez	Penguin Random House	May 16, 2024	Editar	Eliminar
3	El Señor de los Anillos		J.R.R. Tolkien	HarperCollins	May 16, 2024	Editar	Eliminar

**Insertar Datos:**  
Crearemos un archivo form.py en la app, aquí utilizaremos los formularios que vamos a requerir a partir del modelo, crearemos una clase para la lectura y el mapeo de los formularios.

```
libreria > forms.py > LibroForm > Meta
1 from django import forms
2 from .models import Libro
3
4 class LibroForm(forms.ModelForm):
5     class Meta:
6         model = Libro
7         fields = '__all__'
```

En views.py del app agregaremos el archivo creado recientemente que es el forms.py

```
5 from .forms import LibroForm #importamos el formulario LibroForm
```

Ahora modificaremos la funcion de crear:  
Para identificar todos los elementos que se estan enviando desde el formulario

```
16 def crear(request):
17     formulario=LibroForm(request.POST or None,request.FILES or None) #creamos un formulario de LibroForm,
18                                     #y le pasamos el request.POST que contiene los datos del formulario
19     return render(request, 'paginas/libros/crear.html',{'formulario':formulario}) #retornamos un render con la plantilla crear.html,
20                                     #y le pasamos el formulario como contexto
```

Ahora modificaremos el archivo form.html

```
libreria > templates > paginas > libros > dj form.html
1 <form enctype="multipart/form-data" method="post">
2     <!--django utiliza un token con la finalidad de evitar ataques csrf-->
3     {% csrf_token %}
4     {% for campo in formulario %} <!--recorremos el formulario enviado desde la vista-->
5     <div class="mb-3">
6         <label for="" class="form-label">{{campo.label}}</label>
7         <input type="{{campo.field.widget.input_type}}" class="form-control" name="{{campo.name}}" id="" aria-describedby="helpId"
8         placeholder="{{campo.label}}"/>
9         <!--{{campo.field.widget.input_type}} obtiene el tipo de campo que se esta utilizando en el formulario-->
10    </div>
11    <!--imprimimos la parte de errores-->
12    <div class="col-12 help-text">{{campo.errors}}</div>
13
14    {% endfor %}
15    <div class="d-grid gap-2">
16        <input name="" id="" class="btn btn-success" type="submit" value="Enviar Información"/>
17    </div>
18 </form>
```

Que nos da como resultado

Crear Libro

Datos del libro

Titulo del libro

Titulo del libro

Portada del libro

Seleccionar archivo

Sin archivos seleccionados

Autor del libro

Autor del libro

Editorial del libro

Editorial del libro

Fecha de publicacion del libro

Fecha de publicacion del libro

Enviar Información

**Eliminar:** Para esto debemos modificar primeramente en views.py

```
26 def eliminar(request,id):
27     libro = Libro.objects.get(idlibro=id) #obtenemos el libro por el id
28     libro.delete() #eliminamos el libro
29     return redirect('libros') #redireccionamos a la vista libros
```

Lo que realizamos es crear una funcion eliminar que recibira un id, que luego lo buscara en libro y posteriormente lo eliminara manteniendose en la misma pagina.

Ahora en urls.py agregaremos la url para poder eliminar

```
13 path('eliminar/<int:id>/', views.eliminar, name='eliminar'), #definimos la url de eliminar con un parametro id de
```


Dicha ruta enviara el id a eliminar el cual lo recibira el views y posteriormente lo mostrara.

Dando como resultado.

CRUD\_DJANGO Inicio libros Nosotros

## Lista de los libros

Nuevo Libro

#	Título	Portada	Autor	Editorial	Fecha de publicacion	Accion
3	El Señor de los Anillos		J.R.R. Tolkien	HarperCollins	May 16, 2024	<a href="#">Editar</a> <a href="#">Eliminar</a>

Una eliminacion correcta, la cual elimina tambien la imagen del proyecto tal y como se definio al inicio.

**Editar:** Para esto debemos crear una funcion para en views.py

```
24 def editar(request,id):
25     #recuperamos el libro por el id
26     libro = Libro.objects.get(idlibro=id)
27     formulario=LibroForm(request.POST or None, request.FILES or None, instance=libro)
28     if formulario.is_valid() and request.POST:
29         formulario.save()
30         return redirect('libros')
31     return render(request, 'paginas/libros/editar.html',{'formulario':formulario}) #retornamos un render con la planti
```

Recuperamos la informacion con el id recibido

Luego lo almacenamos en la variable formulario, validamos si la informacion es valida y si fue enviada a traves del metodo pos para poderlo guardar

Modificamos el index.html de la carpeta libros

```
<td>
<a name="" id="" href="{% url 'editar' libro.idlibro%}" class="btn btn-primary" role="button">Editar</a>
<a name="" id="" href="{% url 'eliminar' libro.idlibro%}" class="btn btn-danger" role="button">Eliminar</a>
</td>
```

En el cual al dar click enviar nos redireccionara a la vista editar llevando en si el id que se va a modificar.

En urls.py

```
12 path('libros/editar/<int:id>/', views.editar, name='editar'), #definimos la url de editar
```

Definiremos lo que debe realizar, en este caso debe mostrar la siguiente pagina editar con el id.

Lo que nos da como resultado

127.0.0.1:8000/libros/editar/3/

CRUD\_DJANGO Inicio libros Nosotros

inglés español


### Editar Libro

#### Datos del libro

Título del libro

El Señor de los Anillos 2

Portada del libro



Seleccionar archivo Sin archivos seleccionados

Autor del libro

J.R.R. Tolkien

Editorial del libro

HarperCollins

Fecha de publicacion del libro