Paso 1: Instalar Django y dependencias MySQL

Primero, crea un entorno virtual y actívalo:

bash

```
# Crea un entorno virtual
python -m venv env

# Activa el entorno virtual (en Windows usa: env\Scripts\activate)
source env/bin/activate
```

Luego, instala Django y el conector MySQL para Python (MySQLclient):

```
pip install django mysqlclient
```

Paso 2: Crear un nuevo proyecto Django

```
django-admin startproject myproject
cd myproject
```

Paso 3: Configurar MySQL en el proyecto Django

En el archivo settings.py de tu proyecto, configura la base de datos para usar MySQL:

```
# settings.py
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'nombre_de_tu_base_de_datos',
        'USER': 'tu_usuario_mysql',
        'PASSWORD': 'tu_contraseña_mysql',
        'HOST': 'localhost',  # O la dirección de tu servidor MySQL
```

```
        'PORT': '3306',
    }
}
```

Asegúrate de haber creado la base de datos en MySQL:

```
CREATE DATABASE nombre_de_tu_base_de_datos;
```

Paso 4: Crear una aplicación Django

Crea una aplicación dentro del proyecto para gestionar el CRUD:

```
python manage.py startapp myapp
```

Luego, añade la aplicación al archivo settings.py:

python

```
# settings.py
INSTALLED_APPS = [
    # Otras apps de Django
    'myapp',
]
```

Paso 5: Definir el modelo en Django

En el archivo models.py de tu aplicación, define el modelo que representará la tabla en la base de datos. Por ejemplo, crearemos un modelo de **Libro**:

```
# myapp/models.py
from django.db import models
```

```
class Libro(models.Model):
    titulo = models.CharField(max_length=200)
    autor = models.CharField(max_length=100)
    publicado = models.DateField()
    isbn = models.CharField(max_length=13, unique=True)

    def __str__(self):
        return self.titulo
```

Paso 6: Migraciones

Genera las migraciones y aplícalas a la base de datos MySQL:

```
python manage.py makemigrations
python manage.py migrate
```

Paso 7: Crear vistas para el CRUD

Vamos a definir las vistas para crear, leer, actualizar y eliminar libros. En el archivo views.py, crea las funciones correspondientes:

```
python

# myapp/views.py
from django.shortcuts import render, get_object_or_404, redirect
from .models import Libro
from .forms import LibroForm

# Listar todos los libros (Read)
def libro_list(request):
    libros = Libro.objects.all()
    return render(request, 'myapp/libro_list.html', {'libros': libros})
```

```
# Crear un nuevo libro (Create)
def libro_create(request):
    if request.method == 'POST':
        form = LibroForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('libro_list')
    else:
        form = LibroForm()
    return render(request, 'myapp/libro_form.html', {'form': form})

# Actualizar un libro existente (Update)
def libro_update(request, pk):
    libro = get_object_or_404(Libro, pk=pk)
    if request.method == 'POST':
        form = LibroForm(request.POST, instance=libro)
        if form.is_valid():
            form.save()
            return redirect('libro_list')
    else:
        form = LibroForm(instance=libro)
    return render(request, 'myapp/libro_form.html', {'form': form})

# Eliminar un libro (Delete)
def libro_delete(request, pk):
```

```
    libro = get_object_or_404(Libro, pk=pk)
    if request.method == 'POST':
        libro.delete()
        return redirect('libro_list')
    return render(request, 'myapp/libro_confirm_delete.html', {'libro': libro})
```

Paso 8: Formularios

Django facilita el manejo de formularios. Crea un archivo forms.py en tu aplicación y define un formulario basado en el modelo de **Libro**:

python

```
# myapp/forms.py
from django import forms
from .models import Libro

class LibroForm(forms.ModelForm):
    class Meta:
        model = Libro
        fields = ['titulo', 'autor', 'publicado', 'isbn']
```

Paso 9: Crear plantillas

Dentro de tu aplicación, crea un directorio templates/myapp/ y añade las siguientes plantillas:

1. libro_list.html (Listado de libros):

html

```
<h2>Lista de Libros</h2>
<a href="{% url 'libro_create' %}">Agregar nuevo libro</a>
<ul>
```

```
  {% for libro in libros %}
  <li>
    {{ libro.titulo }} - {{ libro.autor }}
    <a href="{% url 'libro_update' libro.pk %}">Editar</a>
    <a href="{% url 'libro_delete' libro.pk %}">Eliminar</a>
  </li>
  {% endfor %}
</ul>
```

2. libro_form.html (Formulario para crear/editar libros):

html

```
<h2>Formulario de Libro</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Guardar</button>
</form>
```

3. libro_confirm_delete.html (Confirmación de eliminación):

html

```
<h2>¿Estás seguro de que quieres eliminar "{{ libro.titulo }}"?</h2>
<form method="post">
  {% csrf_token %}
  <button type="submit">Eliminar</button>
</form>
<a href="{% url 'libro_list' %}">Cancelar</a>
```

Paso 10: Configurar URLs

Configura las URLs para las vistas de tu CRUD en el archivo urls.py de tu aplicación:

python

```
# myapp/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('', views.libro_list, name='libro_list'),
    path('nuevo/', views.libro_create, name='libro_create'),
    path('<int:pk>/editar/', views.libro_update, name='libro_update'),
    path('<int:pk>/eliminar/', views.libro_delete, name='libro_delete'),
]
```

Y añade las URLs de la aplicación en el urls.py principal de tu proyecto:

python

```
# myproject/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('libros/', include('myapp.urls')),  # Incluir las rutas de la app
]
```

Paso 11: Ejecutar el servidor

Finalmente, ejecuta el servidor para probar el proyecto:

bash

python manage.py runserver

Visita http://localhost:8000/libros/ para ver el CRUD en acción.