**Parcial final Programación 4**

paso a paso para crear un CRUD (Create, Read, Update, Delete) sencillo con Django y SQLite. Vamos a crear una aplicación de gestión de libros.

**Requisitos Previos**

* Python 3.8 o superior
* pip (gestor de paquetes de Python)

**Paso 1: Preparación del Entorno Virtual**

**Instalación de Virtual Environment**

bash

Copy

*# Instalar virtualenv*

pip install virtualenv

*# Crear un directorio para el proyecto*

mkdir biblioteca\_django

cd biblioteca\_django

*# Crear entorno virtual*

python -m venv venv

*# Activar el entorno virtual*

*# En Windows*

venv\Scripts\activate

*# En macOS/Linux*

source venv/bin/activate

**Paso 2: Instalación de Django**

bash

Copy

*# Instalar Django*

pip install django

**Paso 3: Crear el Proyecto Django**

bash

Copy

*# Crear nuevo proyecto Django*

django-admin startproject biblioteca\_proyecto

*# Cambiar al directorio del proyecto*

cd biblioteca\_proyecto

*# Crear la aplicación*

python manage.py startapp libros

**Paso 4: Configuración del Modelo**

Abre libros/models.py y define el modelo de Libro:

python

Copy

from django.db import models

class Libro(models.Model):

titulo = models.CharField(max\_length=200)

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

ano\_publicacion = models.IntegerField()

genero = models.CharField(max\_length=50)

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

return self.titulo

**Paso 5: Configuración de la Aplicación**

Abre biblioteca\_proyecto/settings.py y agrega la aplicación a INSTALLED\_APPS:

python

Copy

INSTALLED\_APPS = [

*# ...*

'libros',

]

**Paso 6: Crear Migraciones**

bash

Copy

*# Crear migraciones*

python manage.py makemigrations libros

*# Aplicar migraciones*

python manage.py migrate

**Paso 7: Crear Formulario**

Crea un archivo libros/forms.py:

python

Copy

from django import forms

from .models import Libro

class LibroForm(forms.ModelForm):

class Meta:

model = Libro

fields = ['titulo', 'autor', 'ano\_publicacion', 'genero']

**Paso 8: Vistas (CRUD)**

Abre libros/views.py:

python

Copy

from django.shortcuts import render, redirect, get\_object\_or\_404

from .models import Libro

from .forms import LibroForm

def lista\_libros(request):

libros = Libro.objects.all()

return render(request, 'libros/lista\_libros.html', {'libros': libros})

def crear\_libro(request):

if request.method == 'POST':

form = LibroForm(request.POST)

if form.is\_valid():

form.save()

return redirect('lista\_libros')

else:

form = LibroForm()

return render(request, 'libros/crear\_libro.html', {'form': form})

def editar\_libro(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('lista\_libros')

else:

form = LibroForm(instance=libro)

return render(request, 'libros/editar\_libro.html', {'form': form})

def eliminar\_libro(request, pk):

libro = get\_object\_or\_404(Libro, pk=pk)

if request.method == 'POST':

libro.delete()

return redirect('lista\_libros')

return render(request, 'libros/eliminar\_libro.html', {'libro': libro})

**Paso 9: Configurar URLs**

En biblioteca\_proyecto/urls.py:

python

Copy

from django.contrib import admin

from django.urls import path

from libros import views

urlpatterns = [

path('admin/', admin.site.urls),

path('', views.lista\_libros, name='lista\_libros'),

path('libro/nuevo/', views.crear\_libro, name='crear\_libro'),

path('libro/editar/<int:pk>/', views.editar\_libro, name='editar\_libro'),

path('libro/eliminar/<int:pk>/', views.eliminar\_libro, name='eliminar\_libro'),

]

**Paso 10: Crear Templates**

Crea los siguientes archivos de template:

libros/templates/libros/lista\_libros.html:

html

Copy

<!DOCTYPE html>

<html>

<head>

<title>Lista de Libros</title>

</head>

<body>

<h1>Biblioteca</h1>

<a href="{% url 'crear\_libro' %}">Añadir Nuevo Libro</a>

<ul>

{% for libro in libros %}

<li>

{{ libro.titulo }} - {{ libro.autor }}

<a href="{% url 'editar\_libro' libro.pk %}">Editar</a>

<a href="{% url 'eliminar\_libro' libro.pk %}">Eliminar</a>

</li>

{% endfor %}

</ul>

</body>

</html>

libros/templates/libros/crear\_libro.html:

html

Copy

<!DOCTYPE html>

<html>

<head>

<title>Crear Libro</title>

</head>

<body>

<h1>Añadir Nuevo Libro</h1>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Guardar</button>

</form>

<a href="{% url 'lista\_libros' %}">Volver a la lista</a>

</body>

</html>

libros/templates/libros/editar\_libro.html:

html

Copy

<!DOCTYPE html>

<html>

<head>

<title>Editar Libro</title>

</head>

<body>

<h1>Editar Libro</h1>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Actualizar</button>

</form>

<a href="{% url 'lista\_libros' %}">Cancelar</a>

</body>

</html>

libros/templates/libros/eliminar\_libro.html:

html

Copy

<!DOCTYPE html>

<html>

<head>

<title>Eliminar Libro</title>

</head>

<body>

<h1>Eliminar Libro</h1>

<p>¿Estás seguro de que deseas eliminar "{{ libro.titulo }}"?</p>

<form method="post">

{% csrf\_token %}

<button type="submit">Confirmar Eliminación</button>

</form>

<a href="{% url 'lista\_libros' %}">Cancelar</a>

</body>

</html>

**Paso 11: Ejecutar el Servidor**

bash

Copy

*# Ejecutar servidor de desarrollo*

python manage.py runserver

**Pasos Adicionales (Opcional)**

1. Instalar dependencias para otros desarrolladores:

bash

Copy

*# Crear archivo de requisitos*

pip freeze > requirements.txt

*# Otros desarrolladores pueden instalar con:*

pip install -r requirements.txt

**Características de este CRUD**

* Modelo Libro con campos básicos
* Vistas para listar, crear, editar y eliminar libros
* Formulario basado en modelo
* Templates básicos
* Uso de SQLite como base de datos (predeterminado de Django)

**Consideraciones de Seguridad y Mejoras**

* Añadir autenticación
* Validaciones más robustas
* Estilos CSS
* Manejo de errores
* Paginación

**Troubleshooting**

* Asegúrate de tener el entorno virtual activado
* Verifica que todas las migraciones se hayan aplicado
* Comprueba que la aplicación esté en INSTALLED\_APPS

¡Listo! Ahora tienes un CRUD básico con Django y SQLite. Puedes acceder a la aplicación en http://127.0.0.1:8000/ después de ejecutar el servidor de desarrollo.

**Rúbrica de Evaluación - CRUD Django con SQLite**

**1. Estructura del Proyecto (20 puntos)**

* Estructura de directorios correcta (5 puntos)
* Uso de entorno virtual (5 puntos)
* Archivo requirements.txt presente (5 puntos)
* README.md detallado (5 puntos)

**2. Modelo de Datos (20 puntos)**

* Modelo correctamente definido (10 puntos)
* Campos apropiados y validaciones (5 puntos)
* Método \_\_str\_\_ implementado (5 puntos)

**3. Vistas y Funcionalidad CRUD (30 puntos)**

* Lista de elementos funcional (5 puntos)
* Creación de nuevos elementos (5 puntos)
* Edición de elementos (5 puntos)
* Eliminación de elementos (5 puntos)
* Manejo de formularios (5 puntos)
* Validaciones de formulario (5 puntos)

**4. Templates y Frontend (15 puntos)**

* Templates básicos para cada vista (5 puntos)
* Diseño limpio y funcional (5 puntos)
* Navegación entre vistas (5 puntos)

**5. Configuración y Ejecución (15 puntos)**

* Configuración correcta de settings (5 puntos)
* Migraciones aplicadas correctamente (5 puntos)
* Servidor de desarrollo funcional (5 puntos)

**Documentación y Entrega (15 puntos)**

* Archivo README.md detallado (5 puntos)
* Capturas de pantalla de cada funcionalidad (5 puntos)
* Archivo comprimido con estructura correcta (5 puntos)

**Criterios Específicos para el README.md**

El README.md debe incluir:

1. Título del proyecto
2. Descripción breve
3. Requisitos del sistema
4. Instrucciones de instalación
5. Pasos para configurar el entorno virtual
6. Comandos para instalar dependencias
7. Instrucciones para ejecutar el proyecto
8. Capturas de pantalla de las principales funcionalidades
9. Información de contacto

**Estructura del Archivo Comprimido**

Copy

nombre\_apellido\_crud\_django.zip

│

├── proyecto\_biblioteca/

│ ├── venv/

│ ├── biblioteca\_proyecto/

│ │ ├── \_\_init\_\_.py

│ │ ├── settings.py

│ │ ├── urls.py

│ │ └── wsgi.py

│ ├── libros/

│ │ ├── migrations/

│ │ ├── templates/

│ │ ├── \_\_init\_\_.py

│ │ ├── admin.py

│ │ ├── apps.py

│ │ ├── models.py

│ │ ├── tests.py

│ │ ├── urls.py

│ │ └── views.py

│ ├── manage.py

│ ├── requirements.txt

│ └── README.md

│

└── capturas\_pantalla/

├── lista\_libros.png

├── crear\_libro.png

├── editar\_libro.png

└── eliminar\_libro.png

**Consideraciones Adicionales**

* Se descontarán puntos por errores de código
* La funcionalidad completa es crucial
* La presentación y documentación son importantes
* Se valorará la claridad y prolijo del código

**Formato de Entrega**

* Archivo comprimido (.zip)
* Nombre del archivo: nombre\_apellido\_crud\_django.zip
* Incluir todas las capturas de pantalla
* README.md detallado
* Código funcionando completamente

**Criterios de Evaluación Extra**

* Manejo de errores (+5 puntos)
* Estilos CSS (+5 puntos)
* Validaciones adicionales (+5 puntos)
* Implementación de búsqueda (+5 puntos)

**Penalizaciones**

* Código que no funcione: hasta -20 puntos
* Falta de documentación: hasta -15 puntos
* Errores de configuración: hasta -10 puntos
* No seguir estructura solicitada: hasta -15 puntos

Puntaje Total: 100 puntos

**Observaciones Finales**

* Revisa múltiples veces antes de entregar
* Prueba todos los componentes
* Documenta cualquier dificultad encontrada
* Mantén un código limpio y comentado