

CUE: MEJORES PRÁCTICAS EN DJANGO

Pablo Varas Salamanca

07 de noviembre de 2024, 14:20:01

[GitHub del proyecto](#)

DRILLING: AGREGANDO FUNCIONALIDADES A NUESTRO PROYECTO

EJERCICIO:

- En el menú de **Libros**, debe listar los libros creados dinámicamente por http://localhost:8000/lista_libros/ o por el administrador de Django, y debe presentar la siguiente apariencia:

Site Web Django Inicio Libros Acerca de Contacto Login		
Listado de libros		
Titulo	Autor	Valoración
Django 3 Web Development Cookbook Fourth Edition	Aidas Bendraitis	3250
Two Scoops of Django 3.x	Daniel Feldroy	1570
El libro de Django	Adrian Holovaty	750
Python Web Development with Django	Jeff Forcier	1350
Django for Professionals	William S. Vincent	2100
Django for APIs	William S. Vincent	2540

DESCRIPCIÓN	PRODUCTOS	CONTACTO
Desarrollo de Sitio Web de Django.	Libros	Dirección, calle, País info@example.com + 01 234 567 88 + 01 234 567 89

© 2022 Copyright: SiteWebDjango.com

- En el sistema, como un usuario sin iniciar sesión (no autenticado), ingresa al siguiente enlace: http://localhost:8000/crear_libro. Si logras ingresar libros, ¿qué se debería hacer para limitar el ingreso de libros a sólo usuarios autenticados en el sistema?

DESARROLLO:

1. Listado de libros

La vista `lista_libros` en el archivo `views.py` lista todos los libros creados y se encuentra protegida con el decorador `@login_required`, por lo que solo los usuarios autenticados pueden acceder a ella. También se registró esta vista en el archivo `urls.py` como la ruta `'lista_libros/'`

- Vista `lista_libros`

```
1
2 @login_required # para que solo se pueda acceder si esta logueado
3 def lista_libros(request):
4     libros = Book.objects.all()
5     return render(request, 'lista_libros.html', {'libros': libros})
```

Captura de pantalla del archivo `views.py`

```
@login_required # para que solo se pueda acceder si esta logeado
def lista_libros(request):
    libros = Book.objects.all()
    return render(request, 'lista_libros.html', {'libros': libros})
```

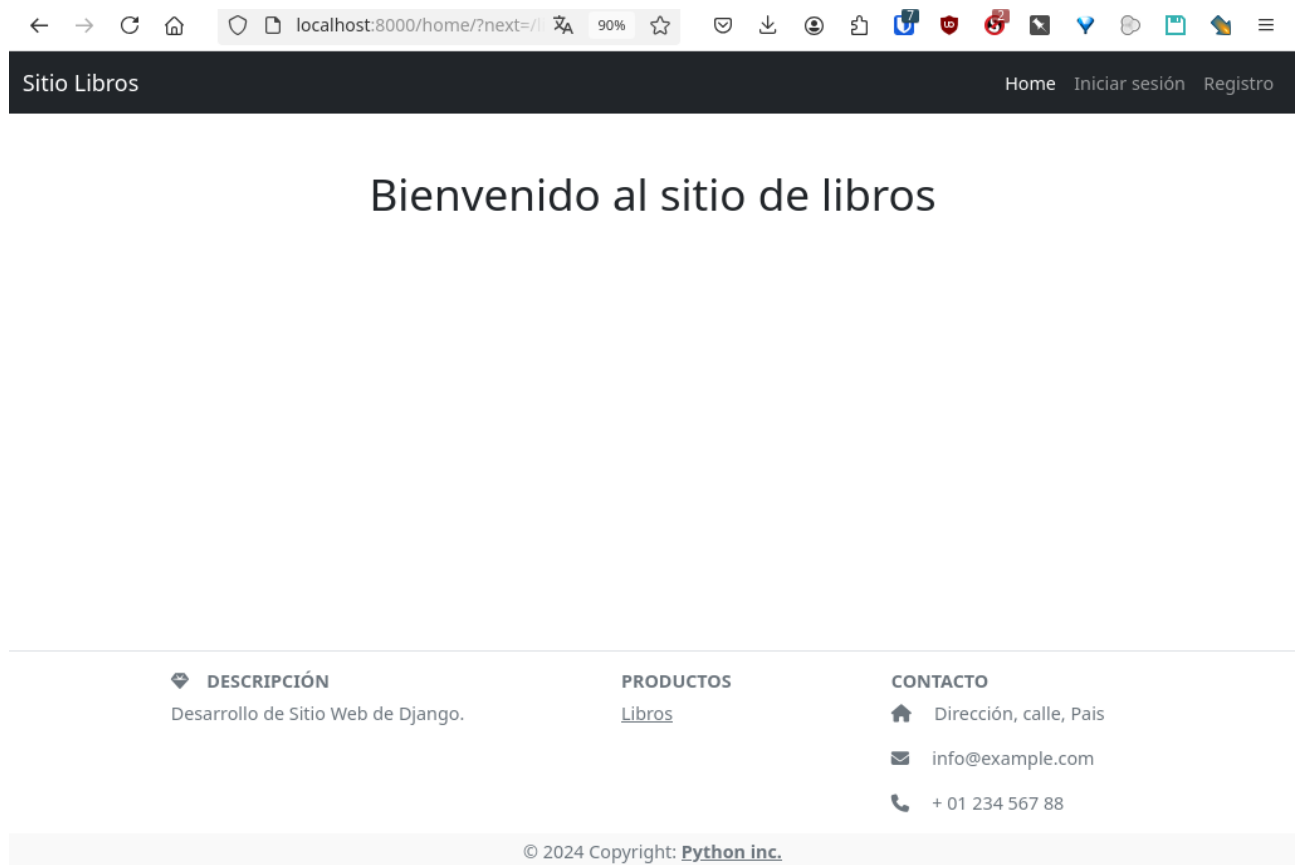
- Registro de la ruta de la vista `'lista_libros/'` en `urls.py`

```
1 path('lista_libros/', lista_libros, name='lista_libros'),
```

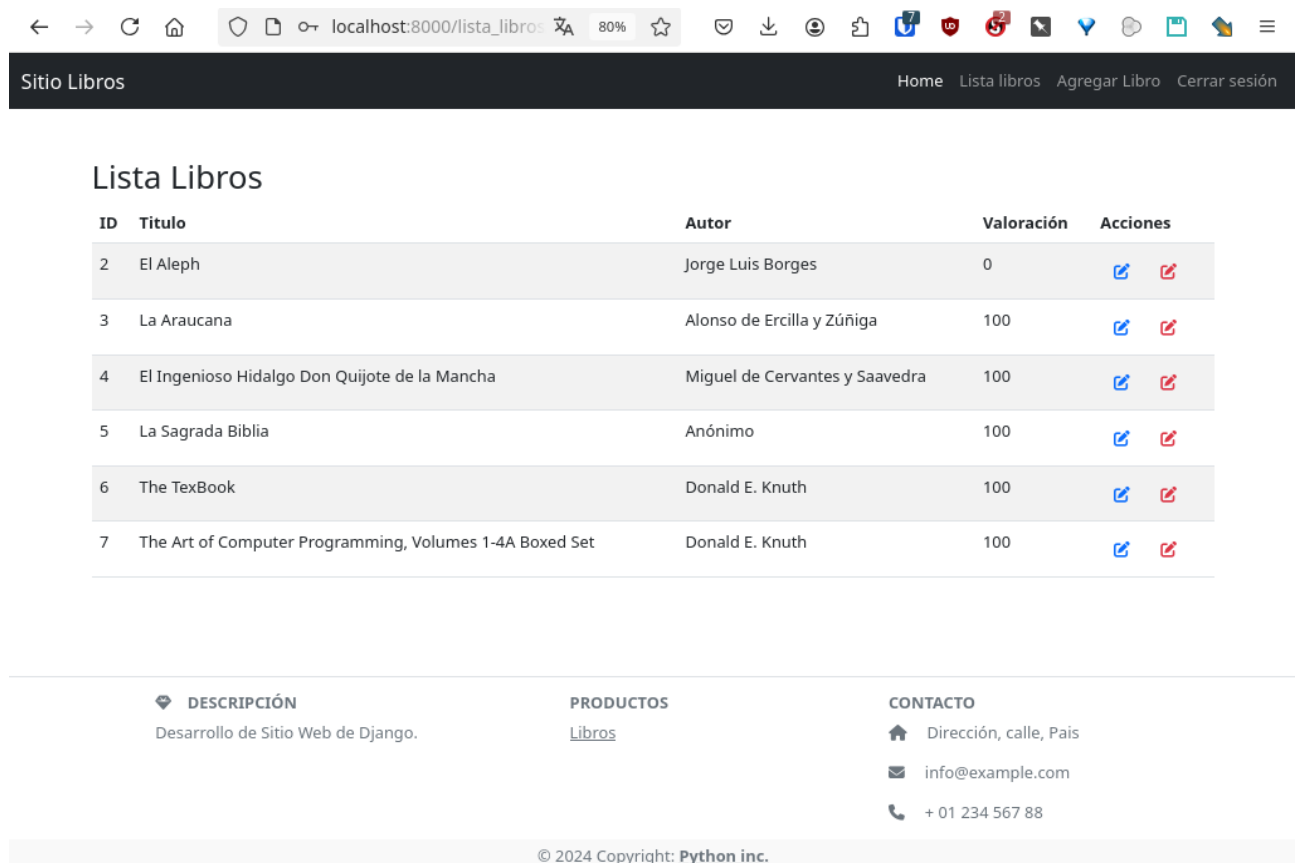
Captura de la ruta de `lista_libros` en `urls.py`

```
from django.urls import path
from . import views
# importar views
from .views import IndexPageView, lista_libros, crear_libro, editar_libro, eliminar_libro
from .views import registro, iniciar_sesion, home_page, cerrar_sesion
urlpatterns = [
    # path(ruta, view, name=nombre_de_la_ruta)
    path('', IndexPageView.as_view(), name='index'),
    path('lista_libros/', lista_libros, name='lista_libros'),
    path('crear_libro/', crear_libro, name='crear_libro'), # registrando ruta para crear libro
    path('editar_libro/<int:libro_id>', editar_libro, name='editar_libro'), # registrando ruta para editar libro
    path('eliminar_libro/<int:libro_id>', eliminar_libro, name='eliminar_libro'), # registrando ruta para eliminar libro
]
```

Si intento ingresar directamente a http://127.0.0.1:8000/lista_libros/ sin haber iniciado sesión con un usuario autorizado me redirige a <http://127.0.0.1:8000/home>



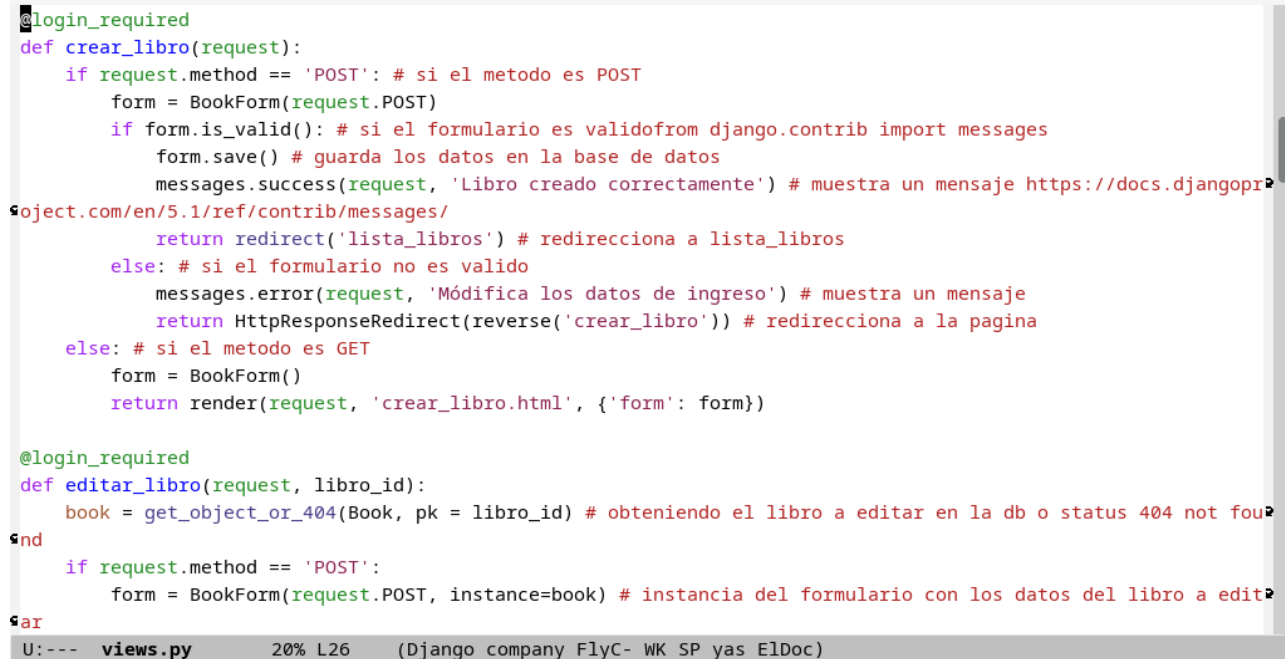
Si accedo a `lista_libros` con un usuario autorizado, se despliega:



2. Restricción de acceso

La vista `crear_libro`, que permite agregar un nuevo libro, también está protegida con `@login_required`, impidiendo el acceso a usuarios no autenticados. Si un usuario no autenticado intenta acceder a `crear_libro`, será redirigido a la página de inicio de sesión antes de poder ingresar un libro, cumpliendo con el requisito de limitar la acción a usuarios autenticados.

```
1  @login_required
2  def crear_libro(request):
3      if request.method == 'POST': # si el metodo es POST
4          form = BookForm(request.POST)
5          if form.is_valid():
6              # si el formulario es válido from django.contrib import messages
7              form.save() # guarda los datos en la base de datos
8              messages.success(request, 'Libro creado correctamente')
9              # muestra un mensaje
10             https://docs.djangoproject.com/en/5.1/ref/contrib/messages/
11             return redirect('lista_libros') # redirecciona a lista_libros
12         else: # si el formulario no es valido
13             messages.error(request, 'Modifica los datos de ingreso')
14             # muestra un mensaje
15             return HttpResponseRedirect(reverse('crear_libro'))
16             # redirecciona a la pagina
17         else: # si el metodo es GET
18             form = BookForm()
19             return render(request, 'crear_libro.html', {'form': form})
```

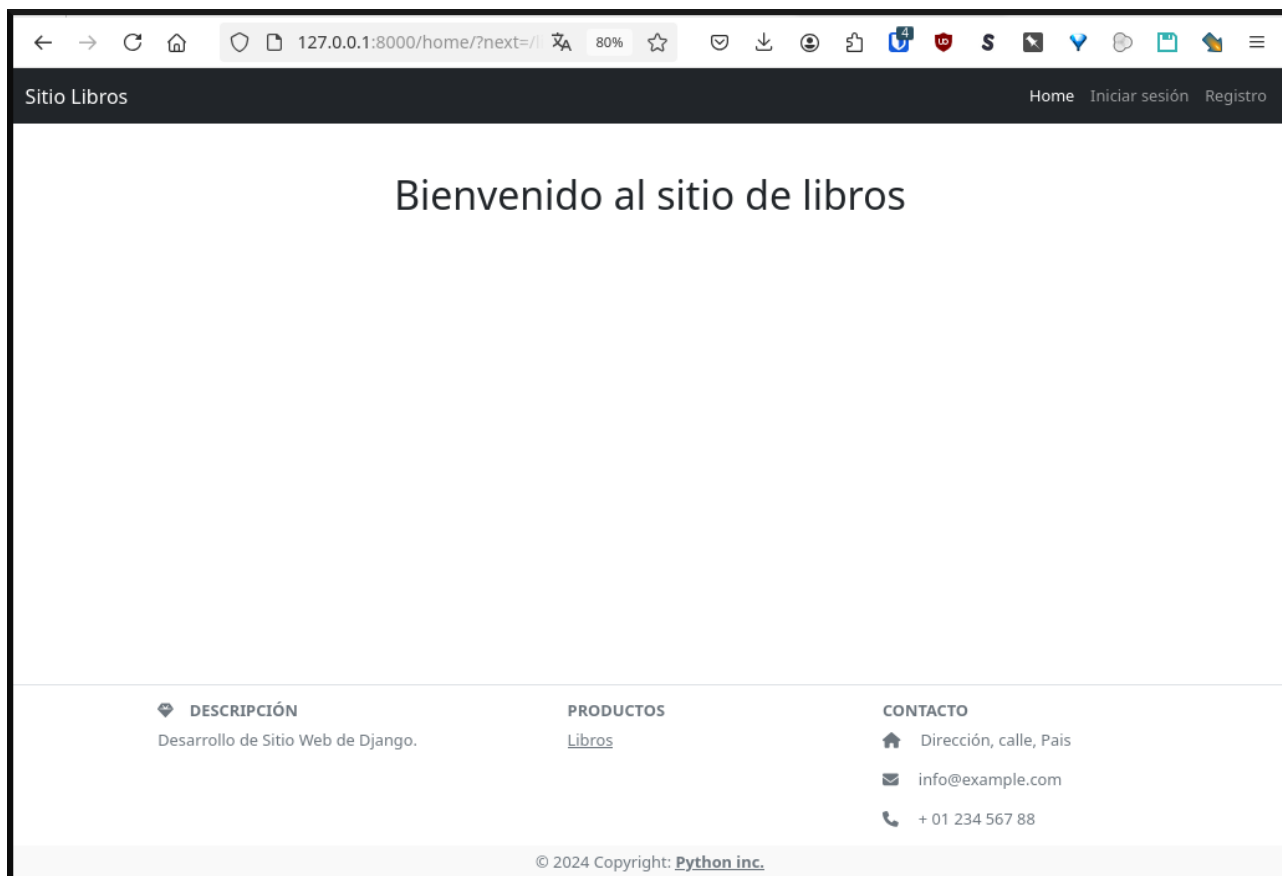


```
@login_required
def crear_libro(request):
    if request.method == 'POST': # si el metodo es POST
        form = BookForm(request.POST)
        if form.is_valid(): # si el formulario es valido from django.contrib import messages
            form.save() # guarda los datos en la base de datos
            messages.success(request, 'Libro creado correctamente') # muestra un mensaje https://docs.djangoproject.com/en/5.1/ref/contrib/messages/
            return redirect('lista_libros') # redirecciona a lista_libros
        else: # si el formulario no es valido
            messages.error(request, 'Modifica los datos de ingreso') # muestra un mensaje
            return HttpResponseRedirect(reverse('crear_libro')) # redirecciona a la pagina
    else: # si el metodo es GET
        form = BookForm()
        return render(request, 'crear_libro.html', {'form': form})

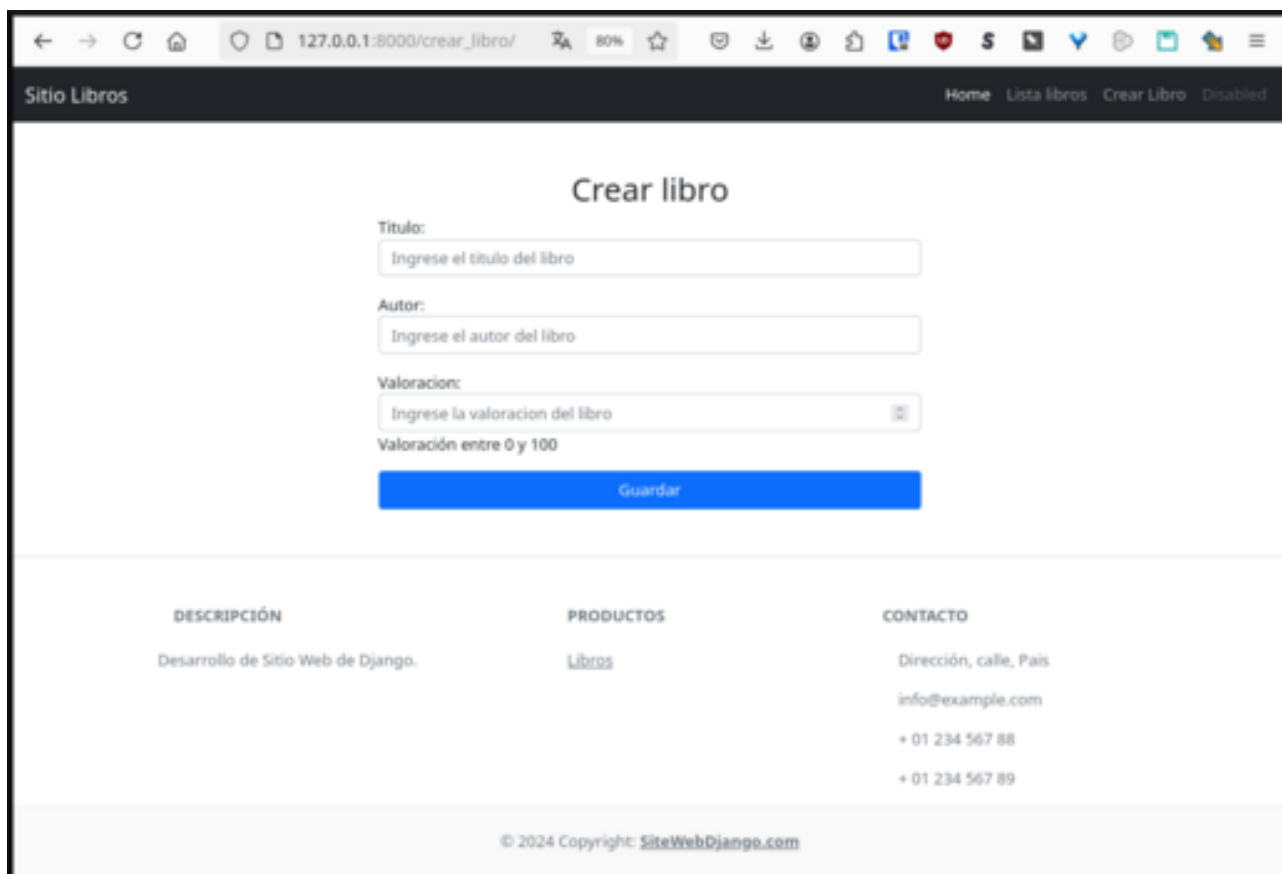
@login_required
def editar_libro(request, libro_id):
    book = get_object_or_404(Book, pk = libro_id) # obteniendo el libro a editar en la db o status 404 not found
    if request.method == 'POST':
        form = BookForm(request.POST, instance=book) # instancia del formulario con los datos del libro a editar
    else:
        form = BookForm(instance=book)
    return render(request, 'editar_libro.html', {'form': form, 'book': book})
```

U: --- views.py 20% L26 (Django company FlyC- WK SP yas ElDoc)

Si intento ingresar directamente a http://127.0.0.1:8000/crear_libro/ sin haber iniciado sesión con un usuario autorizado me redirige a <http://127.0.0.1:8000/home>



Si accedo a `crear_libros` con un usuario autorizado, se despliega:



3. Modelo y formulario:

El modelo Book en `models.py` define los campos necesarios, incluyendo titulo, autor, y valoracion, así como los campos de fecha de creación y modificación, que son útiles para rastrear los cambios. Además, el formulario BookForm en `forms.py` permite la entrada de datos para estos campos, con validaciones aplicadas en el formulario.

- Modelo Book en `models.py`

```
from django.db import models
import datetime # para manejar fechas

# Create your models here.
# https://docs.djangoproject.com/en/5.1/topics/db/models/
# https://www.w3schools.com/django/django_models.php
class Book(models.Model):
    # atributos
    titulo = models.CharField(max_length = 100, null = False)
    autor = models.CharField(max_length = 50, null = False)
    valoracion = models.IntegerField(help_text = 'Valoración entre 0 y 100')
    fecha_creacion = models.DateTimeField(default=datetime.datetime.now())
    fecha_modificacion = models.DateTimeField(default=datetime.datetime.now())

U:--- models.py      Top L8      (Django company FlyC- WK SP yas ElDoc)
```

- Formulario BookForm en `forms.py`

```
from django import forms
from .models import Book

# https://docs.djangoproject.com/en/5.1/topics/forms/
# https://docs.djangoproject.com/en/5.1/ref/forms/widgets/
class BookForm(forms.ModelForm):

    class Meta: # clase meta para definir características del objeto
        model = Book # modelo a utilizar
        fields = ['titulo', 'autor', 'valoracion']
        widgets = {
            'titulo' : forms.TextInput(
                attrs={
                    'class': 'form-control',
                    'placeholder': 'Ingrese el titulo del libro',
                    'required': True
                }),
            'autor' : forms.TextInput(
                attrs={
                    'class': 'form-control',
                    'placeholder': 'Ingrese el autor del libro',
                    'required': True
                }),
        },

-:--- forms.py      Top L22      (Django company FlyC- WK SP yas ElDoc)
```

4. Administración y visualización

El archivo `admin.py` registra el modelo `Book` y añade opciones de visualización y filtrado, lo cual facilita la administración de libros desde el panel de Django.

```
from .models import Book

# Register your models here.
# https://docs.djangoproject.com/en/5.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin

class BookAdmin(admin.ModelAdmin):
    readonly_fields = ('fecha_creacion', 'fecha_modificacion')
    list_display = ('titulo', 'autor', 'valoracion', 'rating')
    list_filter = ('autor', 'valoracion', 'fecha_modificacion')

    def rating(self, obj):
        if obj.valoracion < 1000:
            return 'Baja'
        elif 1000 <= obj.valoracion <= 2500:
            return 'Media'
        else:
            return 'Alta'

admin.site.register(Book, BookAdmin) # registro de los model separados por coma
```

-- admin.py Bot L4 (Django company FlyC- WK SP yas ElDoc)