

CUE: TEMPLATES EN DJANGO

Pablo Varas Salamanca

29 de octubre de 2024

[GitHub del proyecto](#)

DRILLING: FORMULARIO MODELFORM

EJERCICIO:

Para resolver este ejercicio, anteriormente debe haber revisado la lectura y los videos del CUE: Formularios en Django.

En la siguiente actividad deberás crear una vista que permita ingresar un libro con: título, autor y valoración.

Parte 1

1. URL: crear path para la ruta crear_libro/

>app/urls.py

```
{% extends 'layout.html' %}
{% block content %}
    <section class="container mt-5">
        <h2 class="text-center">Crear libro</h2>
        <div class="row justify-content-center">
            <div class="col-md-6">
                <form action="{% url 'crear_libro' %}" method="post">
                    {% csrf_token %}
                    {{ form.as_p }}
                    <input type="submit" value="Guardar" class="btn btn-primary w-100">
                </form>
            </div>
        </div>
    </section>
{% endblock %}
```

~:--- crear_libro.html All L6 (Web Emmet company-box company FlyC- SP yas)

2. View: crear una vista llamada crear_libro, que recibe un request y retorna el template book/crear_libro.html

>app/views.py

```
class IndexPageView(TemplateView): # un view o controlador con una clase
    template_name = 'index.html'

def lista_libros(request):
    libros = Book.objects.all()
    return render(request, 'lista_libros.html', {'libros': libros})

# https://docs.djangoproject.com/en/5.1/topics/forms/
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
            form.save() # guarda los datos en la base de datos
            return redirect('lista_libros')
    else: # si el metodo es GET
        form = BookForm()
        return render(request, 'crear_libro.html', {'form': form})
```

-- views.py Bot L8 (Django company-box company FlyC- SP yas ElDoc)

3. Formulario: crear un formulario para la creación de libros

>app/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 L23 (Django company-box company FlyC- SP yas ElDoc)

4. Template: crear una plantilla llamada crear_libro.html, que recibe el formulario del libro

```
>app/templates/crear_libro.html
```

```
{% extends 'layout.html' %}
{% block content %}
    <section class="container mt-5">
        <h2 class="text-center">Crear libro</h2>
        <div class="row justify-content-center">
            <div class="col-md-6">
                <form action="{% url 'crear_libro' %}" method="post">
                    {% csrf_token %}
                    {{ form.as_p }}
                    <input type="submit" value="Guardar" class="btn btn-primary w-100">
                </form>
            </div>
        </div>
    </section>
{% endblock %}
```

```
--:---- crear_libro.html All L1 (Web Emmet company-box company FlyC- SP yas)
```

5. Activar entorno.

```
source /home/pablo/pablo-env/bin/activate
```

```
pablo@debian: /media/DATAs/Dascargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7/drilling/book_/site_django$ source /home/pablo/pablo-env/bin/activate
(pablo-env) pablo@debian: /media/DATAs/Dascargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7/drilling/book_/site_django$
```

6. Realizar las migraciones con:

```
python manage.py makemigrations
```

```
python manage.py migrate
```

```
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7/drilling/book_/site_django$ py
thon manage.py makemigrations
No changes detected
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7/drilling/book_/site_django$ py
thon manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, book, contenttypes, 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 book.0001_initial... OK
  Applying sessions.0001_initial... OK
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7/drilling/book_/site_django$
```

7. Crear template footer.html.

```
>app/templates/footer.html
```

```
{% load bootstrap5 %}
{% block content %}
<!-- Footer -->
<footer class="d-none d-md-block text-center text-lg-start bg-white text-muted fixed-bottom">
  <!-- Section: Social media -->
  <section class="d-flex justify-content-center justify-content-lg-between p-4 border-bottom">
  </section>
  <section class="">
    <div class="container text-center text-md-start mt-5"> <!-- Grid row -->
      <div class="row mt-3"> <!-- Grid column -->
        <div class="col-md-3 col-lg-4 col-xl-3 mx-auto mb-4">
          <!-- Content -->
          <h6 class="text-uppercase fw-bold mb-4">
            <i class="fas fa-gem me-3 text-grayish"></i>Descripción
          </h6> <p> Desarrollo de Sitio Web de Django. </p>
        </div>
        <div class="col-md-2 col-lg-2 col-xl-2 mx-auto mb-4">
          <!-- Links -->
          <h6 class="text-uppercase fw-bold mb-4"> Productos </h6>
          <p> <a href="{% url 'lista_libros' %}" class="text-reset">Libros</a> </p>
        </div>
        <div class="col-md-4 col-lg-3 col-xl-3 mx-auto mb-md-0 mb-4">
          <!-- Links -->
        </div>
      </div>
    </div>
  </section>
</footer>
```

U: --- footer.html Top L1 (Web Emmet company-box company FlyC- SP yas)

8. Incluirlo en base.html

```
!DOCTYPE html
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  {% load bootstrap5 %}
  <!-- {# Load CSS and JavaScript #} -->
  {% bootstrap_css %}
  {% bootstrap_javascript %}
  <!-- {# Display django.contrib.messages as Bootstrap alerts #} -->
  {% bootstrap_messages %}
  <title>Book</title>
</head>
<body>
  <!-- se incluye el navbar como componente-->
  {% include 'navbar.html' %}
  {% block content %}
  {% endblock %}
  {% include 'footer.html' %}
</body>
</html>
```

U:--- base.html All L1 (Web Emmet company-box company FlyC- SP yas)

9. Ejecutar servidor, ubicarse en la carpeta del proyecto donde se encuentra el archivo =manage.py

python manage.py runserver

```
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7/drilling/book_/site_django$ py
thon manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 28, 2024 - 00:21:34
Django version 4.2.16, using settings 'site_django.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Y obtenemos :

Sitio Libros

Home Lista libros Crear Libro Disabled

Crear libro

Titulo:

Autor:

Valoracion:

Valoración entre 0 y 100

[Guardar](#)

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

© 2024 Copyright: [SiteWebDjango.com](#)

Parte 2

1. URL: crear path para la ruta editar_libro/

```
>app/urls.py
```

```
urlpatterns = [  
    path('editar_libro/<int:libro_id>', editar_libro, name="editar_libro")  
    # registrando ruta para editar libro  
]
```

```
from django.urls import path  
from . import views  
from .views import IndexPageView, lista_libros, crear_libro, editar_libro  
  
urlpatterns = [  
    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  
]
```

--:--- urls.py All L2 (Django company-box company FlyC- SP yas ElDoc)

2. Template: editar listar_libros.html y agregar boton para editar un libro que comunica con el path editar_libro/id

```
{% extends 'layout.html' %}

{% block content %}
    <section class="container mt-5">
        <h2>Lista Libros</h2>
        {% if libros %}
            <!-- tabla con libros -->
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th scope="col">ID</th>
                        <th scope="col">Titulo</th>
                        <th scope="col">Autor</th>
                        <th scope="col">Valoración</th>
                        <th scope="col">Acciones</th>
                    </tr>
                </thead>
                <tbody>
                    {% for libro in libros %}
                        <tr>
                            <td>{{ libro.id }}</td>
                            <td>{{ libro.titulo }}</td>
                            <td>{{ libro.autor }}</td>
```

U:--- lista_libros.html Top L22 (Web Emmet company-box company FlyC- SP yas)

```
{% extends 'layout.html' %}
{% block content %}
    <section class="container mt-5">
        <h2 class="text-center">Editar libro</h2>
        <div class="row justify-content-center">
            <div class="col-md-6">
                <form action="{% url 'editar_libro' libro_id=libro_id %}" method="post">
                    {% csrf_token %}
                    {{ form.as_p }}
                    <input type="submit" value="Guardar" class="btn btn-primary w-100">
                </form>
            </div>
        </div>
    </section>
{% endblock %}
```

:-:--- editar_libro.html All L5 (Web Emmet company-box company FlyC- SP yas)

En la vista editar_libro.html, añadir el ícono de "Guardar" en el botón como se mostró:

```
<input type="submit" value="Guardar" class="btn btn-primary w-100">
```

3. View: crear una vista llamada `editar_libro`, que recibe un request y retorna el template `book/editar_libro.html`

>app/views.py.

Guías en:

- <https://docs.djangoproject.com/en/5.1/topics/http/shortcuts/>
- https://www.geeksforgeeks.org/get_object_or_404-method-in-django-models/

```
        return redirect('lista_libros')
    else: # si el metodo es GET
        form = BookForm()
        return render(request, 'crear_libro.html', {'form': form})

def editar_libro(request, libro_id):
    # https://docs.djangoproject.com/en/5.1/topics/http/shortcuts/
    # https://www.geeksforgeeks.org/get_object_or_404-method-in-django-models/
    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
        if form.is_valid(): # si el formulario es valido
            form.save() # guarda los datos en la base de datos
            return redirect('lista_libros') # redirecciona a lista de libros
        else:
            form = BookForm(instance=book) # instancia del formulario con los datos del libro a editar
            return render(request, 'editar_libro.html', {'form': form, 'libro_id': libro_id}) # renderiza la vista para editar el libro

-:--- views.py      Bot L25      (Django company-box company FlyC- SP yas ElDoc)
```

4. Template: crear una plantilla llamada `editar_libro.html`, que recibe el formulario del libro

>app/templates/editar_libro.html

```
{% extends 'layout.html' %}
{% block content %}
    <section class="container mt-5">
        <h2 class="text-center">Editar libro</h2>
        <div class="row justify-content-center">
            <div class="col-md-6">
                <form action="{% url 'editar_libro' libro_id=libro_id %}" method="post">
                    {% csrf_token %}
                    {{ form.as_p }}
                    <input type="submit" value="Guardar" class="btn btn-primary w-100">
                </form>
            </div>
        </div>
    </section>
{% endblock %}

-:--- editar_libro.html  All L1      (Web Emmet company-box company FlyC- SP yas)
```


5. Fontawesome: agregar icono para boton de guardar cambios en la vista editar_libro.

<https://fontawesome.com/>

En el archivo base.html, el ícono de fontawesome se agrega en el <head>

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  {% load bootstrap5 %}
  <!-- Load CSS and JavaScript -->
  {% bootstrap_css %}
  {% bootstrap_javascript %}
  <!-- Display django.contrib.messages as Bootstrap alerts -->
  {% bootstrap_messages %}
  <!-- fontawesome css -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.6.0/css/all.min.css" integrity="sha512-Kc323vGBEqzTmouAECnVceyQyqdsSiqLQISBL29aUW4U/M7pSPA/gEUZQqv1cwx40nYxTxve5UMg5GT6L4JJg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
  <title>Book</title>
</head>
```

Además, en el <body> se agregó el script de JavaScript de Font Awesome:

```
<body>
  <!-- se incluye el navbar como componente-->
  {% include 'navbar.html' %}
  {% block content %}
  {% endblock %}
  {% include 'footer.html' %}
  <!-- fontawesome js -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.6.0/js/all.min.js" integrity="sha512-6sSYJqDRZGk3b+Yfddh83MzmuP9R7X1QZ6g5aIXhRvR1Y/N/P47jmnkENm7YL3oqsmI6AK+V6AD99uWDnIw==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>
</body>
```

6. Activar entorno

source /home/pablo/pablo-env/bin/activate

```
pablo@debian: /media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.1/book_/site_django$ source /home/pablo/pa
(pablo-env) pablo@debian: /media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.1/book_/site_django$
```

7. Realizar las migraciones de la base de datos

```
python manage.py makemigrations
```

```
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.1/book_/site_django$ python manage.py makemigrations
No changes detected
```

```
python manage.py migrate
```

```
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.1/book_/site_django$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, book, contenttypes, 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 book.0001_initial... OK
  Applying sessions.0001_initial... OK
```

8. Ejecutar servidor, ubicarse en la carpeta del proyecto donde se encuentra el archivo manage.py

```
python manage.py runserver
```

Creamos un libro y seleccionamos el botón al costado del libro:



Se nos muestra la url `http://127.0.0.1:8000/editar_libro/1`

Sitio Libros

Home Lista libros Crear Libro Disabled

Editar libro

Titulo:

Autor:

Valoracion:

Valoración entre 0 y 100

[Guardar](#)

DESCRIPCIÓN
Desarrollo de Sitio Web de Django.

PRODUCTOS
[Libros](#)

CONTACTO
🏠 Dirección, calle, País
✉ info@example.com
☎ +01 234 567 88

© 2024 Copyright: [SiteWebDjango.com](#)

Parte 3

1. URL: crear path para la ruta `eliminar_libro/`

>app/urls.py

```
urlpatterns = [  
    path('eliminar_libro/<int:libro_id>', eliminar_libro, name="eliminar_libro") # registrando r  
]
```

```
from django.urls import path  
from . import views  
# importar views  
from .views import IndexPageView, lista_libros, crear_libro, editar_libro, eliminar_libro  
  
urlpatterns = [  
    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  
]
```

2. Template: editar lista_libros.html y agregar boton para eliminar un libro que comunica con el path eliminar_libro/id

```
<td>{{ libro.valoracion }}</td>
<td>
  <a class="btn" href="{% url 'editar_libro' libro_id=libro.id %}" role="button" data-bs-toggle="modal" data-bs-target="#modalEditar" data-bs-placement="top" title="Editar">
    <i class="fa fa-edit text-primary"></i>
  </a>
  <a class="btn" href="{% url 'eliminar_libro' libro_id=libro.id %}" role="button" data-bs-toggle="modal" data-bs-target="#modalEliminar" data-bs-placement="top" title="Eliminar">
    <i class="fa fa-edit text-danger"></i>
  </a>
</td>
</tr>
{% empty %}
<tr>
  <td colspan="4">No hay libros disponibles</td>
</tr>
{% endfor %}
</tbody>
</table>
{% else %}
<!-- No hay libros -->
<p>No hay libros disponibles</p>
{% endif %}
```

U:--- lista_libros.html 48% L44 (Web Emmet company-box company FlyC- SP yas)

3. Fontawesome: agregar icono para boton de guardar cambios en la vista listar_libros.html

><https://fontawesome.com/>

```
<td>{{ libro.titulo }}</td>
<td>{{ libro.autor }}</td>
<td>{{ libro.valoracion }}</td>
<td>
  <a class="btn" href="{% url 'editar_libro' libro_id=libro.id %}" role="button" data-bs-toggle="modal" data-bs-target="#modalEditar" data-bs-placement="top" title="Editar">
    <i class="fa fa-edit text-primary"></i>
  </a>
  <a class="btn" href="{% url 'eliminar_libro' libro_id=libro.id %}" role="button" data-bs-toggle="modal" data-bs-target="#modalEliminar" data-bs-placement="top" title="Eliminar">
    <i class="fa fa-edit text-danger"></i>
  </a>
</td>
</tr>
{% empty %}
<tr>
  <td colspan="4">No hay libros disponibles</td>
</tr>
{% endfor %}
</tbody>
</table>
{% else %}
<!-- No hay libros -->
```

U:--- lista_libros.html 43% L32 (Web Emmet company-box company FlyC- SP yas)

4. View: crear una vista llamada eliminar_libro, que recibe un request y retorna el template book/editar_libro.html

```
end
    if request.method == 'POST':
        form = BookForm(request.POST, instance=book) # instancia del formulario con los datos del libro a editar
    elif request.method == 'GET':
        if form.is_valid(): # si el formulario es valido
            form.save() # guarda los datos en la base de datos
            return redirect('lista_libros') # redirecciona a lista de libros
        else:
            form = BookForm(instance=book) # instancia del formulario con los datos del libro a editar
            return render(request, 'editar_libro.html', {'form': form, 'libro_id': libro_id}) # renderiza la vista para editar el libro

def eliminar_libro(request, libro_id):
    book = get_object_or_404(Book, pk=libro_id) # obteniendo el libro a eliminar en la db o status 404 not found
    book.delete() # eliminando el libro de la base de datos
    messages.info(request, 'Libro eliminado correctamente') # muestra un mensaje
    return redirect('lista_libros') # redirecciona a lista de libros
```

U:*** views.py Bot L45 (Django company-box company FlyC- SP yas ElDoc)

>app/views.py

- Guías en los enlaces:

- <https://docs.djangoproject.com/en/5.1/topics/http/shortcuts/>
- https://www.geeksforgeeks.org/get_object_or_404-method-in-django-models/

>app/views.py

5. Importar librerías para mensaje

```
from django.contrib import messages
```

```
from django.http import HttpResponseRedirect, HttpResponseRedirect
from django.views.generic import TemplateView
from django.shortcuts import render, redirect, get_object_or_404
from django.urls import reverse
from django.contrib import messages
from .models import Book
from .forms import BookForm

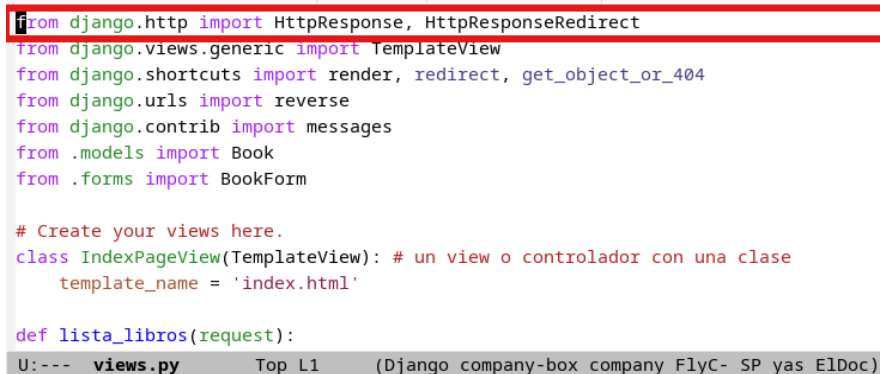
# Create your views here.
class IndexPageView(TemplateView): # un view o controlador con una clase
    template_name = 'index.html'

def lista_libros(request):
```

U:--- views.py Top L1 (Django company-box company FlyC- SP yas ElDoc)

6. Importar librerías HttpResponseRedirect

```
from django.http import HttpResponseRedirect
```



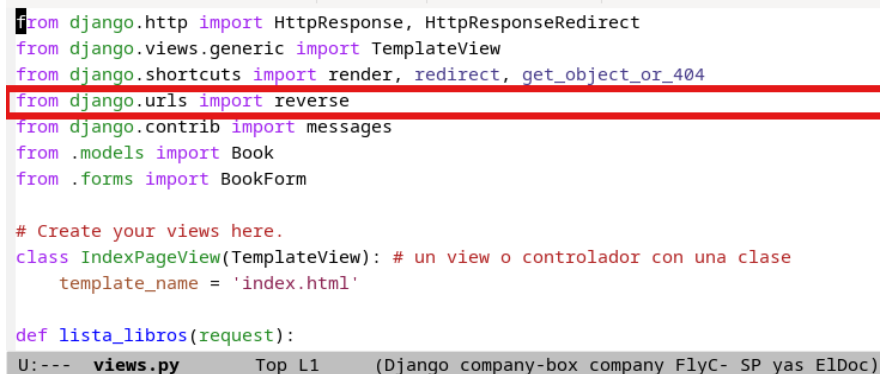
```
from django.http import HttpResponseRedirect
from django.views.generic import TemplateView
from django.shortcuts import render, redirect, get_object_or_404
from django.urls import reverse
from django.contrib import messages
from .models import Book
from .forms import BookForm

# Create your views here.
class IndexPageView(TemplateView): # un view o controlador con una clase
    template_name = 'index.html'

def lista_libros(request):
    U:--- views.py Top L1 (Django company-box company FlyC- SP yas ElDoc)
```

7. Importar librerías para reverse

```
from django.urls import reverse
```



```
from django.http import HttpResponseRedirect
from django.views.generic import TemplateView
from django.shortcuts import render, redirect, get_object_or_404
from django.urls import reverse
from django.contrib import messages
from .models import Book
from .forms import BookForm

# Create your views here.
class IndexPageView(TemplateView): # un view o controlador con una clase
    template_name = 'index.html'

def lista_libros(request):
    U:--- views.py Top L1 (Django company-box company FlyC- SP yas ElDoc)
```

8. Templates: agregar en los templates el tag para desplegar mensajes mediante django.contrib.messages

Display django.contrib.messages as Bootstrap alerts

```
{% bootstrap_messages %}
```



```
{% extends 'layout.html' %}
{% load bootstrap5 %}
{% block content %}
    <section class="container mt-5">
        <h2>Lista Libros</h2>
        <!-- {# Display django.contrib.messages as Bootstrap alerts #} -->
        {% bootstrap_messages %}
        {% if libros %}
            <!-- tabla con libros -->
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th scope="col">ID</th>
    U:--- lista_libros.html Top L3 Git-master (Web Emmet company-box company FlyC- SP yas)
```

9. Views: modificar los views para desplegar los mensajes necesarios de acuerdo al resultado de la operación

9.1 Mensaje para editar en views.py

Con la línea `messages.error(request, 'Modifica los datos de ingreso')` se muestra un mensaje de error al usuario, informándole que debe modificar los datos ingresados. El mensaje se agrega al sistema de mensajes de Django y estará disponible en la siguiente página cargada para que el usuario lo vea.

Con `HttpResponseRedirect` Redirige al usuario a otra URL. Se redirecciona a la página de creación de libros (`crear_libro`). `reverse('crear_libro')`: Usa el nombre de la URL (`crear_libro`) para construir la dirección completa.

```
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
        if form.is_valid(): # si el formulario es valido
            form.save() # guarda los datos en la base de datos
            return redirect('lista_libros') # redirecciona a lista de libros
        else: # si el formulario no es valido
            messages.error(request, 'Modifica los datos de ingreso') # muestra un mensaje
            return HttpResponseRedirect(reverse('editar_libro', args=[book.id])) # redirecciona a la pagina para editar el libro
    else: # si el metodo es GET
        form = BookForm(instance=book) # instancia del formulario con los datos del libro a editar
        return render(request, 'editar_libro.html', {'form': form, 'libro_id': libro_id}) # renderiza la vista para editar el libro
```

9.2 Mensaje para eliminar en views.py

Este código utiliza el sistema de mensajes de Django para mostrar un mensaje de éxito después de guardar los datos.

```
def eliminar_libro(request, libro_id):
    book = get_object_or_404(Book, pk=libro_id) # obteniendo el libro a eliminar en la db o status 404 not found
    book.delete() # eliminando el libro de la base de datos
    messages.info(request, 'Libro eliminado correctamente') # muestra un mensaje
    return redirect('lista_libros') # redirecciona a lista de libros
```

10. Activar entorno

```
source /home/pablo/pablo-env/bin/activate
```

```
pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.2/book_/site_django$ source /home/pablo/pa
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.2/book_/site_django$
```

11. Se realizan las migraciones

```
python manage.py makemigrations
```

```
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.2/book_/site_django$ python ma
nage.py makemigrations
No changes detected
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.2/book_/site_django$
```

python manage.py migrate

```
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.2/book_/site_django$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, book, contenttypes, 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 book.0001_initial... OK
  Applying sessions.0001_initial... OK
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.2/book_/site_django$
```

12. Ejecutar el servidor, ubicarse en la carpeta del proyecto donde se encuentra el archivo manage.py

python manage.py runserver

```
(pablo-env) pablo@debian:/media/DATA/Descargas/BOOTCAMP/Bootcamp/VSC/M6_Django/S7.2/book_/site_django$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 29, 2024 - 16:18:34
Django version 4.2.16, using settings 'site_django.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```


- Se despliega en http://127.0.0.1:8000/lista_libros/

The screenshot shows a web browser at the URL `127.0.0.1:8000/lista_libros/`. The page title is "Sitio Libros". The navigation bar includes links for "Home", "Lista libros", "Crear Libro", and "Disabled". The main content area is titled "Lista Libros" and features a green success message: "Libro creado correctamente". Below this is a table with the following data:

ID	Título	Autor	Valoración	Acciones
1	La Araucana	Alonso de Ercilla y Zúñiga	100	Editar Eliminar

Below the table is a black button labeled "Eliminar". The footer contains three sections: "DESCRIPCIÓN" (Desarrollo de Sitio Web de Django), "PRODUCTOS" (Libros), and "CONTACTO" (Dirección, calle, Pais; info@example.com; + 01 234 567 88). The footer also includes the copyright notice "© 2024 Copyright: SiteWebDjango.com".

- Si seleccionamos Eliminar se elimina el libro y se muestra el mensaje "Libro eliminado correctamente"

The screenshot shows the same web browser at the URL `127.0.0.1:8000/lista_libros/`. The page title is "Sitio Libros". The navigation bar includes links for "Home", "Lista libros", "Crear Libro", and "Disabled". The main content area is titled "Lista Libros" and features a light blue success message: "Libro eliminado correctamente". Below this message, the text "No hay libros disponibles" is displayed. The footer contains the same three sections as the previous screenshot: "DESCRIPCIÓN" (Desarrollo de Sitio Web de Django), "PRODUCTOS" (Libros), and "CONTACTO" (Dirección, calle, Pais; info@example.com; + 01 234 567 88). The footer also includes the copyright notice "© 2024 Copyright: SiteWebDjango.com".