Informe de Laboratorio 08

ema: Django: Relaciones de uno a muchos, muchos a muchos y impresión de pdf y emai

Nota

Estudiante	Escuela	Asignatura
Fernando Miguel Garambel Marín fgarambel@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Laboratorio de Programación Web 2 Semestre: III Código: 1701212

Laboratorio	Tema	Duración
08	Django: Relaciones de uno a muchos, muchos a muchos y impresión de pdf y emails	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 24 de mayo 2024	Al 14 de junio 2024

1. Objetivos

- Implementar en una aplicación en Django el manejo de Bases de datos.
- Utilizar una tabla y relacionarla con muchas tablas.
- Utilizar muchas tablas y relacionarlas con muchas tablas.
- Implementar el envío de emails y la impresión de pdfs desde una aplicación Django

2. Actividades

- Crear un proyecto en Django
- Siga los pasos de los videos para poder implementar la aplicación de Relación de Uno a muchos en una Base de Datos, muchos a muchos, impresión de pdfs y envío de emails.
- Use git y haga los commits necesarios para manejar correctamente la aplicación.

3. Ejercicio Propuestos

- Deberán replicar la actividad de los videos donde se trabaja con Relacion de uno a muchos, de muchos a muchos, impresión de pdfs y envío de emails; adecuándolo desde un proyecto en blanco Django.
- Para ello crear una carpeta dentro del proyecto github colaborativo con el docente, e informar el link donde se encuentra.
- Eres libre de agregar CSS para decorar tu trabajo.
- Ya sabes que el trabajo con Git es obligatorio. Revisa los videos entregados.

4. Equipos, materiales y temas utilizados

- Sistema operativo de 64 bits, procesador basado en x64.
- Latex.
- git version 2.41.0.windows.1
- Cuenta en GitHub con el correo institucional.

5. URL Github, Video

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/FernandoGarambelM/Lab08_Django_relaciones.git`
- URL para el video flipgrid
- `https://flip.com/s/pvZBG1NhvCRn`

6. Replicando la actividad del video - Relacion de uno a mucho y de muchos a muchos

- Primero ingresamos al entorno virtual

Listing 1: Activar el ambiente virtual

```
    test\Scripts\activate
```

- Creamos los modelos que vamos a usar

Listing 2: Codigo de models.py

```
    from django.db import models

# Create your models here.
class Lenguage(models.Model):
    name = models.CharField(max_length=10)

    def __str__(self):
        return self.name

class Framework(models.Model):
    name = models.CharField(max_length=10)
    lenguage = models.ForeignKey(Lenguage, on_delete=models.CASCADE)

    def __str__(self):
        return self.name
class Movie(models.Model):
    name = models.CharField(max_length=10)

    def __str__(self):
        return self.name

class Character(models.Model):
    name = models.CharField(max_length=10)
    movies = models.ManyToManyField(Movie)

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

- Es importante aclarar las relaciones que se crearon en estos modelos:
- One-to-Many (Uno a muchos)
- Entre Framework y Lenguage hay una relación uno a muchos. Cada framework está asociado con un único lenguaje, pero un lenguaje puede tener múltiples frameworks. Esto se logra mediante la clave foránea (ForeignKey) en el modelo Framework.
- Many-to-Many (Muchos a muchos)
- Entre Character y Movie hay una relación muchos a muchos. Un personaje puede aparecer en múltiples películas y una película puede tener múltiples personajes. Esto se logra mediante el campo ManyToManyField en el modelo Character.

- En resumen: Lenguage tiene una lista de Frameworks asociados, Framework pertenece a un solo Lenguage, Character puede estar en varias Movies y viceversa, Movie puede tener varios Characters y viceversa, Estas relaciones permiten modelar de manera eficiente la estructura y las interconexiones de los datos.

Listing 3: Ingresamos a shell de python

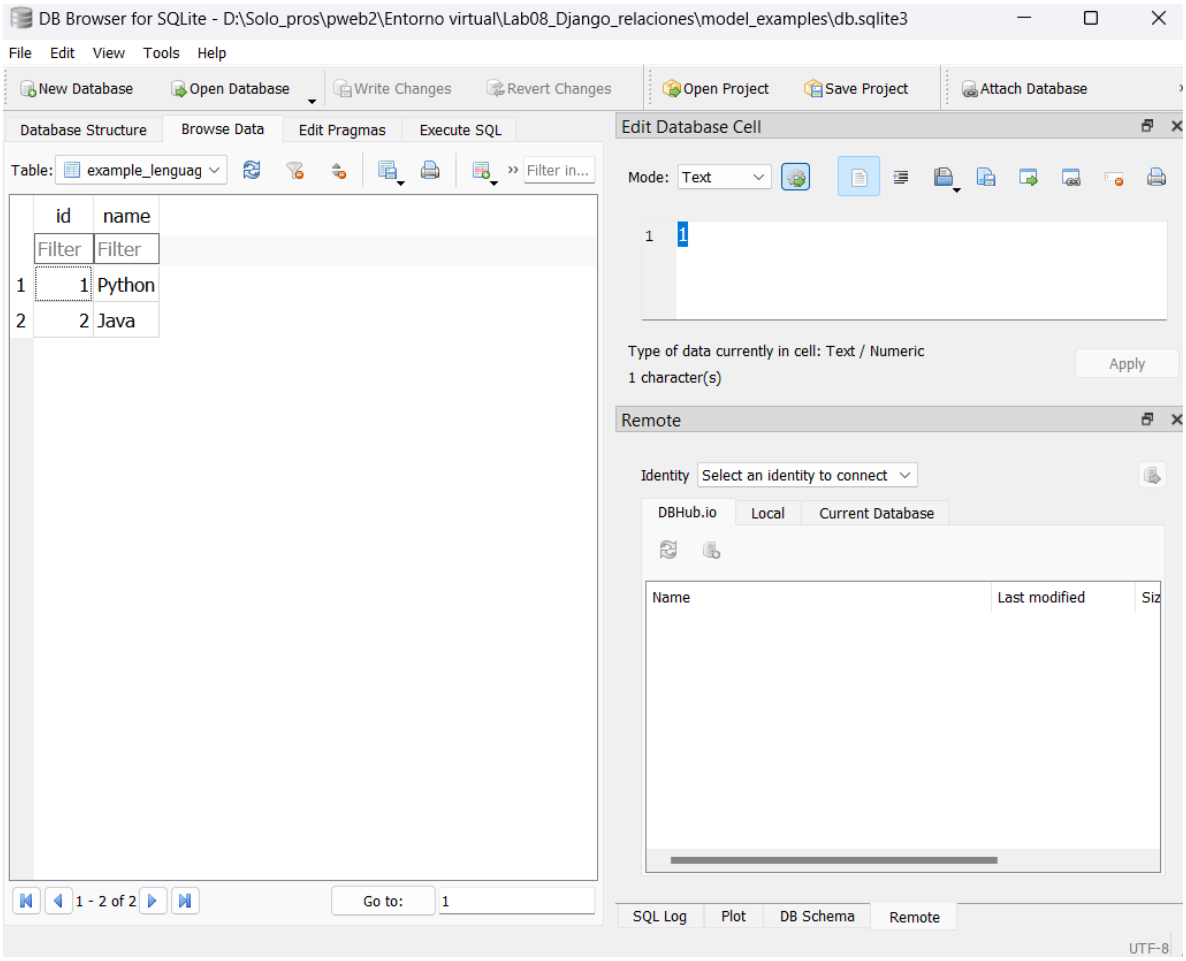
```
python manage.py shell
```

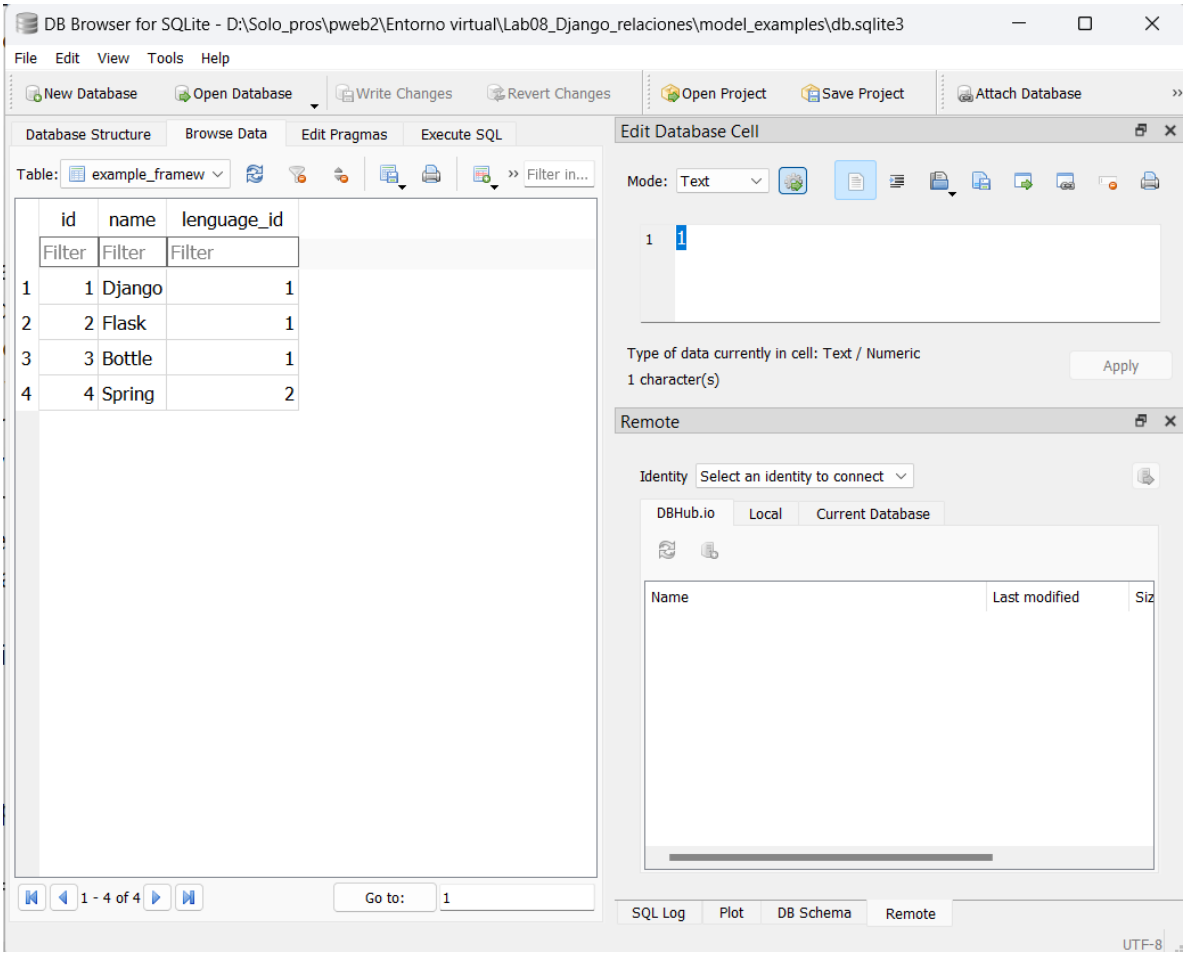
- Ahi ingresaremos los datos para las tablas en base de datos
- Realizamos migraciones para enlazar la base de datos

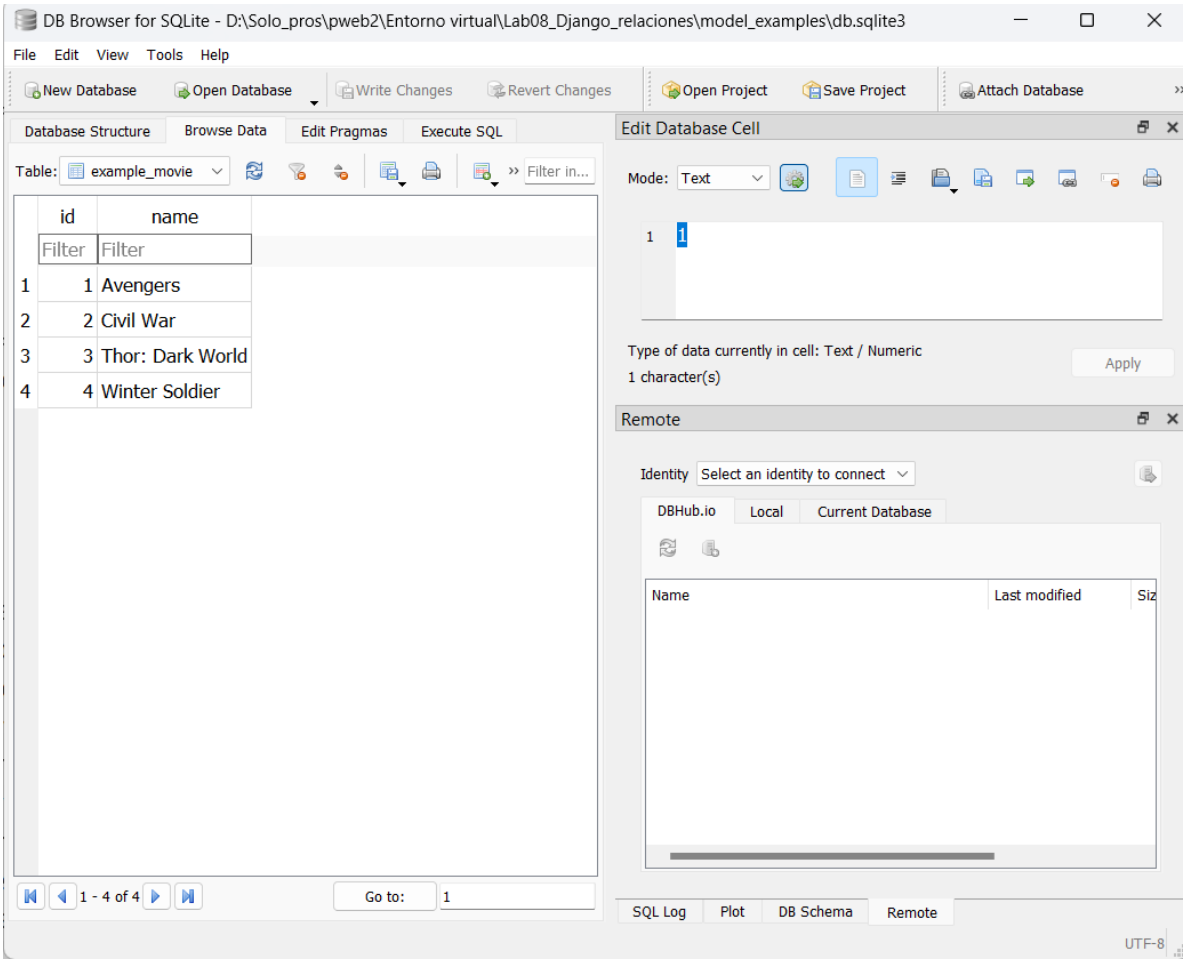
Listing 4: Codigo para realizar migraciones

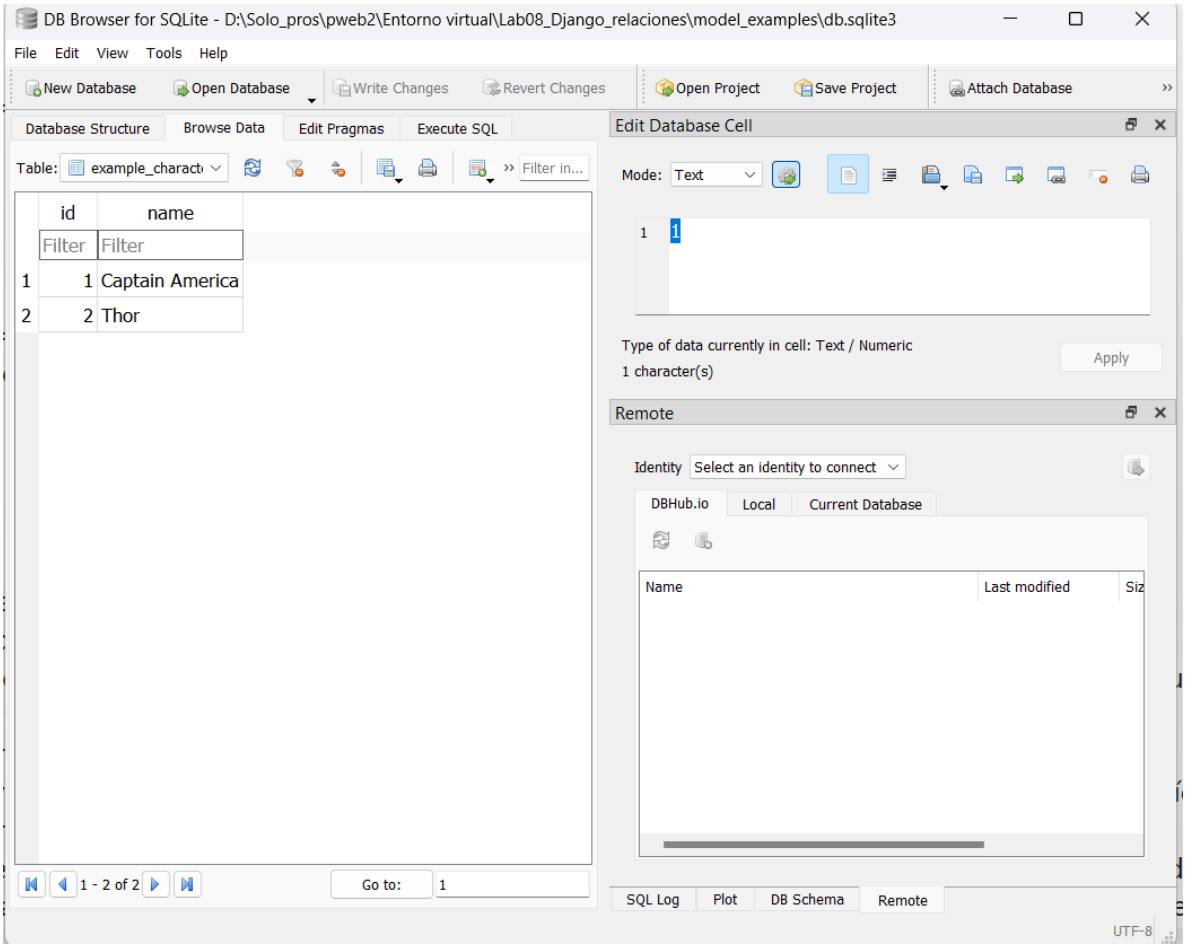
```
python manage.py makemigrations
python manage.py migrations
```

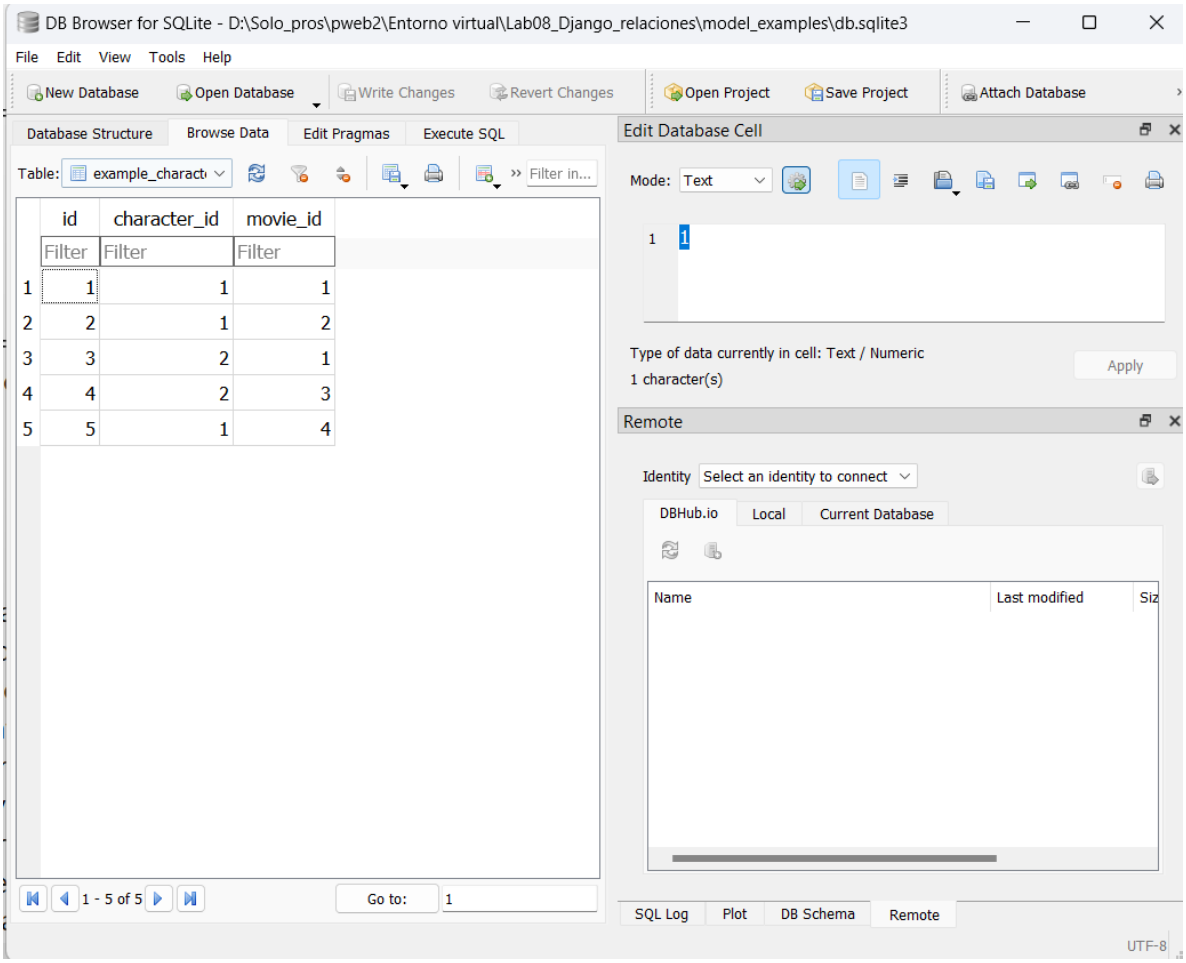
- Despues de seguir estos pasos la plantilla ya funciona por completo
- Ahora veremos las bases de datos




DB Browser for SQLite - D:\Solo_pros\pweb2\Entorno virtual\Lab08_Django_relaciones\model_examples\db.sqlite3
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: example_framew
Filter in...
id name lenguage_id
Filter Filter Filter
1 1 Django 1
2 2 Flask 1
3 3 Bottle 1
4 4 Spring 2
1 - 4 of 4
Go to: 1
Edit Database Cell
Mode: Text
1 1
Type of data currently in cell: Text / Numeric
1 character(s)
Apply
Remote
Identity Select an identity to connect
DBHub.io
Local
Current Database
Name
Last modified
Siz
SQL Log
Plot
DB Schema
Remote
UTF-8


DB Browser for SQLite - D:\Solo_pros\pweb2\Entorno virtual\Lab08_Django_relaciones\model_examples\db.sqlite3
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: example_movie
Filter in...
id
name
Filter
1 1 Avengers
2 2 Civil War
3 3 Thor: Dark World
4 4 Winter Soldier
1 - 4 of 4
Go to: 1
Edit Database Cell
Mode: Text
Type of data currently in cell: Text / Numeric
1 character(s)
Apply
Remote
Identity
Select an identity to connect
DBHub.io
Local
Current Database
Name
Last modified
SQL Log
Plot
DB Schema
Remote
UTF-8


DB Browser for SQLite - D:\Solo_pros\pweb2\Entorno virtual\Lab08_Django_relaciones\model_examples\db.sqlite3
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: example_charact
Filter in...
id
name
Filter
Filter
1 1 Captain America
2 2 Thor
1 - 2 of 2
Go to: 1
Edit Database Cell
Mode: Text
1 1
Type of data currently in cell: Text / Numeric
1 character(s)
Apply
Remote
Identity Select an identity to connect
DBHub.io
Local
Current Database
Name
Last modified
Siz
SQL Log
Plot
DB Schema
Remote
UTF-8



7. Impresion de pdf y envio de email en django

- Instalamos xhtml12pdf para poder descargar pdfs

Listing 5: Instalar xhtml12pdf

```
pip install xhtml12pdf
```

- Luego creamos views.py

Listing 6: Código de views.py

```
from django.http import HttpResponse
from django.core.mail import send_mail
from django.template.loader import get_template
from django.shortcuts import render
from xhtml2pdf import pisa
from .models import Lenguage, Framework, Movie, Character
from django.core.mail import EmailMessage
from io import BytesIO
```

```
9
10  def index(request):
11      return HttpResponse("Hello, world!")
12
13  def generar_pdf(request):
14      template_path = 'plantilla_pdf.html'
15      context = {
16          'lenguages': Lenguage.objects.all(),
17          'frameworks': Framework.objects.all(),
18          'movies': Movie.objects.all(),
19          'characters': Character.objects.all()
20      }
21
22      response = HttpResponse(content_type='application/pdf')
23      response['Content-Disposition'] = 'attachment; filename="datos_modelos.pdf"'
24
25      template = get_template(template_path)
26      html = template.render(context)
27
28      pisa_status = pisa.CreatePDF(html, dest=response)
29
30      if pisa_status.err:
31          return HttpResponse('Hubo un error al generar el PDF: %s' % pisa_status.err)
32      return response
33
34  def enviar_correo(request):
35      context = {
36          'lenguages': Lenguage.objects.all(),
37          'frameworks': Framework.objects.all(),
38          'movies': Movie.objects.all(),
39          'characters': Character.objects.all()
40      }
41
42      if request.method == 'POST':
43          asunto = request.POST['asunto']
44          mensaje = request.POST['mensaje']
45          destinatario = request.POST['destinatario']
46          remitente = 'fgarambel@unsa.edu.pe'
47
48          if 'send_plain' in request.POST:
49              try:
50                  send_mail(asunto, mensaje, remitente, [destinatario])
51                  return HttpResponse('Correo enviado exitosamente')
52              except Exception as e:
53                  return HttpResponse(f'Error al enviar el correo: {str(e)}')
54
55          elif 'send_pdf' in request.POST:
56              template_path = 'plantilla_pdf.html'
57              template = get_template(template_path)
58              html = template.render(context)
59
60              result = BytesIO()
61              pisa_status = pisa.CreatePDF(html, dest=result)
62
63              if pisa_status.err:
64                  return HttpResponse('Hubo un error al generar el PDF: %s' % pisa_status.err)
```

```
65
66              email = EmailMessage(
67                  asunto,
68                  mensaje,
69                  remitente,
70                  [destinatario]
71              )
72
73              email.attach('datos_modelos.pdf', result.getvalue(), 'application/pdf')
74              email.send()
75
76              return HttpResponse('Correo con PDF enviado exitosamente')
77
78      return render(request, 'enviar_correo.html', context)
79
80  def enviar_pdf_por_correo(request):
81      template_path = 'plantilla_pdf.html'
82      context = {
83          'lenguages': Lenguage.objects.all(),
84          'frameworks': Framework.objects.all(),
85          'movies': Movie.objects.all(),
86          'characters': Character.objects.all()
87      }
88
89      template = get_template(template_path)
90      html = template.render(context)
91
92      result = BytesIO()
93      pisa_status = pisa.CreatePDF(html, dest=result)
94
95      if pisa_status.err:
96          return HttpResponse('Hubo un error al generar el PDF: %s' % pisa_status.err)
97
98      email = EmailMessage(
99          'Datos de Modelos',
100         'Adjunto encontraras el PDF con los datos de los modelos.',
101         'tu_correo@gmail.com',
102         ['destinatario@gmail.com']
103     )
104
105     email.attach('datos_modelos.pdf', result.getvalue(), 'application/pdf')
106     email.send()
107
108     return HttpResponse('Correo enviado exitosamente')
```

- Generación de PDFs: Genera un PDF con datos de los modelos y descarga directamente (generar pdf) o envia por correo (enviar pdf por correo).
- Envío de correos: Enviar correos de texto plano o con un PDF adjunto, según lo que elija el usuario en el formulario (enviar correo).
- Los siguientes templates usados son:

Listing 7: Código de views.py

```
<!DOCTYPE html>
<html>
<head>
    <title>Enviar Correo Electronico</title>
    {% load static %}
    <link rel="stylesheet" type="text/css" href="{% static 'css/styles.css' %}">
</head>
<body>
    <div class="container">
        <h1>Enviar Correo Electronico</h1>
        <form method="post">
            {% csrf_token %}
            <label for="asunto">Asunto:</label>
            <input type="text" id="asunto" name="asunto" required><br>
            <label for="mensaje">Mensaje:</label>
            <textarea id="mensaje" name="mensaje" required></textarea><br>
            <label for="destinatario">Destinatario:</label>
            <input type="email" id="destinatario" name="destinatario" required><br>
            <button type="submit" name="send_plain">Enviar</button>
            <button type="submit" name="send_pdf">Enviar con PDF</button>
        </form>

        <h2>Lista de elementos a enviar</h2>
        <h3>Lenguages</h3>
        <ul>
            {% for lenguage in lenguages %}
                <li>{{ lenguage.name }}</li>
            {% endfor %}
        </ul>

        <h3>Frameworks</h3>
        <ul>
            {% for framework in frameworks %}
                <li>{{ framework.name }} ({{ framework.lenguage.name }})</li>
            {% endfor %}
        </ul>

        <h3>Movies</h3>
        <ul>
            {% for movie in movies %}
                <li>{{ movie.name }}</li>
            {% endfor %}
        </ul>

        <h3>Characters</h3>
        <ul>
            {% for character in characters %}
                <li>{{ character.name }} ({{ character.movies.all|join:", " }})</li>
            {% endfor %}
        </ul>
    </div>
</body>
</html>
```

Listing 8: Código de views.py

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Datos de Modelos</title>
</head>
<body>
    <h1>Datos de Lenguages</h1>
    <ul>
        {% for lenguage in lenguages %}
            <li>{{ lenguage.name }}</li>
        {% endfor %}
    </ul>

    <h1>Datos de Frameworks</h1>
    <ul>
        {% for framework in frameworks %}
            <li>{{ framework.name }} - {{ framework.lenguage.name }}</li>
        {% endfor %}
    </ul>

    <h1>Datos de Movies</h1>
    <ul>
        {% for movie in movies %}
            <li>{{ movie.name }}</li>
        {% endfor %}
    </ul>

    <h1>Datos de Characters</h1>
    <ul>
        {% for character in characters %}
            <li>{{ character.name }} - {{ character.movies.all|join:", " }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

8. Referencias

- https://docs.djangoproject.com/es/3.2/
- https://docs.djangoproject.com/es/3.2/ref/models/fields/#field-types