Informe de Laboratorio 07

Tema: Relaciones de uno a muchos, muchos a muchos y impresion de pdf y emails

Nota

Estudiantes	Escuela	Asignatura
Daniela Choquecondo Aspilcueta	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III

Laboratorio	Tema	Duración
07	Relaciones de uno a muchos, muchos a muchos y impresion de pdf y emails	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 04 Julio 2023	Al 14 Julio 2023

1. REPOSITORIO DE GITHUB:

- Link del Repositorio en github: `https://github.com/Dan-Ar5/Pweb-Lab07.git`

2. OBJETIVOS:

- Desarrollarr los ejercicios en los videos presentados.
- Utilizar diferentes librerias para lograr un resultado optimo.
- Aprender y produndizar más el Framework Web Django.
- Comprender más sobre relaciones en las Bases de Datos.

3. TEMAS:

- Proyectos de Django
- Aplicaciones en Django
- Base de Datos
- Creación de PDF
- Envío de Email automatico

4. ACTIVIDADES:

Reproducir las actividades de los videos donde trabajamos:

1. Relación de uno a muchos
2. Relación muchos a muchos
3. Impresión de pdfs
4. Envio de emails
5. Crear su video Flipgrid

5. EJERCICIOS PROPUESTOS:

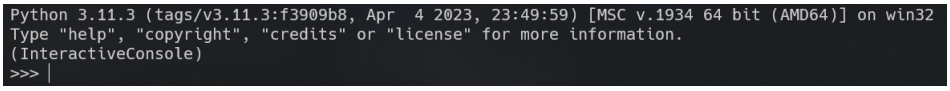
1. Es necesario recrear las acciones presentadas en los videos, donde se ingresan datos y se realizan consultas en una base de datos utilizando Django. Estas acciones se llevan a cabo en una relación de uno a muchos, permitiendo vincular múltiples elementos a un solo elemento principal

   Video de insercion de datos en una BD en relación de uno a muchos (clic)

   Video de querys en una BD en relación de uno a muchos (clic)

2. Luego de replicar el código del video, ingresamos al shell por defecto de Django con el siguiente comando.

```
python .\manage.py shell
```



3. Insertar datos:

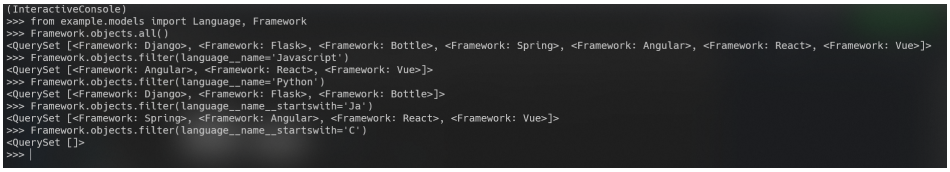
```
from example.models import Language, Framework
javascript = Language(name = 'Javascript')
javascript.save()
angular = Framework(name = 'Angular')
react = Framework(name = 'React')
javascript
angular.language = javascript
react.language = javascript
angular.save()
react.save()
vue = Framework(name='Vue', language=javascript)
vue.save()
```

```
(InteractiveConsole)
>>> from example.models import Language, Framework
>>> javascript = Language(name = 'Javascript')
>>> javascript.save()
>>> angular = Framework(name = 'Angular')
>>> react = Framework(name = 'React')
>>> javascript
<Language: Javascript>
>>> angular.language = javascript
>>> react.language = javascript
>>> angular.save()
>>> react.save()
>>> javascript
<Language: Javascript>
>>> vue = Framework(name='Vue', language=javascript)
>>> vue.save()
```

4. Cambios en la base de datos:

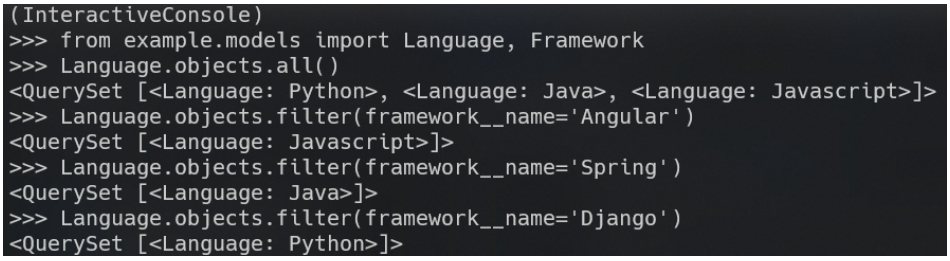
	id	name	language_id
	Filtro	Filtro	Filtro
1	1	Django	1
2	2	Flask	1
3	3	Bottle	1
4	4	Spring	2
5	5	Angular	3
6	6	React	3
7	7	Vue	3

5. Confirmando la relación de Uno a Muchos:




```
from example.models import Language, Framework
Framework.objects.all()
Framework.objects.filter(language__name='Javascript')
Framework.objects.filter(language__name='Python')
Framework.objects.filter(language__name__startswith='Ja')
Framework.objects.filter(language__name__startswith='C')
```

6. Tabla de Language:



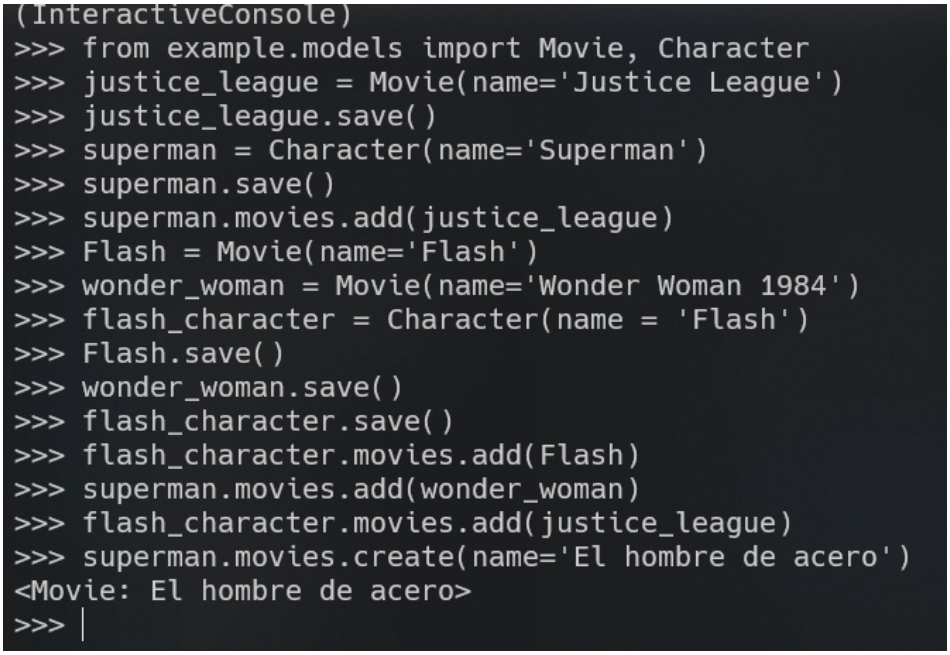

```
from example.models import Language, Framework
Language.objects.all()
Language.objects.filter(framework__name='Angular')
Language.objects.filter(framework__name='Spring')
Language.objects.filter(framework__name='Django')
```

7. Es necesario recrear las acciones realizadas en el video, donde se ingresan datos y se realizan consultas sobre una relación de muchos a muchos en una base de datos utilizando Django.

   Video de insercion de datos en una BD en relación de muchos a muchos (clic)

   Video de querys en una BD en relación de muchos a muchos (clic)

8. Insertamos datos:




```
from example.models import Movie, Character
justice_league = Movie(name='Justice League')
justice_league.save()
superman = Character(name='Superman')
superman.save()
superman.movies.add(justice_league)
flash = Movie(name='Flash')
wonder_woman = Movie(name='Wonder Woman 1984')
flash_character = Character(name = 'Flash')
flash.save()
wonder_woman.save()
flash_character.save()
flash_character.movies.add(flash)
superman.movies.add(wonder_woman)
flash_character.movies.add(justice_league)
superman.movies.create(name='El hombre de acero')
```

	id	name
	Filtro	Filtro
1	1	Captain America
2	2	Thor
3	3	Superman
4	4	Flash

9. Revisamos los combios. Finalmente, llevamos a cabo las consultas para estas tablas y confirmamos su relación de muchos a muchos, siendo un gran ejemplo los vínculos entre Superman y Justice League en la base de datos.

```
(InteractiveConsole)
>>> from example.models import Movie, Character
>>> Character.objects.all()
<QuerySet [<Character: Captain America>, <Character: Thor>, <Character: Superman>, <Character: Flash>]>
>>> Character.objects.filter(movies__name='Justice League')
<QuerySet [<Character: Superman>, <Character: Flash>]>
>>> Movie.objects.filter(character__name='Superman')
<QuerySet [<Movie: Justice League>, <Movie: Wonder Woman 1984>, <Movie: El hombre de acero>]>
>>> superman = Character.objects.get(name='Superman')
>>> superman.movies.all()
<QuerySet [<Movie: Justice League>, <Movie: Wonder Woman 1984>, <Movie: El hombre de acero>]>
>>> justice_league = Movie.objects.get(name='Justice League')
>>> justice_league.character_set.all()
<QuerySet [<Character: Superman>, <Character: Flash>]>
>>> flash = Character.objects.get(name='Flash')
>>> flash.movies.all()
<QuerySet [<Movie: Justice League>, <Movie: Flash>]>
>>>
```

```
from example.models import Movie, Character
Character.objects.all()
Character.objects.filter(movies__name='Justice League')
Movie.objects.filter(character__name='Superman')
superman = Character.objects.get(name='Superman')
```

```
superman.movies.all()
justice_league = Movie.objects.get(name='Justice League')
justice_league.character_set.all()
flash = Character.objects.get(name='Flash')
flash.movies.all()
```

10. Se deberá replicar el código del siguiente video con la intención de aprender sobre la creación e impresión de pdfs con Django.

    Render a Django HTML Template to a PDF file Django Utility CFE Render$_{t}o_{P}DF(clic)$

    [–] el archivo ütils.pyçumple la función de generar el contenido PDF al transformar el código HTML en un formato adecuado para su presentación en ese tipo de documento.

```python
from xhtml2pdf import pisa

def render_to_pdf(template_src, context_dict={}):
    template = get_template(template_src)
    html  = template.render(context_dict)
    result = BytesIO()
    pdf = pisa.pisaDocument(BytesIO(html.encode("ISO-8859-1")), result)
    if not pdf.err:
        return HttpResponse(result.getvalue(), content_type='application/pdf')
    return None
```

    [–] views.py, este es el encargado de generar el pdf, dando el nombre y la información necesario al template para que esté completo.

```python
from django.http import HttpResponse
from django.views.generic import View
from django.template.loader import get_template
from datetime import datetime

from .utils import render_to_pdf

class GeneratePDF(View):
    def get(self, request, *args, **kwargs):
        template = get_template('invoice.html')
        context = {
            "invoice_id": 5621,
            "customer_name": "Reyser Zapata",
            "amount": 1899.99,
            "today": datetime.now(),
        }

        html = template.render(context)
        pdf = render_to_pdf('invoice.html', context)

        if pdf:
            response = HttpResponse(pdf, content_type='application/pdf')
            filename = "Invoice_%s.pdf" %("document")
            content = "inline; filename='%s'" %(filename)
            download = request.GET.get("download")
            if download:
                content = "attachment; filename='%s'" %(filename)
            response['Content-Disposition'] = content
            return response
        return HttpResponse("Not found")
```

[–] El archivo "template.html.ªctúa como la plantilla receptora de información desde la Vista. Esta plantilla se encargará de transformar la información recibida en un documento PDF, listo para su visualización o impresión.

```python
from django.http import HttpResponse
from django.views.generic import View
from django.template.loader import get_template
from datetime import datetime

from .utils import render_to_pdf

class GeneratePDF(View):
    def get(self, request, *args, **kwargs):
        template = get_template('invoice.html')
        context = {
            "invoice_id": 5621,
            "customer_name": "Reyser Zapata",
            "amount": 1899.99,
            "today": datetime.now(),
        }

        html = template.render(context)
        pdf = render_to_pdf('invoice.html', context)

        if pdf:
            response = HttpResponse(pdf, content_type='application/pdf')
            filename = "Invoice_%s.pdf" %("document")
            content = "inline; filename='%s'" %(filename)
            download = request.GET.get("download")
            if download:
                content = "attachment; filename='%s'" %(filename)
            response['Content-Disposition'] = content
            return response
        return HttpResponse("Not found")
```

```html
<html>
    <head>
        <title>Title</title>
        <style type="text/css">
            body {
                font-weight: 200;
                font-size: 14px;
            }
            .header {
                font-size: 20px;
                font-weight: 100;
                text-align: center;
                color: #007cae;
            }
            .title {
                font-size: 22px;
                font-weight: 100;
               /* text-align: right;*/
               padding: 10px 20px 0px 20px;
            }
            .title span {
                color: #007cae;
            }
            .details {
                padding: 10px 20px 0px 20px;
                text-align: left !important;
                /*margin-left: 40%;*/
            }
            .hrItem {
                border: none;
                height: 1px;
                /* Set the hr color */
                color: #333; /* old IE */
                background-color: #fff; /* Modern Browsers */
```

Para ejecutar el ejercicio:

```
1.- py -m venv venv
```

```
2.- pip install -r requirements.txt
```

```
3.- python .\manage.py runserver
```

Nos diriimos al servidor local (`http://127.0.0.1:8000/`)

Se deberá replicar el código del siguiente video con la intención de aprender sobre el envio de Email automaticamente con Django

Sendings Emails in Django (clic)

Se logró replicar el código, un proyecto y una app en Djgango, siendo los archivos más importantes los siguientes:

- [–] views.py, aqui debemos colocar el Asunto, el Cuerpo, el Correo con el que se enviará y el/los correo(s) a donde llegará(destinatario). Para el ejemplo se usará correos temporales como destinatarios.

```python
from django.shortcuts import render
from django.core.mail import send_mail

# Create your views here.
def index(request):
    send_mail('Hola soy Reyser Zapata',
              'Este es un correo enviado automaticamente con Django.',
              'rzapata@unsa.edu.pe',
              ['koferaf572@kameili.com'],
              fail_silently=False)
    return render(request, 'send/index.html')
```

[–] settings.py, aqui debemos configurar el servicio SMTP del cual haremos uso para el envío de correo, en el que debemos de colocar, dominio, puerto y el correo(user) junto con la contraseña con la que se enviará el correo, es importante que estos sean correctos. Algo importante para que funcione con el servicio SMTP de Gmail, es habilitar el acceso de aplicaciones menos seguras en tu cuenta de Gmail. Puedes hacerlo siguiendo estas instrucciones: `https://support.google.com/accounts/answer/6010255`

```python
#settings.py
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = ''
EMAIL_HOST_PASSWORD = ''
EMAIL_USE_TLS = True
EMAIL_USE_SSL = False
```

Para poder ejecutar este ejercicio, necesitaremos hacer lo siguiente:

```
python .\manage.py runserver
```

Una vez hayamos seguido los comandos, podremos abrir el servidor local (`http://127.0.0.1:8000/`) y comprobar si se ha enviado el email

Correo enviado con éxito

[–] Link del video (clic)

Referencias

- https://www.w3schools.com/python/python_reference.asp
- https://docs.python.org/3/tutorial/
- https://docs.djangoproject.com/es/4.2/