

Informe de Laboratorio 07

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

Nota

Estudiantes	Escuela	Asignatura
Reyser Julio Zapata Butron	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/ReyserLyn/Pweb-lab07.git>

2. OBJETIVOS:

- Desarrollar los ejercicios en los videos presentados.
- Utilizar diferentes librerías para lograr un resultado óptimo.
- Aprender y profundizar 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 automático

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:

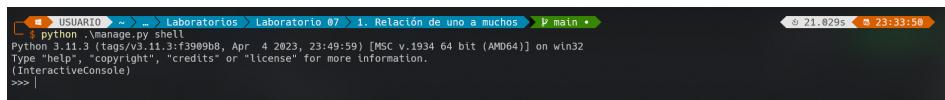
1. Se Deberá replicar las actividades de los video donde se insertar datos y se realizan consultas de una relación de uno a muchos en una base de datos con Django.

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)

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

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



Ahora que estamos dentro de la shell de nuestro proyecto, podemos empezar insertando 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()
```

```

C:\> USUARIO > ~ > .. > Laboratorios > Laboratorio 07 > 1. Relación de uno a muchos > main.py > ~1
$ python manage.py shell
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(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()
>>> |

```

Nos fijamos en los cambios que hubo en la base de datos:

Estructura		Hoja de datos	Editar pragmas	Ejecutar SQL
Tabla: example_language				
<input type="button" value="Filtro"/> Filtro				
1	1	Python		
2	3	Javascript		
3	2	Java		

Estructura		Hoja de datos	Editar pragmas	Ejecutar SQL
Tabla: example_framework				
<input type="button" value="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

Ya que ingresamos datos, podemos hacer las respectivas consultas para la tabla de Framework. Confirmando así que se tiene una 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')
```

Ahora realizamos consultas para la 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')
```

2. Se Deberá replicar las actividades de los video donde se insertar datos y se realizan consultas de una relación de muchos a muchos en una base de datos con 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)

Luego de replicar el código del video, ingresamos al shell e 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')

```

```

C:\USUARIO> ~> ..> Laboratorios > Laboratorio 07 > 2. Relación muchos a muchos > p main •
$ python .\manage.py shell
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> 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')
<Movie: El hombre de acero>
>>> |

```

Ahora revisamos los cambios que hubo en la base de datos:

The screenshot shows a database management interface with the following details:

- Toolbar:** Includes buttons for 'Nueva base de datos', 'Abrir base de datos', 'Guardar cambios', 'Deshacer cambios', 'Estructura', 'Hoja de datos', 'Editar pragmas', and 'Ejecutar SQL'.
- Table Selection:** 'Tabla: example_character'
- Table Data:**

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

Nueva base de datos Abrir base de datos Guardar cambios Deshacer cambios

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: example_movie

	id	name
	Filtro	Filtro
1	1	Avengers
2	2	Civil War
3	3	Thor: Dark World
4	4	Winter Soldier
5	5	Justice League
6	6	Flash
7	7	Wonder Woman 1984
8	8	El hombre de acero

Nueva base de datos Abrir base de datos Guardar cambios Deshacer cambios

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: example_character_movies

	id	character_id	movie_id
	Filtro	Filtro	Filtro
1	1	1	1
2	2	1	2
3	3	2	1
4	4	2	3
5	5	1	4
6	6	3	5
7	7	4	6
8	8	3	7
9	9	4	5
10	10	3	8

Por ultimo, realizamos los querys para estas tablas y confirmamos su relación de muchos a muchos, siendo gran ejemplo, Superman y Justice League

```
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()
```

```
USER@... ~ > ... > Laboratorios > Laboratorio 07 > 2. Relación muchos a muchos > main.py > ?1 ~1
$ python \manage.py shell
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(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>]>
>>> |
```

3. 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_to_PDF(*clic*)

Se logró replicar el código siendo estos los más importantes

[–] `utils.py`, es el encargado de renderizar el pdf, transformando el Html en un PDF.

```
1  from io import BytesIO
2  from django.http import HttpResponse
3  from django.template.loader import get_template
4
5  from xhtml2pdf import pisa
6
7  def render_to_pdf(template_src, context_dict={}):
8      template = get_template(template_src)
9      html = template.render(context_dict)
10     result = BytesIO()
11     pdf = pisa.pisaDocument(BytesIO(html.encode("ISO-8859-1")), result)
12     if not pdf.err:
13         return HttpResponse(result.getvalue(), content_type='application/pdf')
14     return None
15
```

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

```
1  from django.shortcuts import render
2
3  # Create your views here.
4  from django.http import HttpResponseRedirect
5  from django.views.generic import View
6  from django.template.loader import get_template
7  from datetime import datetime
8
9  from .utils import render_to_pdf
10
11 class GeneratePDF(View):
12     def get(self, request, *args, **kwargs):
13         template = get_template('invoice.html')
14         context = {
15             "invoice_id": 5621,
16             "customer_name": "Reyser Zapata",
17             "amount": 1899.99,
18             "today": datetime.now(),
19         }
20
21         html = template.render(context)
22         pdf = render_to_pdf('invoice.html', context)
23
24     if pdf:
25         response = HttpResponseRedirect(pdf, content_type='application/pdf')
26         filename = "Invoice_%s.pdf" %("document")
27         content = "inline; filename='%s'" %(filename)
28         download = request.GET.get("download")
29         if download:
30             content = "attachment; filename='%s'" %(filename)
31         response['Content-Disposition'] = content
32         return response
33     return HttpResponseRedirect("Not found")
```

[–] template.html, es lo que recibirá información de la Vista y se transformará en PDF

```
1  from django.shortcuts import render
2
3  # Create your views here.
4  from django.http import HttpResponseRedirect
5  from django.views.generic import View
6  from django.template.loader import get_template
7  from datetime import datetime
8
9  from .utils import render_to_pdf
10
11 class GeneratePDF(View):
12     def get(self, request, *args, **kwargs):
13         template = get_template('invoice.html')
14         context = {
15             "invoice_id": 5621,
16             "customer_name": "Reyser Zapata",
17             "amount": 1899.99,
18             "today": datetime.now(),
19         }
20
21         html = template.render(context)
22         pdf = render_to_pdf('invoice.html', context)
23
24     if pdf:
25         response = HttpResponseRedirect(pdf, content_type='application/pdf')
26         filename = "Invoice_%s.pdf" %("document")
27         content = "inline; filename='%s'" %(filename)
28         download = request.GET.get("download")
29         if download:
30             content = "attachment; filename='%s'" %(filename)
31             response['Content-Disposition'] = content
32         return response
33     return HttpResponseRedirect("Not found")
```

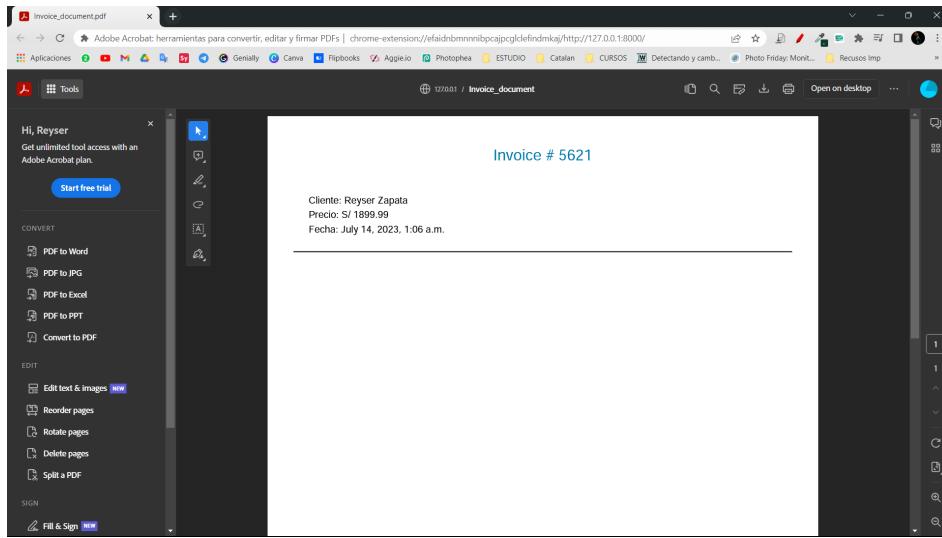
Para poder ejecutar este ejercicio, necesitaremos seguir los siguientes pasos para poder ejecutarlo:

```
py -m venv venv
```

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

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

Una vez hayamos seguido los comandos, podremos abrir el servidor local (<http://127.0.1:8000/>) y ver nuestro PDF



- Se deberá replicar el código del siguiente video con la intención de aprender sobre el envío de Email automáticamente con Django

Sendings Emails in Django (clic)

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

[–] views.py, aquí 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.

```

1  from django.shortcuts import render
2  from django.core.mail import send_mail
3
4  # Create your views here.
5  def index(request):
6      send_mail('Hola soy Reyser Zapata',
7                'Este es un correo enviado automáticamente con Django.',
8                'rzapata@unsa.edu.pe',
9                ['koferaf572@kameilli.com'],
10               fail_silently=False)
11  return render(request, 'send/index.html')
12

```

[–] settings.py, aquí 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>

4. Envio de emails > emailexample >  settings.py > ...

```
126 #settings.py
127 EMAIL_HOST = 'smtp.gmail.com'
128 EMAIL_PORT = 587
129 EMAIL_HOST_USER =
130 EMAIL_HOST_PASSWORD =
131 EMAIL_USE_TLS = True
132 EMAIL_USE_SSL = False
```

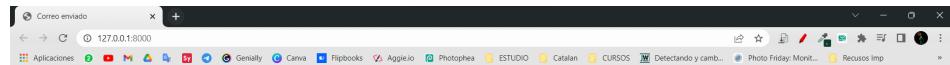
[–] index.html, si todo va bien, esta página se cargará correctamente:

```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  |   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7
8  |   <title>Correo enviado</title>
9  </head>
10 <body>
11 |   <h1>Correo enviado con éxito</h1>
12 </body>
13 </html>
```

Para poder ejecutar este ejercicio, necesitaremos seguir los siguientes pasos para poder ejecutarlo:

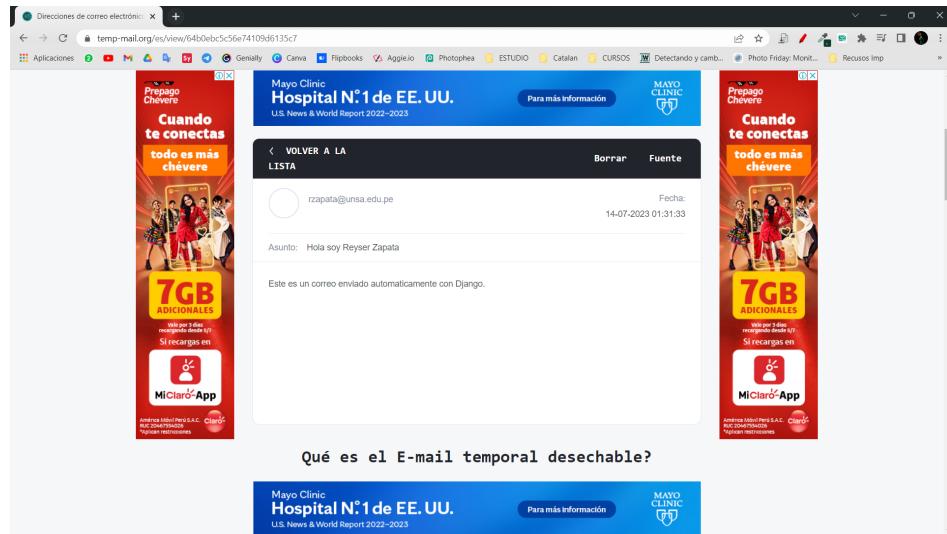
```
python .\manage.py runserver
```

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

The screenshot shows a web browser displaying the Temp Mail website (temp-mail.org/es). The main interface is dark-themed with white text. At the top, it says "Tu dirección de correo electrónico temporal" and displays the generated email address "gisev41461@kamelli.com". Below the address are two small icons: a lock and a trash can. At the bottom of this section are four buttons: "Copiar" (Copy), "Actualizar" (Update), "Cambiar" (Change), and "Borrar" (Delete). Below this, there is a list of recent emails. The first email is from "rzapata@unsa.edu.pe" with the subject "Hola soy Reyser Zapata". The email body contains the text "Recopila firmas electrónicas sobre la marcha con Acrobat Pro DC.". The second email in the list is from "Adobe" with the subject "Recopila firmas electrónicas sobre la marcha con Acrobat Pro DC.".



5. Realizar un video Flipgrid mostrando las ejecuciones de los ejercicios:

[–] Link del video (clic)

COMMITS IMPORTANTES EN GITHUB:

Para este Laboratorio, se necesitó de varios commits, en los cuales se puede apreciar el progreso que hubo para poder completar con éxito este laboratorio. En cada commit se logró implementar un ejercicio de este laboratorio.

AUTOEVALUACIÓN GRUPAL:

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		20	

Referencias

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