

## Informe de Laboratorio 08

**Tema: Relaciones de uno a muchos, muchos a muchos y impresión de pdf y emails**

Nota

Integrantes	Escuela	Asignatura
Miguel Angel Alvarez Choque 20230477 Rodrigo Alexander Fernández Huarca 20230465 Eduardo Joel Cuno Salazar 20231497 Jose Maria Ticona Saure 20233482	Escuela Profesional de Ingeniería de Sistemas	Programación Web II Semestre: I

Laboratorio	Tema	Duración
08	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 19 de junio 2024	Al 21 de junio 2024

## RESULTADOS Y PRUEBAS

### 1. EJERCICIOS RESUELTOS:

#### Ejercicio propuesto:

- *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.*

#### 1.1. Parte 1:

- En esta parte se trabajo la relación de uno a muchos.

### 1.1.1. evidencias:

```
>>> from myapp.models import Language, Framework
>>> python = Language(name='python')
>>> django = Framework(name='Django')
>>> flask = Framework(name='Flask')
>>> python
<Language: Language object (None)>
>>> django.language = python
>>> flask.language = python
>>> bottle = Framework(name='Bottle', language=python)
>>> django.save()

>>> java = Language(name='java')
>>> spring = Framework(name='spring', language=java)
>>> java.save()
```

DB Browser for SQLite - C:\Users\joset\OneDrive\Escritorio\Lab8\Pweb-6\Laboratorio\_08\Ejercicio\_01\myproject\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 Pragnas Execute SQL

Tables: example\_language Filter in any column

id	name
1	python
2	java

Identity Select an identity to connect

DBHub.io Local Current Database

Name

DB Browser for SQLite - C:\Users\joset\OneDrive\Escritorio\Lab8\Pweb-6\Laboratorio\_08\Ejercicio\_01\myproject\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 Pragnas Execute SQL

Tables: example\_framework Filter in any column

id	name	language_id
1	Bottle	1
2	Flask	1
3	Django	1

Identity Select an identity to connect

DBHub.io Local Current Database

Name

### 1.2. Parte 2:

- En esta segunda parte se realizó la relación muchos a muchos, primero se creo el proyecto con su aplicacion y se agregaron los modelos necesarios para el ejercicio.

```
3 class Simple(models.Model):
4     text = models.CharField(max_length=10)
5     number = models.IntegerField(null=True, default=0)
6     url = models.URLField(default='www.example.com')
7
8     def __str__(self):
9         return self.url
10
11 class DateExample(models.Model):
12     the_date = models.DateTimeField()
13
14 class NullExample(models.Model):
15     col = models.CharField(max_length=10, blank=True, null=True)
16
17 class Language(models.Model):
18     name = models.CharField(max_length=10)
19
20     def __str__(self):
21         return self.name
22
23 class Framework(models.Model):
24     name = models.CharField(max_length=10)
25     language = models.ForeignKey(Language, on_delete=models.CASCADE)
26
27     def __str__(self):
28         return self.name
29
30 class Movie(models.Model):
31     name = models.CharField(max_length=10)
32
33     def __str__(self):
34         return self.name
35
36 class Character(models.Model):
37     name = models.CharField(max_length=10)
38     movies = models.ManyToManyField(Movie)
39
40     def __str__(self):
41         return self.name
```

- Se incluyeron los modelos al panel de administración de django

```
1 from django.contrib import admin
2 from .models import Simple, DateExample, NullExample, Language, Framework, Movie, Character
3
4 admin.site.register(Simple)
5 admin.site.register(DateExample)
6 admin.site.register(NullExample)
7 admin.site.register(Language)
8 admin.site.register(Framework)
9 admin.site.register(Movie)
10 admin.site.register(Character)
```

- Se realizaron las migraciones y en la consola de django se empezó a trabajar con los modelos para probar la relación de muchos a muchos.
- Se creó la película Avengers y el personaje 'Captain America'

```
>>> avengers = Movie(name='Avengers')
>>> avengers.save()
>>> captain_america = Character(name='Captain America')
>>> captain_america.save()
```

- Se añade la película a 'captain america'

```
>>> captain_america.movies.add(avengers)
```

- Se crean mas películas

```
>>> civil_war = Movie(name='Civil War')
>>> thor = Movie(name='Thor: Dark World')
```

- Se crea el personaje 'Thor'

```
>>> thor_character = Character(name='Thor')
```

- Se guardan las películas creadas y el personaje 'Thor'

```
>>> civil_war.save()
>>> thor.save()
>>> thor_character.save()
```

- Se añaden las películas a los personajes

```
>>> captain_america.movies.add(civil_war)
>>> thor_character.movies.add(avengers)
>>> thor_character.movies.add(thor)
```

- Se crea y añade la película Winter Soldier a 'captain america' mediante el uso del metodo create

```
>>> captain_america.movies.create(name='Winter Soldier')
```

- A continuación se comprueba que la relaciones entre modelos es correcta y cumple con la regla de muchos a muchos.

- Se filtran personajes para la película 'Civil War'

```
>>> Character.objects.filter(movies__name='Civil War')
QuerySet [<Character: Captain America>]
```

- Se filtran películas para el personaje 'Captain America'

```
>>> Movie.objects.filter(character__name= 'Captain America')
QuerySet [<Movie: Avengers>, <Movie: Civil War>, <Movie: Winter Soldier>]
```

- Se obtiene la instancia de 'Captain America'

```
>>> capatain_america = Character.objects.get(name='Captain America')
>>> capatain_america
<Character: Captain America>
```

- Se verifica las películas de 'Captain America'

```
>>> capatain_america.movies.all()
<QuerySet [<Movie: Avengers>, <Movie: Civil War>, <Movie: Winter Soldier>]>
```

- Se obtiene la instancia de 'Avengers'

```
>>> avengers = Movie.objects.get(name='Avengers')
>>> avengers
<Movie: Avengers>
```

- Se verifican los personajes de 'Avengers'

### 1.2.1. evidencias:

- Primer video del ejercicio 2

```
ca. C:\Windows\System32\cmd.exe - python manage.py shell

>>> from many_to_many.models import Movie, Character
>>> avengers = Movie(name='Avengers')
>>> avengers.save()
>>> captain_america = Character(name='Captain America')
>>> captain_america.save()
>>> captain_america.movies.add(avengers)
>>> civil_war = Movie(name='Civil War')
>>> thor = Movie(name='Thor: Dark World')
>>> civil_war.save()
>>> thor.save()
>>> thor_character = Character(name='Thor')
>>> thor_character.save()
>>> captain_america.movies.add(civil_war)
>>> thor_character.movies.add(avengers)
>>> thor_character.movies.add(thor)
>>> captain_america.movies.create(name='Winter Soldier')
<Movie: Winter Soldier>

>>> avengers.character_set.all()
<QuerySet [<Character: Captain America>, <Character: Thor>]>
```

Segundo video del ejercicio 2

```
>>> Character.objects.filter(movies__name='Civil War')
<QuerySet [<Character: Captain America>]>
>>> Movie.objects.filter(character__name='Captain America')
<QuerySet [<Movie: Avengers>, <Movie: Civil War>, <Movie: Winter Soldier>]>
>>> captain_america = Character.objects.get(name='Captain America')
>>> captain_america.movies.all()
<QuerySet [<Movie: Avengers>, <Movie: Civil War>, <Movie: Winter Soldier>]>
>>> Movie.objects.get(name='Avengers')
<Movie: Avengers>
>>> avengers.character_set.all()
<QuerySet [<Character: Captain America>, <Character: Thor>]>
```

Table: many\_to\_many\_character

	id	name
	Filter	Filter
1	1	Captain America
2	2	Thor

Table: many\_to\_many\_movie

	id	name
	Filter	Filter
1	1	Avengers
2	2	Civil War
3	3	Thor: Dark World
4	4	Winter Soldier

Table: many\_to\_many\_character

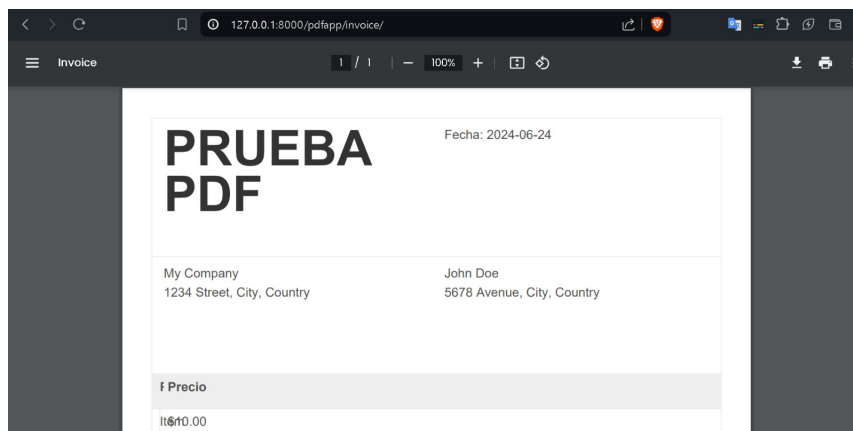
	id	character_id	movie_id
	Filter	Filter	Filter
1	1	1	1
2	2	1	2
3	3	2	1
4	4	2	3
5	5	1	4

### 1.3. Parte 3:

- En esta parte se realizó la impresión de pdfs.

### 1.3.1. Evidencias:

- Se creó la vista *rendertopdf*
- Además se creó la vista secundaria *generateinvoice*
- Se añadió un template *invoice.html* para que sea la estructura del pdf.
- El resultado de esta parte se comparte a continuación



### 1.4. Parte 4:

- En esta parte se realizó el envío de emails.



#### 1.4.1. Evidencias:

- Primero se creó la vista que se encargará del envío del email

```
def enviar_correo(request):
    if request.method == 'POST':
        destinatario = request.POST.get('destinatario')
        asunto = request.POST.get('asunto')
        contenido = request.POST.get('contenido')

        from_email = settings.EMAIL_HOST_USER
        recipient_list = [destinatario]

        try:
            message_html = render_to_string('pdfapp/plantilla_mail.html', {'titulo': asunto, 'contenido': contenido})

            email = EmailMessage(
                subject=asunto,
                body=message_html,
                from_email=from_email,
                to=recipient_list
            )
            email.content_subtype = 'html'
            email.send()

            return HttpResponseRedirect("Correo enviado exitosamente")
        except Exception as e:
            return HttpResponseRedirect(f"Error al enviar el correo: {str(e)}", status=500)

    return render(request, 'pdfapp/formulario_correo.html')
```

- Luego se añadió la información de la cuenta y el tipo de cuenta que se maneja en *settings.py*

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = 'miguelaach123@gmail.com'
EMAIL_HOST_PASSWORD = 'xerw yhpg bpnt oehp'
```

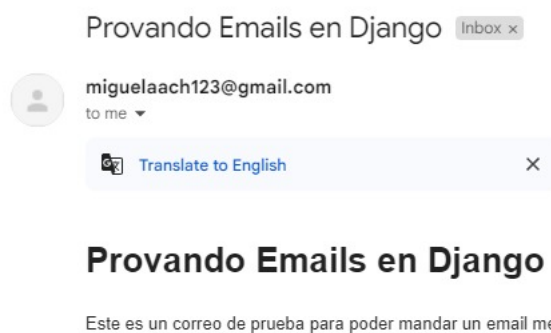
- Ahora se mostrará como se redactó un correo en *formulario\_correo.html* para luego enviarlo



- Se informa que se envió el correo



- Se muestra ahora la captura que efectivamente se envió el email.



## CONCLUSIONES

- Aprender Django te equipa con las habilidades necesarias para desarrollar aplicaciones web modernas y eficientes. Su enfoque en la simplicidad, la rapidez de desarrollo y la seguridad lo convierte en una excelente elección tanto para principiantes como para desarrolladores experimentados.
- En este laboratorio profundizamos en los siguientes aspectos: Relaciones de uno a muchos, muchos a muchos y impresión de pdf y emails. Dominar estos aspectos en Django permite desarrollar aplicaciones web más completas y funcionales, con capacidades avanzadas de manejo de datos, presentación de información y comunicación con los usuarios.

## METODOLOGÍA DE TRABAJO

- Para el presente trabajo se repartió cada parte del trabajo a un integrante del equipo. Los roles principales fueron:
  - Desarrollo de las actividades y ejercicios del laboratorio:
    - José Maria - Parte 1
    - Rodrigo - parte 2
    - Eduardo - parte 3
    - Miguel - parte 4

## REFERENCIAS Y BIBLIOGRAFÍA

Para el trabajo se ocuparon fuentes básicas para entender el uso de Django. Se procede a compartir las fuentes bibliograficas:

- <https://docs.djangoproject.com/es/3.2/>
- <https://docs.djangoproject.com/es/3.2/ref/models/fields/#field-types>
- [https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Tutorial\\_local\\_library\\_website](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Tutorial_local_library_website)

### 1.5. URL'S del repositorio:

- URL del Repositorio GitHub donde se elaboró el trabajo del laboratorio.
- <https://github.com/ELGRANn/Pweb-6.git>
- URL personal de cada integrante del grupo.
  - Miguel Angel Alvarez Choque:
  - [https://github.com/miguelnodjan/pw2\\_24a.git](https://github.com/miguelnodjan/pw2_24a.git)

- Eduardo Joel Cuno Salazar:  
• <https://github.com/ELGRANn/pw2-24a.git>
- Rodrigo Alexander Fernández Huarca:  
• <https://github.com/RdrigoFH/pw2-24a.git>
- Jose Maria Ticona Saure:  
• <https://github.com/joseticonasaure/pw2-24a.git>

## 1.6. Estructura de laboratorio 7

```
+-----laboratorio_8
|---Ejercicio_01
|   |---myproject
|       |---myapp
|           |---migrations
|           |   |---__pycache__
|           |   |---__pycache__
|           |---myproject
|           +-----__pycache__
|-----ejercicio_2
|   |-----ejercicio_2
|   |   |-----__pycache__
|   |   |-----many_to_many
|   |   |-----migrations
|-----myproject
|   |-----myproject
|   |   |-----__pycache__
|   |   |-----pdfapp
|   |       |-----migrations
|   |       |   |-----__pycache__
|   |       |-----templates
|   |       |   |-----pdfapp
|   |       |   |-----__pycache__
```

## 2. Rúbricas

### 2.1. Entregable Informe

Tabla 2: Tipo de Informe

<b>Informe</b>		
<b>Latex</b>	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.	<b>Nota</b>
<b>Observaciones</b>	Respetar la estructura de organización para la ubicación de los entregables. Por cada observación dentro del informe se le descontará puntos. Se debe incluir el código fuente latex del informe	

### 2.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 3: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
<b>2.0</b>	0.5	1.0	1.5	2.0
<b>4.0</b>	1.0	2.0	3.0	4.0

Tabla 4: Rúbrica para contenido del Informe y demostración

	Contenido y demostración	Puntos	Checklist	Estudiante	Profesor
<b>1. GitHub</b>	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	1	
<b>2. Commits</b>	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
<b>3. Código fuente</b>	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1	
<b>4. Ejecución</b>	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
<b>5. Pregunta</b>	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
<b>6. Fechas</b>	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
<b>7. Ortografía</b>	El documento no muestra errores ortográficos.	2	X	1	
<b>8. Madurez</b>	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	2	
<b>Total</b>		20		12	