Informe de Laboratorio 07

Tema: Python - Django

Nota

Estudiante	Escuela	Asignatura
Alejandro Josue Phocco Tapia aphoccot@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 20220589

Laboratorio	Tema	Duración
07	Python - Django	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 29 de Junio 2023	Al 14 Julio 2023

1. Tarea

- Recreacion de los videos proporcionados en el aula virtual sobre relaciones one to many y many to many, asi como la generacion de pdf's y el envio de emails.

2. URL de Repositorio Github

- URL para el laboratorio 07 en el Repositorio GitHub.
- `https://github.com/AlejandroPhoccoTapia/Lab07-PW2-AlejandroPhocco`

3. Ejercicios

3.1. Relaciones One to Many

Veremos los modelos que permiten la relacion de uno a muchos.

Listing 1: models.py de la app

```
from django.db import models

# Create your models here.

class Author(models.Model):
    name = models.CharField(max_length=10)

    def __str__(self):
        return self.name

class Book(models.Model):
    name = models.CharField(max_length=10)
    author = models.ForeignKey(Author, on_delete=models.CASCADE)

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

Aqui la relacion se da mediante el campo ForeignKey.

3.2. Relaciones Many to Many

Veremos los modelos que permiten la relacion de uno a muchos.

Listing 2: models.py de la app

```
from django.db import models

# Create your models here.

class Class(models.Model):
    name = models.CharField(max_length=10)

    def __str__(self):
        return self.name

class Student(models.Model):
    name = models.CharField(max_length=10)
    nameClass = models.ManyToManyField(Class)

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

Aqui la relacion se da mediante el campo ManyToManyField.

3.3. Generaciónn de pdf

- Analizando las url de la app

Listing 3: Archivo urls.py

```
    from django.urls import path
    from . import views

    urlpatterns = [
      path('pdf/', views.GeneratePDF.as_view(), name='PDF'),
```

```
        ]
```

Como vemos tendremos una url "pdf"que sera la que imprima un pdf como tal haciendo uso de una view; esto teniendo en cuenta que importaremos las urls de la aplicacion pdf.

- En la app pdf, analizaremos el archivo utils.py

Listing 4: Archivo pdf/utils.py

```
from io import BytesIO
from django.http import HttpResponse
from django.template.loader import get_template

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
```

En esta función toma una plantilla HTML, la renderiza con los datos del contexto y la convierte en un archivo PDF utilizando xhtml2pdf. Luego, devuelve una respuesta HTTP que contiene el PDF generado.

- Vistas de la app

Listing 5: Archivo pdf/views.py

```
from django.shortcuts import render

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

from .utils import render_to_pdf #created in step 4

class GeneratePdf(View):
    def get(self, request, *args, **kwargs):
        template = get_template('invoice.html')
        context = {
             'today': "Today",
             'amount': 1339.99,
            'customer_name': 'Cooper Mann',
            'invoice_id': 1233434,
        }
        html = template.render(context)
        pdf = render_to_pdf('invoice.html',context)
        if pdf:
            response = HttpResponse(pdf, content_type='application/pdf')
            filename = "Invoice_%s.pdf"%("12341231")
```

```
                    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")
```

Esta vista basada en clase permite generar y devolver un archivo PDF a partir de una plantilla HTML cuando se realiza una solicitud HTTP GET a la vista. El archivo PDF puede ser descargado o mostrado en el navegador según los parámetros proporcionados en la URL de la solicitud. A resaltar que se importa la función render_to_pdf desde un archivo utils.py, que se supone que contiene una función personalizada para generar un archivo PDF a partir de una plantilla HTML.

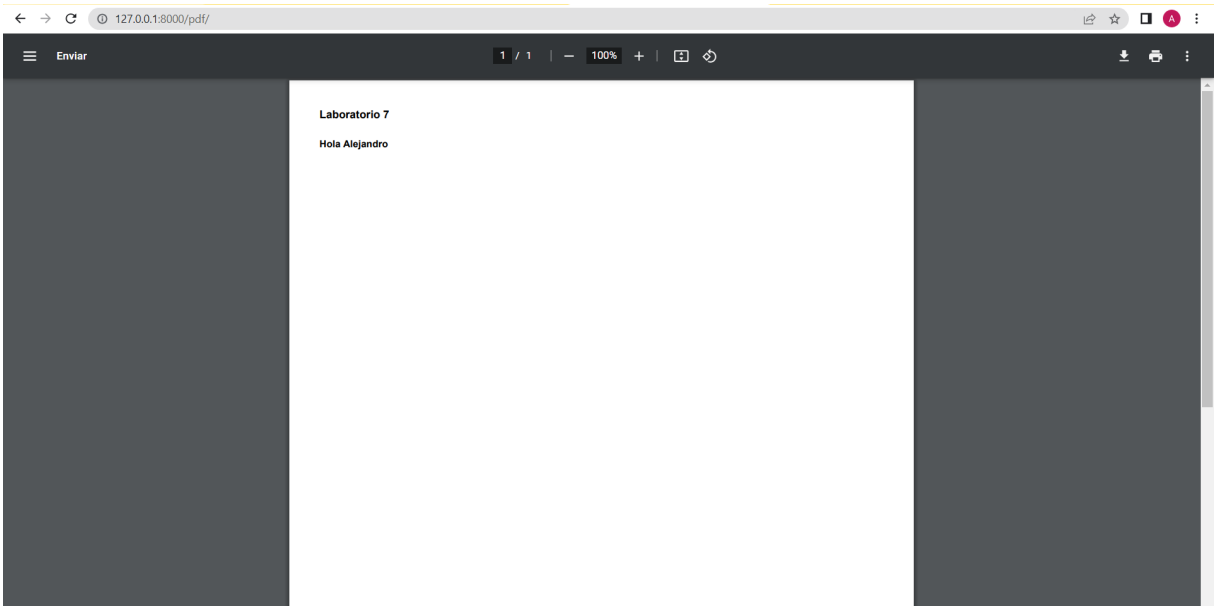
- HTML de la app

Listing 6: Archivo TemplateToPDF/templates/pdf.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Enviar</title>
</head>
<body>
  <h1>Laboratorio {{lab}}</h1>
  <h2>Hola {{name}}</h2>
</body>
</html>
```

Esta es una plantilla basica que utiliza el numero de laboratorio y el nombre del alumno

- Demostracion en navegador



Observamos que el pdf se creo efectivamente con los datos enviados como contexto.

3.4. Envio de email

- Analizando la configuracion del proyecto para enviar emails

Listing 7: Archivo settings.py

```
EMAIL_HOST = 'smtp.gmail.com' #Estoy usando el servicio de gmail
EMAIL_PORT = 587
EMAIL_HOST_USER = 'aphoccot@unsa.edu.pe' #Uso de mi correo institucional
EMAIL_HOST_PASSWORD = 'qilregazjfiwoebh'
EMAIL_USE_TLS = True
EMAIL_USE_SSL = False
```

Aqui configuraremos el servicio de mensajeria que utilizaremos, el puerto que utiliza y la cuenta que usara para enviar los correos.

- En la app SendingEmails, analizaremos el archivo urls.py

Listing 8: Archivo SendingEmails/urls.py

```
from django.urls import path, include
from . import views

urlpatterns = [
  path('email/', views.send),
]
```

Crea una ruta email y llama a la funcion send de su vista que se encargara de enivar el mail.

- Analizaremos el archivo views.py

Listing 9: Archivo SendingEmails/views.py

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

def send(request):
    send_mail('EMAIL PRUEBA DJANGO',
              'Sirve :V',
              'aphoccot@unsa.edu.pe',
              ['pocochuk12345@gmail.com'],
              fail_silently=False)
    return render(request, 'email.html')
```

Esta vista en Django utiliza la función send_mail y luego retorna la plantilla emal.html

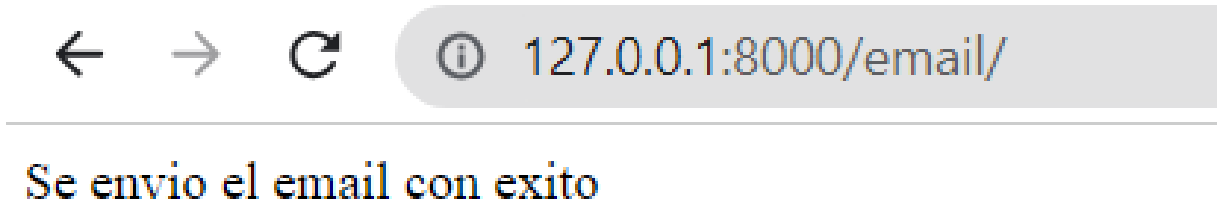
- Analizaremos el archivo templates/email.html

Listing 10: Archivo SendingEmails/templates/email.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Envio de Email</title>
</head>
<body>
  Se envio el email con exito
</body>
</html>
```

Esta es una plantilla simple que contiene un mensaje que confirma que el correo ha sido enviado.

- Demostracion en navegador



Vemos que la pagina nos confirma que el correo ha sido enviado.

Comprobamos que efectivamente llego correo, con el contenido que le especificamos.

Link donde esta alojado el video (FlipGrid):
GRUPO: `https://flip.com/groups/14644257/topics/37115679/responses/430314063`

4. Cuestionario

- Sin cuestionario -

5. Conclusiones

- Django es un excelente framework que nos facilita la creacion de tablas con relacion de uno a muchos o de muchos a muchos.
- Djando ofrece una gran facilidad para crear pdfs y para enviar email personalizados, de una forma sencilla, efectiva y comoda.

6. Referencias

- https://www.w3schools.com/python/python_reference.asp
- https://docs.djangoproject.com/en/4.2/topics/email/
- https://www.scaler.com/topics/django/relationships-in-django-models/
- https://docs.djangoproject.com/en/4.2/howto/outputting-pdf/