

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

## INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
<b>ASIGNATURA:</b>	Programación Web 2				
<b>TÍTULO DE LA PRÁCTICA:</b>					
<b>NÚMERO DE PRÁCTICA:</b>	07	<b>AÑO LECTIVO:</b>	2023A	<b>NRO. SEMESTRE:</b>	III
<b>FECHA DE PRESENTACIÓN</b>	14 Jul 2023	<b>HORA DE PRESENTACIÓN</b>	23:59		
<b>INTEGRANTE (s):</b> Maxs Sebastian Joaquin Forocca Mamani				<b>NOTA:</b>	
<b>DOCENTE(s):</b> Anibal Sardon					

SOLUCIÓN Y RESULTADOS
<p><b>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</b></p> <p>Enlace a los videos:</p> <p>Video 1 Relaciones: <a href="https://flip.com/s/Bs4HsSgSy-2S">https://flip.com/s/Bs4HsSgSy-2S</a></p> <p>Video 2 Pdf/Emails: <a href="https://flip.com/s/uVVrm1HNBaoV">https://flip.com/s/uVVrm1HNBaoV</a></p> <p><b>Relación de uno a muchos</b></p> <ul style="list-style-type: none"> <li>- Código: Para realizar la relación de uno a muchos en bases de datos de Django, se inspeccionaron los videos del Laboratorio y se creó una versión similar en el archivo models.py de la aplicación: "Aplicacion1", del proyecto: "Proyecto", del presente trabajo. Donde se tiene a la Clase Lenguaje y Framework, en la cual los frameworks tienen un lenguaje con el que trabajan y los lenguajes pueden tener varios frameworks, se utiliza "models.ForeignKey()" para hacer la relación de uno a muchos.</li> </ul>

```
5 # One to Many
6 class Lenguaje(models.Model):
7     name = models.CharField(max_length = 50)
8
9     def __str__(self):
10         | return self.name
11
12 class Framework(models.Model):
13     name = models.CharField(max_length = 150)
14     lenguaje = models.ForeignKey(Lenguaje, on_delete = models.CASCADE)
15
16     def __str__(self):
17         | return self.name
18
19
```

- **Agregar y Consultas:** Para agregar y consultar en la DB, se hizo las respectivas migraciones y se abrió un shell donde se importa las clases de Lenguaje y Framework. Posterior a ello se crean objetos de las clases importadas, se guardan en la base de datos y se relacionan a los frameworks con sus respectivos lenguajes. Para realizar consultas, se utiliza "objects.filter" para Lenguaje y Framework especificando el nombre o inicial de nombre en los parámetros para hacer el filtrado y obtener la información solicitada.

```
(env) PS C:\Universidad\UNSA\2023A\LAB_PMEB2\individual\LAB-PMEB2-Ind\Laboratorio07\Proyecto> 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 Aplicacion1.models import Lenguaje, Framework
>>> python = Lenguaje(name='Python')
>>> python.save()
>>> django = Framework(name='Django')
>>> flask = Framework(name = 'Flask')
>>> python
<Lenguaje: Python>
>>> django.lenguaje = python
>>> flask.lenguaje = python
>>> django.save()
>>> flask.save()
>>> java = Lenguaje(name = 'Java')
>>> java.save()
>>> spring = Framework(name = 'Spring', lenguaje = java)
>>> spring.save()
>>> javascript = Lenguaje(name = 'JavaScript')
>>> javascript.save()
>>> react = Framework(name = 'React', lenguaje = javascript)
>>> react.save()
>>> Framework.objects.filter(lenguaje__name = 'Python')
<QuerySet [(<Framework: Django>, <Framework: Flask>)]>
>>> Framework.objects.filter(lenguaje__name = 'Java')
<QuerySet [(<Framework: Spring>)]>
>>> Framework.objects.filter(lenguaje__name = 'JavaScript')
<QuerySet [(<Framework: React>)]>
>>> Framework.objects.filter(lenguaje__name_startswith = 'Java')
<QuerySet [(<Framework: Spring>, <Framework: React>)]>
>>> Lenguaje.objects.filter(framework__name = 'Django')
<QuerySet [(<Lenguaje: Python>)]>
>>> Lenguaje.objects.filter(framework__name = 'Java')
<QuerySet []>
>>> Lenguaje.objects.filter(framework__name = 'Spring')
<QuerySet [(<Lenguaje: Java>)]>
>>> Lenguaje.objects.filter(framework__name = 'React')
<QuerySet [(<Lenguaje: JavaScript>)]>
>>>
```

- Tabla de Lenguajes: En la siguiente imagen se muestra la tabla de Lenguajes después de agregar la información a la base de datos en el shell, para ello se utilizó DB Browser for SQLite.

Estructura   Hoja de datos   Editar pragmas   Ejecutar SQL			
Tabla: Aplicacion1_lenguaje			
	id	name	
	Filtro	Filtro	
1	1	Python	
2	2	Java	
3	3	JavaScript	

- Tabla de Frameworks: En la siguiente imagen se muestra la tabla de Frameworks después de agregar la información a la base de datos en el shell, para ello se utilizó DB Browser for SQLite. Y se puede evidenciar la relación de uno a muchos en la columna "lenguaje\_id".

Estructura   Hoja de datos   Editar pragmas   Ejecutar SQL			
Tabla: Aplicacion1_framework			
	id	name	lenguaje_id
	Filtro	Filtro	Filtro
1	1	Django	1
2	2	Flask	1
3	3	Spring	2
4	4	React	3

### Relación muchos a muchos

- Código: Para realizar la relación de muchos a muchos en bases de datos de Django, se inspeccionaron los videos del Laboratorio y se creó una versión similar. Donde se

tiene a la Clase Movie (película) y Character (personaje), en la cual las películas pueden tener varios personajes y los personajes pueden pertenecer a varias películas, se utiliza "models.ManyToManyField()" para hacer la relación de muchos a muchos.

```

21 #Many to Many
22
23 class Movie(models.Model):
24     name = models.CharField(max_length = 100)
25
26     def __str__(self):
27         return self.name
28
29 class Character(models.Model):
30     name = models.CharField(max_length = 50)
31     movies = models.ManyToManyField(Movie)
32
33     def __str__(self):
34         return self.name

```

- **Agregar y Consultas:** Para agregar y consultar en la DB, se hizo las respectivas migraciones y se abrió un shell donde se importa las clases de Movie y Character. Posterior a ello se crean objetos de las clases importadas, se guardan en la base de datos con ".save()" y se relacionan a los Character con sus respectivas Movie utilizando ".add()" o creando directamente con ".create()". Para realizar consultas, se utiliza "objects.filter" para Movie y Character especificando el nombre en los parámetros para hacer el filtrado y obtener la información solicitada.

```

(env) PS C:\Universidad\UNSA\2023A\LAB PWEB2\individual\LAB-PWEB2-Ind\Laboratorio07\Proyecto> 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 Aplicacion1.models import Movie, Character
>>> narnia1 = Movie(name='Las Cronicas de Narnia: El leon, la bruja y el armario')
>>> narnia1.save()
>>> edmund = Character(name='Edmund')
>>> edmund.save()
>>> edmund.movies.add(narnia1)
>>> narnia2 = Movie(name='Las Cronicas de Narnia: El principe Caspian')
>>> narnia2.save()
>>> caspian = Character(name='Caspian')
>>> caspian.save()
>>> caspian.movies.add(narnia2)
>>> edmund.movies.add(narnia2)
>>> eustace = Character(name='Eustace')
>>> eustace.save()
>>> eustace.movies.create(name='Las Cronicas de Narnia: La travesia del viajero del alba')
<Movie: Las Cronicas de Narnia: La travesia del viajero del alba>
>>> Character.objects.filter(movies__name='Las Cronicas de Narnia: El principe Caspian')
File "<console>", line 1
    Character.objects.filter(movies__name='Las Cronicas de Narnia: El principe Caspian')
    ^
SyntaxError: incomplete input
>>> Character.objects.filter(movies__name='Las Cronicas de Narnia: El principe Caspian')
<QuerySet [(<Character: Caspian>, <Character: Edmund>)]>
>>> Movie.objects.filter(character__name='Edmund')
<QuerySet [(<Movie: Las Cronicas de Narnia: El leon, la bruja y el armario>, <Movie: Las Cronicas de Narnia: El principe Caspian>)]>
>>> personaje1 = Character.objects.get(name='Edmund')
>>> personaje1
<Character: Edmund>
>>> personaje1.movies.all()
<QuerySet [(<Movie: Las Cronicas de Narnia: El leon, la bruja y el armario>, <Movie: Las Cronicas de Narnia: El principe Caspian>)]>
>>> pelicula2 = Movie.objects.get(name='Las Cronicas de Narnia: El principe Caspian')
>>> pelicula2
<Movie: Las Cronicas de Narnia: El principe Caspian>
>>> pelicula2.character_set.all()
<QuerySet [(<Character: Caspian>, <Character: Edmund>)]>

```

- Tabla de Movie: En la siguiente imagen se muestra la tabla de Movie después de agregar la información a la base de datos en el shell, para ello se utilizó DB Browser for SQLite.

Estructura

Hoja de datos


Editar pragmas


Ejecutar SQL


Tabla:


Aplicacion1\_movie


▼


























Filtrar en cualquier columna

	id	name
	Filtro	Filtro
1	1	Las Cronicas de Narnia: El leon, la bruja y el armario
2	2	Las Cronicas de Narnia: El principe Caspian
3	3	Las Cronicas de Narnia: La travesia del viajero del alba

- Tabla de Character: En la siguiente imagen se muestra la tabla de Character después de agregar la información a la base de datos en el shell, para ello se utilizó DB Browser for SQLite.

Estructura

Hoja de datos

Editar pragmas

Ejecutar SQL

Tabla:

Aplicacion1\_character

- Tabla de Movie\_Character: En la siguiente imagen se puede evidenciar la relación de muchos a muchos de Movie y Character, para ello se tiene las columnas de "movie\_id" y "character\_id". Esta tabla se crea automáticamente después de migrar con la relación Many To Many.

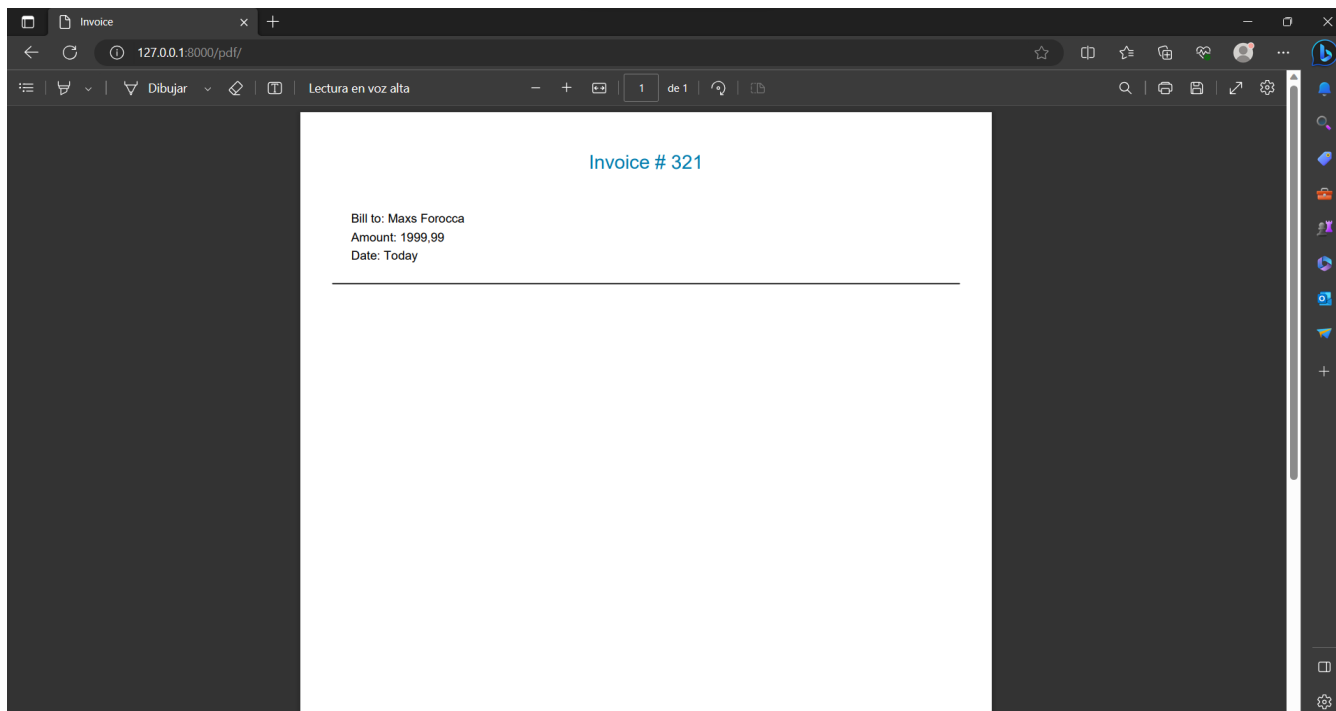
Estructura    Hoja de datos    Editar pragmas    Ejecutar SQL			
Tabla: Aplicacion1_character_movies			
	id	character_id	movie_id
	Filtro	Filtro	Filtro
1	1	1	1
2	2	2	2
3	3	1	2
4	4	3	3

## Impresión de pdfs


- Código: Para realizar la impresión de Pdf en Django, se inspeccionó el video del Laboratorio y se ejecutó el siguiente comando en la consola "pip install --pre xhtml2pdf" para poder trabajar con PDFs en Django. En la el archivo "views.py" de Proyecto se importa View, render\_to\_pdf y get\_template, además se crea la clase GeneratePDF que capta la plantilla con el nombre "invoice.html" (con la modificacion en settings.py para trabajar con plantillas), se crea un contexto para el template y se renderiza a pdf. En el caso de haberse renderizado, se le otorga un nombre, contenido y al momento de descargar el pdf, se le da un nombre de descarga por defecto.

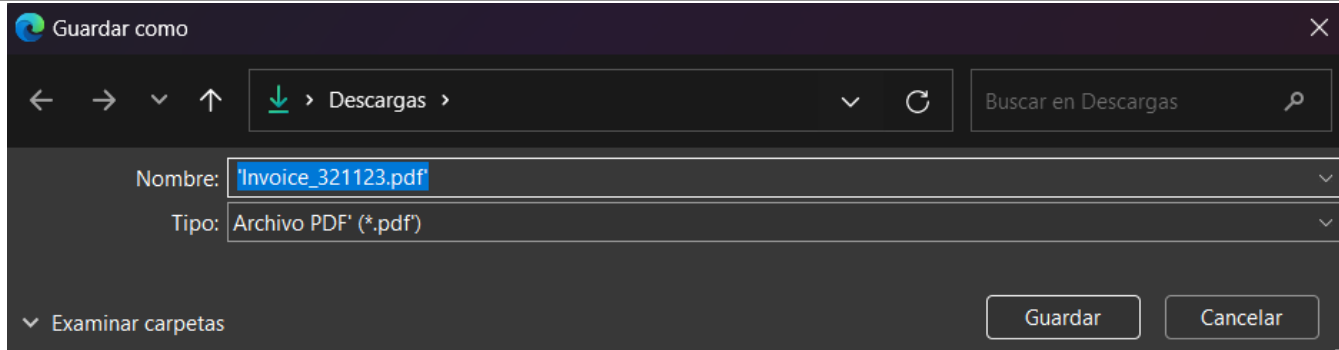
```
7 class GeneratePDF(View):
8     def get(self, request, *args, **kwargs):
9         template = get_template('invoice.html')
10        context = {
11            "invoice_id": 321,
12            "customer_name": "Maxs Forocca",
13            "amount": 1999.99,
14            "today": "Today",
15        }
16        html = template.render(context)
17        pdf = render_to_pdf('invoice.html', context)
18        if pdf:
19            response = HttpResponse(pdf, content_type = 'application/pdf')
20            filename = "Invoice_%s.pdf" %("321123")
21            content = "inline; filename='%s'" %(filename)
22            download = request.GET.get("download")
23            if download:
24                content = "attachment; filename='%s'" %(filename)
25            response['Content-Disposition'] = content
26            return response
27        return HttpResponse("No Encontrado")
28
```

- Ejecución: Para mostrar la ejecución, se otorgó la URL (en urls.py) de "pdf/", donde se muestra la página del template Invoice renderizado a pdf.



- Descarga: Al momento de querer descargar el Pdf, se puede evidenciar el nombre por defecto del archivo Pdf.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>



## Envío de emails

- Código: Para realizar el envío de Emails en Django, se inspeccionó el video del Laboratorio y se creó la aplicación: "send". En el archivo views.py de la aplicación se importó send\_mail y se creó la función index, que utiliza send\_mail para enviar un mensaje email, en el primer campo se indica el asunto, en el segundo el contenido y dentro de corchetes, en forma de String se coloca el correo que recibirá el mensaje. Para ello se utilizó un correo temporal de la página "temp-mail". Finalmente se enviará el template index.

```

4 # Create your views here.
5 def index(request):
6
7     send_mail('Hola desde Mi Proyecto de Django',
8             'Hola, este es un mensaje automatico.',
9             'maxs.sebas@gmail.com',
10            ['retac43400@msback.com'],
11            fail_silently=False)
12
13     return render(request, 'index.html')
```


- Settings: Para poder enviar un Email desde Django, se agregó a settings.py el host, puerto, ssl, tls, correo y contraseña.

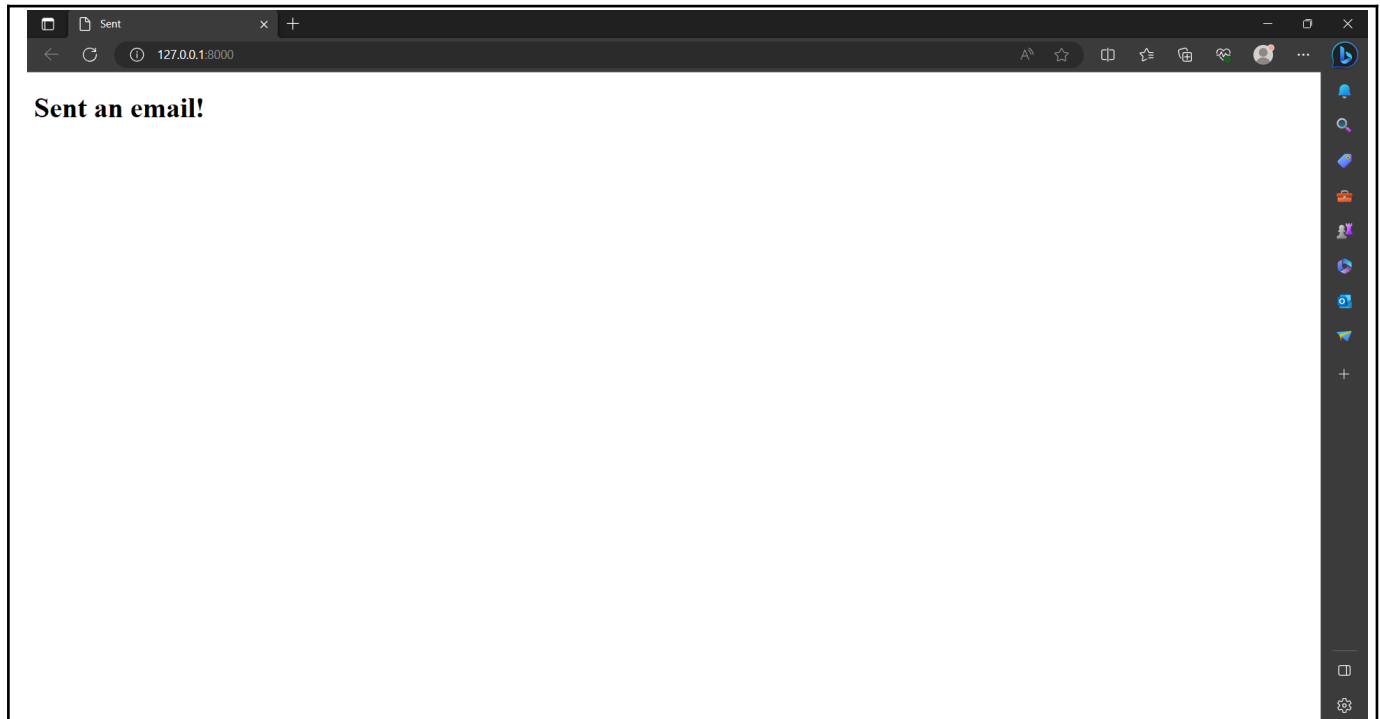
```

123 EMAIL_BACKEND="django.core.mail.backends.smtp.EmailBackend"
124 EMAIL_HOST = 'smtp.gmail.com'
125 EMAIL_PORT = 587
126 EMAIL_HOST_USER = 'maxs.sebas@gmail.com'
127 EMAIL_HOST_PASSWORD = 'contrasenia' #contrasenia de aplicacion, verificacion en dos pasos
128 EMAIL_USE_TLS = True
129 EMAIL_USE_SSL = False
```

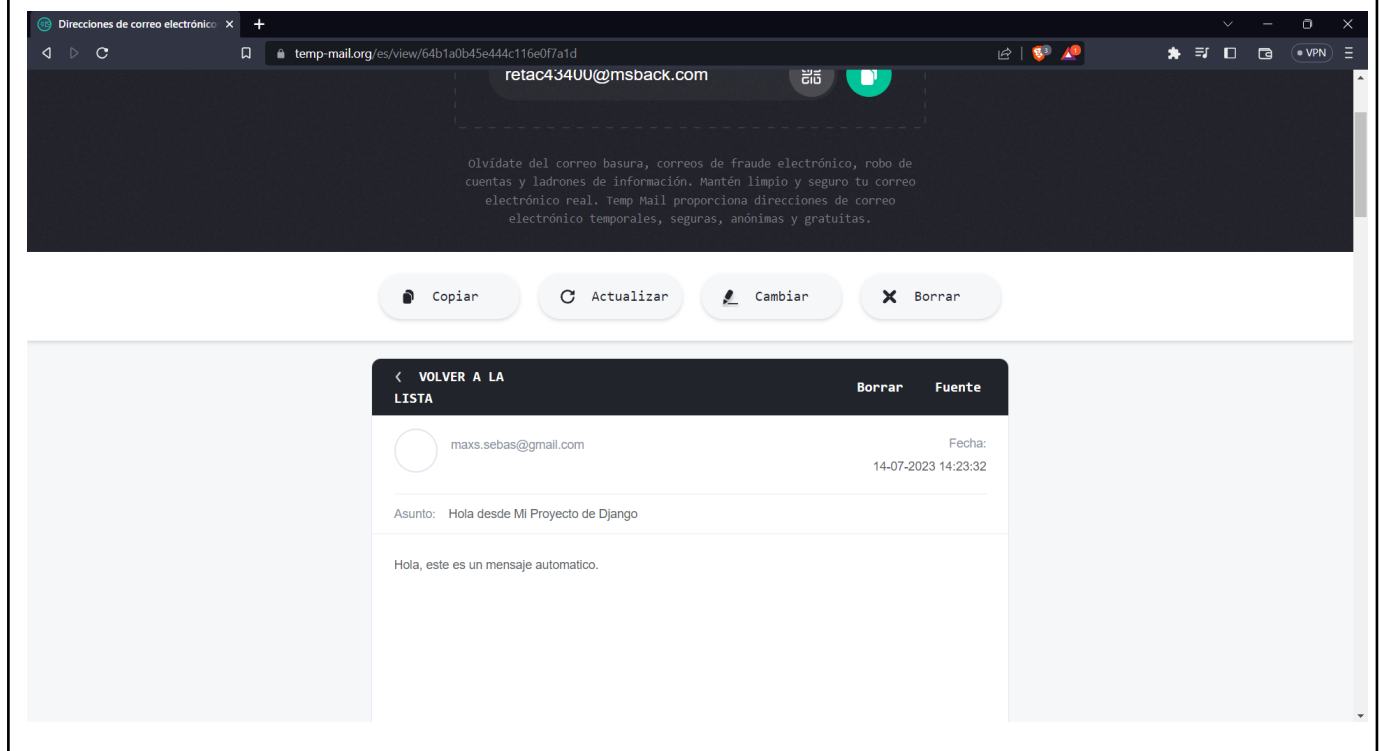
- Envío: Al momento de correr el servidor, y dirigirse a la ruta por defecto (se especifico en urls.py para enviar el email), se muestra el template index mostrando el mensaje de que se envió al correo.



	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 9



- Recepción: Cuando nos dirigimos al correo temporal se puede evidenciar que se recibió exitosamente el correo enviado desde Django.



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 10</p>

## Commits

- Commits importantes al momento de realizar el laboratorio.

```
commit bc0216272604bfd811b836be51b855e67dbf3b9a
Author: MaxsForocca <mforocca@unsa.edu.pe>
Date: Tue Jul 11 20:40:49 2023 -0500
```

carpeta templates, modificar settings

```
commit b77429869bc2b0b201bacf9985762b8731a6a361
Author: MaxsForocca <mforocca@unsa.edu.pe>
Date: Sun Jul 9 00:02:36 2023 -0500
```

creacion de proyecto y aplicacion1

```
commit 003485405aa96ab02a6a0fc4ac29deb1f7a2473b
Author: MaxsForocca <mforocca@unsa.edu.pe>
Date: Fri Jul 14 20:59:35 2023 -0500
```

ejercicios terminados, agrego mas imagenes

```
commit 70904ec3d46200898ddb67fd11e23322fc26b89
Author: MaxsForocca <mforocca@unsa.edu.pe>
Date: Fri Jul 14 14:42:19 2023 -0500
```

se agrego imagenes de pdf y emails

```
commit 0d80688e5288cee74bf239339e7251a4aee65214
Author: MaxsForocca <mforocca@unsa.edu.pe>
Date: Fri Jul 14 14:41:20 2023 -0500
```

se agrego imagenes y se completo emails

```
commit ff06d995a6176f4327517cb50a44782838ef6d21
Author: MaxsForocca <mforocca@unsa.edu.pe>
Date: Thu Jul 13 22:48:27 2023 -0500
```

send email Django

```
commit 3f0f5744d6e2fd791efe26b11fce0729604d3e27
Author: MaxsForocca <mforocca@unsa.edu.pe>
Date: Thu Jul 13 10:26:41 2023 -0500
```

complete templates to pdf in django

```
commit 64e417e7e3d49aff638f1ea51050cc13d2d4aa6f
Author: MaxsForocca <mforocca@unsa.edu.pe>
Date: Wed Jul 12 23:26:06 2023 -0500
```

avance de templates to pdf in django

## II. SOLUCIÓN DEL CUESTIONARIO

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 11</p>

III. CONCLUSIONES

RETROALIMENTACIÓN GENERAL

REFERENCIAS Y BIBLIOGRAFÍA