Laboratorio 07
Uno a muchos, muchos a muchos, email y PDF

Melsy Melany Huamaní Vargas

14 de julio de 2023

Profesor	Escuela	Asignatura
Anibal Sardon Paniagua	Ingeniería de Sistemas	Programación Web 2

Laboratorio	Tema	Duración
07	Uno a muchos, mucho a muchos, email y pdf	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	04/07/2023	14/07/2023

DESARROLLO

ENLACE GITHUB Y GIT

Este trabajo se está presentando en el Github: `https://github.com/mhuamanivar/PW2-HuamaniV-Lab07` y el .git es: `https://github.com/mhuamanivar/PW2-HuamaniV-Lab07.git`, además el video de flipgrip es el siguiente: `https://flip.com/s/Ms_fZ__6sTyz`.

CREANDO PROYECTO E INSTALANDO

Se utiliza el sistema Windows para instalar las herramientas necesarias y crear el proyecto. En este caso, se escogió una aplicación library para que se muestran libros.

- Primero se verifica la versión de Python.

```
C:\Users\melsy>mkvirtualenv my_env7
created virtual environment CPython3.11.1.final.0-64 in 3903ms
  creator CPython3Windows(dest=C:\Users\melsy\Envs\my_env7, clear=False, no_vcs_ignore=
      False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy
      , app_data_dir=C:\Users\melsy\AppData\Local\pypa\virtualenv)
    added seed packages: pip==23.1.2, setuptools==67.8.0, wheel==0.40.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,
      PowerShellActivator,PythonActivator
```

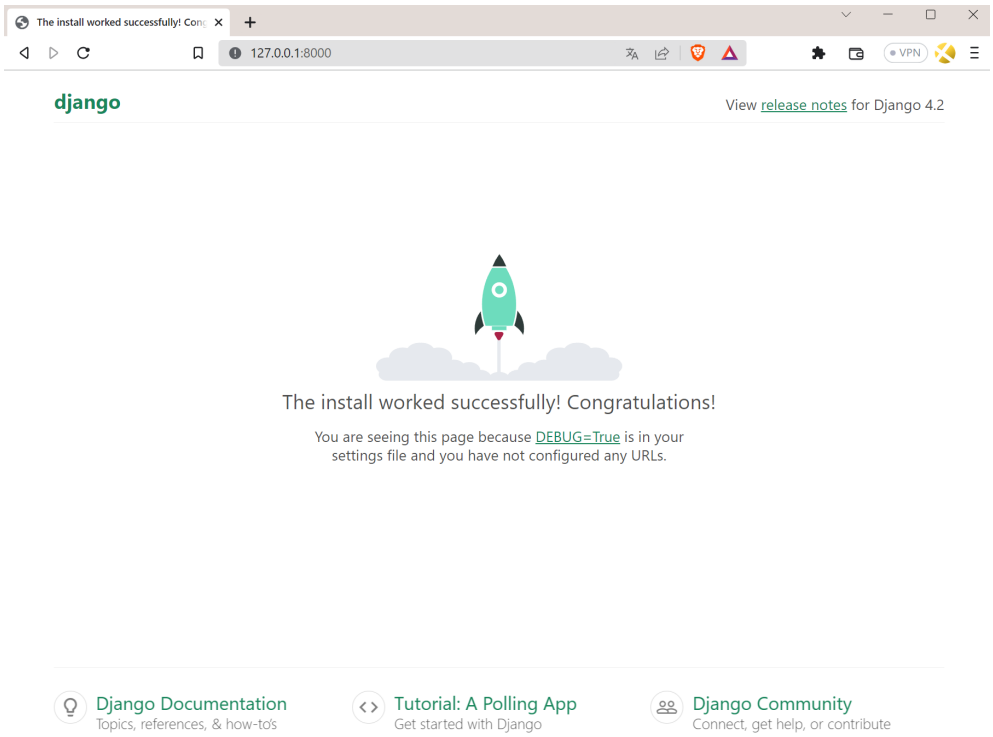
- Se instala Django y xhtml2pdf

```
(my_env7) C:\Users\melsy>pip install django
(my_env7) C:\Users\melsy>pip install --pre xhtml2pdf
```

- Se crea el proyecto dentro de la carpeta ‘projects‘ y dentro de esta se crea la aplicación ‘library‘ para usarla más adelante.

```
(my_env7) C:\Users\melsy>cd projects
(my_env7) C:\Users\melsy\projects> django-admin startproject proyecto
(my_env7) C:\Users\melsy\projects>cd proyecto
(my_env7) C:\Users\melsy\projects\proyecto>python manage.py startapp library
```

- Se corre el servidor y se puede ver lo siguiente



CREANDO LOS MODELOS, UNO A MUCHOS Y MUCHOS A MUCHOS

- Primero se añade la aplicación creada en el archivo 'settings.py' de 'lab07'.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'library' # Se agrega esta linea
]
```

- Se crea el modelo 'Author' que tiene relación de uno a muchos con el modelo 'Book'.

```
class Author(models.Model):
    name = models.CharField(max_length=100)

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

- Ahora se crea el modelo 'Publisher' que en este caso tiene relación de muchos a muchos con el modelo 'Book'.

```
class Publisher(models.Model):
    name = models.CharField(max_length=100)
```

```
3
4       def __str__(self):
5           return self.name
```

- Se tiene el modelo 'Book' el cual tiene relación uno a muchos y muchos a muchos.

```
1   class Book(models.Model):
2       title = models.CharField(max_length=100)
3       author = models.ForeignKey(Author, on_delete=models.CASCADE)
4       publisher = models.ManyToManyField(Publisher)
5       summary = models.TextField()
6
7       def __str__(self):
8           return self.title
```

- Se hacen las migraciones de los modelos del proyecto.

```
(my_env7) C:\Users\melsy\projects\proyecto>python manage.py makemigrations
Migrations for 'library':
  library\migrations\0001_initial.py
    - Create model Author
    - Create model Publisher
    - Create model Book
(my_env7) C:\Users\melsy\projects\proyecto>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, library, sessions
Running migrations:
  Applying library.0001_initial... OK
```

CONFIGURANDO VIEWS, EMAIL Y PDF

- Se crean las vistas para ver la lista de libro y los detalles del libro seleccionado en 'views.py' de library.

```
1   from django.shortcuts import render
2   from .models import Book
3
4   def book_list(request):
5       books = Book.objects.all()
6       return render(request, 'library/book_list.html', {'books': books})
7
8   def book_detail(request, book_id):
9       book = get_object_or_404(Book, id=book_id)
10      return render(request, 'library/book_detail.html', {'book': book})
```

- Se crea la función 'generate_pdf' para generar el PDF con los detalles de un libro y se agregan los siguientes import en 'views.py' de library.

```
1   from django.shortcuts import render, get_object_or_404
2   from django.http import HttpResponse
3   from django.template.loader import get_template
4   from xhtml2pdf import pisa
5   from .models import Book
6
7   def generate_pdf(request, book_id):
```

```
    book = get_object_or_404(Book, id=book_id)
    template_path = 'library/invoice.html'
    context = {
        'book': book,
    }
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = 'attachment; filename="
        book_details.pdf"'

    template = get_template(template_path)
    html = template.render(context)

    pisa_status = pisa.CreatePDF(html, dest=response)
    if pisa_status.err:
        return HttpResponse('Error al generar el PDF', status=500)
    return response
```

- Se crea la función 'send_email' en 'views.py' de library, para enviar un email el cual utiliza la función anterior para enviar el PDF con los detalles del libro.

```
from django.shortcuts import render, get_object_or_404
from django.http import HttpResponse
from django.template.loader import get_template
from django.core.mail import EmailMessage
from xhtml2pdf import pisa
from .models import Book

def send_email(request, book_id):
    if request.method == 'POST':
        pdf = generate_pdf(request, book_id)
        email_address = request.POST.get('email')

        email = EmailMessage(
            'Detalles del libro',
            'Adjunto encontraras los detalles del libro en formato PDF
                .',
            'melsy@gmail.com',
            [email_address]
        )

        email.attach('book_details.pdf', pdf.getvalue(), 'application/
            pdf')
        email.send()

        return render(request, 'library/email_sent.html')

    book = get_object_or_404(Book, id=book_id)
    return render(request, 'library/book_detail.html', {'book': book})
```

- En el archivo 'settings.py' de proyecto se colocan las credenciales del email para enviar el correo, par esto se utilizó la página de 'Mailtrap' el cual brinda estas credenciales para pruebas de correo.

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'sandbox.smtp.mailtrap.io'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'bd45a037fa56d1'
EMAIL_HOST_PASSWORD = '9a7f42a21e19bf'
```

```
EMAIL_USE_TLS = True
EMAIL_USE_SSL = False
```

CONFIGURANDO URLS

- Se colocan las urls de library en el archivo de ‘urls.py‘ de proyecto para que puedan ser reconocidas.

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('library/', include('library.urls')),
]
```

- Se colocan las urls en el archivo de ‘urls.py‘ de library para que sean reconocidas posteriormente.

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

app_name = 'library'

urlpatterns = [
    path('books/', views.book_list, name='book_list'),
    path('book/<int:book_id>/', views.book_detail, name='book_detail')
        ,
    path('book/<int:book_id>/pdf/', views.generate_pdf, name='
        generate_pdf'),
    path('book/<int:book_id>/email/', views.send_email, name='
        send_email'),
]
```

CREANDO LOS ARCHIVOS HTML

- En primer lugar, se modifica el archivo ‘settings.py‘ de proyecto para que reconozca los archivos html.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')], # Se agrega
            esta linea
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
```

```
            },
        },
    ]
```

- Se crea el archivo 'book_list.html' con lo siguiente.

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista de libros</title>
</head>
<body>
    <h1>Lista de libros</h1>
    <ul>
    {% for book in books %}
        <li><a href="{% url 'library:book_detail' book.id %}">{{ book.
            title }}</a></li>
    {% empty %}
        <li>No hay libros disponibles.</li>
    {% endfor %}
    </ul>
</body>
</html>
```

- Se crea el archivo 'book_detail.html' con lo siguiente.

```
<!DOCTYPE html>
<html>
<head>
    <title>Detalles del libro</title>
</head>
<body>
    <h1>Detalles del libro</h1>
    <h2>{{ book.title }}</h2>
    <p>Autor: {{ book.author.name }}</p>
    <p>Publicado por:</p>
    <ul>
        {% for publisher in book.publisher.all %}
            <li>{{ publisher.name }}</li>
        {% endfor %}
    </ul>
    <p>Resumen: {{ book.summary }}</p>
    <form action="{% url 'library:send_email' book.id %}" method="POST
        ">
        {% csrf_token %}
        <label for="email">Enviar PDF por email:</label><br>
        <input type="email" id="email" name="email"><br>
        <input type="submit" value="Enviar">
    </form>
    <a href="{% url 'library:generate_pdf' book.id %}">Descargar en
        PDF</a>

</body>
</html>
```

- Se crea el archivo 'invoice.html' para mostrar en PDF el contenido requerido.

```
<!DOCTYPE html>
<html>
<head>
    <title>Detalles del libro</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;
            padding: 10px 20px 0px 20px;
        }
        .details {
            padding: 10px 20px 0px 20px;
            text-align: left !important;
        }
        .hrItem {
            border: none;
            height: 1px;
            color: #333;
            background-color: #fff;
        }
    </style>
</head>
<body>
    <div class='wrapper'>
        <div class='header'>
            <p class='title'>Detalles del libro</p>
        </div>
        <div class='details'>
            <p>Autor: {{ book.author.name }}</p>
            <p>Publicado por:
            {% for publisher in book.publisher.all %}
                {{ publisher.name }}{% if not forloop.last %}, {%
                    endif %}
            {% empty %}
                Sin publicaciones
            {% endfor %}
            </p>
            <p>Resumen: {{ book.summary }}</p>
            <hr class='hrItem' />
        </div>
    </div>
</body>
</html>
```

- Se crea el archivo 'email_sent.html' donde se confirma que el correo ha sido enviado.

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Confirmacion de envio de correo</title>
</head>
<body>
    <h1>Correo electronico enviado</h1>
    <p>Se ha enviado el correo electronico con los detalles del libro
       en formato PDF.</p>
</body>
</html>
```

CONFIGURANDO PAGINA DE ADMIN

- En primer lugar se modifica ‘admin.py‘ de library para que se puedan crear los modelos de author, book y publisher.

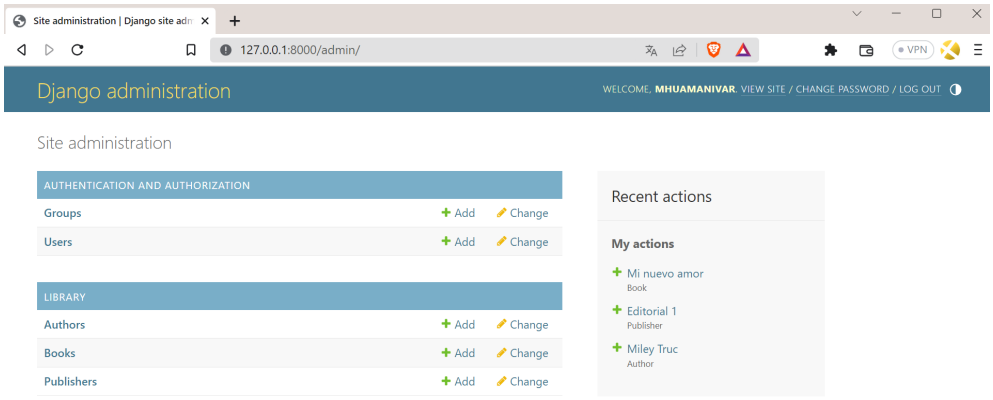
```
from django.contrib import admin
from .models import Book, Author, Publisher

@admin.register(Book)
class BookAdmin(admin.ModelAdmin):
    pass

@admin.register(Author)
class AuthorAdmin(admin.ModelAdmin):
    pass

@admin.register(Publisher)
class PublisherAdmin(admin.ModelAdmin):
    pass
```

- Se crea el superusuario para acceder a la página del administrador.

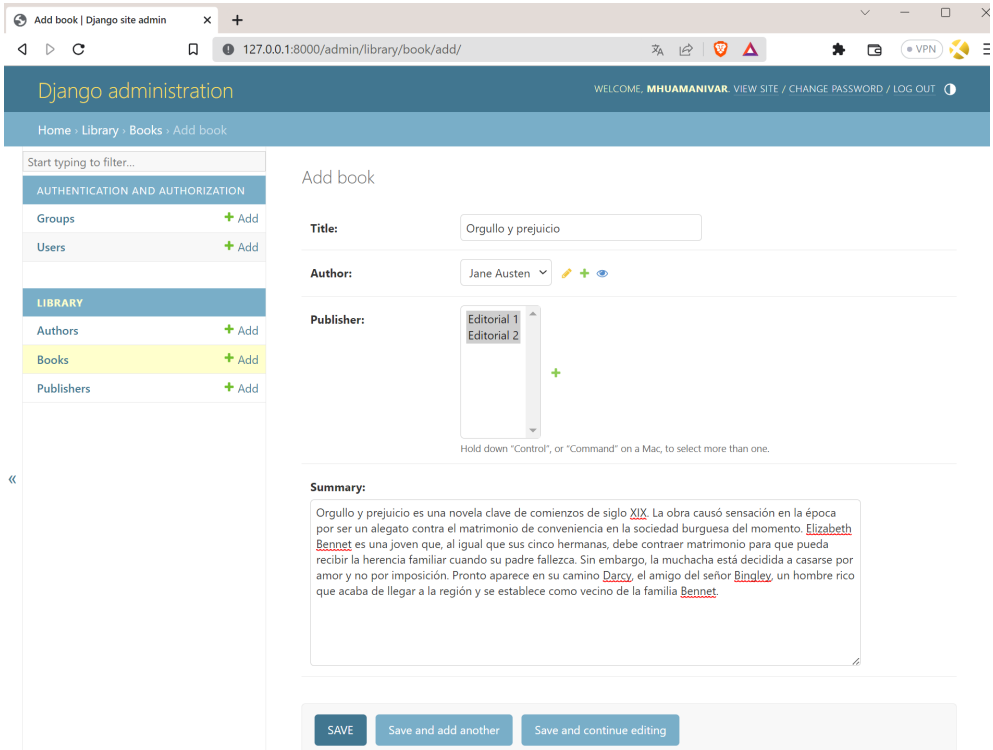
```
(my_env7) C:\Users\melsy\projects\proyecto>python manage.py createsuperuser
Username (leave blank to use 'melsy'): mhuamanivar
Email address: mhuamanivar@unsa.edu.pe
Password:
Password (again):
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

VIENDO PAGINA DE ADMIN

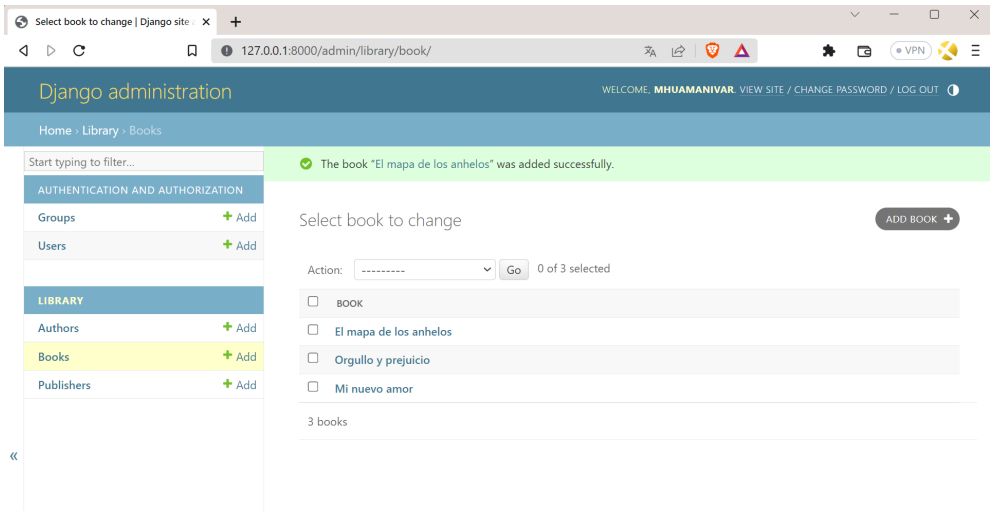
- Se va a la página del admin



- Luego se da en ‘add‘ para añadir un libro, y colocamos los detalles importantes, se crea tres libros como ejemplo.

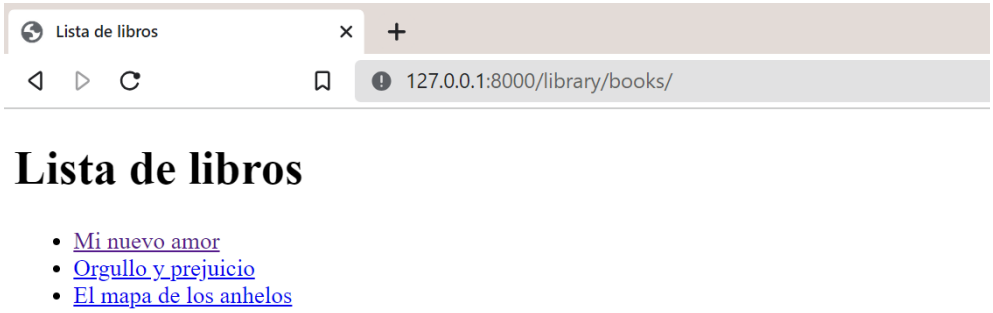


- Despues se puede ver la lista de libros creados en la pagina del admin.

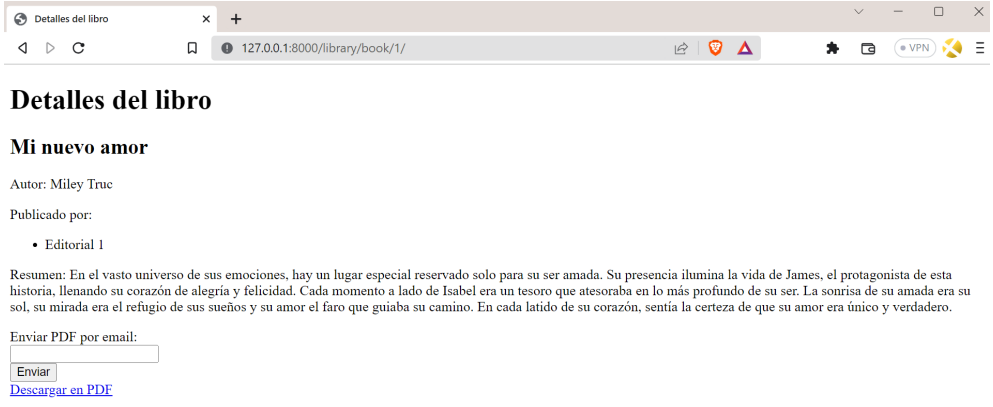


VIENDO LA PAGINA PRINCIPAL

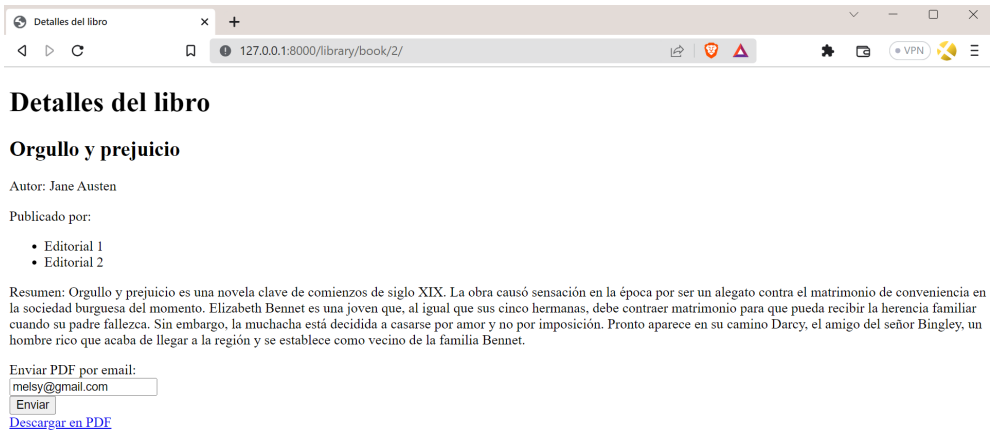
- Se va a la pagina 'http://127.0.0.1:8000/library/books/' y se ve la lista de libros



- Se da click a uno de ellos y se pueden ver los detalles del libro.

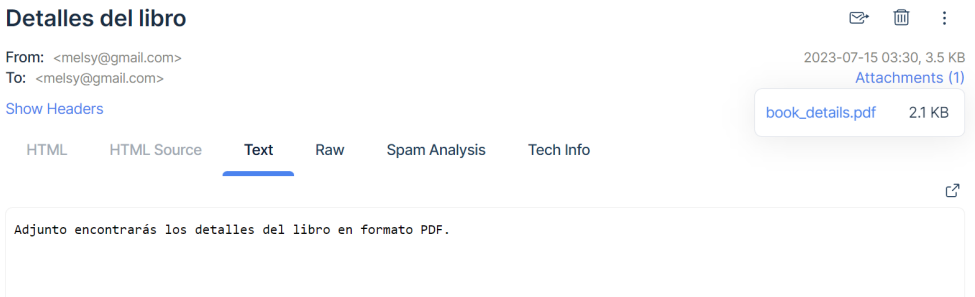


- Para enviar un correo, se inserta el correo al que se desea enviar

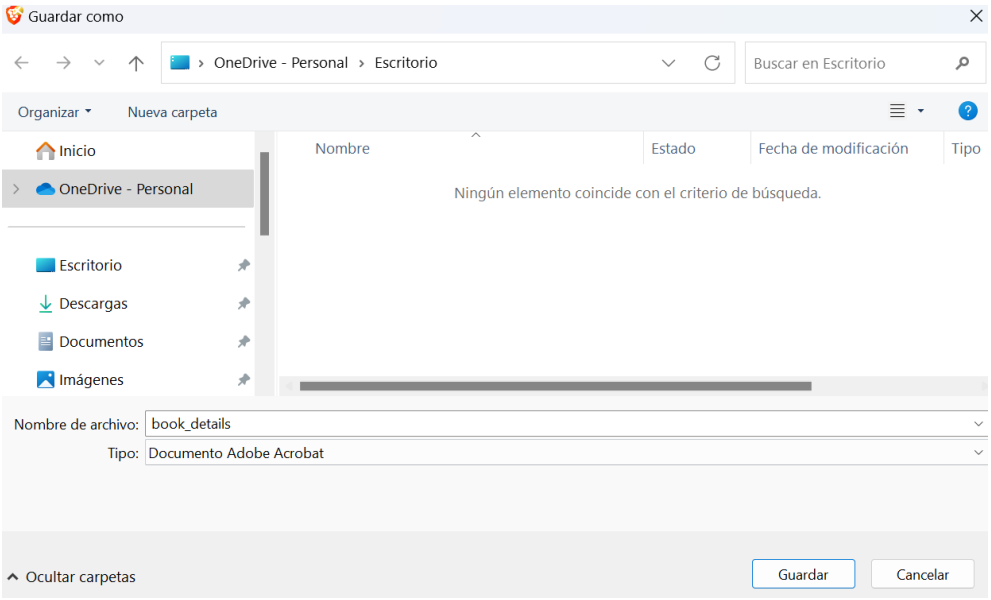


- Posteriormente se puede ver la pagina que menciona que se ha enviado correctamente.

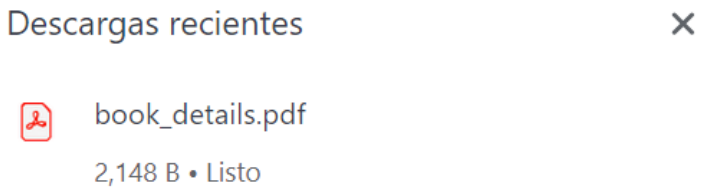
- Como se eligió trabajar con Mailtrap, entonces se va a la página y se puede ver que ha sido enviado correctamente el mensaje, además se ve de quien para quien va dirigido y el archivo que ha sido enviado en ‘Attachments‘.



- Por otro lado, si se quiere enviar un PDF, entonces se hace click en ‘Descargar en PDF‘ y le damos en guardar en la carpeta que queremos.



- Se confirma que ha sido guardado con el navegador.



- Finalmente se puede ver el archivo en formato PDF, que contiene los detalles del libro.