Reto Práctico #1 – Máster en Microservicios Django + DRF

Consultor: Héctor Lozada

Fecha de Asignación: 03-05-2025

Fecha de Entrega: -

Valor: 100 pts.

Reto Práctico: Desarrollo de un microservicio de gestión de documentos con Django y DRF

En este reto, los estudiantes desarrollarán un microservicio básico para la gestión de documentos utilizando Django y Django REST Framework (DRF). El microservicio debe permitir subir, listar, consultar metadatos, descargar y eliminar archivos. El objetivo es poner en práctica la estructuración de proyectos, uso de Git y ramas, configuración del entorno, modelado de datos, despliegue y creación de endpoints RESTful.

Punto 1: Inicialización del Repositorio y Entorno de Desarrollo

Objetivo: Crear un entorno aislado de desarrollo y un repositorio versionado.

1. Crear un repositorio público en GitHub con el nombre exacto: django-document-service.
2. Clonar el repositorio en su máquina local.
3. Crear una nueva rama de desarrollo: feat/microservice.
4. Establecer la versión de Python 3.11 usando pyenv.

```
pyenv local 3.11.x
```

5. Crear un entorno virtual utilizando uno de los métodos aprendidos en clase virtualenv

```
python -m venv .venv
source .venv/bin/activate
```

6. Instalar Django y Django REST Framework como dependencias del proyecto.

```
pip install django djangorestframework
pip freeze > requirements.txt
```

7. Hacer commit y push de los cambios a la rama feat/microservice.

Punto 2: Crear Proyecto Django y App de Documentos

Objetivo: Crear el proyecto principal con Django y la app base documents. También se integrará la app en el proyecto y se confirmará la estructura esperada.

1. Desde la rama feat/scaffolding crear un nuevo proyecto llamado docservice
2. Crear una app llamada documents
3. Agregar 'documents' a la variable INSTALLED_APPS de settings.py.
4. Realizar commit y push de los cambios hacia la rama feat/scaffolding.
5. Fusionar los cambios feat/scaffolding con la rama main

Punto 3: Modelo Document y Registro en Admin

Aquí se define el modelo principal Document, que almacenará archivos y sus metadatos. Se realizarán las migraciones, se registrará en el administrador y se configurará la base de datos PostgreSQL externa.

1. Desde la rama main, crear una nueva rama llamada feat/microservice.
2. Crear el modelo Document en documents/models.py con los campos: file, name, uploaded_at, size, content_type.

```
from django.db import models

class Document(models.Model):
    file = models.FileField(upload_to="documents/")
    name = models.CharField(max_length=255)
    uploaded_at = models.DateTimeField(auto_now_add=True)
    size = models.PositiveIntegerField()
    content_type = models.CharField(max_length=100)

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

3. Registrar el modelo Document en documents/admin.py.
4. Realizar migraciones con makemigrations y migrate.
5. Modificar settings.py para conectar el proyecto a una base de datos PostgreSQL local o externa.

Punto 4: Endpoints RESTful con DRF

Objetivo: Crear los endpoints necesarios usando DRF para permitir el manejo de archivos. Implementar vistas con APIView, serializadores, rutas y configurar el almacenamiento local en MEDIA_ROOT.

1. Crear serializers.py en la app documents y definir DocumentSerializer.
2. Implementar vistas en views.py usando APIView para los siguientes endpoints:
   - POST /documents/: subir archivo
   - GET /documents/: listar archivos
   - GET /documents/{id}/: obtener metadatos
   - GET /documents/{id}/download/: descargar archivo
   - DELETE /documents/{id}/: eliminar archivo
3. Crear documents/urls.py e incluirlo en docservice/urls.py.
4. Configurar MEDIA_ROOT y MEDIA_URL en settings.py.
5. Agregar static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT) en urls.py.
6. Opcional: integrar django-storages y AWS S3 para almacenamiento externo.

Punto 5: Despliegue con Docker y Kubernetes

En este último paso, el objetivo es contenerizar el microservicio y preparar su despliegue usando Kubernetes. Se deben crear los manifiestos y archivos necesarios para un entorno productivo moderno.

1. Crear un archivo Dockerfile para contenerizar el microservicio Django.
2. Crear docker-compose.yml con servicios para la app Django y PostgreSQL.
3. Crear el archivo deployment.yaml para el deployment de Kubernetes.
4. Crear el archivo service.yaml para exponer el microservicio.
5. Crear el archivo ingress.yaml para enrutamiento HTTP externo.
6. Crear el archivo secret.yaml con variables de entorno sensibles.
7. Verificar despliegue en Minikube o clúster remoto (opcional).

Evidencias:

El repositorio puede ser creado de forma pública o privada, en caso de ser pública compartir el repositorio a través de una URL vía correo electrónico a

hlozada@outlook.com, en caso de elegir repositorio privado puede agregarme directamente como colaborador en el proyecto.

Puntaje Total (100 puntos):

- Punto 1: Configuración del entorno y Git – 20 puntos
- Punto 2: Scaffolding y estructura inicial – 20 puntos
- Punto 3: Modelo, admin y base de datos – 20 puntos
- Punto 4: Endpoints RESTful con DRF – 20 puntos
- Punto 5: Contenerización y despliegue – 20 puntos