2 sesiones

Práctica 1: Preparación de la Base de Datos

José María Guirao (jmguirao@ugr.es)

Durante las prácticas de la asignatura iremos haciendo gradualmete una aplicación de tienda virtual. Empezaremos en esta primera práctica por rellenar la BD con datos de prueba, y hacer algunas consultas.

Usaremos MongoDB como base de datos. Es una base de datos NO-SQL, orientada a documentos, que entre otras características, no necesita normalización, y los registros no tienen que ser uniformes entre ellos. Para utilizar **MongoDB** desde python usaremos el cliente síncrono PyMongo. En este tutorial tenemos una breve introducción.

Usaremos también la librería Pydantic para poder tener un esquema de la BD contra el que poder validar los datos.

Los datos inciales los traemos del **api** https://fakestoreapi.com/products. Usaremos la librería requests, que sirve para hacer peticiones http (`requests`) desde python.

Después añadiremos compras, referenciando los productos por su índice de monogdb

Un bosquejo de la aplicación:

```
# Seed.py
from pydantic import BaseModel, FilePath, Field, EmailStr
from pymongo import MongoClient
from pprint import pprint
from datetime import datetime
from typing import Any
import requests
```

```
# https://requests.readthedocs.io/en/latest/
def getProductos(api):
    response = requests.get(api)
    return response.json()

# Esquema de la BD
# https://docs.pydantic.dev/latest/
# con anotaciones de tipo https://docs.python.org/3/library/typing.html
# https://docs.pydantic.dev/latest/usage/fields/

class Nota(BaseModel):
    puntuación: float = Field(ge=0., lt=5.)
    cuenta: int = Field(ge=1)

class Producto(BaseModel):
    _id: Any
    nombre: str
    precio: float
    descripción: str
    categoría: str
    imágen: FilePath | None
    rating: Nota

class Compra(BaseModel):
    _id: Any
    usuario: EmailStr
    fecha: datetime
    productos: list


dato = {
    'nombre': "MBJ Women's Solid Short Sleeve Boat Neck V ",
    'precio': 9.85,
    'descripción': '95% RAYON 5% SPANDEX, Made in USA or Imported, Do Not Ble
    'categoría': "women's clothing",
    'imágen': None,
    'rating': {'puntuación': 4.7, 'cuenta': 130}
}

# Valida con el esquema:
# daría error si no corresponde algún tipo
producto = Producto(**dato)
```

```
print(producto.descripción)
pprint(producto.model_dump()) # Objeto -> python dict


# Conexión con la BD
# https://pymongo.readthedocs.io/en/stable/tutorial.html
client = MongoClient('mongo', 27017)

tienda_db = client.tienda                    # Base de Datos
productos_collection = tienda_db.productos  # Colección

productos_collection.insert_one(producto.model_dump())

print(productos_collection.count_documents({}))

# todos los productos
lista_productos_ids = []
for prod in productos_collection.find():
    pprint(prod)
    print(prod.get('_id'))   # Autoinsertado por mongo
    lista_productos_ids.append(prod.get('_id'))

print(lista_productos_ids)

nueva_compra = {
    'usuario': 'fulanito@correo.com',
    'fecha': datetime.now(),
    'productos': lista_productos_ids
}

# valida
compra = Compra(**nueva_compra)
pprint(compra.model_dump())
# añade a BD
compras_collection = tienda_db.compras  # Colección
compras_collection.insert_one(compra.model_dump())

for com in compras_collection.find():
    pprint(com)


...
```

```
# productos = getProductos('https://fakestoreapi.com/products')
# for p in productos:
#   print(p)
```

Las imágenes las almacenamos en un directorio, y en el campo 'imágen' anotamos el nombre del archivo, que debe ser único.

Contenedores

Los archivos serían ahora:

```
.
├── data
├── e-commerce
│   ├── imágenes
│   ├── Dockerfile
│   ├── requirements.txt
│   └── Seed.py
└── docker-compose.yml
```

y

docker-compose.yml:

```
# docker-compose.yml
services:
  app:
    build: ./e-commerce
    volumes:
      - ./e-commerce:/e-commerce
    depends_on:
      - mongo

  mongo:
    image: mongo:6.0
    ports:
      - 27017:27017
    volumes:
      - ./data:/data/db
```

El archivo **Dockerfile**:

```
# Dockerfile
FROM python:3.11-alpine

WORKDIR /e-commerce
COPY . /e-commerce
RUN pip install -r requirements.txt
```

y el archivo **requirements.txt** para la instalación con `pip` de las librerías de python en el contenedor **app**:

```
# requirements.txt
pymongo==4.5
pydantic==2.3
requests==2.31
email-validator==2.0
typing-extensions==4.8
```

Consultas:

- Electrónica entre 100 y 200€, ordenados por precio
- Productos que contengan la palabra 'pocket' en la descripción
- Productos con puntuación mayor de 4
- Ropa de hombre, ordenada por puntuación
- Facturación total
- Facturación por categoría de producto

Referencias:

- query documents
- regex search

Para nota

- Hacer una copia de seguridad de la BD (mongodump)
- Añadir una validación adiccional para asegurarse que el contenido en el campo 'nombre' empieza por mayúscula (Field validators)