Laboratorio 13: MongoDB aplicado a la Gestión de Tiendas Online

Prof. Heider Sanchez
ACLs: Ana María Accilio, Sebastián Loza

P1: (8 pts) Gestión de Productos y Pedidos

Objetivo: Aplicar los conceptos fundamentales de MongoDB para diseñar y manipular colecciones de productos y pedidos de un e-commerce, incluyendo la creación de estructuras de documentos, inserción de datos con arrays y documentos anidados, consultas básicas, etc.

> **Herramienta requerida:** Utilizar un cliente gráfico de MongoDB como **MongoDB Compass** o **DataGrip** para realizar todas las operaciones de esta sección. Esto permitirá visualizar mejor la estructura de los documentos, validar los resultados de las consultas y adjuntar las evidencias en el informe.

1. Crear la colección de `productos`, `clientes` y `pedidos`:
2. Insertar datos:

- Agregar datos sintéticos a dichas colecciones. Debe poblar con al menos 100 datos en cada colección.
- Usar arrays y documentos anidados (ejemplo, lista de productos en cada pedido).
- Tomar como ejemplo de datos los archivos "productos.json", "clientes.json" y "pedidos.json".

4. Consultas:

- Obtener todos los clientes del país de Chile.
- Obtener los pedidos de un cliente específico (por el ID).
- Obtener los pedidos con un total superior a cierto valor.
- Contar el número total de pedidos entre un rango de fechas
- ¿Cuáles son los k productos más vendidos?
- ¿Quienes son los clientes más frecuentes?

5. Reto adicional:
    - Crear un índice en el campo `fecha_pedido` y `id_cliente` para acelerar las consultas y proponer alguna consulta que utilice estos índices.

P2: (12 pts) Gestión de Productos con Python

Objetivo: Desarrollar una aplicación completa en Python que integre MongoDB con APIs web externas, implementando operaciones CRUD avanzadas, consultas de agregación y manejo de datos del mundo real para un sistema de inventario de un e-commerce.

> **Herramienta requerida:** Utilizar **Python** con las librerías `pymongo`, `requests` y `json` para permitir la gestión automatizada de datos MongoDB y la integración con servicios web externos.

1. (6 pts) Cargar datos desde una API Web Pública

- Asegúrate de tener las siguientes librerías instaladas `jsonify`, `request`, y `pymongo`.
- Crear una clase en Python `MongoDBHandler` para concentrar todos los métodos de extracción y manipulación de datos. Se le proporciona un código base de dicha clase.
  - En el constructor de la clase debe colocar el nombre de la Base de datos y de las Colecciones (`categorias`, `productos`).
  - Crear un método para cargar todas las categorías de los productos desde la API Web de categorías (https://dummyjson.com/products/categories).
  - Crear un método para cargar todos productos (194 en total) categoría por categoría. La url para consultar los productos de cada categoría se encuentra en la misma colección de categorías previamente cargada. Utilizar el parámetro limit para traer todos los elementos.
  - La siguiente url trae los primeros 5 productos de la categoría `tablet`: https://dummyjson.com/products/category/tablets?limit=0
- A continuación, un ejemplo de la clase `MongoDBHandler`:

```python
import requests
from pymongo import MongoClient

class MongoDBHandler:
    def __init__(self):
        self.mongo_uri = "mongodb://localhost:27018/"
        self.db_name = "pucp_store"
        self.categories_name="categorias"
        self.productos_name="productos"
        self.connect_mongo()

    # Conectar a MongoDB
    def connect_mongo(self):
        try:
            client = MongoClient(self.mongo_uri)
            self.db = client[self.db_name] if client else None
            print(f"Conectado a la base de datos '{self.db_name}' en MongoDB.")
        except Exception as e:
            print(f"Error al conectar a MongoDB: {e}")

    # Obtener datos JSON desde la URL
    def fetch_json_data(self, url):
        try:
            response = requests.get(url)
            response.raise_for_status()
            # Lanza una excepción si el código de estado no es 200
            return response.json()  # Devuelve los datos JSON
        except requests.exceptions.HTTPError as errh:
            print("HTTP Error:", errh)
        except requests.exceptions.ConnectionError as errc:
            print("Error de conexión:", errc)
        except requests.exceptions.Timeout as errt:
            print("Timeout Error:", errt)
        except requests.exceptions.RequestException as err:
            print("Error desconocido:", err)
        return None

    # Obtener todos las categorias
    def fetch_all_categories(self, base_url):
        # TODO
        pass

    # Obtener todos los productos
    def fetch_all_products(self):
        # TODO
        pass
```

- Ejecutar la carga de datos:

```
# Crear instancia de MongoDBHandler
mongo_handler = MongoDBHandler()

# URL del JSON para las categorías y productos
categories_url = "https://dummyjson.com/products/categories"

# 1. Cargar todas las categorías
mongo_handler.fetch_all_categories(categories_url)

# 2. Cargar todos los productos de cada categoría
mongo_handler.fetch_all_products()
```

2. (6 pts) Agregar operaciones CRUD y filtrado personalizado

- Agregar funcionalidades de manipulación y consulta de datos a la clase `MongoDBHandler`:

```python
# Crear instancia de MongoDBHandler
mongo_handler = MongoDBHandler()

# Crear índice en campo especifico
mongo_handler.crear_indice(campo = "nombre")

# Crear productos
mongo_handler.crear_producto({"nombre": "Producto 1", "precio": 100})
mongo_handler.crear_producto({"nombre": "Producto 2", "precio": 200})
mongo_handler.crear_producto({"nombre": "Producto 3", "precio": 300})

# Leer productos
productos = mongo_handler.obtener_productos()
print("Lista de productos:", productos)

# Leer producto específico
producto = mongo_handler.obtener_producto({"_id": productos[0]['_id']})
print("Producto obtenido:", producto)

# Actualizar un producto
if productos:
    mongo_handler.actualizar_producto({"_id": productos[0]['_id']},
                                      {"precio": 120})

# Eliminar un producto
if productos:
    mongo_handler.eliminar_producto({"_id": productos[0]['_id']})

# Consultas básicas
print("Productos con precio 100:", mongo_handler.obtener_productos_por_precio(100))
print("Productos que contienen '1' en el nombre:",
mongo_handler.obtener_productos_por_nombre("5G"))

# Consultas agregadas
print("Precio promedio de productos:", mongo_handler.precio_promedio())
print("Total de productos:", mongo_handler.contar_productos())
print("El producto con mayor stock por categoría:",
mongo_handler.mayor_stock_categoria())
```

Entregable

Presentar un informe con las siguientes evidencias organizadas de manera clara:

Parte 1: MongoDB con Cliente Gráfico (8 pts)

- **Código:** Consultas MongoDB utilizadas.
- **Capturas:** Screenshots del cliente gráfico mostrando colecciones, documentos, consultas e índices.
- **Explicación:** Descripción breve de cada operación.

Parte 2: Python + MongoDB (12 pts)

- **Código:** Un Notebook con la clase `MongoDBHandler` completa y funcional.
- **Capturas:** Ejecución del programa mostrando conexión, carga de datos y operaciones CRUD.
- **Resultados:** Salidas de las consultas agregadas.

> **Importante:** Capturas legibles, código bien formateado y contenido organizado secuencialmente.