

TPO

Motor de Base de Datos MongoDB con Aplicación de Software Python



Objetivo:

- Aprender a conectar MongoDB con Python.
- Practicar operaciones CRUD en MongoDB usando Python.
- Explorar consultas avanzadas y manipulación de datos con PyMongo.
- Aplicar conocimientos en un caso práctico real.
- Evidenciar cada uno de los puntos solicitados como enunciado.

Requisitos previos

1. MongoDB instalado (incluyendo MongoDB Shell y Compass).
2. Python instalado con la librería `pymongo`.
Se instala con:
`pip install pymongo`
 -
3. Un dataset a elección como propuesta del equipo de trabajo JSON para cargar en MongoDB.

Parte 0: Entorno de Trabajo: Instalación

Evidenciar en qué contexto se trabajará tanto para MongoDB, como Python (Notebook con políticas de seguridad, sistema operativo, etc., es decir, tomar en cuenta los aspectos y feedback mencionados en la clase sobre la instalación de MongoDB).

Parte 1: Instalación y Conexión de MongoDB con Python

Conectar Python con MongoDB

- 1) Crear un script en Python que establezca una conexión con MongoDB y liste las bases de datos disponibles.
- 2) Crear una base de datos y una colección en base a la propuesta del equipo.

Parte 2: Cargar y Consultar Datos en MongoDB

- 3) Insertar documentos en la colección (manualmente).
- 4) Consultar todos los documentos realizados.
- 5) Realizar 7 propuestas de filtros sobre los documentos trabajados.

Parte 3: Actualizar y Eliminar Datos

- 6) Realizar 7 propuestas de actualización sobre los documentos trabajados.
- 7) Realizar 7 propuestas de eliminación sobre los documentos trabajados.

NOTA: Tener en cuenta la aplicación de “filtros”.

Parte 4: Proyecto Final - Análisis de Datos en MongoDB

Caso práctico:

- 8) Investigar herramientas de autogeneración de datos a través de Script para simulación de datos en MongoDB. El mismo se deberá dejar como evidencia en la entrega bajo un archivo **TemaSeleccionadoPorElGrupo.json** (dataset ficticio).

Una vez realizado dicho paso, se deberá:

- Cargar el archivo JSON en MongoDB.

Se puede hacer desde MongoDB Compass o con Python:

```
1  import json
2
3  ✓ with open("empleados.json") as file:
4      data = json.load(file)
5      collection.insert_many(data)
6  print("Datos importados correctamente")
7
```

- 9) Sobre dichos datos generados (tema a elección del equipo de trabajo), se deberá realizar la propuesta de 5 consultas, por ejemplo del tipo:
- a) Listar empleados mayores de 30 años con salario superior a 7000.
 - b) Contar cuántos empleados hay en cada cargo.
 - c) Obtener el promedio de salario por cargo.
- 10) Exportar los resultados a un archivo CSV, y mostrar habilidades con manejo de pandas de python, por ejemplo: Usar **pandas** para guardar la salida en CSV.

```
1  import pandas as pd
2
3  empleados = list(collection.find({}, {"_id": 0}))
4  df = pd.DataFrame(empleados)
5  df.to_csv("empleados_exportados.csv", index=False)
6  print("Datos exportados correctamente")
7
```

Entrega del TPO

Cada equipo deberá subir un informe en formato PDF con:


- Código utilizado en Python.
- Capturas de pantalla de MongoDB Compass mostrando los datos.
- Resultados de las consultas y análisis realizados.

Fechas

Fecha de entrega Enunciado:

 Hoy, Clase 03, 27 de marzo 2025 Turno Tarde

Fecha límite de entrega:

 Clase 24 de abril 2025, a las 7:00 AM en el canal de Teams de cada equipo de trabajo.

EXTRA: Conexión de MongoDB con una API y Dashboard

Objetivo:

- Crear una API en Flask para exponer los datos de MongoDB.
- Consumir los datos desde un dashboard en Power BI o Tableau.

Paso a paso:

- ♦ Primero, instala Flask y PyMongo si aún no lo tienes:

Sección extra para el TP, donde aprenderás a conectar MongoDB con una API en Flask y luego a visualizar los datos en Power BI o Tableau.

EXTRA: Conexión de MongoDB con una API y Dashboard

Objetivo:

- Crear una API en Flask para exponer los datos de MongoDB.
- Consumir los datos desde un dashboard en Power BI o Tableau.

1 Crear una API en Flask que exponga datos de MongoDB

- ♦ Primero, instala Flask y PyMongo si aún no lo tienes:

```
pip install flask pymongo
```

- ♦ Luego, crea un archivo `app.py` con el siguiente código:

```
from flask import Flask, jsonify
```

```
from pymongo import MongoClient
```

```
app = Flask(__name__)
```

```
# Conexión con MongoDB
```

```
client = MongoClient("mongodb://localhost:27017/")
```

```
db = client["empresa"]
```

```
collection = db["empleados"]
```

```
@app.route('/empleados', methods=['GET'])
```

```
def get_empleados():
```

```
    empleados = list(collection.find({}, {"_id": 0})) # Excluye el _id para evitar problemas en JSON
```

```
    return jsonify(empleados)
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

- ♦ Ejecuta la API:

```
python app.py
```

- ♦ Prueba en el navegador:

- Abre <http://127.0.0.1:5000/empleados> y verás los datos en formato JSON.

Adaptar el tema a la propuesta del equipo

2 Conectar MongoDB con Power BI o Tableau

 En Power BI

1. Abre Power BI Desktop.
2. Ve a Obtener Datos → URL Web.

Ingresa la URL de la API:

<http://127.0.0.1:5000/empleados>

3. Carga los datos y crea visualizaciones como:

- Gráfico de barras: Comparación de salarios por cargo.
- Tabla dinámica: Cantidad de empleados por departamento.

 En Tableau

1. Abre Tableau y selecciona Conectar a Servidor → JSON.
2. Ingresa la URL de la API.
3. Carga los datos y crear visualizaciones interactivas.

Entrega del Extra

Cada equipo deberá:

- Capturar la pantalla de la API funcionando.
- Mostrar el dashboard en Power BI o Tableau con las visualizaciones.

ASPECTOS IMPORTANTES:

CRUD es un acrónimo que significa "Crear, Leer, Actualizar y Eliminar". En MongoDB, las operaciones CRUD son las acciones básicas para interactuar con la base de datos.

Operaciones CRUD en MongoDB

- **Crear:** Insertar nuevos documentos en la base de datos
- **Leer:** Consultar documentos en la base de datos
- **Actualizar:** Modificar documentos existentes en la base de datos
- **Eliminar:** Eliminar documentos de la base de datos