



INSTITUTO TECNOLÓGICO DEL PUTUMAYO



IES Vigilada por:
Educación

PROGRAMA INGENIERIA DE SISTEMAS

Página 1 | 55

INFORME - MONGODB CRUD

Presentado por: Cristian Camilo Vallejos Bastidas

Presentado: Ing. Brayan Arcos



El **Saber** como **Arma** de **Vida**



PROGRAMA INGENIERIA DE SISTEMAS

Página 2 | 55

Índice

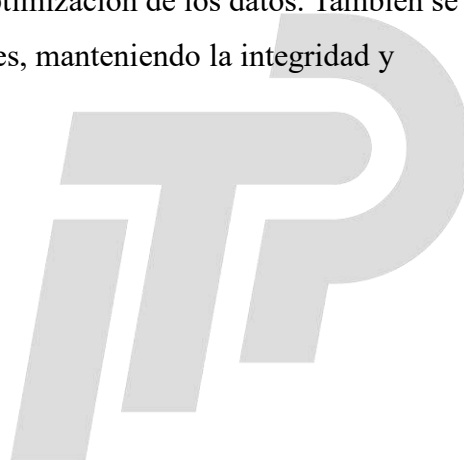
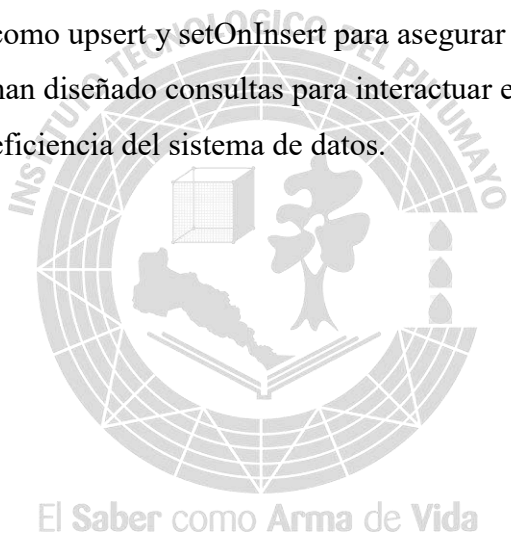
1. Resumen Ejecutivo	3
2. Introducción.....	4
2.1 Contexto y Motivación.....	4
2.2 Alcance del Informe	4
2.3 Objetivos	5
3. Metodología.....	6
3.1 Herramientas Utilizadas	6
3.2 Procedimientos.....	6
4. Desarrollo del Informe.....	8
4.1 Descripción de la Base de Datos de Datos Personal	8
4.3 Métodos de captura:	12
4.4 Consultas	17
5. Análisis y Discusión	51
6. Conclusiones.....	53
7. Recomendaciones	54
8. Referencias	55

PROGRAMA INGENIERIA DE SISTEMAS

Página 3 | 55

1. Resumen Ejecutivo

En esta práctica, se ha desarrollado un sistema de reserva de vuelos utilizando MongoDB, una base de datos NoSQL orientada a documentos y también un taller práctico de MongoDB relacionado con el tema CRUD. El objetivo principal ha sido modelar de manera eficiente las relaciones entre usuarios, aeropuertos, vuelos y reservas, aplicando técnicas de normalización y diseño de esquemas adecuados para simular un sistema realista de reservas de vuelos. Durante el desarrollo del sistema, se han implementado diversas operaciones de inserción y actualización de documentos mediante funciones como `updateOne()`, `updateMany()`, aplicando operadores clave como `upsert` y `setOnInsert` para asegurar la coherencia y optimización de los datos. También se han diseñado consultas para interactuar entre las colecciones, manteniendo la integridad y eficiencia del sistema de datos.



PROGRAMA INGENIERIA DE SISTEMAS

Página 4 | 55

2. Introducción

2.1 Contexto y Motivación

El uso de bases de datos NoSQL, como MongoDB, es cada vez más relevante en el ámbito académico y profesional debido a su flexibilidad para manejar datos no estructurados y semiestructurados. Realizar prácticas de implementación de bases de datos es importante en la formación de los futuros profesionales de áreas como Ingeniería de Sistemas. Esta práctica, centrada en el desarrollo de un sistema de reserva de vuelos en MongoDB, permite a los estudiantes poner en práctica conceptos clave como la creación de colecciones, la normalización de datos, y la ejecución de consultas avanzadas, todo en un entorno que simula situaciones del mundo real.

2.2 Alcance del Informe

Este informe tiene los siguientes aspectos de MongoDB en el marco de la práctica:

- **Diseño de la base de datos:** Creación de colecciones para usuarios, aeropuertos, vuelos y reservas, incluyendo la configuración de relaciones entre ellas.
- **Consultas CRUD:** Implementación de consultas para la creación, lectura, actualización y eliminación de datos, usando operadores avanzados para optimizar las operaciones.
- **Actualización de datos:** Uso de operadores como upsert, setOnInsert, each, y set para realizar actualizaciones eficientes y manejos de datos en tiempo real.
- **Normalización y embebido de datos:** Explicación sobre el uso de referencias y documentos embebidos para representar relaciones entre colecciones y su impacto en el diseño.

PROGRAMA INGENIERIA DE SISTEMAS

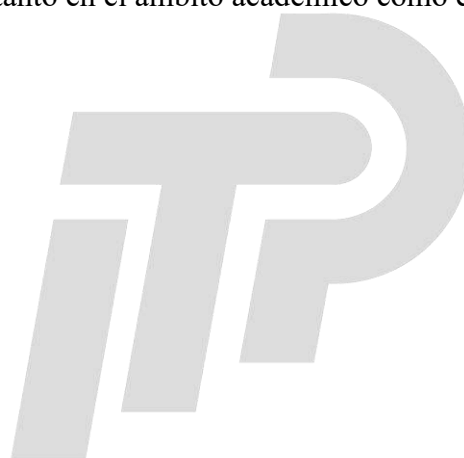
Página 5 | 55

2.3 Objetivos

- Desarrollar un sistema de reserva de vuelos en MongoDB para aplicar y consolidar los conocimientos sobre bases de datos NoSQL.
- Implementar un esquema de diseño de base de datos adecuado, que permita el manejo eficiente de datos mediante la normalización y el uso de referencias.
- Proporcionar ejemplos de consultas CRUD y actualizaciones avanzadas que permitan interactuar con la base de datos de manera óptima.
- Demostrar cómo MongoDB puede ser una herramienta clave en la construcción de sistemas escalables, aplicable a diversos proyectos tanto en el ámbito académico como en la industria.



El **Saber** como **Arma de Vida**



PROGRAMA INGENIERIA DE SISTEMAS

Página 6 | 55

3. Metodología

3.1 Herramientas Utilizadas

Para llevar a cabo esta práctica de diseño y gestión de la base de datos en MongoDB, se utilizaron las siguientes herramientas:

- **MongoDB Compass:** Utilizado para la creación y administración visual de las colecciones, permitiendo observar de manera gráfica las relaciones entre los datos y su estructura.
- **MongoDB Database Tools:** Herramienta empleada para la exportación e importación de bases de datos, facilitando la manipulación y respaldo de los datos generados.
- **Studio 3T:** Un entorno gráfico avanzado para la creación de consultas complejas, análisis de datos y visualización de resultados, lo que permitió agilizar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar).

3.2 Procedimientos

Creación de la Base de Datos Personal: En esta fase inicial, se procedió a la creación de la estructura de la base de datos con las colecciones necesarias, utilizando MongoDB Compass y siguiendo un esquema relacional para garantizar la integridad de los datos.

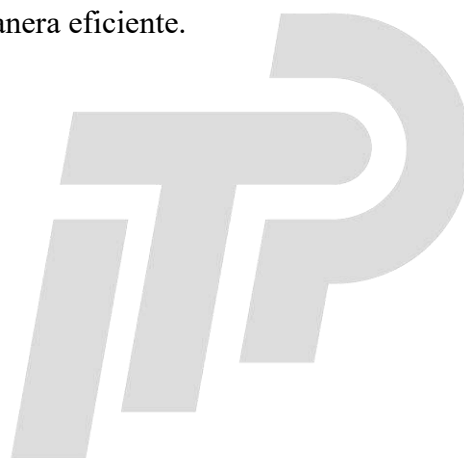
- **Usuarios:** Se creó una colección para almacenar la información básica de los usuarios del sistema de reservas.
- **Aeropuertos:** Una colección para representar los distintos aeropuertos disponibles en el sistema.
- **Vuelos:** Se implementó una colección para almacenar los vuelos, vinculados a los aeropuertos mediante relaciones 1 a muchos.
- **Reservas:** Finalmente, una colección para gestionar las reservas de los usuarios, relacionando usuarios y vuelos.

PROGRAMA INGENIERIA DE SISTEMAS

Página 7 | 55

Realización del taller: En esta parte, el taller consistía en una serie de pasos para agregar datos en archivos JSON a colecciones que se tenían que crear en MongoDB Compass, pero primero tenían que extraer información mediante una pagina web llamada: jsongrid.com para así visualizar la información y extraerla para las consultas del CRUD que se iba ir desarrollando.

Métodos de Captura: Para la creación de la base de datos, se utilizó un enfoque de captura de datos desde archivos **JSON** previamente estructurados. Estos archivos contenían la información correspondiente a cada colección (usuarios, aeropuertos, vuelos, reservas) y fueron ingresados a MongoDB mediante herramientas como **MongoDB Compass** y **Studio 3T**. Esto permitió una inserción masiva de los documentos, asegurando la consistencia y exactitud en los datos sin necesidad de ingresar la información manualmente, lo que facilitó el proceso de carga de datos y permitió manejar grandes volúmenes de información de manera eficiente.



4. Desarrollo del Informe

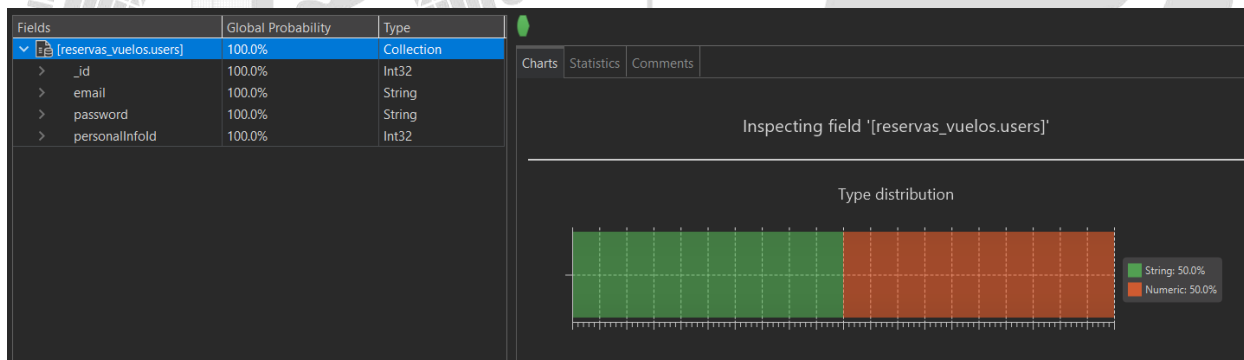
4.1 Descripción de la Base de Datos de Datos Personal

1. Colección: users

- **_id** (Int32): Identificador único de cada usuario.
- **email** (String): Correo electrónico del usuario, que actúa como un identificador adicional.
- **password** (String): Contraseña del usuario para acceso al sistema.
- **personalInfoId** (Int32): Referencia a la colección de información personal, estableciendo una relación entre el usuario y sus datos personales.

Distribución de Tipos:

- **String**: 50%
- **Numeric**: 50%



2. Colección: personal_info

- **_id** (Int32): Identificador único para cada conjunto de información personal.
- **address** (Object): Contiene detalles sobre la dirección del usuario, como calle, ciudad y país.
- **birthDate** (String): Fecha de nacimiento del usuario.
- **fullName** (Object): Información del nombre completo, que podría descomponerse en nombres y apellidos.
- **nationality** (String): Nacionalidad del usuario.

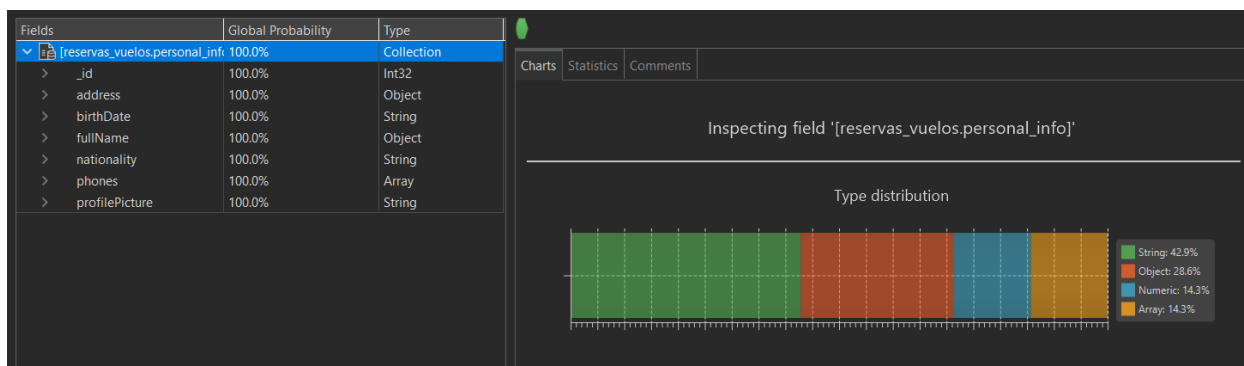
PROGRAMA INGENIERIA DE SISTEMAS

Página 9 | 55

- **phones** (Array): Lista de números telefónicos que pertenecen al usuario.
- **profilePicture** (String): URL o ruta de la imagen de perfil del usuario.

Distribución de Tipos:

- **String**: 42.9%
- **Object**: 28.6%
- **Numeric**: 14.3%
- **Array**: 14.3%

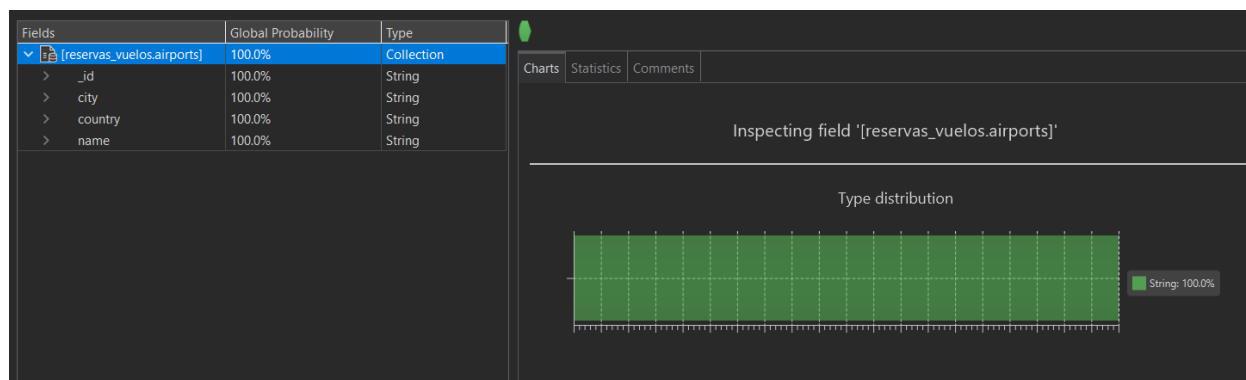


3. Colección: reservas_vuelos.airports

- **_id** (String): Identificador único de cada aeropuerto.
- **city** (String): Nombre de la ciudad donde se encuentra el aeropuerto.
- **country** (String): País donde se encuentra el aeropuerto.
- **name** (String): Nombre del aeropuerto.

Distribución de Tipos:

- **String**: 100%

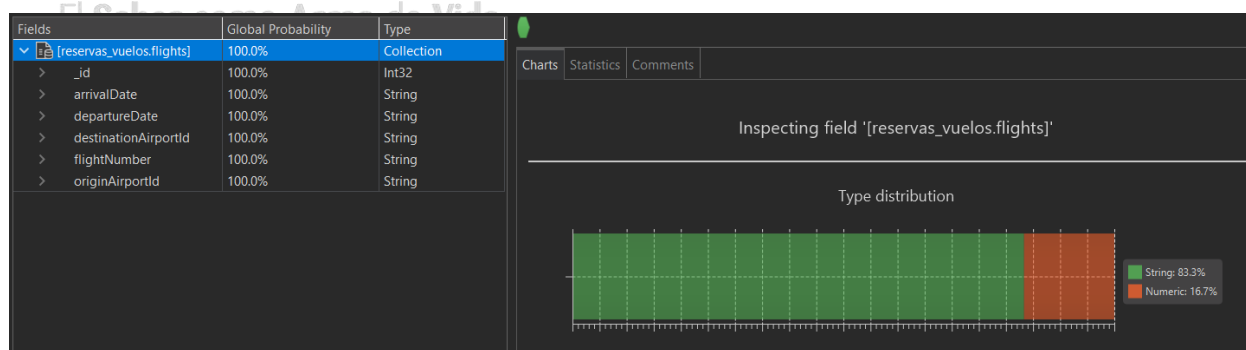


4. Colección: reservas_vuelos.flights

- **_id** (Int32): Identificador único de cada vuelo.
- **arrivalDate** (String): Fecha y hora de llegada del vuelo.
- **departureDate** (String): Fecha y hora de salida del vuelo.
- **destinationAirportId** (String): Código del aeropuerto de destino.
- **flightNumber** (String): Número del vuelo.
- **originAirportId** (String): Código del aeropuerto de origen.

Distribución de Tipos:

- String: 83.3%
- Numeric: 16.7%



PROGRAMA INGENIERIA DE SISTEMAS

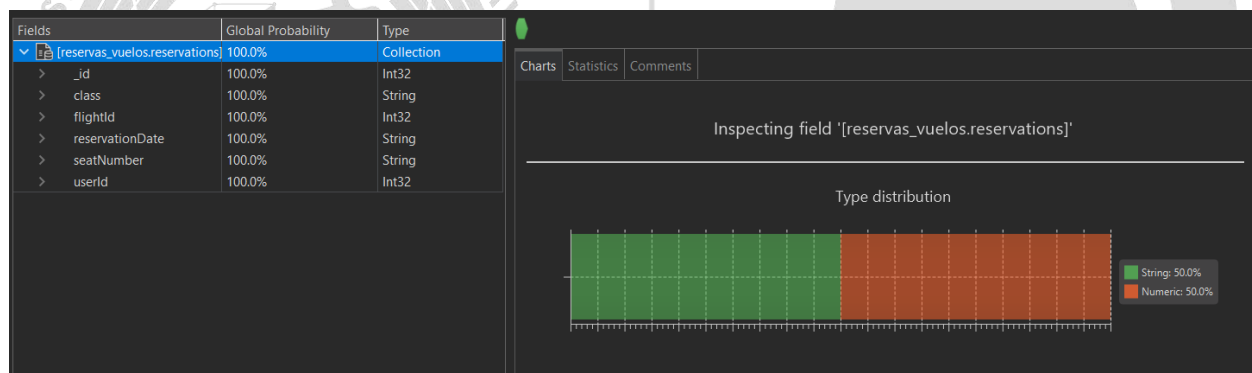
Página 11 | 55

5. Colección: reservas_vuelos.reservations

- **_id** (Int32): Identificador único de cada reserva.
- **class** (String): Clase en la que se reserva el asiento (por ejemplo, Economy o Business).
- **flightId** (Int32): Referencia al vuelo reservado.
- **reservationDate** (String): Fecha en la que se realizó la reserva.
- **seatNumber** (String): Número de asiento reservado.
- **userId** (Int32): Identificador del usuario que realizó la reserva.
-

Distribución de Tipos:

- String: 50%
- Numeric: 50%



PROGRAMA INGENIERIA DE SISTEMAS

Página 12 | 55

4.3 Métodos de captura:

Preparación de la Base de Datos de reservas_vuelos

1. Cargar los datos de los ficheros JSON en la base de datos Reservas:

- Abre MongoDB Compass y conecta a tu servidor de MongoDB.
- Si aún no tienes una base de datos llamada reservas_vuelos, crea una nueva base de datos con ese nombre.

2. Colección: Users:

- Carga los datos del fichero documento_users.json en la colección users.
- En MongoDB Compass, selecciona la base de datos Reservas, luego haz clic en "Crear Colección" y nómbrala users.
- Haz clic en "Importar Datos" y selecciona el fichero documento_users.json. Asegúrate de que el formato sea correcto y confirma la carga.

3. Colección: Personal_Info:

- Carga los datos del fichero documento_personalinfo.json en la colección personal_info.
- Repite el proceso anterior: selecciona la base de datos Reservas, crea la colección personal_Info, y luego importa el fichero correspondiente.

4. Colección: Flights:

- Carga los datos del fichero documento_flights.json en la colección flights.
- Sigue el mismo procedimiento para crear la colección y cargar el fichero.

5. Colección: Airports:

- Carga los datos del fichero documento_airports.json en la colección airports.
- Crea la colección y carga los datos de manera similar.

6. Colección: Reservations:

- Carga los datos del fichero documento_reservations.json en la colección reservations.
- Crea la colección y realiza la carga de datos.

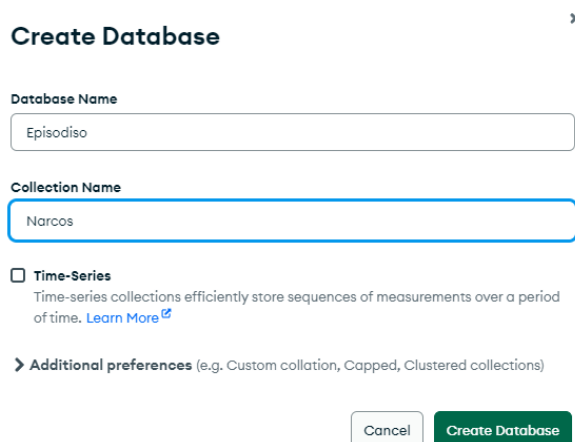
PROGRAMA INGENIERIA DE SISTEMAS

P á g i n a 13 | 55

Preparación de la Base de Datos de Episodios (Taller Practico):

PREPARA LOS DATOS

- 1) Carga los datos del fichero “ejercicio_00.json” en la base de datos llamada Episodios, sobre la colección Narcos.



- 2) Visualiza el contenido del fichero “ejercicio_01.json” con la herramienta online <https://jsongrid.com/json-viewer>. Observa la estructura del documento y recupera solo la lista de episodios que contienen dicho fichero.

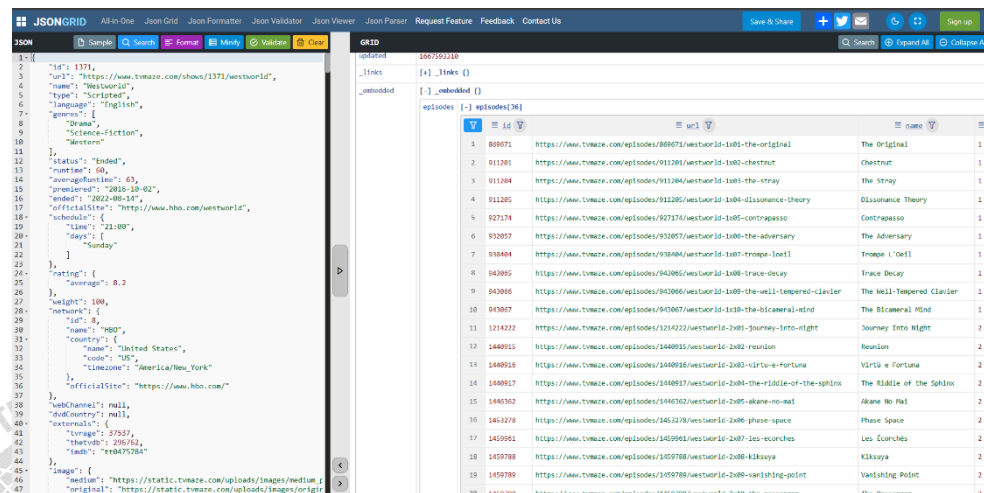
Con la lista de episodios crea otro fichero JSON llamado “episodios_westworld.json”.

El Saber como Arma de Vida

PROGRAMA INGENIERIA DE SISTEMAS

Página 14 | 55

Usando MongoDB Compass carga dicho fichero en la colección “**Westworld**” dentro de la base de datos Episodios.



The screenshot displays the JSONGRID application interface. On the left, a JSON document for 'Westworld' is shown, detailing its metadata and episode list. On the right, a table view of the 'episodes' collection is displayed, listing episode numbers, titles, and URLs.

id	url	name
000071	https://www.twmaze.com/episodes/000071/westworld-1x01-the-original	The Original
011201	https://www.twmaze.com/episodes/011201/westworld-1x02-chestnut	Chestnut
011204	https://www.twmaze.com/episodes/011204/westworld-1x03-the-stray	The Stray
011205	https://www.twmaze.com/episodes/011205/westworld-1x04-dissonance-theory	Dissonance Theory
027174	https://www.twmaze.com/episodes/027174/westworld-1x05-contrapasso	Contrapasso
032057	https://www.twmaze.com/episodes/032057/westworld-1x06-the-adversary	The Adversary
030404	https://www.twmaze.com/episodes/030404/westworld-1x07-trope-joeil	Trope Joeil
043005	https://www.twmaze.com/episodes/043005/westworld-1x08-trace-decay	Trace Decay
043006	https://www.twmaze.com/episodes/043006/westworld-1x09-the-well-tempered-clavier	The Well-Tempered Clavier
043007	https://www.twmaze.com/episodes/043007/westworld-1x10-the-bicameral-mind	The Bicameral Mind
1214222	https://www.twmaze.com/episodes/1214222/westworld-2x01-journey-into-night	Journey Into Night
1440915	https://www.twmaze.com/episodes/1440915/westworld-2x02-reunion	Reunion
1440916	https://www.twmaze.com/episodes/1440916/westworld-2x03-virtu-e-fortuna	Virtu e Fortuna
1440917	https://www.twmaze.com/episodes/1440917/westworld-2x04-the-ridle-of-the-sphinx	The Riddle of the Sphinx
1440918	https://www.twmaze.com/episodes/1440918/westworld-2x05-akane-no-mai	Akane No Mai
1453278	https://www.twmaze.com/episodes/1453278/westworld-2x06-phase-space	Phase Space
1450961	https://www.twmaze.com/episodes/1450961/westworld-2x07-les-courches	Les Courches
1450968	https://www.twmaze.com/episodes/1450968/westworld-2x08-kikuya	Kikuya
1450969	https://www.twmaze.com/episodes/1450969/westworld-2x09-vanishing-point	Vanishing Point
1450970	https://www.twmaze.com/episodes/1450970/westworld-2x10-the-seasoner	The Seasoner

El Saber como Arma de Vida

PROGRAMA INGENIERIA DE SISTEMAS

Página 15 | 55

localhost:27017 > Episodios > Westworld

Documents 1 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

ADD DATA EXPORT DATA UPDATE DELETE

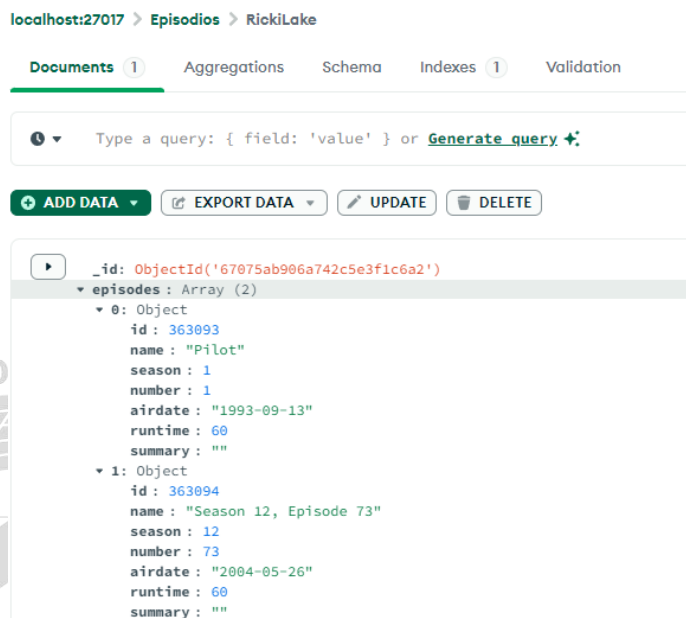
```
{
  "_id": "ObjectId('6707596f06a742c5e3f1c6a0')",
  "episodes": [
    {
      "id": 869671,
      "name": "The Original",
      "season": 1,
      "number": 1,
      "airdate": "2016-10-02",
      "runtime": 68,
      "rating": {
        "summary": "A woman named Dolores is a free spirit in the Old West..."
      }
    },
    {
      "id": 911201,
      "name": "Chestnut",
      "season": 1,
      "number": 2,
      "airdate": "2016-10-09",
      "runtime": 60,
      "rating": {
        "summary": "Bernard suspects that someone is sabotaging the hosts..."
      }
    },
    {
      "id": 911204,
      "name": "The Stray",
      "season": 1,
      "number": 3,
      "airdate": "2016-10-16",
      "runtime": 60,
      "rating": {
        "summary": "Bernard continues to investigate Dolores' supposed malfunction..."
      }
    },
    {
      "id": 911205,
      "name": "Dissonance Theory",
      "season": 1,
      "number": 4,
      "airdate": "2016-10-23",
      "runtime": 60,
      "rating": {
        "summary": "While Dolores joins William and Logan on their adventure..."
      }
    }
  ]
}
```

El Saber como Arma de Vida

PROGRAMA INGENIERIA DE SISTEMAS

Página 16 | 55

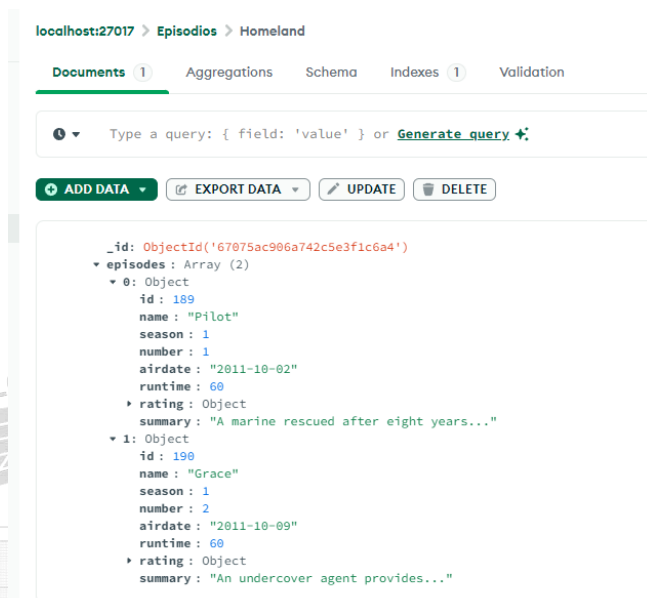
- 3) Repite el paso 2 con el fichero “ejercicio_02.json” pero esta vez la colección debe llamarse “RickiLake”.



PROGRAMA INGENIERIA DE SISTEMAS

Página 17 | 55

- 4) Repite el paso 2 con el fichero “ejercicio_03.json” pero esta vez la colección debe llamarse “Homeland”.



4.4 Consultas

Base de datos reservas_vuelos:

El Saber como Arma de Vida
CREATE

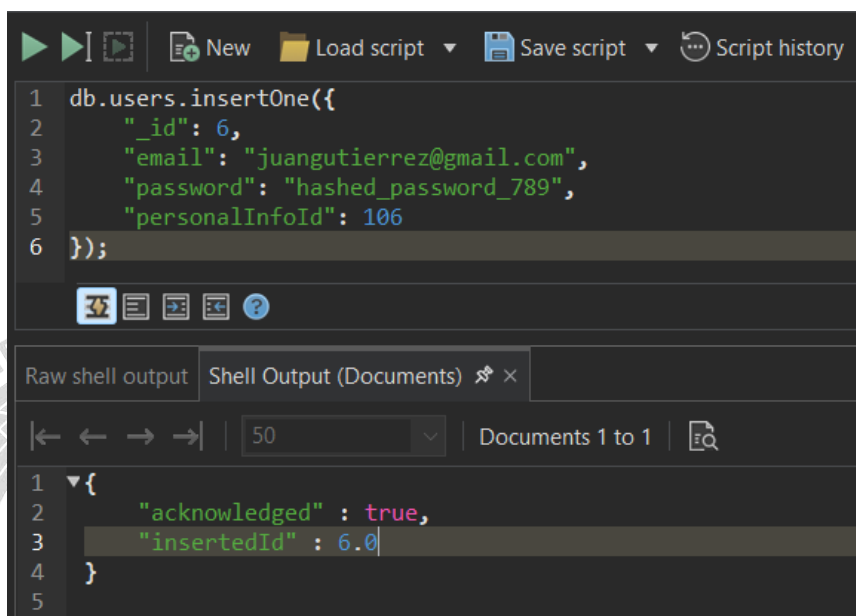
Consulta 1:

```
db.users.insertOne({
  "_id": 6,
  "email": "juangutierrez@gmail.com",
  "password": "hashed_password_789",
  "personalInfoId": 106
});
```

PROGRAMA INGENIERIA DE SISTEMAS

Página 18 | 55

Explicación: Se inserta un nuevo documento en la colección users con un identificador único (_id = 6). Se añaden los campos email, password y personalInfoId al documento. Esta operación crea un nuevo usuario en la base de datos.



```
1 db.users.insertOne({
2   "_id": 6,
3   "email": "juangutierrez@gmail.com",
4   "password": "hashed_password_789",
5   "personalInfoId": 106
6 });
```

Raw shell output | Shell Output (Documents) ×

Documents 1 to 1

```
1 {
2   "acknowledged" : true,
3   "insertedId" : 6.0
4 }
5
```

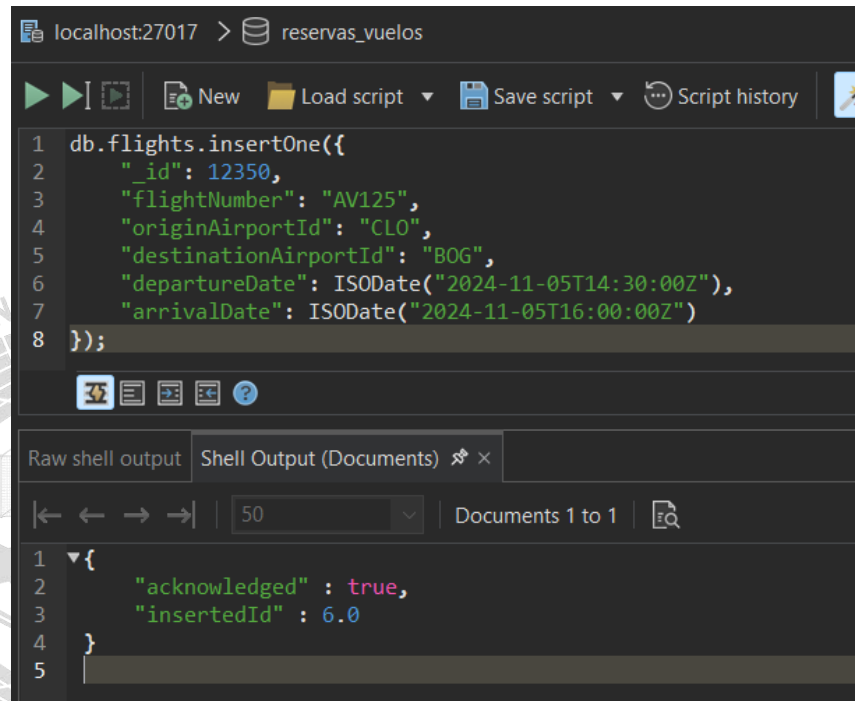
Consulta 2:

```
db.flights.insertOne({
  "_id": 12350,
  "flightNumber": "AV125",
  "originAirportId": "CLO",
  "destinationAirportId": "BOG",
  "departureDate": ISODate("2024-11-05T14:30:00Z"),
  "arrivalDate": ISODate("2024-11-05T16:00:00Z")
});
```

PROGRAMA INGENIERIA DE SISTEMAS

Página 19 | 55

Explicación: Se inserta un nuevo vuelo en la colección flights con el identificador `_id = 12350`. El vuelo tiene información sobre el número de vuelo, el aeropuerto de origen (CLO), el aeropuerto de destino (BOG), la fecha de salida y la de llegada.



```
localhost:27017 > use reservas_vuelos

1 db.flights.insertOne({
2   "_id": 12350,
3   "flightNumber": "AV125",
4   "originAirportId": "CLO",
5   "destinationAirportId": "BOG",
6   "departureDate": ISODate("2024-11-05T14:30:00Z"),
7   "arrivalDate": ISODate("2024-11-05T16:00:00Z")
8 });

Raw shell output Shell Output (Documents) x
1 {
2   "acknowledged" : true,
3   "insertedId" : 6.0
4 }
5
```

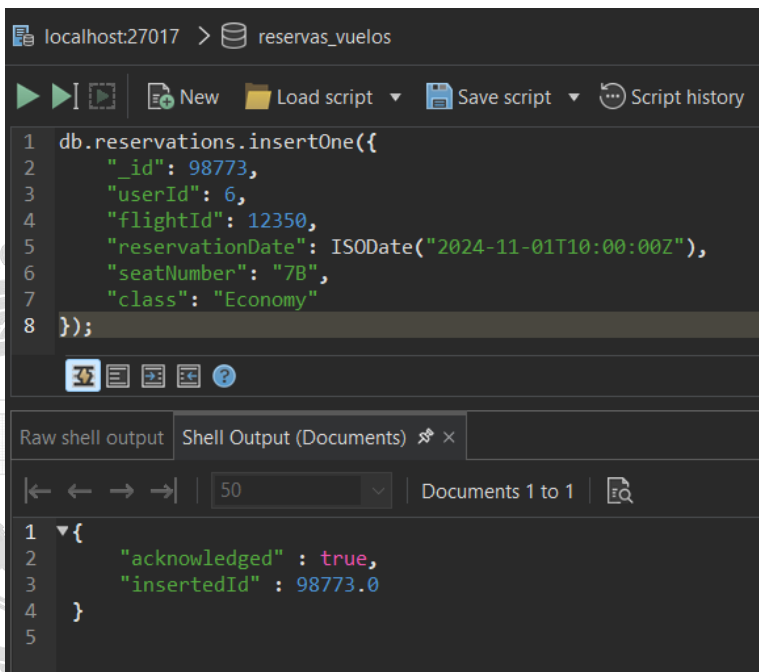
El Saber como Arma de Vida Consulta 3:

```
db.reservations.insertOne({
  "_id": 98773,
  "userId": 6,
  "flightId": 12350,
  "reservationDate": ISODate("2024-11-01T10:00:00Z"),
  "seatNumber": "7B",
  "class": "Economy"
});
```

PROGRAMA INGENIERIA DE SISTEMAS

P á g i n a 20 | 55

Explicación: Esta consulta inserta una nueva reserva en la colección reservations. Se registra una reserva con el usuario de `_id = 6` en el vuelo con `_id = 12350`, incluyendo información como la fecha de reserva, el número de asiento y la clase de vuelo.



```
localhost:27017 > reservas_vuelos

1 db.reservations.insertOne({
2   "_id": 98773,
3   "userId": 6,
4   "flightId": 12350,
5   "reservationDate": ISODate("2024-11-01T10:00:00Z"),
6   "seatNumber": "7B",
7   "class": "Economy"
8 });

Raw shell output  Shell Output (Documents) ✖ ×

1 {
2   "acknowledged" : true,
3   "insertedId" : 98773.0
4 }
5
```

El Saber como Arma de Vida

PROGRAMA INGENIERIA DE SISTEMAS

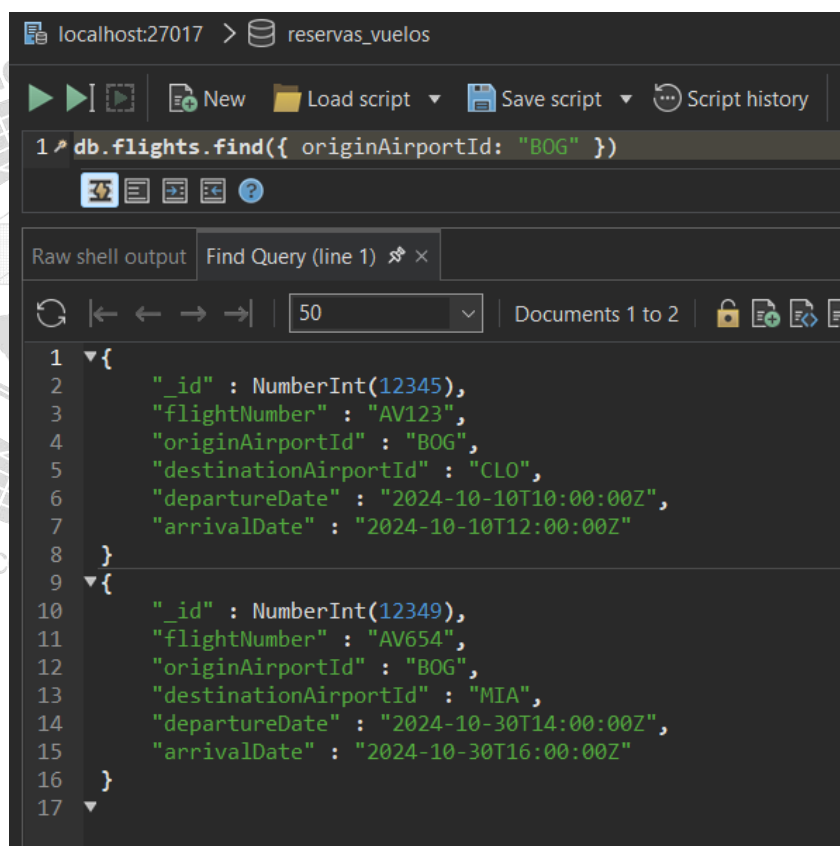
Página 21 | 55

READ

Consulta 4:

```
db.flights.find({ originAirportId: "BOG" });
```

Explicación: Esta consulta busca todos los vuelos que tienen como aeropuerto de origen "BOG" (Bogotá). Se devuelve una lista de documentos que cumplan con esta condición.



The screenshot shows a MongoDB shell interface with the following details:

- Database: `reservas_vuelos`
- Query: `1 db.flights.find({ originAirportId: "BOG" })`
- Output: A list of 2 documents (1 to 2) returned.
- Document 1:

```
{
  "_id" : NumberInt(12345),
  "flightNumber" : "AV123",
  "originAirportId" : "BOG",
  "destinationAirportId" : "CLO",
  "departureDate" : "2024-10-10T10:00:00Z",
  "arrivalDate" : "2024-10-10T12:00:00Z"
}
```
- Document 2:

```
{
  "_id" : NumberInt(12349),
  "flightNumber" : "AV654",
  "originAirportId" : "BOG",
  "destinationAirportId" : "MIA",
  "departureDate" : "2024-10-30T14:00:00Z",
  "arrivalDate" : "2024-10-30T16:00:00Z"
}
```

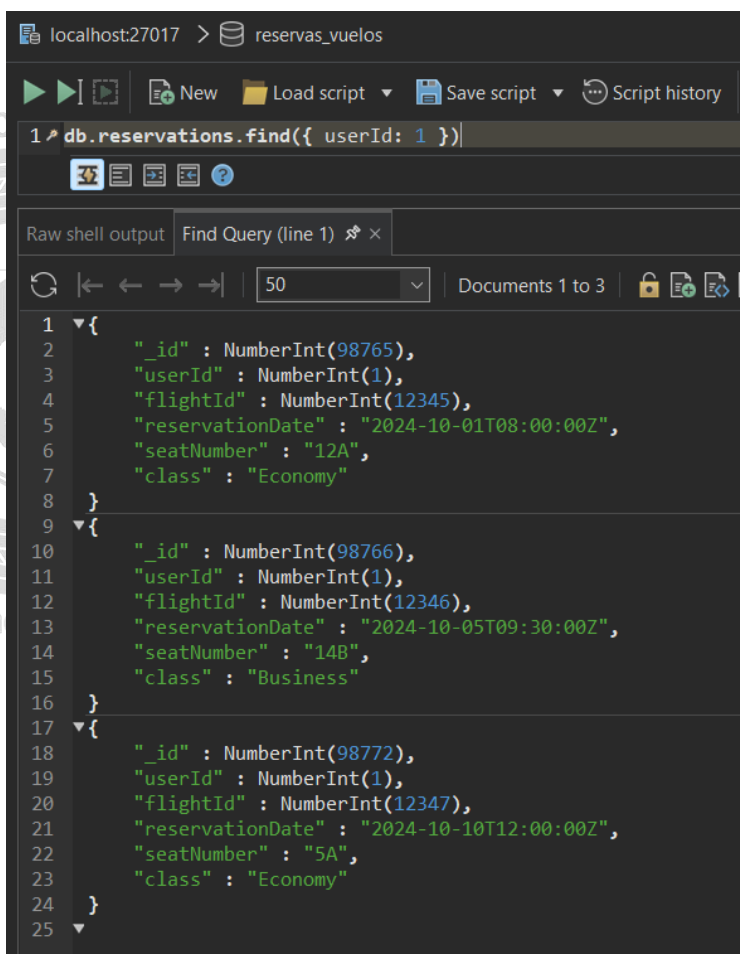
PROGRAMA INGENIERIA DE SISTEMAS

P á g i n a 22 | 55

Consulta 5:

```
db.reservations.find({ userId: 1 });
```

Explicación: Se buscan todas las reservas en la colección reservations que estén asociadas con el usuario con `userId = 1`. Esta consulta devuelve todas las reservas realizadas por ese usuario en particular.



```
localhost:27017 > use reservas_vuelos
1 db.reservations.find({ userId: 1 })

Raw shell output Find Query (line 1) x
50 Documents 1 to 3

1 {
2   "_id" : NumberInt(98765),
3   "userId" : NumberInt(1),
4   "flightId" : NumberInt(12345),
5   "reservationDate" : "2024-10-01T08:00:00Z",
6   "seatNumber" : "12A",
7   "class" : "Economy"
8 }
9 {
10  "_id" : NumberInt(98766),
11  "userId" : NumberInt(1),
12  "flightId" : NumberInt(12346),
13  "reservationDate" : "2024-10-05T09:30:00Z",
14  "seatNumber" : "14B",
15  "class" : "Business"
16 }
17 {
18  "_id" : NumberInt(98772),
19  "userId" : NumberInt(1),
20  "flightId" : NumberInt(12347),
21  "reservationDate" : "2024-10-10T12:00:00Z",
22  "seatNumber" : "5A",
23  "class" : "Economy"
24 }
25
```

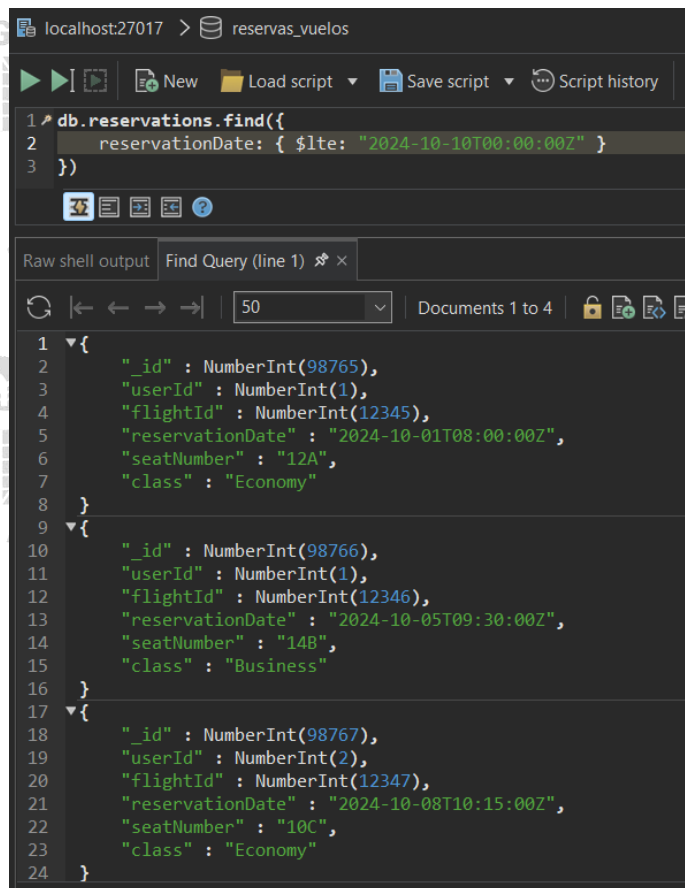
PROGRAMA INGENIERIA DE SISTEMAS

Página 23 | 55

Consultas 6:

```
db.reservations.find({
  reservationDate: { $lte: "2024-10-10T00:00:00Z" }
});
```

Explicación: Esta consulta recupera todas las reservas que fueron realizadas antes o hasta el 10 de octubre de 2024. El operador \$lte significa "menor o igual a", lo que permite filtrar por fechas anteriores a la especificada.



The screenshot shows a MongoDB shell interface with the following content:

```
localhost:27017 > reservas_vuelos
```

Buttons: New, Load script, Save script, Script history

```
1 db.reservations.find({
2   reservationDate: { $lte: "2024-10-10T00:00:00Z" }
3 })
```

Raw shell output: Find Query (line 1)

50 Documents 1 to 4

```
1 {
2   "_id" : NumberInt(98765),
3   "userId" : NumberInt(1),
4   "flightId" : NumberInt(12345),
5   "reservationDate" : "2024-10-01T08:00:00Z",
6   "seatNumber" : "12A",
7   "class" : "Economy"
8 }
9
10 {
11   "_id" : NumberInt(98766),
12   "userId" : NumberInt(1),
13   "flightId" : NumberInt(12346),
14   "reservationDate" : "2024-10-05T09:30:00Z",
15   "seatNumber" : "14B",
16   "class" : "Business"
17 }
18
19 {
20   "_id" : NumberInt(98767),
21   "userId" : NumberInt(2),
22   "flightId" : NumberInt(12347),
23   "reservationDate" : "2024-10-08T10:15:00Z",
24   "seatNumber" : "10C",
25   "class" : "Economy"
26 }
```

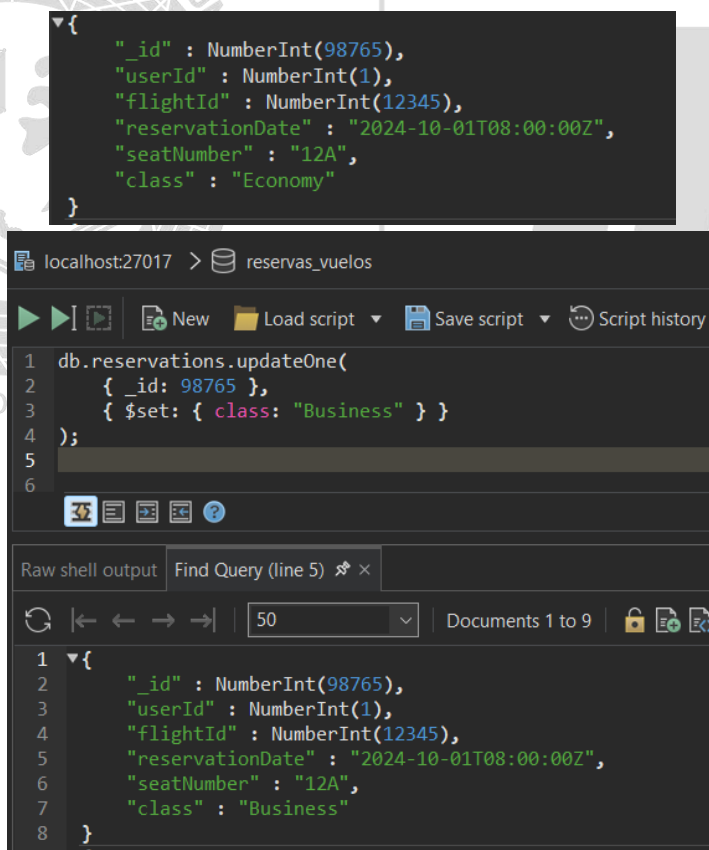
PROGRAMA INGENIERIA DE SISTEMAS

Página 24 | 55

UPDATE**Consulta 7:**

```
db.reservations.updateOne(  
  { _id: 98765 },  
  { $set: { class: "Business" } }  
);
```

Explicación: Esta consulta busca la reserva con `_id = 98765` y actualiza el campo `class`, cambiando su valor a `"Business"`. El uso de `$set` asegura que solo se actualice este campo sin modificar el resto del documento.



```
{  
  "_id" : NumberInt(98765),  
  "userId" : NumberInt(1),  
  "flightId" : NumberInt(12345),  
  "reservationDate" : "2024-10-01T08:00:00Z",  
  "seatNumber" : "12A",  
  "class" : "Economy"  
}
```

```
localhost:27017 > reservas_vuelos  
New Load script Save script Script history  
1 db.reservations.updateOne(  
2   { _id: 98765 },  
3   { $set: { class: "Business" } }  
4 );  
5  
6  
Raw shell output Find Query (line 5)  
50 Documents 1 to 9  
1 {  
2   "_id" : NumberInt(98765),  
3   "userId" : NumberInt(1),  
4   "flightId" : NumberInt(12345),  
5   "reservationDate" : "2024-10-01T08:00:00Z",  
6   "seatNumber" : "12A",  
7   "class" : "Business"  
8 }
```

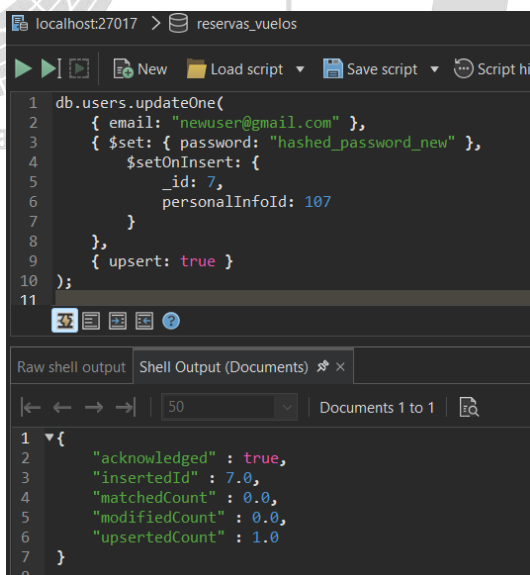
PROGRAMA INGENIERIA DE SISTEMAS

Página 25 | 55

Consulta 8:

```
db.users.updateOne(  
  { email: "newuser@gmail.com" },  
  {  
    $set: { password: "hashed_password_new" },  
    $setOnInsert: {  
      _id: 7,  
      personalInfoId: 107  
    }  
  },  
  { upsert: true }  
);
```

Explicación: En esta consulta, se intenta actualizar el documento que tiene el email "newuser@gmail.com". Si se encuentra, se actualiza el campo password. Si no existe, la opción upsert: true crea un nuevo documento con los valores especificados en \$setOnInsert, incluyendo _id = 7 y personalInfoId = 107.



```
localhost:27017 > use reservas_vuelos  
1 db.users.updateOne(  
2   { email: "newuser@gmail.com" },  
3   { $set: { password: "hashed_password_new" },  
4     $setOnInsert: {  
5       _id: 7,  
6       personalInfoId: 107  
7     }  
8   },  
9   { upsert: true }  
10 );  
11
```

Raw shell output | Shell Output (Documents) ✖

```
1 {  
2   "acknowledged" : true,  
3   "insertedId" : 7.0,  
4   "matchedCount" : 0.0,  
5   "modifiedCount" : 0.0,  
6   "upsertedCount" : 1.0  
7 }  
8
```

PROGRAMA INGENIERIA DE SISTEMAS

Página 26 | 55

Comprobación:

```
{
  "_id" : NumberInt(7),
  "email" : "newuser@gmail.com",
  "password" : "hashed_password_new",
  "personalInfoId" : NumberInt(107)
}
```

Consulta 9:

Explicación: Aquí se actualiza el documento de la colección personal_info con _id = 101. Se cambia el nombre propio a "Brayan Updated" usando \$set. Además, se agrega un nuevo número de teléfono al array phones usando \$push con el operador \$each, que permite agregar múltiples elementos a un array.

```
{
  "_id" : NumberInt(101),
  "fullName" : {
    "firstName" : "Brayan",
    "firstLastName" : "Arcos",
    "secondLastName" : "Burbano"
  },
  "address" : {
    "street" : "Calle 123",
    "city" : "Bogotá",
    "state" : "Cundinamarca",
    "zipCode" : "110111"
  },
  "phones" : [
    NumberInt(314567894),
    NumberInt(8909877)
  ],
  "profilePicture" : "https://www.bancodeimagenesgratis.com/imagen.jpg",
  "birthDate" : "1995-05-15T00:00:00Z",
  "nationality" : "Colombiana"
}
```


PROGRAMA INGENIERIA DE SISTEMAS

Página 27 | 55

Comprobación:

```
1 db.personal_info.updateOne(  
2   { _id: 101 },  
3   { $set: { "fullName.firstName": "Brayan Updated" },  
4     $push: { phones: { $each: [3112345678] } }  
5   }  
6 )  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23
```

Raw shell output Find Query (line 8) ✖

50 Documents 1 to 1

```
2   "id" : NumberInt(101),  
3   "fullName" : {  
4     "firstName" : "Brayan Updated",  
5     "firstLastName" : "Arcos",  
6     "secondLastName" : "Burbano"  
7   },  
8   "address" : {  
9     "street" : "Calle 123",  
10    "city" : "Bogotá",  
11    "state" : "Cundinamarca",  
12    "zipCode" : "110111"  
13  },  
14  "phones" : [  
15    NumberInt(314567894),  
16    NumberInt(8909877),  
17    3112345678.0  
18  ],  
19  "profilePicture" : "https://www.bancodeimagenesgratis.com/imagen.jpg",  
20  "birthDate" : "1995-05-15T00:00:00Z",  
21  "nationality" : "Colombiana"  
22 }  
23
```

El Saber como Arma de Vida

PROGRAMA INGENIERIA DE SISTEMAS

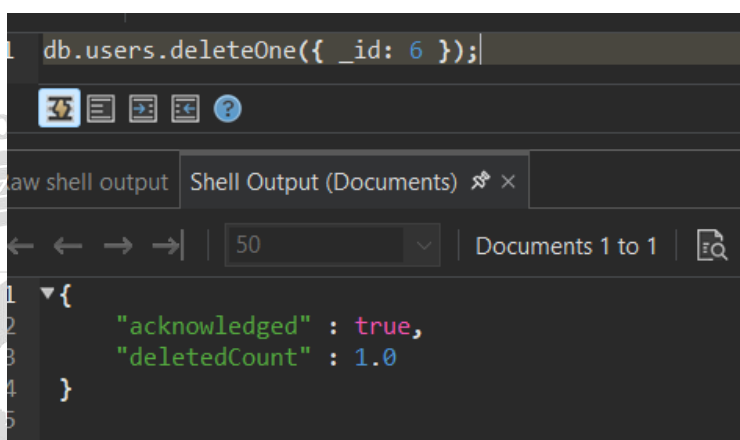
Página 28 | 55

DELETE

Consulta 10

```
db.users.deleteOne({ _id: 6 })
```

Explicación: Se elimina el documento con `_id = 6` de la colección `users`. La consulta borra un solo documento que cumpla con esta condición.



```
1 db.users.deleteOne({ _id: 6 });
```

Shell Output (Documents)

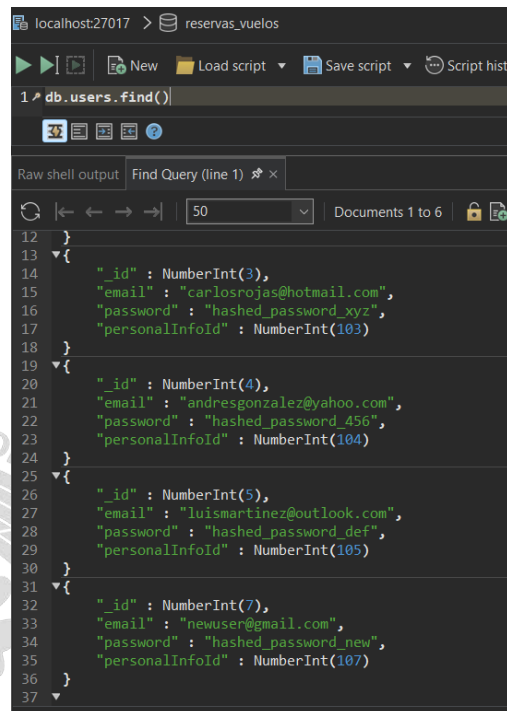
```
1 {
2   "acknowledged" : true,
3   "deletedCount" : 1.0
4 }
5
```

El Saber como Arma de Vida

PROGRAMA INGENIERIA DE SISTEMAS

Página 29 | 55

Comprobación:



```
localhost:27017 > use reservas_vuelos
1 db.users.find()

Raw shell output Find Query (line 1) x
Documents 1 to 6
12 }
13 {
14   "_id" : NumberInt(3),
15   "email" : "carlosrojas@hotmail.com",
16   "password" : "hashed_password_xyz",
17   "personalInfoId" : NumberInt(103)
18 }
19 {
20   "_id" : NumberInt(4),
21   "email" : "andresgonzalez@yahoo.com",
22   "password" : "hashed_password_456",
23   "personalInfoId" : NumberInt(104)
24 }
25 {
26   "_id" : NumberInt(5),
27   "email" : "luismartinez@outlook.com",
28   "password" : "hashed_password_def",
29   "personalInfoId" : NumberInt(105)
30 }
31 {
32   "_id" : NumberInt(7),
33   "email" : "newuser@gmail.com",
34   "password" : "hashed_password_new",
35   "personalInfoId" : NumberInt(107)
36 }
37 }
```

Consultas 11:

```
db.reservations.deleteMany({ userId: 1 })
```

El Saber como Arma de Vida

Explicación: Esta consulta elimina todas las reservas que pertenecen al usuario con `userId = 1` en la colección `reservations`. El uso de `deleteMany` asegura que todos los documentos coincidentes sean eliminados.

PROGRAMA INGENIERIA DE SISTEMAS

Página 30 | 55

```
localhost:27017 > reservas_vuelos

1 db.reservations.find({ userId: 1 })

Raw shell output Find Query (line 1) ✕ ×

1 { "_id" : NumberInt(98765),
2   "userId" : NumberInt(1),
3   "flightId" : NumberInt(12345),
4   "reservationDate" : "2024-10-01T08:00:00Z",
5   "seatNumber" : "12A",
6   "class" : "Business"
7 }
8
9 { "_id" : NumberInt(98766),
10  "userId" : NumberInt(1),
11  "flightId" : NumberInt(12346),
12  "reservationDate" : "2024-10-05T09:30:00Z",
13  "seatNumber" : "14B",
14  "class" : "Business"
15 }
16
17 { "_id" : NumberInt(98772),
18  "userId" : NumberInt(1),
19  "flightId" : NumberInt(12347),
20  "reservationDate" : "2024-10-10T12:00:00Z",
21  "seatNumber" : "5A",
22  "class" : "Economy"
23 }
24
25
```

Respuesta:

```
1 db.reservations.deleteMany({ userId: 1 });

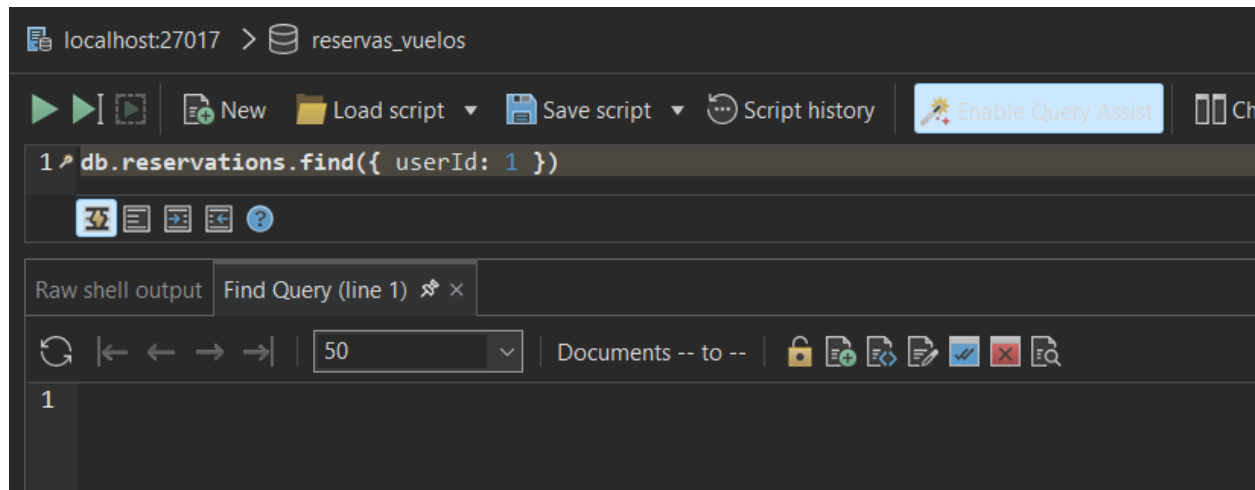
Raw shell output Shell Output (Documents) ✕ ×

1 { "acknowledged" : true,
2   "deletedCount" : 3.0
3 }
4
5
```

PROGRAMA INGENIERIA DE SISTEMAS

Página 31 | 55

Comprobación:



The screenshot shows a MongoDB shell interface. The top bar indicates the connection is to 'localhost:27017' and the current database is 'reservas_vuelos'. The command prompt shows a query: `db.reservations.find({ userId: 1 })`. Below the command, the 'Raw shell output' tab is active, displaying the results of the query. The output is a single document, represented by a large, empty rectangular box, indicating that the query returned one result.



PROGRAMA INGENIERIA DE SISTEMAS

Página 32 | 55

Taller Practico Mongo CRUD:

TRABAJA CON READ**1. db.grades.find({ student_id: 2 })**

Busca todos los documentos donde el student_id sea igual a 2.

grades > student_id			
_id	student_id	class_id	scores
50b59cd75bed7...	2	25	[5 elements]
50b59cd75bed7...	2	27	[4 elements]
50b59cd75bed7...	2	24	[4 elements]

2. db.grades.find({ "scores.0.score": { \$lte: 10 } })

Busca documentos donde el primer elemento en el array scores (índice 0) tenga un campo score menor o igual a 10.

grades > student_id			
_id	student_id	class_id	scores
50b59cd75bed7...	0	24	[4 elements]
50b59cd75bed7...	2	25	[5 elements]
50b59cd75bed7...	3	13	[6 elements]
50b59cd75bed7...	3	16	[3 elements]
50b59cd75bed7...	4	5	[5 elements]
50b59cd75bed7...	4	12	[4 elements]
50b59cd75bed7...	6	22	[4 elements]
50b59cd75bed7...	8	29	[3 elements]
50b59cd75bed7...	11	25	[5 elements]
50b59cd75bed7...	12	13	[6 elements]
50b59cd75bed7...	15	6	[6 elements]
50b59cd75bed7...	20	12	[4 elements]
50b59cd75bed7...	23	27	[6 elements]
50b59cd75bed7...	23	24	[6 elements]

3. db.grades.find({ "scores.4.score": { \$lte: 10 } })

PROGRAMA INGENIERIA DE SISTEMAS

Página 33 | 55

Busca documentos donde el quinto elemento en el array scores (índice 4) tenga un campo score menor o igual a 10.

_id	student_id	class_id	scores
50b59cd75bed7...	0	5	[6 elements]
50b59cd75bed7...	13	22	[6 elements]
50b59cd75bed7...	15	21	[5 elements]
50b59cd75bed7...	15	22	[5 elements]
50b59cd75bed7...	17	19	[5 elements]
50b59cd75bed7...	19	22	[5 elements]
50b59cd75bed7...	24	22	[6 elements]
50b59cd75bed7...	30	0	[6 elements]
50b59cd75bed7...	41	0	[5 elements]
50b59cd75bed7...	42	18	[6 elements]

4. `db.grades.find({ "scores.5.score": { $lte: 10 } })`

Busca documentos donde el sexto elemento en el array scores (índice 5) tenga un campo score menor o igual a 10.

_id	student_id	class_id	scores
50b59cd75bed7...	6	8	[6 elements]
50b59cd75bed7...	12	4	[6 elements]
50b59cd75bed7...	35	18	[6 elements]
50b59cd75bed7...	36	23	[6 elements]
50b59cd75bed7...	41	18	[6 elements]

5. `db.grades.find({ "scores.6.score": { $lte: 10 } })`

Busca documentos donde el séptimo elemento en el array scores (índice 6) tenga un campo score menor o igual a 10.

no devuelve ningún resultado es porque "scores.6.score" está intentando acceder al séptimo elemento de un arreglo (indexado desde 0), pero muchos de los documentos en tu colección no tienen tantos elementos en el arreglo scores.

0 documents selected

PROGRAMA INGENIERIA DE SISTEMAS

Página 34 | 55

6. `db.grades.find({ "scores.0.score": { $gte: 60, $lte: 61 } })`

Busca documentos donde el primer elemento en el array scores (índice 0) tenga un score entre 60 y 61 inclusive.

grades > student_id			
_id	student_id	class_id	scores
50b59cd75bed7...	0	27	[3 elements]
50b59cd75bed7...	4	8	[5 elements]
50b59cd75bed7...	25	0	[4 elements]
50b59cd75bed7...	30	29	[4 elements]

7. `db.grades.find({ "scores.0.score": { $gte: 60, $lte: 61 } }).sort({ student_id: 1 })`

Busca los mismos documentos que en la consulta 6, pero los ordena en función del **student_id** en orden ascendente.

grades > student_id			
_id	student_id	class_id	scores
50b59cd75bed7...	0	27	[3 elements]
50b59cd75bed7...	4	8	[5 elements]
50b59cd75bed7...	25	0	[4 elements]
50b59cd75bed7...	30	29	[4 elements]

8. `db.grades.find({ student_id: 2, class_id: 24 })`

Busca documentos donde el **student_id** sea 2 y el **class_id** sea 24.

grades > scores			
_id	student_id	class_id	scores
50b59cd75bed7...	2	24	[4 elements]

9. `db.grades.find({ class_id: 20, $and: [{ "scores.0.score": { $gte: 15 } }, { "scores.0.score": { $lte: 30 } }] })`

Busca documentos donde el **class_id** sea 20, El primer elemento en el array scores (índice 0) tenga un score entre 15 y 30 inclusive (usando \$and para combinar ambas condiciones).

grades > _id			
_id	student_id	class_id	scores
50b59cd75bed7...	46	20	[5 elements]

PROGRAMA INGENIERIA DE SISTEMAS

Página 35 | 55

Obtenemos Todos los registros donde la clase sea 20 y la primera calificación en el array scores esté entre 15 y 30.

10. `db.grades.find({ scores: { $elemMatch: { type: 'quiz', score: { $gte: 50 } } } })`

Usa `$elemMatch` para buscar documentos donde al menos un elemento en el array scores sea de tipo quiz y tenga un score mayor o igual a 50.

grades > student_id			
_id	student_id	class_id	scores
50b59cd75bed7...	0	28	[6 elements]
50b59cd75bed7...	0	5	[6 elements]
50b59cd75bed7...	0	7	[5 elements]
50b59cd75bed7...	0	27	[3 elements]
50b59cd75bed7...	0	10	[4 elements]
50b59cd75bed7...	1	18	[3 elements]
50b59cd75bed7...	1	28	[5 elements]
50b59cd75bed7...	1	13	[4 elements]
50b59cd75bed7...	2	27	[4 elements]
50b59cd75bed7...	3	10	[4 elements]
50b59cd75bed7...	3	9	[6 elements]
50b59cd75bed7...	3	12	[4 elements]

11. `db.grades.find({ scores: { $elemMatch: { type: 'exam', score: { $gte: 50 } } } })`

Usa `$elemMatch` para buscar documentos donde al menos un elemento en el array scores sea de tipo exam y tenga un score mayor o igual a 50

grades > student_id			
_id	student_id	class_id	scores
50b59cd75bed7...	0	2	[5 elements]
50b59cd75bed7...	0	5	[6 elements]
50b59cd75bed7...	0	16	[5 elements]
50b59cd75bed7...	0	6	[6 elements]
50b59cd75bed7...	0	27	[3 elements]
50b59cd75bed7...	0	11	[4 elements]
50b59cd75bed7...	1	18	[3 elements]
50b59cd75bed7...	1	16	[5 elements]
50b59cd75bed7...	1	13	[4 elements]

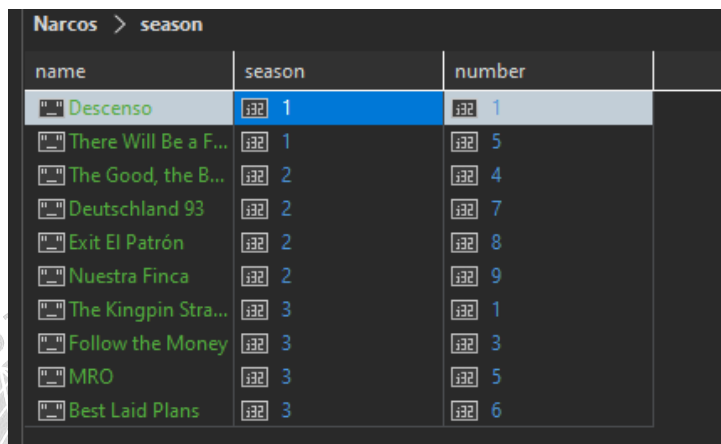
PROGRAMA INGENIERIA DE SISTEMAS

Página 36 | 55

Punto 3

1. `db.Narcos.find({ runtime: { $gte: 55 } }, { _id:0, name:1, season:1, number:1 })`

Una lista de episodios con sus nombres, temporada y número si tienen una duración de 55 minutos o más.



Narcos > season		
name	season	number
Descenso	1	1
There Will Be a F...	1	5
The Good, the B...	2	4
Deutschland 93	2	7
Exit El Patrón	2	8
Nuestra Finca	2	9
The Kingpin Stra...	3	1
Follow the Money	3	3
MRO	3	5
Best Laid Plans	3	6

2. `db.Narcos.find({ runtime: { $gte: 15 } }, { _id:0, season:1, number:1 }).sort({ season:1, number:-1 })`

PROGRAMA INGENIERIA DE SISTEMAS

Página 37 | 55

Lista de episodios que duren más de 15 minutos, ordenados por temporada y en cada temporada del último episodio al primero.

season	number
1	10
1	9
1	8
1	7
1	6
1	5
1	4
1	3
1	2
1	1
2	10
2	9
2	8
2	7
2	6
2	5
2	4
2	3
2	2
2	1

3	10
3	9
3	8
3	7
3	6
3	5
3	4
3	3
3	2
3	1

3. `db.Narcos.find({ season: { $type: 'number' } })`

Todos los documentos (episodios) donde el campo season sea un número. Se excluyen episodios si el campo es de otro tipo (por ejemplo, string o no existe).

_id	id	url	name	season	number	type	airdate	airtime	airstamp	runtime	re
6707570b06a742...	203469	https://www.tv...	Descenso	1	1	regular	2015-08-28		2015-08-28T12...	57	CC
6707570b06a742...	208978	https://www.tv...	The Sword of S...	1	2	regular	2015-08-28		2015-08-28T12...	47	CC
6707570b06a742...	208979	https://www.tv...	The Men of AL...	1	3	regular	2015-08-28		2015-08-28T12...	47	CC
6707570b06a742...	208980	https://www.tv...	The Palace in F...	1	4	regular	2015-08-28		2015-08-28T12...	44	CC
6707570b06a742...	208981	https://www.tv...	There Will Be a ...	1	5	regular	2015-08-28		2015-08-28T12...	55	CC
6707570b06a742...	208982	https://www.tv...	Explosivos	1	6	regular	2015-08-28		2015-08-28T12...	50	CC
6707570b06a742...	208983	https://www.tv...	You Will Cry Te...	1	7	regular	2015-08-28		2015-08-28T12...	51	CC
6707570b06a742...	208984	https://www.tv...	La Gran Mentira	1	8	regular	2015-08-28		2015-08-28T12...	51	CC
6707570b06a742...	208985	https://www.tv...	La Catedral	1	9	regular	2015-08-28		2015-08-28T12...	51	CC
6707570b06a742...	208986	https://www.tv...	Despegue	1	10	regular	2015-08-28		2015-08-28T12...	45	CC
6707570b06a742...	832098	https://www.tv...	Free at Last	2	1	regular	2016-09-02		2016-09-02T12...	53	CC
6707570b06a742...	832099	https://www.tv...	Cambalache	2	2	regular	2016-09-02		2016-09-02T12...	47	CC
6707570b06a742...	832100	https://www.tv...	Our Man in Ma...	2	3	regular	2016-09-02		2016-09-02T12...	47	CC
6707570b06a742...	832101	https://www.tv...	The Good, the ...	2	4	regular	2016-09-02		2016-09-02T12...	56	CC
6707570b06a742...	832102	https://www.tv...	The Enemies of...	2	5	regular	2016-09-02		2016-09-02T12...	52	CC
6707570b06a742...	832103	https://www.tv...	Los Pepes	2	6	regular	2016-09-02		2016-09-02T12...	54	CC
6707570b06a742...	832256	https://www.tv...	Deutschland 93	2	7	regular	2016-09-02		2016-09-02T12...	57	CC
6707570b06a742...	832257	https://www.tv...	Exit El Patron	2	8	regular	2016-09-02		2016-09-02T12...	55	CC
6707570b06a742...	832258	https://www.tv...	Nuestra Finca	2	9	regular	2016-09-02		2016-09-02T12...	57	CC
6707570b06a742...	832259	https://www.tv...	Al Fin Cayó!	2	10	regular	2016-09-02		2016-09-02T12...	53	CC
6707570b06a742...	1249969	https://www.tv...	The Kingpin Str...	3	1	regular	2017-09-01		2017-09-01T12...	55	CC
6707570b06a742...	1285381	https://www.tv...	The Cali KGB	3	2	regular	2017-09-01		2017-09-01T12...	49	CC
6707570b06a742...	1285382	https://www.tv...	Follow the Mo...	3	3	regular	2017-09-01		2017-09-01T12...	59	CC

PROGRAMA INGENIERIA DE SISTEMAS

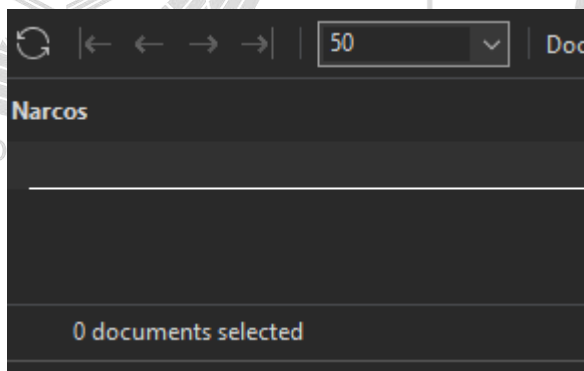
Página 38 | 55

4. `db.Narcos.find({ rating: { $exists: 1 } })`

Busca los episodios que tengan el campo **rating** (calificación) presente en el documento, sin importar su valor o tipo.

Narcos > id											
id	id	url	name	season	number	type	airdate	airtime	airstamp	runtime	tz
[id] 6707570b06a742...	203469	https://www.tvmaze.com/episodes/203469/narcos-1x01-descenso	The Sword of S...	1	2	regular	2015-08-28		2015-08-28T12:...	47	tz
[id] 6707570b06a742...	208978	https://www.tv...	The Men of Al...	1	3	regular	2015-08-28		2015-08-28T12:...	47	tz
[id] 6707570b06a742...	208980	https://www.tv...	The Palace in F...	1	4	regular	2015-08-28		2015-08-28T12:...	44	tz
[id] 6707570b06a742...	208981	https://www.tv...	There Will Be a ...	1	5	regular	2015-08-28		2015-08-28T12:...	55	tz
[id] 6707570b06a742...	208982	https://www.tv...	Explosivos	1	6	regular	2015-08-28		2015-08-28T12:...	50	tz
[id] 6707570b06a742...	208983	https://www.tv...	You Will Cry Te...	1	7	regular	2015-08-28		2015-08-28T12:...	51	tz
[id] 6707570b06a742...	208984	https://www.tv...	La Gran Mentira	1	8	regular	2015-08-28		2015-08-28T12:...	51	tz
[id] 6707570b06a742...	208985	https://www.tv...	La Catedral	1	9	regular	2015-08-28		2015-08-28T12:...	51	tz
[id] 6707570b06a742...	208986	https://www.tv...	Despegue	1	10	regular	2015-08-28		2015-08-28T12:...	45	tz
[id] 6707570b06a742...	832098	https://www.tv...	Free at Last	2	1	regular	2016-09-02		2016-09-02T12:...	53	tz
[id] 6707570b06a742...	832099	https://www.tv...	Cambalache	2	2	regular	2016-09-02		2016-09-02T12:...	47	tz
[id] 6707570b06a742...	832100	https://www.tv...	Our Man in Ma...	2	3	regular	2016-09-02		2016-09-02T12:...	47	tz
[id] 6707570b06a742...	832101	https://www.tv...	The Good, the ...	2	4	regular	2016-09-02		2016-09-02T12:...	46	tz
[id] 6707570b06a742...	832102	https://www.tv...	The Enemies of...	2	5	regular	2016-09-02		2016-09-02T12:...	52	tz
[id] 6707570b06a742...	832103	https://www.tv...	Los Pepes	2	6	regular	2016-09-02		2016-09-02T12:...	54	tz
[id] 6707570b06a742...	832256	https://www.tv...	Deutschland 93	2	7	regular	2016-09-02		2016-09-02T12:...	57	tz
[id] 6707570b06a742...	832257	https://www.tv...	Exit El Patrón	2	8	regular	2016-09-02		2016-09-02T12:...	55	tz
[id] 6707570b06a742...	832258	https://www.tv...	Nuestra Finca	2	9	regular	2016-09-02		2016-09-02T12:...	57	tz
[id] 6707570b06a742...	832259	https://www.tv...	Al Fin Cayó!	2	10	regular	2016-09-02		2016-09-02T12:...	53	tz
[id] 6707570b06a742...	1249969	https://www.tv...	The Kingpin Str...	3	1	regular	2017-09-01		2017-09-01T12:...	55	tz
[id] 6707570b06a742...	1285381	https://www.tv...	The Cali KGB	3	2	regular	2017-09-01		2017-09-01T12:...	49	tz
[id] 6707570b06a742...	1285382	https://www.tv...	Follow the Mo...	3	3	regular	2017-09-01		2017-09-01T12:...	59	tz

5. `db.Narcos.find({ rating: { $exists: 1 }, rating: { $type: "string" } })`



la consulta no arrojaría ningún resultado, ya que el campo **rating** no es del tipo **string**, sino un objeto.

PROGRAMA INGENIERIA DE SISTEMAS

Página 39 | 55

Punto 4

1. `db.grades.distinct("student_id")`

devuelve una lista única de todos los valores diferentes de `student_id` en la colección `grades`. Esto es útil si deseas saber cuántos estudiantes diferentes están presentes en la base de datos o si quieres obtener una lista sin repeticiones de los estudiantes.



Array	
[Index]	
0	123 0.0
1	123 1.0
2	123 2.0
3	123 3.0
4	123 4.0
5	123 5.0
6	123 6.0
7	123 7.0
8	123 8.0
9	123 9.0
10	123 10.0
11	123 11.0
12	123 12.0
13	123 13.0
14	123 14.0
15	123 15.0



2. `db.grades.countDocuments()`

PROGRAMA INGENIERIA DE SISTEMAS

P á g i n a 40 | 55

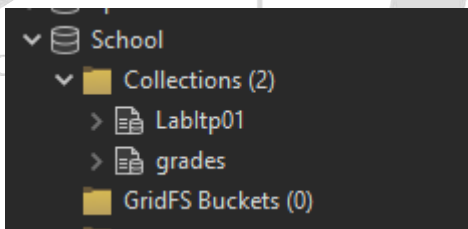
Esta consulta devuelve el número total de documentos en la colección grades. Es útil para saber cuántos registros o entradas hay en la base de datos

```
56
57 The find query will be run with Query Assist.
58
59 [
60   0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
61   11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
62   22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
63   33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
64   44, 45, 46, 47, 48, 49
65 ]
66 280
67 280
68
```

TRABAJA CON CREATE

1. En la BD **School** crea la colección **LabItp**:

- **db.createCollection("LabItp01")**



- **db.LabItp01.insert({ _id:1, name: "pepe", phone: 123456, class: [20, 22, 25] })**

PROGRAMA INGENIERIA DE SISTEMAS

Página 41 | 55

LabItp01 > name			
_id	name	phone	class
2	juanito	654789	[3 elements]
3	carlito	639852	[2 elements]
5	anita	852741	[1 elements]

- `db.LabItp01.insertOne({_id:2, name: "juanito", phone: 654789, class: [10, 12, 15] })`

Inserta un documento similar, pero con `_id: 2` para el estudiante juanito con su número de teléfono y clases asociadas.

LabItp01 > name			
_id	name	phone	class
2	juanito	654789	[3 elements]
3	carlito	639852	[2 elements]
5	anita	852741	[1 elements]

- `db.LabItp01.insertMany([{ _id:3, name: "carlito", phone: 639852, class: [11, 10] }, { _id:4, name: "camilito", phone: 741258, class: [15] }, { _id:5, name: "anita", phone: 852741, class: [10] }, { _id:5, name: "joselito", phone: 1254896, class: [55, 458, 236, 20, 22, 10, 15] }])`

El Inserta varios documentos en la colección **LabItp01**. Uno de estos documentos tiene un problema: hay dos documentos con el mismo `_id: 5`, lo que causará un error de duplicación de clave al intentar insertar el segundo (de joselito).

- `db.LabItp01.find({ class: 10 })`

PROGRAMA INGENIERIA DE SISTEMAS

Página 42 | 55

Esta consulta devuelve todos los documentos donde el campo class contiene el valor 10

LabItp01 > name			
_id	name	phone	class
2	juanito	654789	[3 elements]
3	carlito	639852	[2 elements]
5	anita	852741	[1 elements]

- **db.LabItp02.insertOne({ name: "carolita" })**

Inserta un documento en la colección LabItp02 con solo el campo name: "carolita".

Result > insertedId	
acknowledged	insertedId
true	67076e1fb7fda...

- **db.LabItp02.insertOne({ name: "carolita", information: { classroom: "room_01", locker: 12 }, age: 25 })**

Inserta un segundo documento en la colección LabItp02, pero esta vez con información adicional: information es un objeto que contiene classroom y locker, además del campo age.

acknowledged	insertedId
true	67076e55b7fda...

- **db.LabItp02.find()**

Esta consulta devolverá todos los documentos insertados en la colección LabItp02. En este caso, devolverá los dos documentos relacionados con carolita que insertaste previamente.

PROGRAMA INGENIERIA DE SISTEMAS

Página 43 | 55

LabItp02 > name

_id	name	information	age
<code>id 67076d8fb7fda5...</code>	<code>" carolita"</code>		
<code>id 67076e1fb7fda5...</code>	<code>" carolita"</code>		
<code>id 67076e55b7fda5...</code>	<code>" carolita"</code>	<code>{ 2 fields }</code>	<code>25</code>

TRABAJA CON UPDATE

En la colección LabItp01 realiza las siguientes actualizaciones

- `db.LabItp01.updateOne({ _id: 7 }, { $set: { virtues: ['cheerful', 'funny', 'comprehensive', 'sociable', 'respectful'] } })`

Esta consulta busca el documento con `_id: 7` en LabItp01. Si lo encuentra, agrega (o actualiza) el campo `virtues` con un arreglo que contiene los valores

Result > insertedId

acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
<code>T/F true</code>	<code>null</code>	<code>123 0.0</code>	<code>123 0.0</code>	<code>123 0.0</code>

Si el documento con `_id: 7` no existe, no se realiza ninguna acción. añadimos el id 7

```
db.LabItp01.updateOne(
  { _id: 7 },
  {
    $set: { name: "jhojan", age: 20, virtues: [], information: {} },
    $currentDate: { lastModified: true }
  },
  { upsert: true }
)
```

Result > insertedId

acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
<code>T/F true</code>	<code>null</code>	<code>123 1.0</code>	<code>123 1.0</code>	<code>123 0.0</code>

PROGRAMA INGENIERIA DE SISTEMAS

Página 44 | 55

- **db.LabItp01.updateOne({ _id: 7 }, { \$set: { information: { classroom: "room_A", locker: 15 }, age: 18 } })**

Result > insertedId				
acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
<input checked="" type="checkbox"/> true	<input type="checkbox"/> null	123 1.0	123 1.0	123 0.0

- **db.LabItp01.updateOne({ _id: 7 }, { \$set: { virtues: ['cheerful', 'funny', 'comprehensive', 'sociable', 'respectful'] }, \$currentDate: { lastModified: true } })**

Result > insertedId				
acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
<input checked="" type="checkbox"/> true	<input type="checkbox"/> null	123 1.0	123 1.0	123 0.0

- **db.LabItp01.updateOne({ _id: 7 }, { \$set: { information: { classroom: "room_A", locker: 15 }, age: 18 }, \$currentDate: { lastModified: true } })**

Result > insertedId				
acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
<input checked="" type="checkbox"/> true	<input type="checkbox"/> null	123 1.0	123 1.0	123 0.0

- **db.LabItp01.updateOne({ _id: 10 }, { \$set: { name: "Joan", age: 19, virtues: [], information: {} }, \$currentDate: { lastModified: true } }, { upsert: true })**

En esta consulta, si no existe un documento con `_id: 10`, lo crea (gracias a la opción `upsert: true`). Si existe, actualiza los campos `name`, `age`, `virtues` (un arreglo vacío), e `información` (un objeto vacío). Además, agrega o actualiza el campo `lastModified` con la fecha actual.

PROGRAMA INGENIERIA DE SISTEMAS

Página 45 | 55

Result > insertedId				
acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
T/F true	123 10.0	123 0.0	123 0.0	123 1.0

- Actualiza los documentos con `_id` 1 – 6 y agrega el campo `virtues` con un array que contenga un único valor, el que decidas de la lista siguiente: ['cheerful', 'funny', 'comprehensive', 'sociable', 'respectful'].

```
db.LabItp01.updateMany(
  { _id: { $gte: 1, $lte: 6 } },
  { $set: { virtues: ['sociable'] } }
)
```

Esta consulta busca todos los documentos cuyo `_id` está entre 1 y 6 (inclusive), y les agrega (o actualiza) el campo `virtues` con un valor que puedes elegir de la lista ['cheerful', 'funny', 'comprehensive', 'sociable', 'respectful']. Si el campo `virtues` ya existe en alguno de esos documentos, será sobrescrito.

Result > insertedId				
acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
T/F true	123 null	123 5.0	123 5.0	123 0.0

- Actualiza todos los documentos con una única instrucción y agrega el siguiente campo: `status: 'A'`.

El Saber como Arma de

```
db.LabItp01.updateMany(
  {},
  { $set: { status: 'A' } }
)
```

Esta consulta actualiza todos los documentos de la colección `LabItp01` y les agrega el campo `status` con el valor 'A'.

Result > insertedId				
acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
T/F true	123 null	123 6.0	123 6.0	123 0.0

- Actualiza los documentos de “pepe” y “camilito” y agrega el siguiente campo: role: 'student'.

```
db.LabItp01.updateMany(
  { name: { $in: ['pepe', 'camilito'] } },
  { $set: { role: 'student' } }
)
```

Esta consulta actualiza los documentos donde el campo name es igual a 'pepe' o 'camilito', y les agrega el campo role con el valor 'student'.

Result > insertedId				
acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
<input checked="" type="checkbox"/> true	<input type="checkbox"/> null	123 2.0	123 2.0	123 0.0



El Saber como Arma de Vida TRABAJA CON DELETE

En la colección LabItp01 realiza las siguientes actualizaciones:

1. `db.LabItp01.deleteOne({ name: "carlito" })`

Result > deletedCount	
acknowledged	deletedCount
<input checked="" type="checkbox"/> true	123 1.0

LabItp01 > name		
_id	name	ph
123 1	pepe	132
123 2	juanito	132
123 4	camilito	132
123 5	anita	132
123 10	Joan	132
123 7	jhojan	132

PROGRAMA INGENIERIA DE SISTEMAS

Página 47 | 55

Esta consulta elimina **un solo** documento en la colección **LabItp01** donde el campo **name** es "carlito". Si hay varios documentos que coinciden, solo se eliminará el **primer** documento que MongoDB encuentre

2. db.grades.deleteOne({ student_id: 0 })

Result > deletedCount	
acknowledged	deletedCount
<input checked="" type="checkbox"/> true	123 1.0

_id	student_id	class_id	scores
50b59cd75bed7...	0	28	[6 elements]
50b59cd75bed7...	0	5	[6 elements]
50b59cd75bed7...	0	16	[5 elements]
50b59cd75bed7...	0	24	[4 elements]
50b59cd75bed7...	0	30	[5 elements]
50b59cd75bed7...	0	7	[5 elements]
50b59cd75bed7...	0	6	[6 elements]
50b59cd75bed7...	0	27	[3 elements]
50b59cd75bed7...	0	11	[4 elements]
50b59cd75bed7...	0	10	[4 elements]

Elimina un solo documento de la colección grades donde student_id es igual a 0. Si hay varios documentos con student_id: 0, solo se eliminará el primero que MongoDB encuentre.

3. db.grades.deleteMany({ student_id: 0 })

Result > deletedCount	
acknowledged	deletedCount
<input checked="" type="checkbox"/> true	123 10.0

grades > student_id			
_id	student_id	class_id	scores
50b59cd75bed7...	1	18	[3 elements]
50b59cd75bed7...	1	22	[5 elements]
50b59cd75bed7...	1	28	[5 elements]
50b59cd75bed7...	1	16	[5 elements]
50b59cd75bed7...	1	13	[4 elements]
50b59cd75bed7...	2	25	[5 elements]
50b59cd75bed7...	2	27	[4 elements]

Esta consulta elimina todos los documentos en la colección grades donde student_id sea igual a 0.

4. db.grades.remove({ student_id: 1 }, {justOne: true})

Elementos actuales

_id	student_id	class_id	scores
[id] 50b59cd75bed76f46522c359		[id] 18	[] [3 elements]
[id] 50b59cd75bed7...	[id] 1	[id] 22	[] [5 elements]
[id] 50b59cd75bed7...	[id] 1	[id] 28	[] [5 elements]
[id] 50b59cd75bed7...	[id] 1	[id] 16	[] [5 elements]
[id] 50b59cd75bed7...	[id] 1	[id] 13	[] [4 elements]

Result > deletedCount	
acknowledged	deletedCount
T/F true	[id] 1.0

_id	student_id	class_id	scores
[id] 50b59cd75bed7...	[id] 1	[id] 22	[] [5 elements]
[id] 50b59cd75bed7...	[id] 1	[id] 28	[] [5 elements]
[id] 50b59cd75bed7...	[id] 1	[id] 16	[] [5 elements]
[id] 50b59cd75bed7...	[id] 1	[id] 13	[] [4 elements]

Esta consulta usa el comando remove (que es más antiguo) para eliminar un solo documento de la colección grades donde student_id es igual a 1. La opción justOne: true asegura que solo un documento será eliminado, incluso si hay más de uno que cumpla con el filtro.

5. db.grades.remove({ student_id: 1 })

acknowledged	deletedCount
T/F true	[id] 4.0

grades > class_id			
_id	student_id	class_id	scores
[id] 50b59cd75bed7...	[id] 2	[id] 25	[] [5 elements]
[id] 50b59cd75bed7...	[id] 2	[id] 27	[] [4 elements]
[id] 50b59cd75bed7...	[id] 2	[id] 24	[] [4 elements]
[id] 50b59cd75bed7...	[id] 3	[id] 10	[] [4 elements]

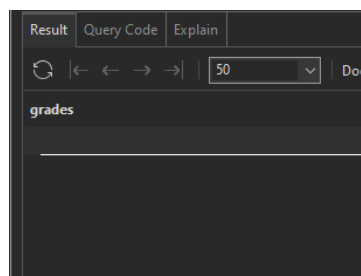
Esta consulta elimina todos los documentos en la colección grades donde student_id sea igual a 1. A diferencia de la consulta anterior, no está limitada a eliminar un solo documento.

PROGRAMA INGENIERIA DE SISTEMAS

Página 49 | 55

6. `db.grades.remove({})`

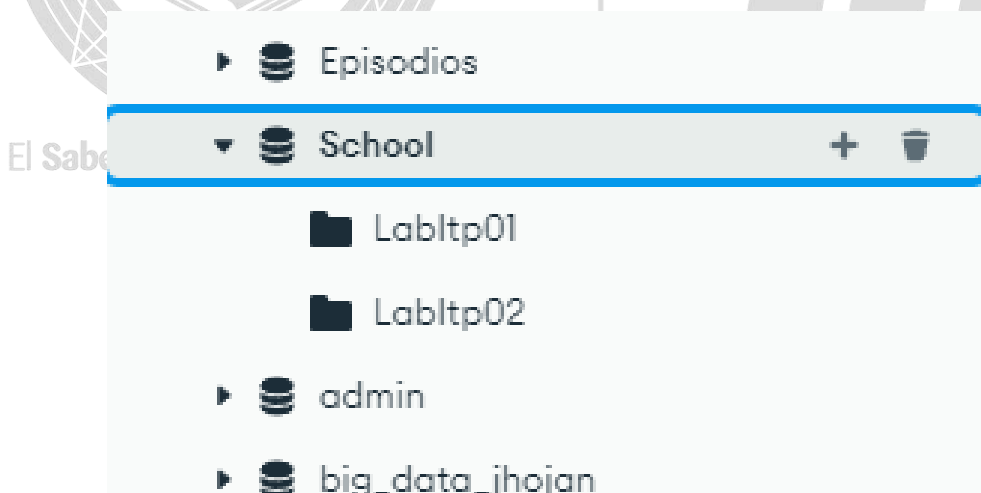
Result > deletedCount	
acknowledged	deletedCount
True	123 264.0



Esta consulta elimina todos los documentos en la colección `grades`. El filtro vacío `{}` indica que no hay condiciones, por lo que se eliminarán todos los registros presentes en la colección.

7. `db.grades.drop()`

Esta consulta elimina por completo la colección `grades`, junto con todos sus documentos y su estructura



PROGRAMA INGENIERIA DE SISTEMAS

Página 50 | 55

```
15
16 db.grades.drop()
17

Raw shell output

Restart the MongoDB Shell Use legacy shell Clear raw shell output Pin new results

11 //
12 // - Ctrl+Enter runs the current selection, or the statement at the
13 // current cursor position if nothing is selected
14 // - ESC offers you to restart the shell should you want to cancel the
15 // execution of the current command or script
16 // - "cls" clears the raw shell output tab
17 // - Run Shell Queries via Studio 3T for full access to query results
18 // -----
19
20 Current Mongosh Log ID: 67087c90a372f90b09072f09
21 Using Mongosh: 2.2.5
22
23 For mongosh info see: https://docs.mongodb.com/mongosh-shell/
24
25 >
26 > School
27 The find query will be run with Query Assist.
28 { acknowledged: true, deletedCount: 1 }
29 The find query will be run with Query Assist.
30 { acknowledged: true, deletedCount: 0 }
31 { acknowledged: true, deletedCount: 1 }
32 { acknowledged: true, deletedCount: 10 }
33 DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findAndDelete, or bulkWrite.
34 { acknowledged: true, deletedCount: 1 }
35 { acknowledged: true, deletedCount: 4 }
36 { acknowledged: true, deletedCount: 264 }
37 true
38
39
40
```



El Saber como Arma de Vida



PROGRAMA INGENIERIA DE SISTEMAS

Página 51 | 55

5. Análisis y Discusión

En esta práctica, se llevó a cabo el diseño y la implementación de una base de datos para un sistema de reservas de vuelos y un taller practico para introducir a CRUD en MongoDB.

1. Estructura de las Colecciones:

- Se crearon cinco colecciones principales: Users, Personal_Info, Flights, Airports, y Reservations. Cada colección contiene datos específicos que permiten un acceso estructurado a la información.
- La creación de colecciones separadas para diferentes tipos de datos facilita la organización y el manejo de información. Esto es especialmente importante en sistemas de reservas, donde los datos pueden ser extensos y variados.

2. Integración de Datos:

- Los campos comunes, como personalInfoId en la colección Users, permiten la integración de información entre las colecciones. Esto no solo garantiza la consistencia de los datos, sino que también optimiza las consultas realizadas en el sistema.

3. Resultados de Consultas:

- Al realizar consultas en las colecciones, se observó que la estructura de la base de datos permite recuperar información de manera rápida y eficiente. Por ejemplo, obtener todos los vuelos reservados por un usuario específico, lo que demuestra la eficacia del diseño implementado.

4. Cumplimiento de Objetivos:

- Los resultados obtenidos se alinean con los objetivos establecidos al inicio de la práctica. Se buscaba crear una base de datos que no solo almacenara información, sino que también facilitara su recuperación y manejo.

PROGRAMA INGENIERIA DE SISTEMAS

Página 52 | 55

- La normalización y la definición clara de relaciones entre colecciones lograron este objetivo, permitiendo un acceso eficiente a los datos, así como un mantenimiento y una actualización más sencillos.

5. Visualización de Datos:

- La visualización de la estructura de la base de datos a través de herramientas como Studio 3T y MongoDB Compass permitió una mejor comprensión de las relaciones entre las colecciones. Esto es fundamental para el análisis posterior y para la toma de decisiones en el desarrollo del sistema.



PROGRAMA INGENIERIA DE SISTEMAS

Página 53 | 55

6. Conclusiones

La creación de múltiples colecciones, cada una especializada en un tipo específico de dato, ha permitido una organización clara y un manejo eficiente de la información. Esto es fundamental para mantener la integridad de los datos y facilitar su recuperación.

La aplicación de principios de normalización y la definición de relaciones entre las colecciones han demostrado ser efectivas para reducir la redundancia de datos. Al embeber información clave, como los detalles personales de los usuarios dentro de la colección correspondiente, se logró optimizar el rendimiento de las consultas.

La capacidad de realizar consultas complejas de manera sencilla, gracias a la estructura de la base de datos, facilita el acceso a la información necesaria para los usuarios. Esto es esencial en un sistema de reservas, donde la rapidez y la precisión son cruciales para la satisfacción del cliente.



Las herramientas empleadas, como MongoDB Compass y Studio 3T, han permitido visualizar y analizar la estructura de la base de datos de manera intuitiva. Esta visualización es invaluable para comprender las interacciones entre las diferentes colecciones y para planificar futuras mejoras en el sistema.

El Saber como Arma de Vida

PROGRAMA INGENIERIA DE SISTEMAS

Página 54 | 55

7. Recomendaciones

Se recomienda que se continúe explorando y profundizando en el uso de MongoDB y otras bases de datos NoSQL. Dada la creciente demanda de aplicaciones que manejan grandes volúmenes de datos y la necesidad de flexibilidad en el diseño de bases de datos, el conocimiento de sistemas NoSQL se ha vuelto cada vez más necesario. Las prácticas realizadas en este informe no solo han proporcionado una comprensión teórica sobre el diseño y manejo de bases de datos, sino que también han permitido experimentar con herramientas prácticas que son esenciales en el entorno laboral actual. La formación continua de estos talleres sobre estas tecnologías es recomendable, al igual que la realización de proyectos prácticos que involucren la implementación de bases de datos NoSQL en aplicaciones del mundo real. El unirse a comunidades en línea y foros especializados puede enriquecer el aprendizaje, al permitir compartir experiencias y colaborar en proyectos. Explorar otras tecnologías NoSQL, como Cassandra y Redis, también puede abrir nuevas oportunidades profesionales.

El **Saber** como **Arma de Vida**



INSTITUTO TECNOLÓGICO DEL PUTUMAYO



IES Vigilada por:
Educación

PROGRAMA INGENIERIA DE SISTEMAS

Página 55 | 55

8. Referencias

Repositorio (*Carpeta CRUD_MONGO*): https://github.com/Camilo138/repository_sql.git



El **Saber** como **Arma** de **Vida**

