

Consultas y Diseño de Bases de Datos en MongoDB

Presentado Por:

Daniel Felipe Mora Rosas

Presentado A:

Lic. Brayan Arcos

Instituto Tecnológico Del Putumayo

17 de octubre del 2024

Mocoa – Putumayo

Contenido

Resumen Ejecutivo	4
Introducción	5
Contexto y Motivación	5
Alcance	6
Objetivos	7
Metodología	8
Herramientas Utilizadas en la Creación de la Base de Datos	8
Herramientas Utilizadas en el Análisis de las Consultas	9
Procedimientos.....	10
Creación de mi base de datos.....	10
Análisis de las Consultas	11
Métodos para capturar los datos	11
Desarrollo del Informe	12
Descripción de la Base de Datos de Datos Personal.....	12
Diseño de Base de Datos Personal.....	14
Normalización.....	14
Cardinalidad y Relaciones	14
Métodos de captura	16
Consultas.....	19

READ.....	19
CREATE	24
UPDATE	27
DELETE	30
Análisis y Discusión	32
Conclusión	33
Referencia	34

Resumen Ejecutivo

Este informe cubre dos tareas principales en MongoDB. La primera fue ejecutar varias consultas en bases de datos y explicar su funcionamiento. La segunda fue diseñar una base de datos para una plataforma de streaming, donde se aplicaron relaciones 1 a 1, 1 a muchos y muchos a muchos, usando funciones como **updateOne()**, **upsert**, y **setOnInsert**. También describimos las herramientas que usamos y los pasos que seguimos.

Introducción

Contexto y Motivación

Este informe se realizó para practicar el uso de MongoDB en dos escenarios: ejecutar consultas en bases de datos existentes y crear una base de datos desde cero.

La motivación principal de este informe es aplicar conceptos importantes de MongoDB, como las consultas y las relaciones entre colecciones. Esto nos permite comprender mejor cómo podemos mejorar el rendimiento y la estructura de una base de datos en proyectos reales.

Alcance

Este informe trata del como nosotros como estudiantes trabajamos con MongoDB. Como primer lugar estaríamos realizando unas consultas sobre bases de datos ya existentes para entender su funcionamiento y analizar los resultados. En segundo lugar, se crea una base de datos para una plataforma de streaming, donde mencionaremos diferentes tipos de relaciones entre colecciones. El objetivo es brindar una visión sobre una forma (de muchas formas) de como podemos manejar de MongoDB.

Objetivos

- Realizar consultas en MongoDB para obtener información de las bases de datos.
- Diseñar una base de datos sobre una plataforma de streaming, teniendo en cuenta las relaciones entre colecciones.
- Usar actualizaciones de datos usando métodos como **updateOne()**, **upsert**, y **setOnInsert**.
- Evaluar los resultados de las consultas y el diseño de la base de datos para mejorar la comprensión del uso de MongoDB.

Metodología

Herramientas Utilizadas en la Creación de la Base de Datos

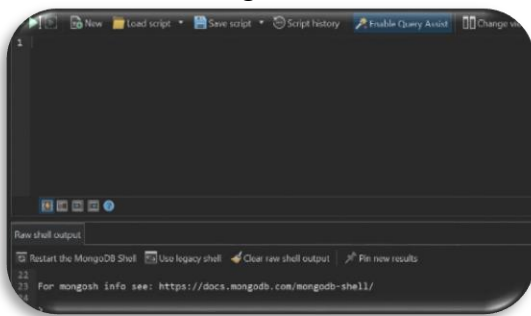
1. Studio 3T



2. Mongo DB



3. Mongoddb-shell



4. Git-Hub



Herramientas Utilizadas en el Análisis de las Consultas

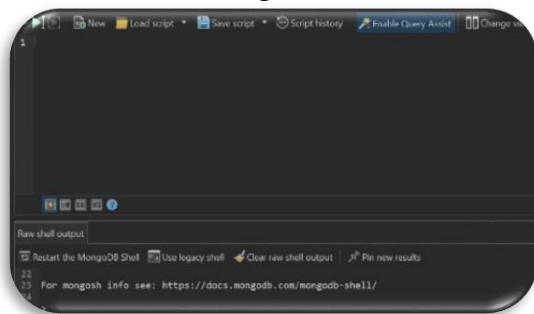
1. Studio 3T



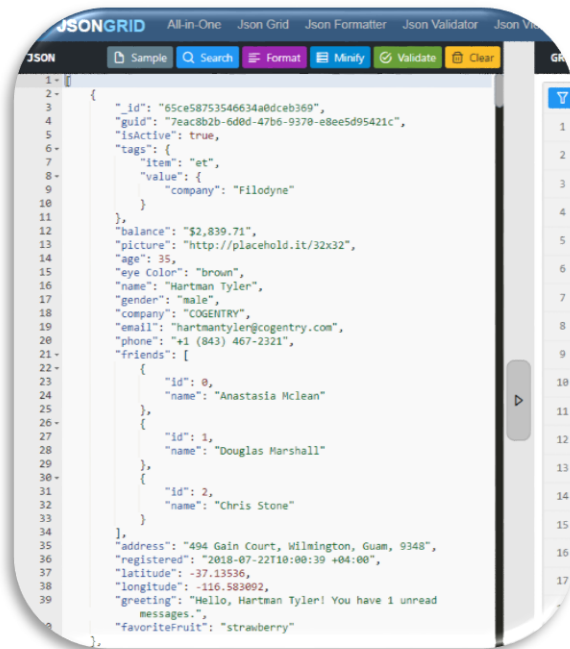
2. Mongo DB



4. Mongoddb-shell



3. JsonGrid



Procedimientos

Pasos para llevar a cabo el análisis y el trabajo

Primero, la creación de la base de datos sobre el streaming, y luego, la ejecución de consultas en las bases de datos que nos dieron.

Creación de mi base de datos

Lo primero que hice fue diseñar y armar una base de datos sencilla sobre una plataforma de streaming. Para esto, tuve que definir bien las colecciones, sus relaciones (1 a 1, 1 a muchos y muchos a muchos), y luego implementarlas en MongoDB. El proceso fue algo así:

- **Definir la idea:** Pensar como funcionaría la plataforma de streaming y qué tipo de datos necesitaría almacenar.
- **Diseñar el modelo de datos:** Nos aseguramos que las colecciones estuvieran bien relacionadas, para que la base de datos funcione sin problemas y que no encontremos redundancia.
- **Limpiar y preparar el código JSON:** Antes de cargar los datos en MongoDB, utilicé una herramienta online, **JsonGrid Online**, para limpiar y verificar el formato del código JSON. Esto para escoger el código que tendría el contenido de la base de datos
- **Implementar en MongoDB:** Usando las herramientas de MongoDB Shell y Studio 3T pudimos crear las colecciones e insertar datos.ç

Análisis de las Consultas

La segunda parte del trabajo fue ejecutar varias consultas en las dos bases de datos que nos dieron. Para esto, utilicé herramientas como MongoDB Shell, MongoDB Compass y Studio 3T, que me ayudaron a hacer las consultas y a ver los resultados. Los pasos que seguí fueron:

- **Cargar las bases de datos:** Lo primero fue importar los archivos de las bases de datos.
- **Hacer las consultas:** Ejecutamos todas las consultas que se pedían, algunas de ellas como las de `find()` y `aggregate()`, para ver cómo se comportaban los datos.
- **Explicar el funcionamiento:** Una vez que vimos los resultados, analizamos y los explicamos, relacionando cada consulta con lo que estaba almacenado en las bases de datos.

Métodos para capturar los datos

Los datos los obtuve ejecutando las consultas directamente en las bases de datos. Cada resultado lo fui guardando y analizando para entender mejor cómo funcionaba la base de datos y cómo las consultas influían en los datos. Para hacerlo más eficiente, guardé las consultas en scripts que luego podía volver a ejecutar sin tener que escribir todo de nuevo.

Desarrollo del Informe

Descripción de la Base de Datos de Datos Personal

Colección roles

La colección `roles` está diseñada para almacenar los diferentes roles que pueden tener los usuarios en la plataforma. Cada rol tiene un identificador único y un nombre descriptivo. Esto permite gestionar los permisos y accesos de los usuarios de manera eficiente.

Colección users

La colección `users` almacena la información personal y de acceso de los usuarios. Cada documento incluye un identificador único, el correo electrónico, un hash de la contraseña, y un perfil que contiene detalles como el nombre del canal, nombre y apellido, dirección y números de teléfono. Además, se incluyen los roles asignados y a quiénes sigue el usuario.

Colección paymentMethods

La colección `paymentMethods` contiene información sobre los métodos de pago de los usuarios. Cada documento incluye un identificador único, el ID del usuario asociado, y detalles del método de pago como el número de tarjeta, fecha de expiración y dirección de facturación.

Colección subscriptions

La colección `subscriptions` almacena información sobre las suscripciones de los usuarios a streamers. Cada documento incluye el ID del usuario que se suscribe, el ID del streamer, el nivel de la suscripción, fechas de inicio y expiración, el método de pago utilizado y el estado de la suscripción.

Colección `giftedSubscriptions`

La colección `giftedSubscriptions` permite gestionar las suscripciones que son regaladas por un usuario a otros. Cada documento incluye el ID del usuario que regala, los IDs de los usuarios que reciben las suscripciones, el ID del streamer, el nivel de la suscripción, fechas de inicio y expiración, y un mensaje opcional.

Colección `streams`

La colección `streams` almacena información sobre las transmisiones en vivo realizadas por los usuarios. Cada documento incluye el ID del usuario que transmite, el título y la descripción del stream, la categoría, etiquetas, estado de transmisión en vivo, y un registro de los mensajes del chat.

Colección `streamViewers`

La colección `streamViewers` registra la información sobre los espectadores de cada transmisión. Cada documento incluye el ID del stream y una lista de IDs de los usuarios que están viendo la transmisión.

Colección `chatMessages`

La colección `chatMessages` almacena los mensajes enviados en el chat durante las transmisiones. Cada documento incluye el ID del usuario que envió el mensaje, el ID del stream, el contenido del mensaje y la marca de tiempo.

Diseño de Base de Datos Personal

Modelo de Datos: Normalización y cardinalidad

Normalización

Para normalizar la base de datos proporcionada, opté por separar entidades como los métodos de pago y las suscripciones en colecciones independientes, en lugar de embebedas dentro de los documentos de usuarios. Esta decisión permite evitar la duplicación de información, facilitando la reutilización de datos cuando varios usuarios comparten un mismo método de pago o tipo de suscripción. Además, al mantener estas entidades en colecciones separadas, se simplifica la actualización y el mantenimiento de la información, ya que los cambios se realizan en un único lugar sin afectar múltiples documentos.

Cardinalidad y Relaciones

Relación entre users y roles: Es de muchos a muchos, ya que un usuario puede tener múltiples roles, y un rol puede ser asignado a varios usuarios. Esto se implementa como un array de roleIds en la colección users.

- **Ejemplo:** El campo roles: [1] en un documento de users indica que el usuario tiene el rol asociado al _id 1 (usuario).

Relación entre users y follows: Es de muchos a muchos, ya que un usuario puede seguir a múltiples usuarios y, a su vez, puede ser seguido por otros. Se maneja mediante un array de follows en la colección users, donde se almacena el _id de los usuarios seguidos.

- **Ejemplo:** El campo follows: [2, 3, 4] en un documento de users indica que el usuario sigue a los usuarios con _id 2, 3 y 4.

Relación entre users y subscriptions: Es de uno a muchos, ya que un usuario puede estar suscrito a múltiples streamers. Se representa en la colección subscriptions, donde el campo userId es una referencia al usuario que realiza la suscripción, y el campo streamerId indica el streamer al que se suscribe.

- **Ejemplo:** El documento { "userId": 1, "streamerId": 2, "tier": 1 } en subscriptions muestra que el usuario 1 está suscrito al streamer 2 con el nivel 1 de suscripción.

Relación entre giftedSubscriptions y users: Es de uno a muchos, ya que un usuario puede regalar suscripciones a múltiples usuarios. El campo userIds en giftedSubscriptions almacena un array con los _id de los usuarios que reciben las suscripciones.

- **Ejemplo:** El campo userIds: [4, 5, 6] en un documento de giftedSubscriptions muestra que los usuarios 4, 5 y 6 recibieron suscripciones regaladas.

Métodos de captura

Para crear la base de datos y las colecciones en MongoDB a partir de los archivos JSON proporcionados, seguimos un procedimiento sencillo utilizando código para automatizar el proceso.

- **Base de datos:** Inicialmente, creamos una nueva base de datos donde almacenaremos las diferentes colecciones relacionadas con usuarios, roles, métodos de pago, suscripciones, transmisiones, entre otros.
- **Colecciones:** Luego, por cada entidad definida en los archivos JSON (por ejemplo, users, roles, paymentMethods, subscriptions), se crea una colección dentro de la base de datos. Estos documentos JSON se insertan directamente en las colecciones utilizando un script, que puede estar basado en JavaScript, Python, o comandos de MongoDB.

Explicación del código (se adjuntará la imagen en el informe):

A través del código, usamos el método `insertMany()` para añadir los documentos a cada colección.

Para cada archivo JSON, lo cargamos en la colección correspondiente


```

db.roles.insertMany([
  {
    "_id": 1,
    "roleName": "Usuario"
  },
  {
    "_id": 2,
    "roleName": "Admin"
  }
])

```

```

db.users.insertMany([
  {
    "_id": 1,
    "email": "juanperez@gmail.com",
    "passwordHash": "hash_del_password",
    "profile": {
      "channelName": "roadtomaster",
      "firstName": "Juan Miguel",
      "lastName": "Pérez",
      "address": "Calle Falsa 123",
      "phones": ["123456789", "987654321"]
    },
    "roles": [1],
    "follows": [2, 3, 4, 5, 6],
    "createdAt": new Date("2024-10-15T00:00:00Z"),
    "updatedAt": new Date("2024-10-15T00:00:00Z")
  }
])

```

```

db.paymentMethods.insertMany([
  {
    "_id": 1,
    "userId": 1,
    "cardNumber": "4111111111111111",
    "expirationDate": "12/26",
    "cvv": "123",
    "billingAddress": "Calle Falsa 123",
    "createdAt": new Date("2024-10-15T00:00:00Z"),
    "updatedAt": new Date("2024-10-15T00:00:00Z")
  }
])

```

```

db.subscriptions.insertMany([
  {
    "_id": 1,
    "userId": 1,
    "streamerId": 2,
    "tier": 1,
    "startedAt": new Date("2024-10-15T00:00:00Z"),
    "expiresAt": new Date("2024-11-15T00:00:00Z"),
    "paymentMethodId": 1,
    "status": "active",
    "timestamp": new Date("2024-10-15T00:00:00Z")
  }
])

```

```

db.giftedSubscriptions.insertMany([
  {
    "_id": 1,
    "giftedBy": 1,
    "userIds": [4, 5, 6, 7, 8, 9, 10, 11, 12, 13],
    "streamerId": 2,
    "tier": 1,
    "startedAt": new Date("2024-10-15T00:00:00Z"),
    "expiresAt": new Date("2024-11-15T00:00:00Z"),
    "message": "¡Disfruta tu suscripción!",
    "timestamp": new Date("2024-10-15T00:00:00Z")
  }
])

```

```

db.streams.insertMany([
  {
    "_id": 101,
    "userId": 1,
    "title": "Stream de gaming",
    "description": "Jugando un juego de estrategia",
    "category": "Gaming",
    "tags": ["estrategia", "multijugador"],
    "isLive": true,
    "chat": [
      { "userId": 1,
        "message": "¡Sigue así, eres increíble!",
        "timestamp": new Date("2024-10-16T01:00:00Z") },
      { "userId": 4,
        "message": "¡Espero que te guste la suscripción!",
        "timestamp": new Date("2024-10-16T01:05:00Z") }
    ],
    "createdAt": new Date("2024-10-15T00:00:00Z"),
    "updatedAt": new Date("2024-10-15T01:00:00Z")
  }
])

```

```
db.streamViewers.insertMany([
  {
    "_id": 101,
    "streamId": 101,
    "viewers": [1, 2, 3, 200],
    "timestamp": new Date("2024-10-17T01:00:00Z")
  }
])
```

```
db.chatMessages.insertMany([
  {
    "_id": 1,
    "userId": 1,
    "streamId": 101,
    "message": "¡Sigue así, eres increíble!",
    "timestamp": new Date("2024-10-16T01:00:00Z")
  }
])
```

Consultas

READ

Consulta	Función																											
db.grades.find({ student_id: 2 })	De la colección grades, busca los estudiantes con id 2																											
<table><tr><th>_id</th><th>student_id</th><th>class_id</th><th>scores</th></tr><tr><td>50b59cd75bed7...</td><td>2</td><td>25</td><td>[5 elements]</td></tr><tr><td>50b59cd75bed7...</td><td>2</td><td>27</td><td>[4 elements]</td></tr><tr><td>50b59cd75bed7...</td><td>2</td><td>24</td><td>[4 elements]</td></tr></table>		_id	student_id	class_id	scores	50b59cd75bed7...	2	25	[5 elements]	50b59cd75bed7...	2	27	[4 elements]	50b59cd75bed7...	2	24	[4 elements]											
_id	student_id	class_id	scores																									
50b59cd75bed7...	2	25	[5 elements]																									
50b59cd75bed7...	2	27	[4 elements]																									
50b59cd75bed7...	2	24	[4 elements]																									
db.grades.find({ "scores.0.score": { \$lte: 10 } })	Busca en la colección grades el primer elemento de scores, que sea menor igual a 10																											
<div>grades > scores > 0 > score</div> <table><tr><th>{Document id}</th><th>type</th><th>score</th></tr><tr><td>50b59cd75bed7...</td><td>exam</td><td>0.59987351892...</td></tr><tr><td>50b59cd75bed7...</td><td>exam</td><td>7.33833450353...</td></tr><tr><td>50b59cd75bed7...</td><td>exam</td><td>4.44443575902...</td></tr><tr><td>50b59cd75bed7...</td><td>exam</td><td>0.65430967860...</td></tr><tr><td>50b59cd75bed7...</td><td>exam</td><td>7.22285465008...</td></tr><tr><td>50b59cd75bed7...</td><td>exam</td><td>5.23116601873...</td></tr><tr><td>50b59cd75bed7...</td><td>exam</td><td>5.46672768849...</td></tr><tr><td>50b59cd75bed7...</td><td>exam</td><td>8.94201896826...</td></tr></table>		{Document id}	type	score	50b59cd75bed7...	exam	0.59987351892...	50b59cd75bed7...	exam	7.33833450353...	50b59cd75bed7...	exam	4.44443575902...	50b59cd75bed7...	exam	0.65430967860...	50b59cd75bed7...	exam	7.22285465008...	50b59cd75bed7...	exam	5.23116601873...	50b59cd75bed7...	exam	5.46672768849...	50b59cd75bed7...	exam	8.94201896826...
{Document id}	type	score																										
50b59cd75bed7...	exam	0.59987351892...																										
50b59cd75bed7...	exam	7.33833450353...																										
50b59cd75bed7...	exam	4.44443575902...																										
50b59cd75bed7...	exam	0.65430967860...																										
50b59cd75bed7...	exam	7.22285465008...																										
50b59cd75bed7...	exam	5.23116601873...																										
50b59cd75bed7...	exam	5.46672768849...																										
50b59cd75bed7...	exam	8.94201896826...																										
db.grades.find({ "scores.4.score": { \$lte: 10 } })	Busca en la colección grades el quinto elemento de scores, que sea menor igual a 10																											
<div>grades > scores > 4 > type</div> <table><tr><th>{Document id}</th><th>type</th><th>score</th></tr><tr><td>50b59cd75bed7...</td><td>homework</td><td>1.23735944117...</td></tr></table>		{Document id}	type	score	50b59cd75bed7...	homework	1.23735944117...																					
{Document id}	type	score																										
50b59cd75bed7...	homework	1.23735944117...																										
db.grades.find({ "scores.5.score": { \$lte: 10 } })	Busca en la colección grades el sexto elemento de scores, que sea menor igual a 10																											
<div>grades > scores > 5 > type</div> <table><tr><th>{Document id}</th><th>type</th><th>score</th></tr><tr><td>50b59cd75bed7...</td><td>homework</td><td>6.24943037646...</td></tr></table>		{Document id}	type	score	50b59cd75bed7...	homework	6.24943037646...																					
{Document id}	type	score																										
50b59cd75bed7...	homework	6.24943037646...																										

db.grades.find({ "scores.6.score": { \$lte: 10 } })	Busca en la colección grades el septimo elemento de scores, que sea menor igual a 10									
Los campos que solicitan, no los tenemos, por ende no hay resultado de salida										
db.grades.find({ "scores.0.score": { \$gte: 60, \$lte: 61 } })	Busca en la colección grades el primer elemento de scores, debe estar entre mayor igual a 60 y menor igual a 61									
<div>grades > scores > 0 > type</div> <table><tr><th>{Document id}</th><th>type</th><th>score</th></tr><tr><td>"50b59cd75bed7...</td><td>exam</td><td>60.1947363615...</td></tr><tr><td>"50b59cd75bed7...</td><td>exam</td><td>60.1391248993...</td></tr></table>		{Document id}	type	score	"50b59cd75bed7...	exam	60.1947363615...	"50b59cd75bed7...	exam	60.1391248993...
{Document id}	type	score								
"50b59cd75bed7...	exam	60.1947363615...								
"50b59cd75bed7...	exam	60.1391248993...								
db.grades.find({ "scores.0.score": { \$gte: 60, \$lte: 61 } }) .sort({ student_id: 1 })	Busca en la colección grades el primer elemento de scores, debe estar entre mayor igual a 60 y menor igual a 61, y con el .sort() decimos que ordene los resultado en este caso en forma ascendente(1) a través del id del estudiante									
Los campos que solicitan, no los tenemos, por ende no hay resultado de salida										
db.grades.find({ student_id: 2, class_id: 24 })	Busca específicamente al estudiante de id 2 pero de la clase de id 24									
<div>grades > class_id</div> <table><tr><th>_id</th><th>student_id</th><th>class_id</th><th>scores</th></tr><tr><td>"50b59cd75bed7...</td><td>2</td><td>24</td><td>[4 elements]</td></tr></table>		_id	student_id	class_id	scores	"50b59cd75bed7...	2	24	[4 elements]	
_id	student_id	class_id	scores							
"50b59cd75bed7...	2	24	[4 elements]							
db.grades.find({ class_id: 20, \$and: [{ "scores.0.score": { \$gte: 15 } }, { "scores.0.score": { \$lte: 30 } }] })	Buscamos un estudiante de la clase id 20, donde tiene que tener el primer score entre 15 y 30									
Los campos que solicitan, no los tenemos, por ende no hay resultado de salida										
db.grades.find({ scores: { \$elemMatch: { type: 'quiz', score: { \$gte: 50 } } } })	Busca el estudiante que en scores en el tipo quiz la nota sea mayor igual a 50									

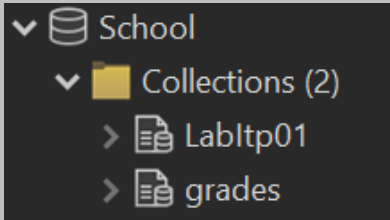
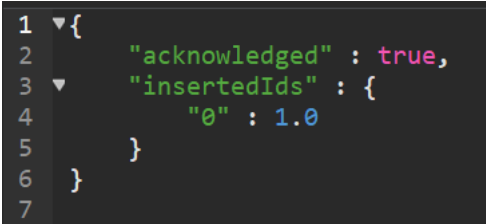
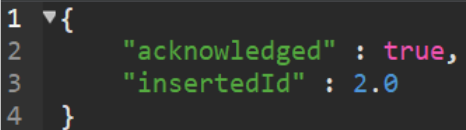
<div>grades > scores > 1 > type</div> <table><tr><th>{Document id}</th><th>type</th><th>score</th></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 79.2896265042...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 60.4769945611...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 64.1596621001...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 55.9800328152...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 69.0817586839...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 77.8923336619...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 65.3994293203...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 66.2422007124...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 94.1422265283...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 82.4870509624...</td></tr><tr><td> 50b59cd75bed7...</td><td> quiz</td><td> 70.2733455818...</td></tr></table>		{Document id}	type	score	50b59cd75bed7...	quiz	79.2896265042...	50b59cd75bed7...	quiz	60.4769945611...	50b59cd75bed7...	quiz	64.1596621001...	50b59cd75bed7...	quiz	55.9800328152...	50b59cd75bed7...	quiz	69.0817586839...	50b59cd75bed7...	quiz	77.8923336619...	50b59cd75bed7...	quiz	65.3994293203...	50b59cd75bed7...	quiz	66.2422007124...	50b59cd75bed7...	quiz	94.1422265283...	50b59cd75bed7...	quiz	82.4870509624...	50b59cd75bed7...	quiz	70.2733455818...			
{Document id}	type	score																																						
50b59cd75bed7...	quiz	79.2896265042...																																						
50b59cd75bed7...	quiz	60.4769945611...																																						
50b59cd75bed7...	quiz	64.1596621001...																																						
50b59cd75bed7...	quiz	55.9800328152...																																						
50b59cd75bed7...	quiz	69.0817586839...																																						
50b59cd75bed7...	quiz	77.8923336619...																																						
50b59cd75bed7...	quiz	65.3994293203...																																						
50b59cd75bed7...	quiz	66.2422007124...																																						
50b59cd75bed7...	quiz	94.1422265283...																																						
50b59cd75bed7...	quiz	82.4870509624...																																						
50b59cd75bed7...	quiz	70.2733455818...																																						
<div>db.grades.find({ scores: { \$elemMatch: { type: 'exam', score: { \$gte: 50 } } } })</div>	Busca el estudiante que en scores en el tipo exam la nota sea mayor igual a 50																																							
<div>grades > scores > 0 > type</div> <table><tr><th>{Document id}</th><th>type</th><th>score</th></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 57.9294711257...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 88.2295067423...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 59.1805667559...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 56.8198151386...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 60.1947363615...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 58.8329741110...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 63.0973787710...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 80.6212442791...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 68.9337029758...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 70.3295399202...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 76.0187667451...</td></tr><tr><td> 50b59cd75bed7...</td><td> exam</td><td> 88.8082254274...</td></tr></table>		{Document id}	type	score	50b59cd75bed7...	exam	57.9294711257...	50b59cd75bed7...	exam	88.2295067423...	50b59cd75bed7...	exam	59.1805667559...	50b59cd75bed7...	exam	56.8198151386...	50b59cd75bed7...	exam	60.1947363615...	50b59cd75bed7...	exam	58.8329741110...	50b59cd75bed7...	exam	63.0973787710...	50b59cd75bed7...	exam	80.6212442791...	50b59cd75bed7...	exam	68.9337029758...	50b59cd75bed7...	exam	70.3295399202...	50b59cd75bed7...	exam	76.0187667451...	50b59cd75bed7...	exam	88.8082254274...
{Document id}	type	score																																						
50b59cd75bed7...	exam	57.9294711257...																																						
50b59cd75bed7...	exam	88.2295067423...																																						
50b59cd75bed7...	exam	59.1805667559...																																						
50b59cd75bed7...	exam	56.8198151386...																																						
50b59cd75bed7...	exam	60.1947363615...																																						
50b59cd75bed7...	exam	58.8329741110...																																						
50b59cd75bed7...	exam	63.0973787710...																																						
50b59cd75bed7...	exam	80.6212442791...																																						
50b59cd75bed7...	exam	68.9337029758...																																						
50b59cd75bed7...	exam	70.3295399202...																																						
50b59cd75bed7...	exam	76.0187667451...																																						
50b59cd75bed7...	exam	88.8082254274...																																						
<div>db.grades.distinct("student_id")</div>	devuelve una lista única (sin duplicados) de todos los valores del campo student_id																																							
<div>1 ▾[2 0.0, 3 1.0, 4 2.0, 5 3.0, 6 4.0, 7 5.0, 8 6.0, 9 7.0, 10 8.0, 11 9.0 12]</div>																																								
<div>db.grades.countDocuments()</div>	Nos da el número total de documentos que existen en la colección grades																																							
<div>switched to db School [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] 65</div>																																								

- Narcos

Consulta	Función																																																												
<pre>db.Narcos.find({ runtime: { \$gte: 55 } }, { _id:0, name:1, season:1, number:1 })</pre>	Buscamos los capítulos que tengan runtime mayor igual a 55 y que solo nos muestre name, season y number																																																												
<div><div>Narcos > season</div><table><tr><th>name</th><th>season</th><th>number</th></tr><tr><td> The Good, the B...</td><td> 2</td><td> 4</td></tr><tr><td> There Will Be a F...</td><td> 1</td><td> 5</td></tr><tr><td> Deutschland 93</td><td> 2</td><td> 7</td></tr><tr><td> Exit El Patrón</td><td> 2</td><td> 8</td></tr><tr><td> Descenso</td><td> 1</td><td> 1</td></tr><tr><td> Nuestra Finca</td><td> 2</td><td> 9</td></tr><tr><td> The Kingpin Strat...</td><td> 3</td><td> 1</td></tr><tr><td> Follow the Money</td><td> 3</td><td> 3</td></tr><tr><td> MRO</td><td> 3</td><td> 5</td></tr><tr><td> Best Laid Plans</td><td> 3</td><td> 6</td></tr></table></div>		name	season	number	The Good, the B...	2	4	There Will Be a F...	1	5	Deutschland 93	2	7	Exit El Patrón	2	8	Descenso	1	1	Nuestra Finca	2	9	The Kingpin Strat...	3	1	Follow the Money	3	3	MRO	3	5	Best Laid Plans	3	6																											
name	season	number																																																											
The Good, the B...	2	4																																																											
There Will Be a F...	1	5																																																											
Deutschland 93	2	7																																																											
Exit El Patrón	2	8																																																											
Descenso	1	1																																																											
Nuestra Finca	2	9																																																											
The Kingpin Strat...	3	1																																																											
Follow the Money	3	3																																																											
MRO	3	5																																																											
Best Laid Plans	3	6																																																											
<pre>db.Narcos.find({ runtime: { \$gte: 15 } }, {id:0, season:1, number:1}).sort({ season:1, number:-1 })</pre>	Buscamos los capítulos que tengan runtime mayor igual a 15 y que solo nos muestre season y number; y ademas las season se deben organizar de forma ascendente y los number de forma descendente																																																												
<div><div>Narcos > number</div><table><tr><th>season</th><th>number</th></tr><tr><td> 1</td><td> 10</td></tr><tr><td> 1</td><td> 9</td></tr><tr><td> 1</td><td> 8</td></tr><tr><td> 1</td><td> 7</td></tr><tr><td> 1</td><td> 6</td></tr><tr><td> 1</td><td> 5</td></tr><tr><td> 1</td><td> 4</td></tr><tr><td> 1</td><td> 3</td></tr><tr><td> 1</td><td> 2</td></tr><tr><td> 1</td><td> 1</td></tr><tr><td> 2</td><td> 10</td></tr><tr><td> 2</td><td> 9</td></tr></table></div>		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																																		
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																																																												
<pre>db.Narcos.find({ season: { \$type: 'number' } })</pre>	Buscamos los capítulos donde el campo season sea tipo numero																																																												
<div><div>Narcos > id</div><table><tr><th>_id</th><th>id</th><th>url</th><th>name</th><th>season</th></tr><tr><td> 208978</td><td> 208978</td><td> https://www.tv...</td><td> The Sword of Si...</td><td> 1</td></tr><tr><td> 670994d2bb499...</td><td> 832100</td><td> https://www.tv...</td><td> Our Man in Ma...</td><td> 2</td></tr><tr><td> 670994d2bb499...</td><td> 832101</td><td> https://www.tv...</td><td> The Good, the B...</td><td> 2</td></tr><tr><td> 670994d2bb499...</td><td> 208981</td><td> https://www.tv...</td><td> There Will Be a F...</td><td> 1</td></tr><tr><td> 670994d2bb499...</td><td> 832102</td><td> https://www.tv...</td><td> The Enemies of ...</td><td> 2</td></tr><tr><td> 670994d2bb499...</td><td> 832103</td><td> https://www.tv...</td><td> Los Pepes</td><td> 2</td></tr><tr><td> 670994d2bb499...</td><td> 832256</td><td> https://www.tv...</td><td> Deutschland 93</td><td> 2</td></tr><tr><td> 670994d2bb499...</td><td> 832099</td><td> https://www.tv...</td><td> Cambalache</td><td> 2</td></tr><tr><td> 670994d2bb499...</td><td> 832257</td><td> https://www.tv...</td><td> Exit El Patrón</td><td> 2</td></tr><tr><td> 670994d2bb499...</td><td> 203469</td><td> https://www.tv...</td><td> Descenso</td><td> 1</td></tr><tr><td> 670994d2bb499...</td><td> 832258</td><td> https://www.tv...</td><td> Nuestra Finca</td><td> 2</td></tr></table></div>		_id	id	url	name	season	208978	208978	https://www.tv...	The Sword of Si...	1	670994d2bb499...	832100	https://www.tv...	Our Man in Ma...	2	670994d2bb499...	832101	https://www.tv...	The Good, the B...	2	670994d2bb499...	208981	https://www.tv...	There Will Be a F...	1	670994d2bb499...	832102	https://www.tv...	The Enemies of ...	2	670994d2bb499...	832103	https://www.tv...	Los Pepes	2	670994d2bb499...	832256	https://www.tv...	Deutschland 93	2	670994d2bb499...	832099	https://www.tv...	Cambalache	2	670994d2bb499...	832257	https://www.tv...	Exit El Patrón	2	670994d2bb499...	203469	https://www.tv...	Descenso	1	670994d2bb499...	832258	https://www.tv...	Nuestra Finca	2
_id	id	url	name	season																																																									
208978	208978	https://www.tv...	The Sword of Si...	1																																																									
670994d2bb499...	832100	https://www.tv...	Our Man in Ma...	2																																																									
670994d2bb499...	832101	https://www.tv...	The Good, the B...	2																																																									
670994d2bb499...	208981	https://www.tv...	There Will Be a F...	1																																																									
670994d2bb499...	832102	https://www.tv...	The Enemies of ...	2																																																									
670994d2bb499...	832103	https://www.tv...	Los Pepes	2																																																									
670994d2bb499...	832256	https://www.tv...	Deutschland 93	2																																																									
670994d2bb499...	832099	https://www.tv...	Cambalache	2																																																									
670994d2bb499...	832257	https://www.tv...	Exit El Patrón	2																																																									
670994d2bb499...	203469	https://www.tv...	Descenso	1																																																									
670994d2bb499...	832258	https://www.tv...	Nuestra Finca	2																																																									

<pre>db.Narcos.find({ rating: { \$exists: 1 } })</pre>	Buscamos los capítulos que tengan el campo rating																																																																																				
<div>Narcos > id</div> <table><tr><th></th><th>type</th><th>airdate</th><th>airtime</th><th>airstamp</th><th>runtime</th><th>rating</th></tr><tr><td></td><td> regular</td><td> 2015-08-28</td><td></td><td> 2015-08-28T12:...</td><td> 47</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2016-09-02</td><td></td><td> 2016-09-02T12:...</td><td> 47</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2016-09-02</td><td></td><td> 2016-09-02T12:...</td><td> 56</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2015-08-28</td><td></td><td> 2015-08-28T12:...</td><td> 55</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2016-09-02</td><td></td><td> 2016-09-02T12:...</td><td> 52</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2016-09-02</td><td></td><td> 2016-09-02T12:...</td><td> 54</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2016-09-02</td><td></td><td> 2016-09-02T12:...</td><td> 57</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2016-09-02</td><td></td><td> 2016-09-02T12:...</td><td> 47</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2016-09-02</td><td></td><td> 2016-09-02T12:...</td><td> 55</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2015-08-28</td><td></td><td> 2015-08-28T12:...</td><td> 57</td><td> (1 fields)</td></tr><tr><td></td><td> regular</td><td> 2016-09-02</td><td></td><td> 2016-09-02T12:...</td><td> 57</td><td> (1 fields)</td></tr></table>			type	airdate	airtime	airstamp	runtime	rating		regular	2015-08-28		2015-08-28T12:...	47	(1 fields)		regular	2016-09-02		2016-09-02T12:...	47	(1 fields)		regular	2016-09-02		2016-09-02T12:...	56	(1 fields)		regular	2015-08-28		2015-08-28T12:...	55	(1 fields)		regular	2016-09-02		2016-09-02T12:...	52	(1 fields)		regular	2016-09-02		2016-09-02T12:...	54	(1 fields)		regular	2016-09-02		2016-09-02T12:...	57	(1 fields)		regular	2016-09-02		2016-09-02T12:...	47	(1 fields)		regular	2016-09-02		2016-09-02T12:...	55	(1 fields)		regular	2015-08-28		2015-08-28T12:...	57	(1 fields)		regular	2016-09-02		2016-09-02T12:...	57	(1 fields)
	type	airdate	airtime	airstamp	runtime	rating																																																																															
	regular	2015-08-28		2015-08-28T12:...	47	(1 fields)																																																																															
	regular	2016-09-02		2016-09-02T12:...	47	(1 fields)																																																																															
	regular	2016-09-02		2016-09-02T12:...	56	(1 fields)																																																																															
	regular	2015-08-28		2015-08-28T12:...	55	(1 fields)																																																																															
	regular	2016-09-02		2016-09-02T12:...	52	(1 fields)																																																																															
	regular	2016-09-02		2016-09-02T12:...	54	(1 fields)																																																																															
	regular	2016-09-02		2016-09-02T12:...	57	(1 fields)																																																																															
	regular	2016-09-02		2016-09-02T12:...	47	(1 fields)																																																																															
	regular	2016-09-02		2016-09-02T12:...	55	(1 fields)																																																																															
	regular	2015-08-28		2015-08-28T12:...	57	(1 fields)																																																																															
	regular	2016-09-02		2016-09-02T12:...	57	(1 fields)																																																																															
<pre>db.Narcos.find({ rating: { \$exists: 1 }, rating: { \$type: "string" } })</pre>	Buscamos los capítulos que tengan el campo rating y solo nos dara el capitulo con rating tipo strung																																																																																				
Los campos que solicitan, no los tenemos, por ende no hay resultado de salida																																																																																					

CREATE

Consulta	Función
db.createCollection("LabItp01")	Creamos una nueva colección en nuestra base School
	
db.LabItp01.insert({ _id:1, name: "pepe", phone: 123456, class: [20, 22, 25] })	inserta un nuevo documento en la colección LabItp01
	
db.LabItp01.insertOne({ _id:2, name: "juanito", phone: 654789, class: [10, 12, 15] })	inserta un solo documento en la colección LabItp01
	
db.LabItp01.insertMany([{ _id:3, name: "carlito", phone: 639852, class: [11, 10] }, { _id:4, name: "camilito", phone: 741258, class: [15] },]	inserta varios documentos a la vez en la colección LabItp01. En este caso, insertamos 4 documentos (cada uno con sus propios datos).


```
{
  _id:5,
  name: "anita",
  phone: 852741,
  class: [ 10] },
{
  _id:5,
  name: "joselito",
  phone: 1254896,
  class: [ 55, 458, 236,20, 22, 10,
15]}
}]
)
```

LabItp01 > name

_id	name	phone	class
1	pepe	123456	[3 elements]
2	juanito	654789	[3 elements]
3	carlito	639852	[2 elements]
4	camilito	741258	[1 elements]
5	anita	852741	[1 elements]

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

busca todos los documentos en la colección LabItp01 donde el campo class sea igual a 10

LabItp01 > class > 0

{Document id}	0	1	2
2	10	12	15
3	11	10	
5	10		

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

agrega un nuevo documento, con nombre ‘carolita’

```
1  ▾{
2    "acknowledged" : true,
3    "insertedId" : ObjectId("670f8c44cdb955da42072f8a")
4  }
```

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

agrega un nuevo documento a la colección LabItp02

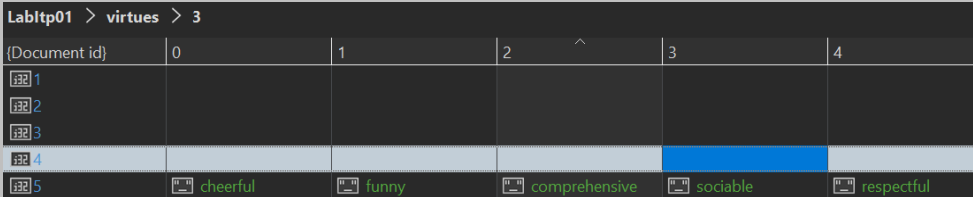
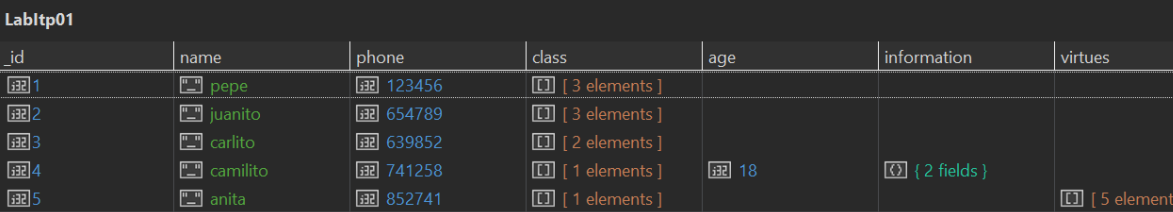
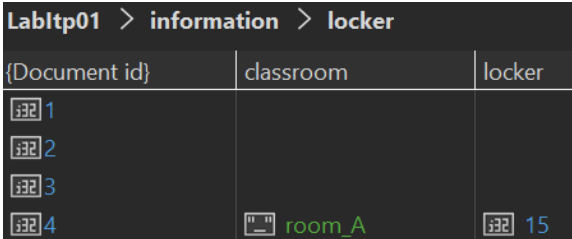
```
1  ▾{
2    "acknowledged" : true,
3    "insertedId" : ObjectId("670f8c82cdb955da42072f8b")
4  }
```

db.LabItp02.find()

Nos muestra el contenido de la coleccion

LabItp02 > age			
_id	name	information	age
670f8c44c2b955...	carolita		
670f8c82cdb955...	carolita	{ 2 fields }	25

UPDATE

Consulta	Función
<pre>db.LabItp01.updateOne({ _id: 5 }, { \$set: { virtues: ['cheerful', 'funny', 'comprehensive', 'sociable', 'respectful'] } })</pre>	<p>El estudiante de id 5, le vamos actualizar agregándole el campo virtues que tiene un array con 5 elementos</p>
	
<pre>db.LabItp01.updateOne({ _id: 4 }, { \$set: { information: { classroom:"room_A", locker: 15 }, age: 18 } })</pre>	<p>El estudiante de id 4, le vamos actualizar agregándole el campo age e information que tiene un array con 2 elementos</p>
	
	
<pre>db.LabItp01.updateOne({ _id: 3 }, { \$set: { virtues: ['cheerful', 'funny','comprehensive', 'sociable', 'respectful']}, lastModified: new Date() })</pre>	<p>El estudiante de id 3, le vamos actualizar agregándole el campo virtues que tiene un array con 5 elementos y además agregamos otro campo llamado lastModified donde guardaremos la fecha en la que</p>

<pre>\$currentDate: { lastModified: true } })</pre>	hacemos el update por medio de \$currentDate
-------------------------------------------------------	------------------------------------------------------------

LabItp01

_id	name	phone	class	lastModified	virtues
1	pepe	123456	[3 elements]		
2	juanito	654789	[3 elements]		
3	carlito	639852	[2 elements]	2024-10-16T11:...	[5 elements]
4	camilito	741258	[1 elements]		
5	anita	852741	[1 elements]		[5 elements]

```
db.LabItp01.updateOne(
  { _id: 2 }, {
    $set: {
      information: {
        classroom: "room_A", locker: 15 }, age:
      18 },
    $currentDate: { lastModified: true }
  })
```

El estudiante de id 2, le vamos actualizar agregándole el campo **age**, **information** que tiene un **array** con 2 elementos y además agregamos otro campo llamado **lastModified** donde guardaremos la fecha en la que hacemos el **update** por medio de **\$currentDate**

LabItp01

_id	name	phone	class	age	information	lastModified
1	pepe	123456	[3 elements]			
2	juanito	654789	[3 elements]	18	[2 fields]	2024-10-16T11:...
3	carlito	639852	[2 elements]			2024-10-16T11:...
4	camilito	741258	[1 elements]	18	[2 fields]	
5	anita	852741	[1 elements]			

```
db.LabItp01.updateOne(
  { _id: 1 }, {
    $set: { name: "Joan", age: 19,
      virtues:[], information: {} }, $currentDate:
    {
      lastModified: true }},
  { upsert: true } )
```

El estudiante de id 1, le vamos actualizar el nombre de **pepe** a **Joan**, agregándole el campo **age**, tambien el campo **virtues** que tiene un **array** con 0 elementos, tambien el campo **information** que tiene un **array** con 0 elementos, agregamos otro campo llamado **lastModified** donde guardaremos la fecha en la que hacemos el **update** por medio de **\$currentDate** y si el user con id 1 **NO** existe, entonces **upsert:true** crea un nuevo elemento a la coleccion

LabItp01

_id	name	phone	class	age	information	lastModified	virtues
1	Joan	123456	[3 elements]	19	[0 fields]	2024-10-16T11:...	[0 elements]
4	camilito	741258	[1 elements]	18	[2 fields]		
2	juanito	654789	[3 elements]	18	[2 fields]	2024-10-16T11:...	
5	anita	852741	[1 elements]				[5 elements]
3	carlito	639852	[2 elements]			2024-10-16T11:...	[5 elements]

```
db.LabItp01.updateMany(
  { _id: { $gte: 1, $lte: 6 } },
```

Actualiza los documentos con **_id** 1 – 6 y agrega el campo **virtues** con un

<pre>{ \$set: { virtues: ['cheerful'] } })</pre>	array que contenga un único valor, el que decidas de la lista siguiente: ['cheerful', 'funny', 'comprehensive', 'sociable', 'respectful'].
---------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------

LabItp01							
id	name	phone	class	age	information	lastModified	virtues
1	Joan	123456	[3 elements]	19	{ 0 fields }	2024-10-16T11:...	[1 element]
2	juanito	654789	[3 elements]	18	{ 2 fields }	2024-10-16T11:...	[1 element]
4	camilito	741258	[1 elements]	18	{ 2 fields }		[1 element]
5	anita	852741	[1 elements]				[1 element]

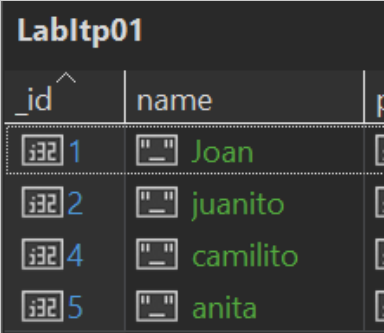
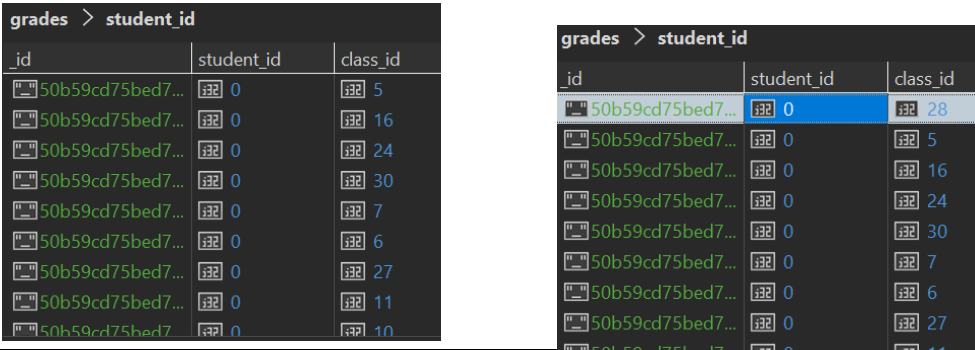
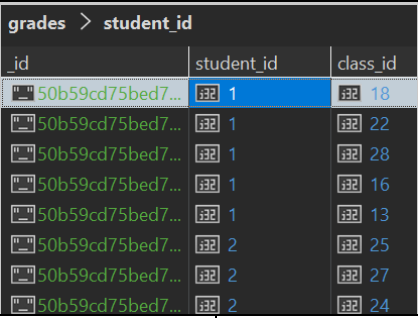
<pre>db.LabItp01.updateMany({}, { \$set: { status: 'A' } })</pre>	Actualiza todos los documentos con una única instrucción y agrega el siguiente campo: status: 'A'.
------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------

LabItp01								
id	name	phone	class	age	information	lastModified	virtues	status
1	Joan	123456	[3 elements]	19	{ 0 fields }	2024-10-16T11:...	[1 elements]	
2	juanito	654789	[3 elements]	18	{ 2 fields }	2024-10-16T11:...	[1 elements]	
4	camilito	741258	[1 elements]	18	{ 2 fields }		[1 elements]	
5	anita	852741	[1 elements]				[1 elements]	

<pre>db.LabItp01.updateMany({ name: { \$in: ["pepe", "camilito"] } }, { \$set: { role: 'student' } })</pre>	Actualiza los documentos de “pepe” y “camilito” y agrega el siguiente campo: role: 'student'.
------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------

LabItp01									
id	name	phone	class	age	information	lastModified	virtues	status	role
1	Joan	123456	[3 elements]	19	{ 0 fields }	2024-10-16T11:...	[1 elements]	A	
2	juanito	654789	[3 elements]	18	{ 2 fields }	2024-10-16T11:...	[1 elements]	A	
4	camilito	741258	[1 elements]	18	{ 2 fields }		[1 elements]	A	student
5	anita	852741	[1 elements]				[1 elements]	A	

DELETE

Consulta	Función
<code>db.LabItp01.deleteOne({ name: "carlito" })</code>	Vamos a eliminar el elemento con name carlito
	
<code>db.grades.deleteOne({ student_id: 0 })</code>	Elimina el primer documento que tenga el student_id igual a 0.
	
<code>db.grades.deleteMany({ student_id: 0 })</code>	Elimina todos los elementos que tengan student_id: 0 .
	
<code>db.grades.remove({ student_id: 1 }, { justOne: true })</code>	Va a eliminar elementos con student_id de 1, pero la función remove por defecto elimina todos los documentos que coinciden con el filtro. Si se usa justOne: true , MongoDB intentará eliminar solo el primer documento que coincida con el filtro

<table><tr><th>_id</th><th>student_id</th></tr><tr><td>"50b59cd75bed7..."</td><td>1</td></tr><tr><td>"50b59cd75bed7..."</td><td>1</td></tr><tr><td>"50b59cd75bed7..."</td><td>1</td></tr><tr><td>"50b59cd75bed7..."</td><td>1</td></tr><tr><td>"50b59cd75bed7..."</td><td>2</td></tr></table>		_id	student_id	"50b59cd75bed7..."	1	"50b59cd75bed7..."	1	"50b59cd75bed7..."	1	"50b59cd75bed7..."	1	"50b59cd75bed7..."	2												
_id	student_id																								
"50b59cd75bed7..."	1																								
"50b59cd75bed7..."	1																								
"50b59cd75bed7..."	1																								
"50b59cd75bed7..."	1																								
"50b59cd75bed7..."	2																								
db.grades.remove({ student_id: 1 })	Elimina todos los elementos que tengan student_id = 1																								
<table><tr><th colspan="2">grades > student_id</th></tr><tr><th>_id</th><th>student_id</th></tr><tr><td>"50b59cd75bed7..."</td><td>2</td></tr><tr><td>"50b59cd75bed7..."</td><td>2</td></tr><tr><td>"50b59cd75bed7..."</td><td>2</td></tr><tr><td>"50b59cd75bed7..."</td><td>3</td></tr><tr><td>"50b59cd75bed7..."</td><td>3</td></tr></table>		grades > student_id		_id	student_id	"50b59cd75bed7..."	2	"50b59cd75bed7..."	2	"50b59cd75bed7..."	2	"50b59cd75bed7..."	3	"50b59cd75bed7..."	3										
grades > student_id																									
_id	student_id																								
"50b59cd75bed7..."	2																								
"50b59cd75bed7..."	2																								
"50b59cd75bed7..."	2																								
"50b59cd75bed7..."	3																								
"50b59cd75bed7..."	3																								
db.grades.remove({ })	Vacía la colección grades																								
<table><tr><th>grades</th></tr><tr><td></td></tr><tr><td></td></tr></table>		grades																							
grades																									
db.grades.drop()	Elimina la colección de la bd																								
<table><tr><td>✓</td><td>Database icon</td><td>School</td></tr><tr><td>✓</td><td>Folder icon</td><td>Collections (2)</td></tr><tr><td>></td><td>Document icon</td><td>Labltp01</td></tr><tr><td>></td><td>Document icon</td><td>Labltp02</td></tr><tr><td></td><td>Folder icon</td><td>Views (0)</td></tr><tr><td></td><td>Folder icon</td><td>GridFS Buckets (0)</td></tr><tr><td></td><td>Folder icon</td><td>System (0)</td></tr><tr><td>✓</td><td>Database icon</td><td>admin</td></tr></table>		✓	Database icon	School	✓	Folder icon	Collections (2)	>	Document icon	Labltp01	>	Document icon	Labltp02		Folder icon	Views (0)		Folder icon	GridFS Buckets (0)		Folder icon	System (0)	✓	Database icon	admin
✓	Database icon	School																							
✓	Folder icon	Collections (2)																							
>	Document icon	Labltp01																							
>	Document icon	Labltp02																							
	Folder icon	Views (0)																							
	Folder icon	GridFS Buckets (0)																							
	Folder icon	System (0)																							
✓	Database icon	admin																							

Análisis y Discusión

Al correr las consultas en la base de datos, pudimos notar varias cosas. Primero que nada, la estructura de la base de datos está bien pensada y los datos que hemos agregado se reflejan correctamente en las consultas, demostrando que podemos trabajar con dichos datos sin problemas. Cada campo que hemos actualizado o agregado aparece correctamente, como las virtudes, las clases, y los detalles de la información de cada estudiante. Esto está relacionado con el objetivo de realizar consultas en MongoDB para obtener información relevante de las bases de datos. También nos permitió ver en acción cómo las consultas funcionan sobre estructuras de datos bien diseñadas.

Además, el diseño de la base de datos y su organización en colecciones y documentos nos permitió comprender mejor cómo gestionar relaciones dentro de una base de datos NoSQL, cumpliendo el segundo objetivo del informe. Al aplicar métodos como `updateOne()` y `upsert`, vimos claramente cómo actualizar documentos y añadir nuevos campos, lo cual no solo mejoró la base de datos, sino que también reforzó el entendimiento sobre el manejo de MongoDB en la manipulación de datos.

Conclusión

Para concluir, el desarrollo de la base de datos personal en MongoDB y la ejecución de consultas nos permitió aplicar y aprender conceptos fundamentales sobre bases de datos NoSQL. Al diseñar la base de datos, conseguimos organizar los datos de manera eficiente, lo que facilitó tanto la consulta de información como su actualización. Cada consulta realizada nos ayudó a confirmar que el diseño alcanzaba los objetivos propuestos y nos permitió gestionar información compleja de forma clara y sencilla.

Referencia

- <https://www.mongodb.com/docs/manual/>
- <https://jsongrid.com/>
- <https://www.mongodb.com/docs/manual/reference/operator/query/>