



Bootcamp

Desarrollo Web Full Stack

Nivel Avanzado
Mg. Inés María Oliveros Hernández
Sesión 16 - 05 de Junio de 2024

UT TALENTOTECH

Tabla de contenidos

1

Base de Datos MongoDB

2

Estructura, tipos de datos BSON

3

Lenguaje MQL, funciones CRUD

4

Ejercicios en MongoDB Compass y Consola

Objetivos



1.

Comprender la estructura y tipos de datos BSON para una correcta modelización y rendimiento en MongoDB.



2.

Dominar MQL y las funciones CRUD para manipular datos eficientemente en MongoDB en consola de comandos de comandos



3.

Realizar ejercicios en MongoDB Compass para la gestión de bases de datos MongoDB mediante la interfaz gráfica y consola.

Base de Datos No Relacional MongoDB



mongoDB



TIC

TALENTO
TECH





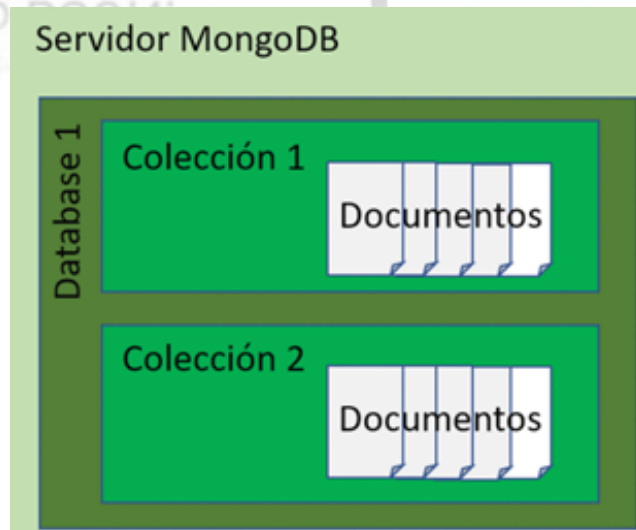
Qué es MongoDB

MongoDB es una base de datos NoSQL orientada a documentos. Almacena datos en un tipo de formato JSON llamado BSON.

Origen de MongoDB

Fundada el año **2007** por Dwight Merriman, Eliot Horowitz y Kevin Ryan, el equipo de la empresa DoubleClick.

La empresa tenía problemas de escalabilidad y agilidad en un producto que publicaba **400.000 anuncios por segundo**, MongoDB se diseñó para brindar agilidad y rapidez para procesos de Big data.



Estructura y tipos de datos BSON

```
{  
  name: "al",  
  age: 18,  
  status: "D",  
  groups: [ "politics", "news" ]  
}
```

Collection

¿Cómo se almacenan los datos?

```
{  
  field1: value1,  
  field2: value2,  
  field3: value3,  
  ...  
  fieldN: valueN  
}
```

```
{  
  "student": {  
    "name": "John",  
    "class": "Intermediate",  
    "address": {  
      "street": "2293 Example Street",  
      "City": "Chicago",  
      "State": "IL"  
    }  
  }  
}
```

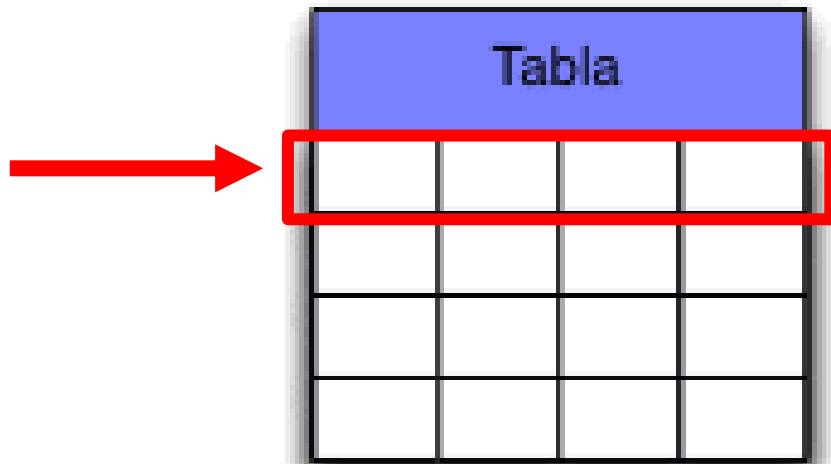
```
{  
  "nombre": "María Gómez",  
  "edad": 18,  
  "carrera": "Ciencias de la Computación"  
}
```

```
{  
  "nombre": "Pedro Rodríguez",  
  "edad": 22,  
  "carrera": "Ingeniería Informática",  
  "materias": ["Matemáticas", "Física", "Programación"]  
}
```

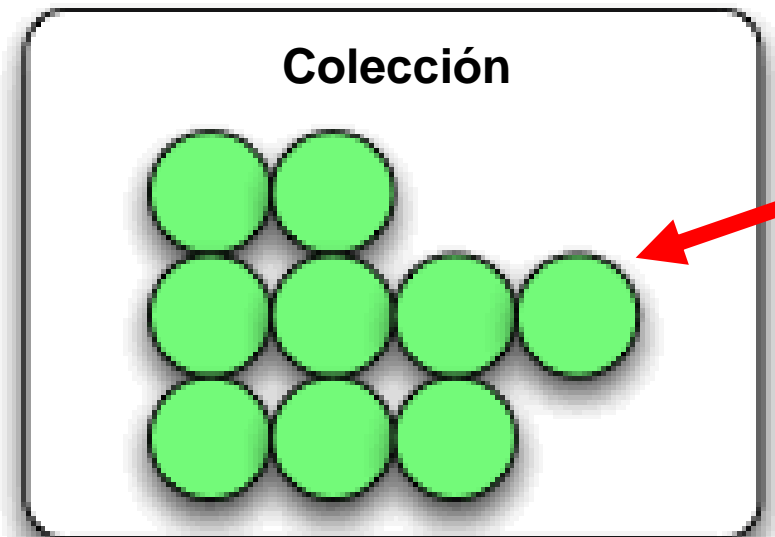
MongoDB almacena datos en **documentos flexibles** JSON/BSON.

Estructura de gestión de la información

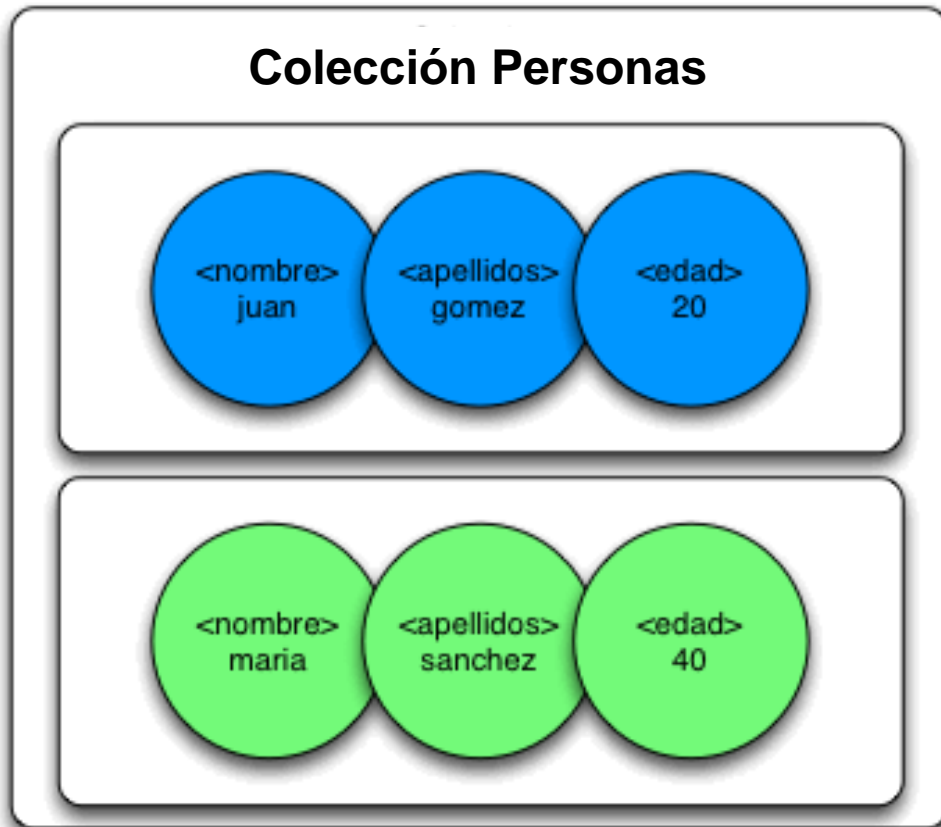
Las tablas en SQL almacenan
registros (filas)



Las colecciones almacenan
documentos



¿Qué son Documentos?



Son **almacenados en formato BSON** (Binary JSON) que es una representación binaria de mapas basados en JSON.

Los documentos están compuestos por pares de **clave : valor** y **cada documento puede tener variaciones importantes con el documento anterior**, puede tener más de una clave valor o nombres distintos.

BSON: Binary JSON, es un formato de intercambio de datos binario utilizado principalmente por la base de datos NoSQL MongoDB. Se basa en la estructura de JSON (JavaScript Object Notation), pero lo representa de manera binaria para optimizar el almacenamiento, la transferencia y la recuperación de datos.

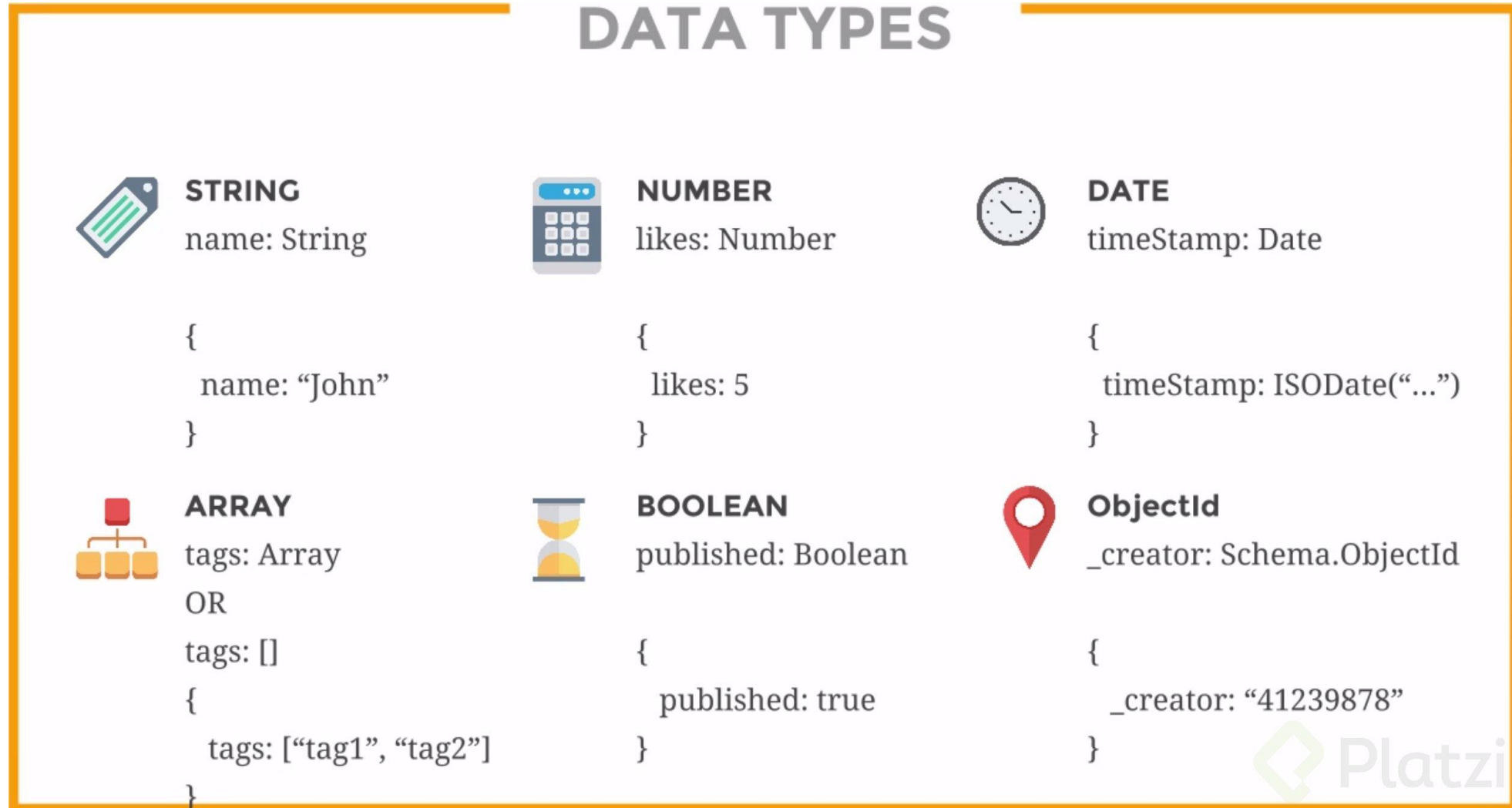
Los tipos de datos soportados por MongoDB son:

- ❖ **String:** cualquier cadena de caracteres codificada en utf-8. Debe ir entre comillas dobles.
- ❖ **Number:** valores numéricos formados por dígitos (con o sin punto decimal). No debe ir entre comillas dobles.
- ❖ **Boolean:** valor lógico (true o false). Sin comillas.
- ❖ **Timestamp:** un dato de 64bits que representa una fecha con hora.
- ❖ **Date:** un dato de 32bits que representa una fecha.
- ❖ **null:** se refiere a un valor nulo o vacío.
- ❖ **undefined:** indica un dato que no ha sido definido.
- ❖ **Array:** agrupación de elementos a manera de lista indizada a partir de la posición 0. En MongoDB se pueden usar los métodos de iteración de arreglos disponibles en JS.

MongoDB está basado en un tipo de estructura de datos compleja llamada JSON, que al ser almacenada en el servidor es convertida en otro formato similar pero binario llamado BSON.

La estructura (o esquema) de los datos en MongoDB **no necesita estar definida en algún lugar**, sino que justo al momento de insertar un dato o grupo de datos, Mongo infiere dinámicamente la estructura a partir del formato de cada uno de los valores introducidos, lo único requerido es que los datos se correspondan con un objeto JSON correctamente formado.

Tipos de Datos BSON



Lenguaje MQL (MongoDB Query Lenguaje) - CRUD

C R U D

CREATE

READ

UPDATE

DELETE



MongoDB Query Lenguaje (MQL)

MQL es el lenguaje de consulta y manipulación de información **nativo** de la base de datos **NoSQL MongoDB**. Permite a los usuarios interactuar con la base de datos, realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) y recuperar información específica de los documentos almacenados.

Características principales de MQL:

- **Sintaxis similar a JSON:** MQL utiliza una sintaxis similar a JSON, lo que facilita su aprendizaje y uso para desarrolladores familiarizados con JSON.
- **Consultas flexibles:** Permite realizar consultas simples y complejas para filtrar, ordenar y agrupar datos.
- **Soporte para operadores y funciones:** Incluye una amplia gama de operadores y funciones para manipular y procesar datos.
- **Integración con agregaciones:** Permite realizar operaciones de agregación de datos, como cálculos estadísticos y transformaciones de datos.

CRUD

Ejemplo de **crear** un **documento**: Crea un nuevo documento en la colección '**users**' con el nombre '**sue**', la edad de 26 años y el estado 'pending'.

(a) Create

```
db.users.insertOne(  ← collection
{
  name: "sue",        ← field: value
  age: 26,             ← field: value
  status: "pending"   ← field: value
}                    } document
)
```

- **db.users.insertOne()**: comando utilizado para insertar un nuevo documento en la colección 'users'.
- El documento tiene tres campos: **name**, **age** y **status**.
- **name: "sue"**: campo **name** del documento y su valor es "sue".
- **age: 26**: campo **age** del documento y su valor es 26.
- **status: "pending"**: campo **status** del documento y su valor es "pending".

Ejemplo de leer un documento: Encuentra los nombres y direcciones de los usuarios mayores de 18 años en la colección 'users', y muestra los primeros 5 resultados.

(b) Read

```
db.users.find(  
  { age: { $gt: 18 } },  
  { name: 1, address: 1 }  
) .limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

- **db.users.find():** comando principal para buscar documentos en la colección "users".
- **{ age: { \$gt: 18 } }:** es el filtro de consulta. Indica que solo se deben buscar documentos donde el campo "age" sea mayor que 18. \$gt es un operador de comparación que significa "mayor que".
- **{ name: 1, address: 1 }:** especifica qué campos se deben incluir en los resultados de la búsqueda. Aquí, "name" y "address" se establecen en 1, lo que significa que se incluirán en los resultados. Si en lugar de 1 hubiera un 0, significaría que esos campos no se incluirían.
- **.limit(5):** el número de documentos devueltos por la consulta a 5, solo se devolverán los primeros 5 documentos que cumplan con los criterios de la consulta.

CRUD

Ejemplo de **actualizar documentos**: Actualiza todos los documentos en la colección 'users' donde la edad es mayor o igual a 18, estableciendo el campo 'status' en 'reject'.

(c) Update

```
db.users.updateMany(  
  { age: { $lt: 18 } },  
  { $set: { status: "reject" } }  
)
```

← collection
← update filter
← update action

- **db.users.updateMany()**: comando utilizado para actualizar múltiples documentos en la colección 'users'.
- El primer argumento de **updateMany()** es un objeto de filtro que especifica qué documentos se deben actualizar. **{age: { \$gte: 18 } }** significa que se actualizarán todos los documentos donde el campo 'age' sea mayor o igual a 18.
- El segundo argumento de **updateMany()** es un objeto de actualización que especifica qué cambios se deben realizar en los documentos que cumplen con el filtro. **{ \$set: { status: "reject" } }** indica que se debe establecer el campo 'status' en 'reject' para estos documentos, el estado de estos documentos será cambiado a "rechazado" o "denegado".

CRUD

Ejemplo de **borrar documentos**: Elimina todos los documentos en la colección 'users' donde el campo 'status' es igual a 'reject'."

(d) Delete

```
db.users.deleteMany(  
  { status: "reject" }  
)
```

← collection
← delete filter

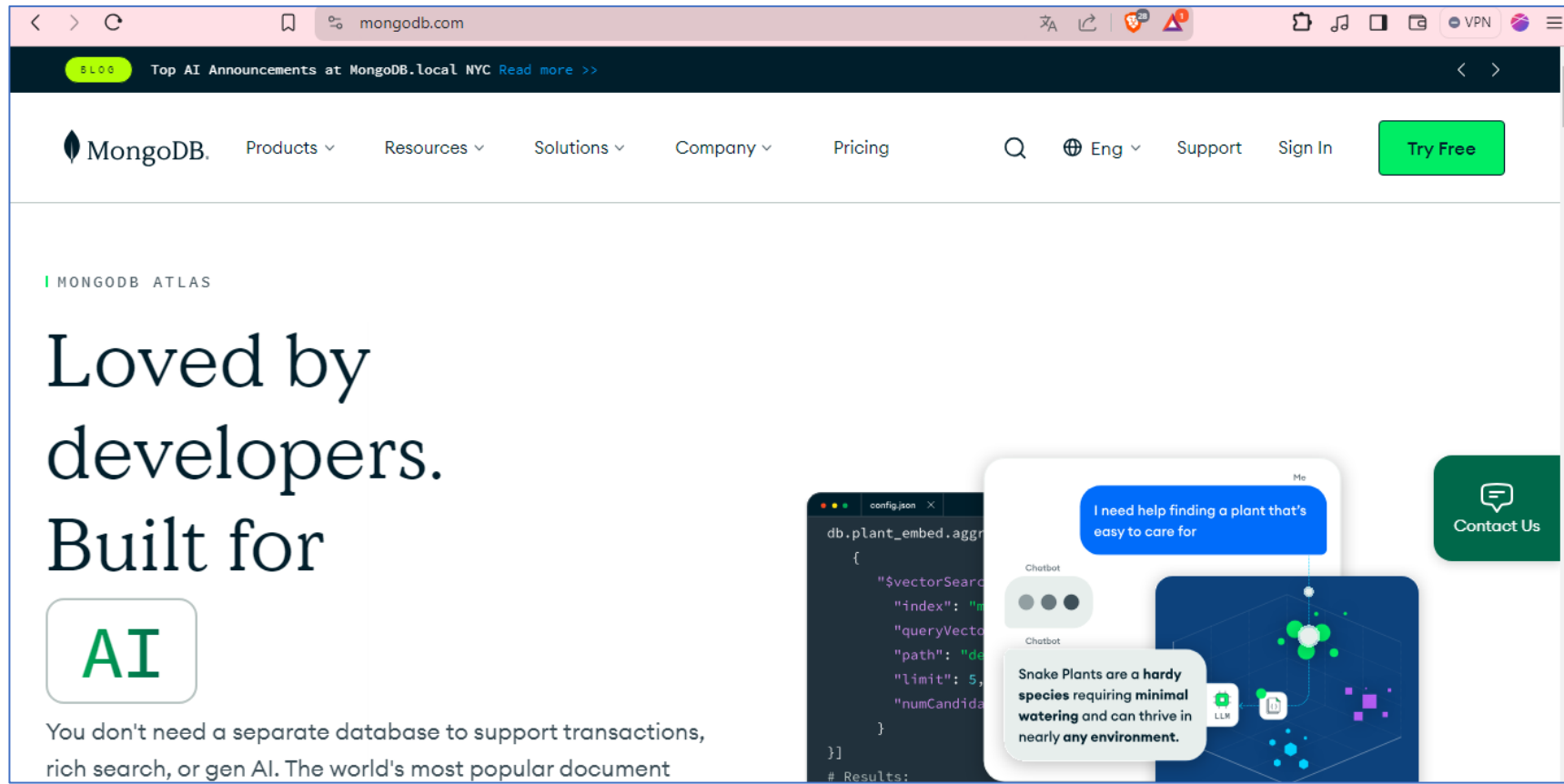
- **db.users.deleteMany()**: comando utilizado para eliminar múltiples documentos en la colección 'users'.
- El argumento de **deleteMany()** es un objeto de filtro que especifica qué documentos se deben eliminar.
- **{ status: "reject" }** indica que se **eliminarán todos** los documentos donde el campo 'status' sea igual a 'reject'.

Taller MongoDB Compass

- Descarga de MongoDB y Compass
- Configuración
- Comandos comunes para trabajar con **colecciones** en una consola de comandos de comandos.
- Ejercicios en Compass

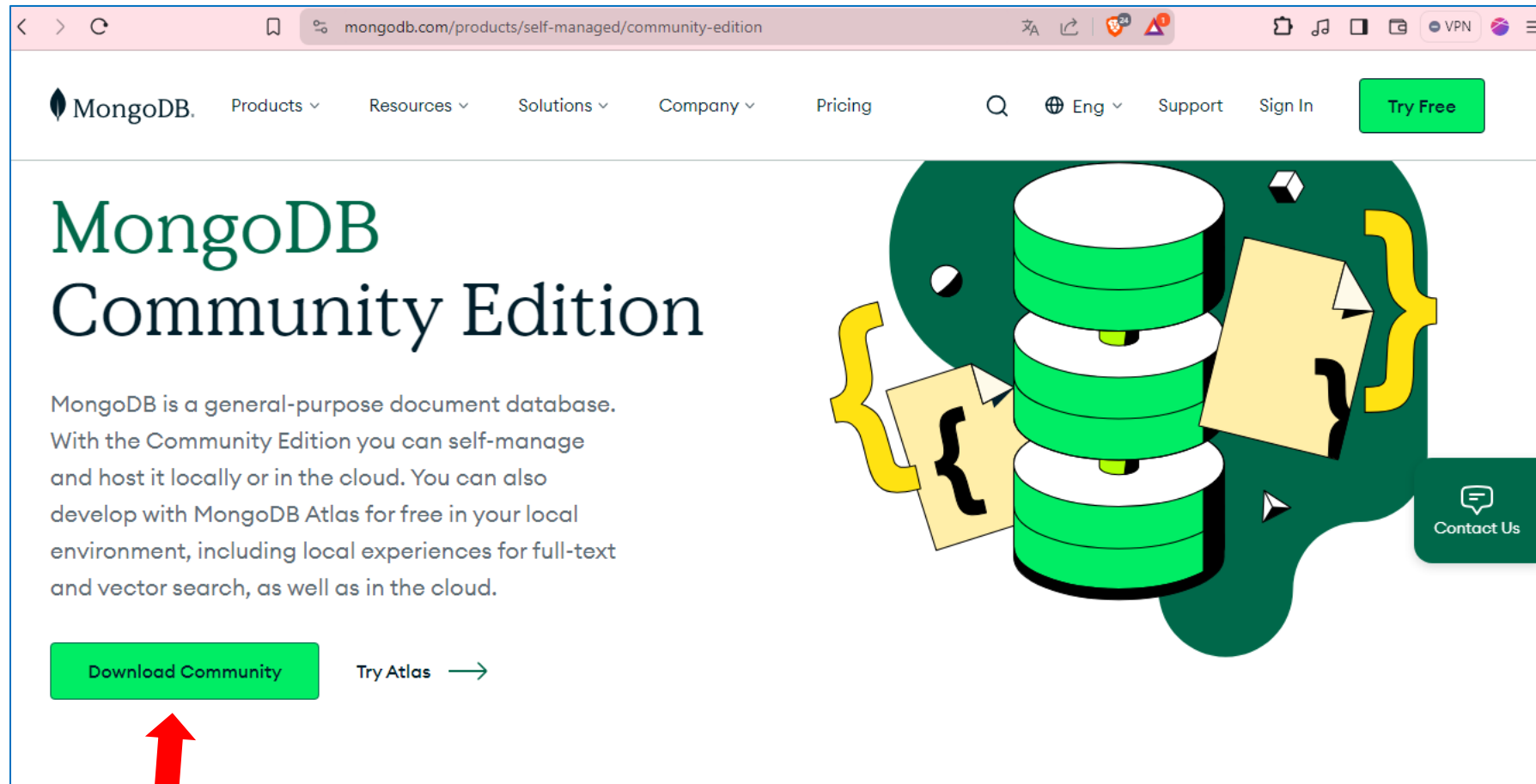
Descarga de MongoDB y Compass

www.mongodb.com



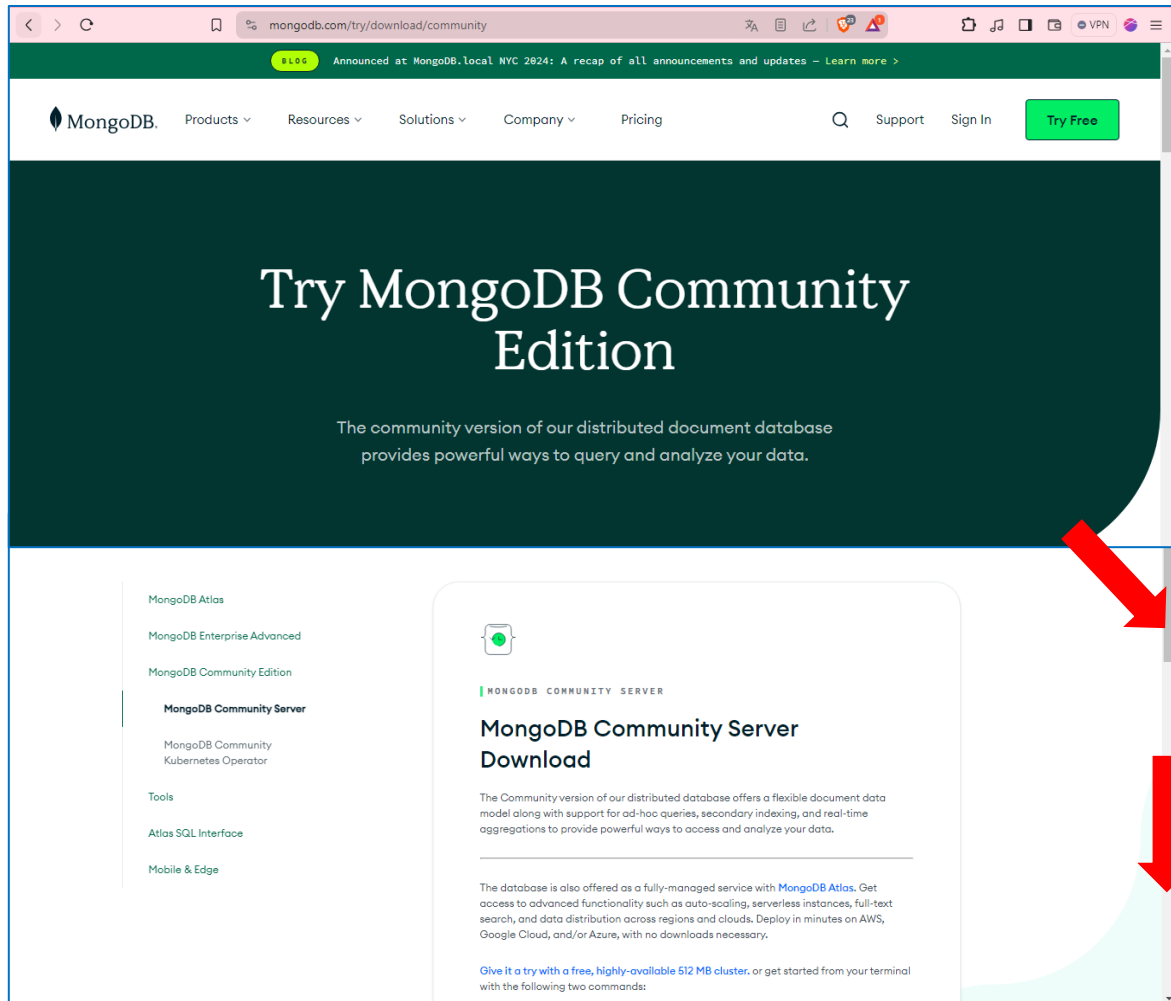
Descarga de MongoDB y Compass

The screenshot shows the MongoDB website (mongodb.com) with a red arrow pointing to the 'Products' menu in the navigation bar. The 'Products' dropdown menu is open, displaying various categories: PLATFORM (Atlas), PLATFORM SERVICES (Database, Search, Vector Search, Stream Processing), TOOLS (Compass, Integrations, Relational Migrator), and SELF MANAGED (Enterprise Advanced, Community Edition). A second red arrow points to the 'Community Edition' link under the SELF MANAGED section, which includes the text 'Develop locally with MongoDB'. To the right of the dropdown, there are sections for 'Build with MongoDB Atlas', 'Test Enterprise Advanced', and 'Try Community Edition', each with a 'Sign Up' or 'Download' button. The bottom of the page features a quote: 'You don't need a separate database to support transactions, rich search, or gen AI. The world's most popular document'.

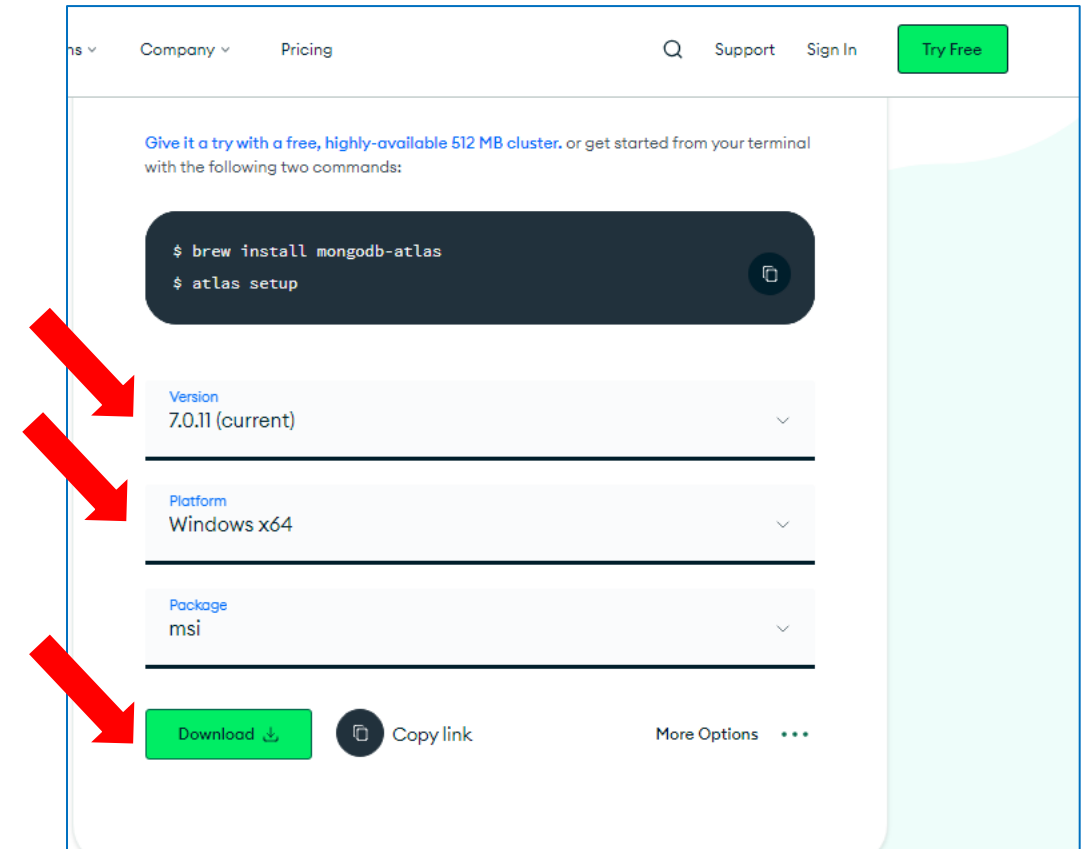


The screenshot shows the MongoDB website's 'Community Edition' page. The browser address bar displays 'mongodb.com/products/self-managed/community-edition'. The navigation bar includes links for Products, Resources, Solutions, Company, Pricing, a search icon, language settings (Eng), Support, Sign In, and a 'Try Free' button. The main heading is 'MongoDB Community Edition'. Below it, a paragraph states: 'MongoDB is a general-purpose document database. With the Community Edition you can self-manage and host it locally or in the cloud. You can also develop with MongoDB Atlas for free in your local environment, including local experiences for full-text and vector search, as well as in the cloud.' To the right of the text is a graphic of a green database cylinder with yellow curly braces and a yellow document icon. At the bottom left, there is a green 'Download Community' button and a 'Try Atlas' link with a right-pointing arrow. A red arrow points to the 'Download Community' button. A 'Contact Us' button is visible in the bottom right corner of the page content.

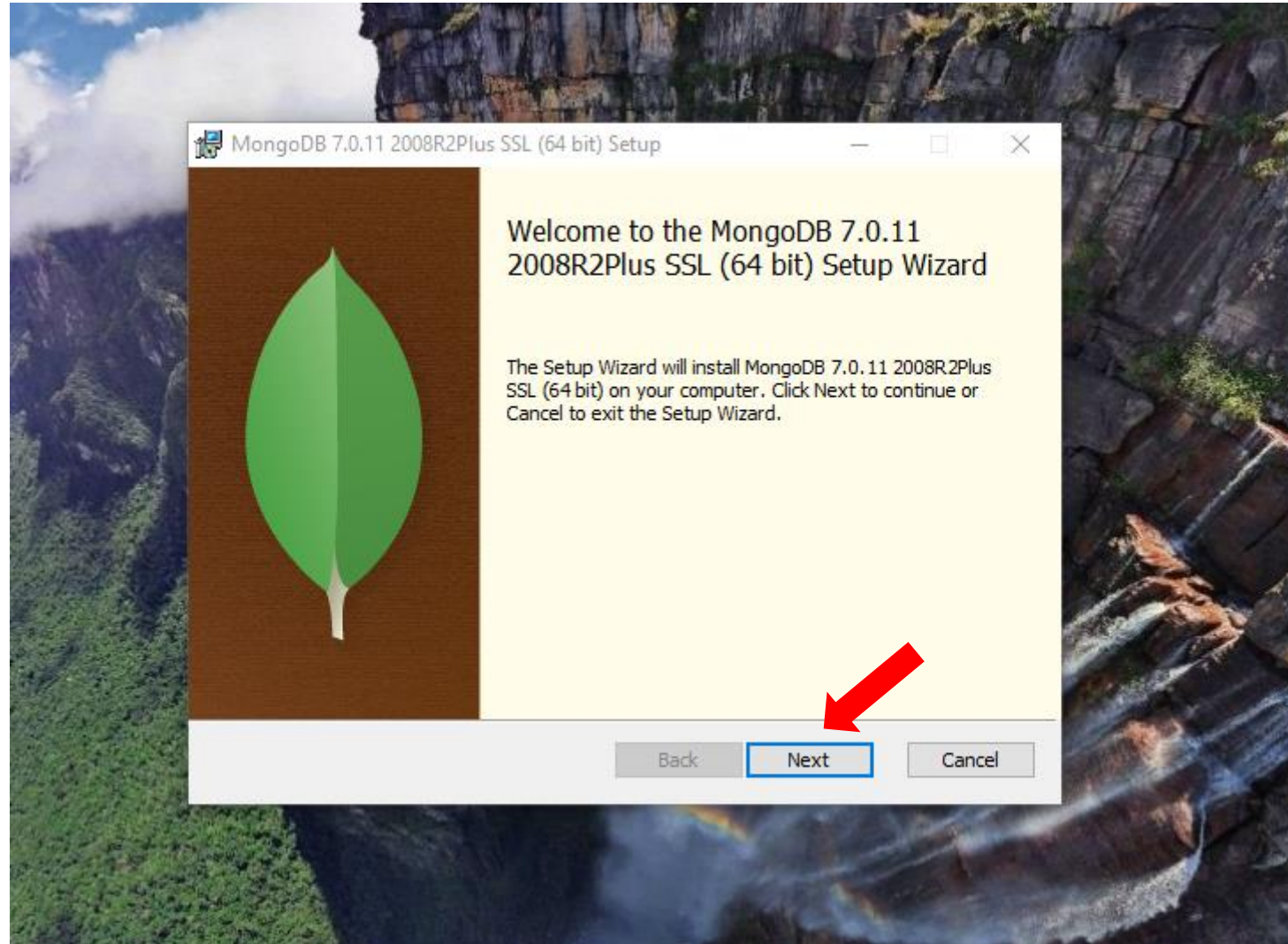
Descarga de MongoDB y Compass

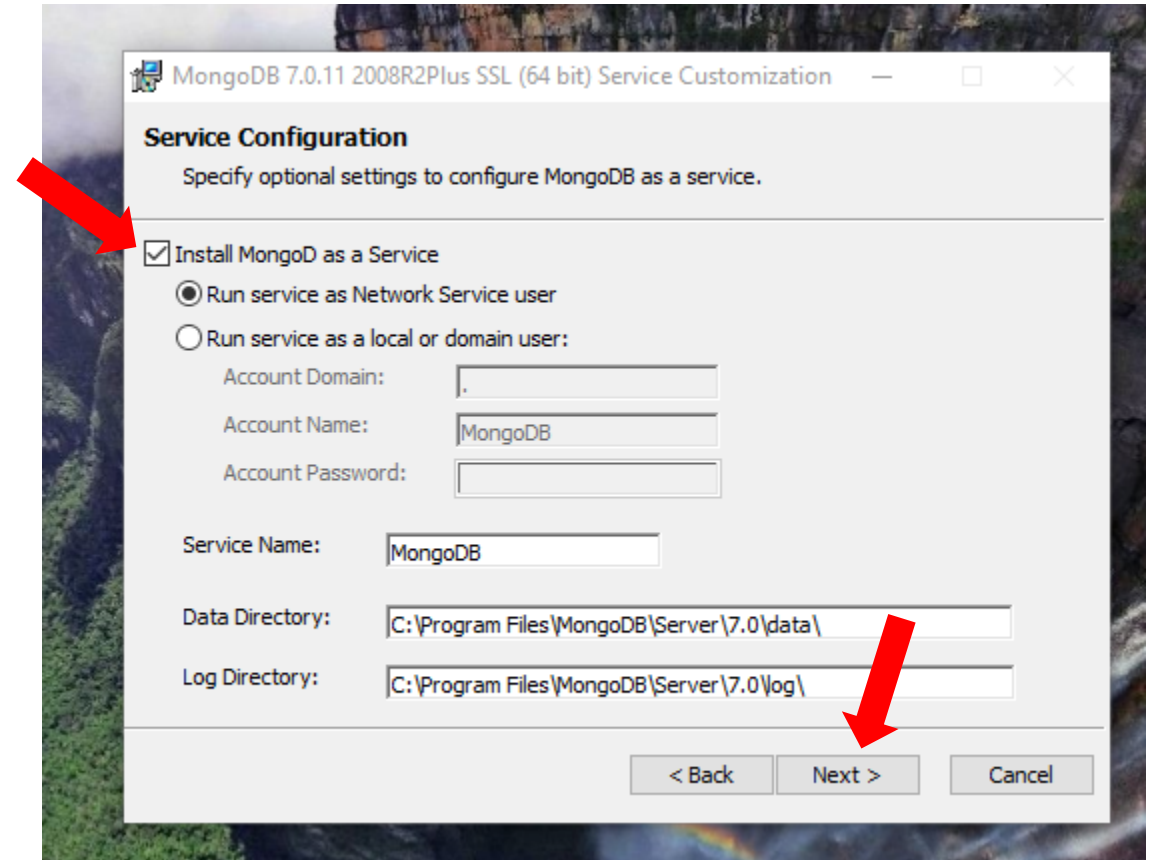
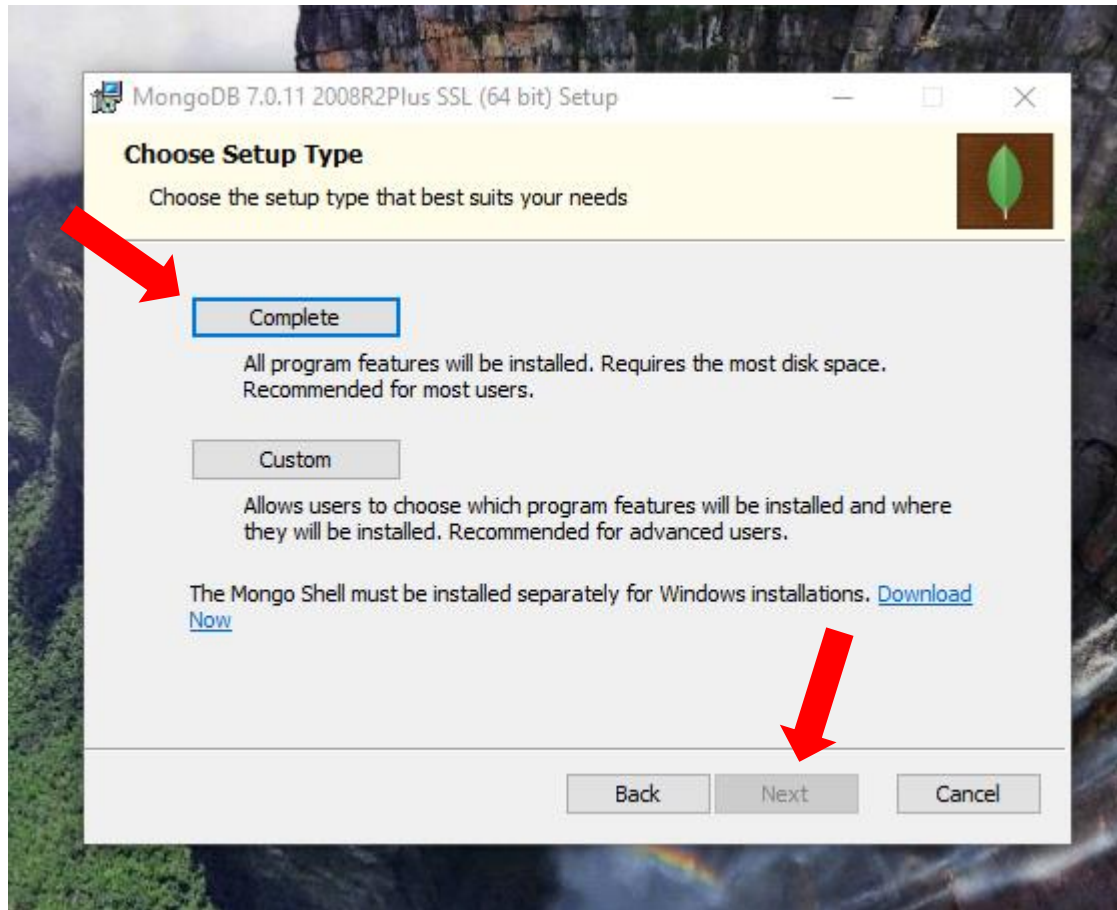


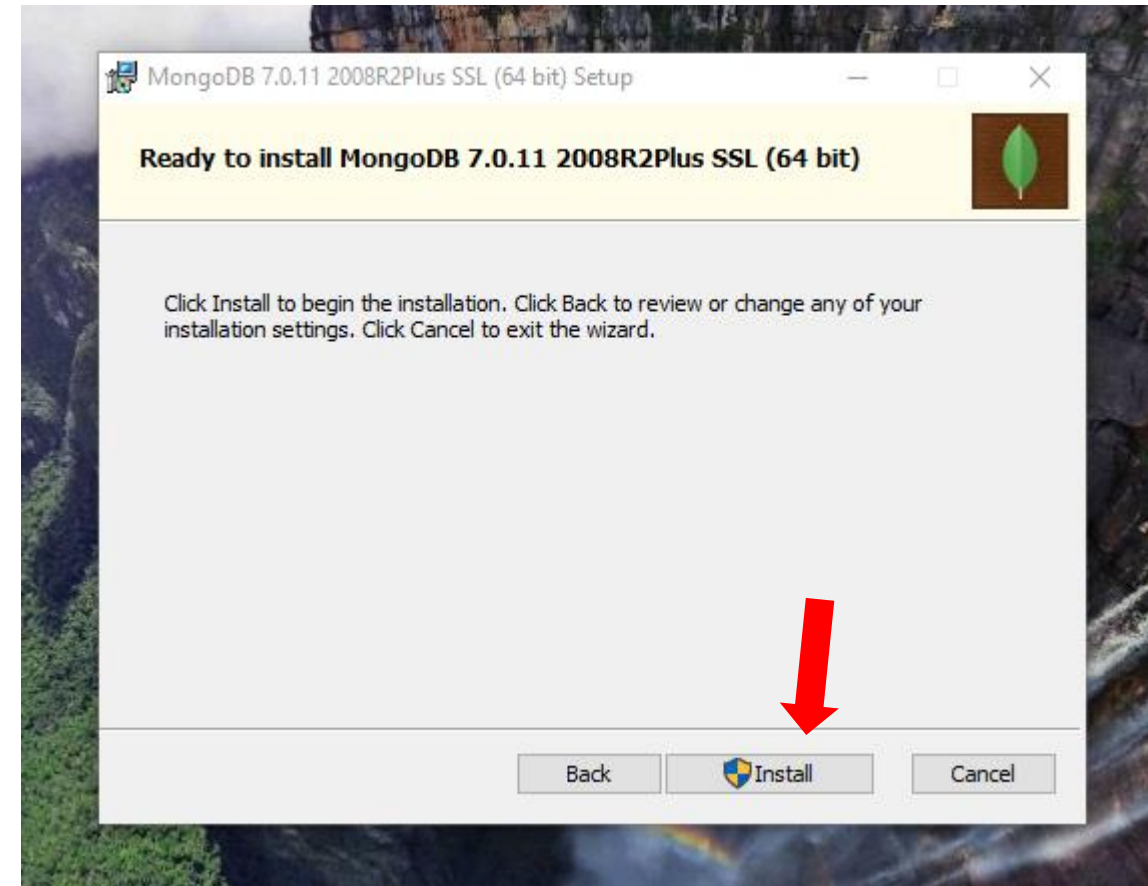
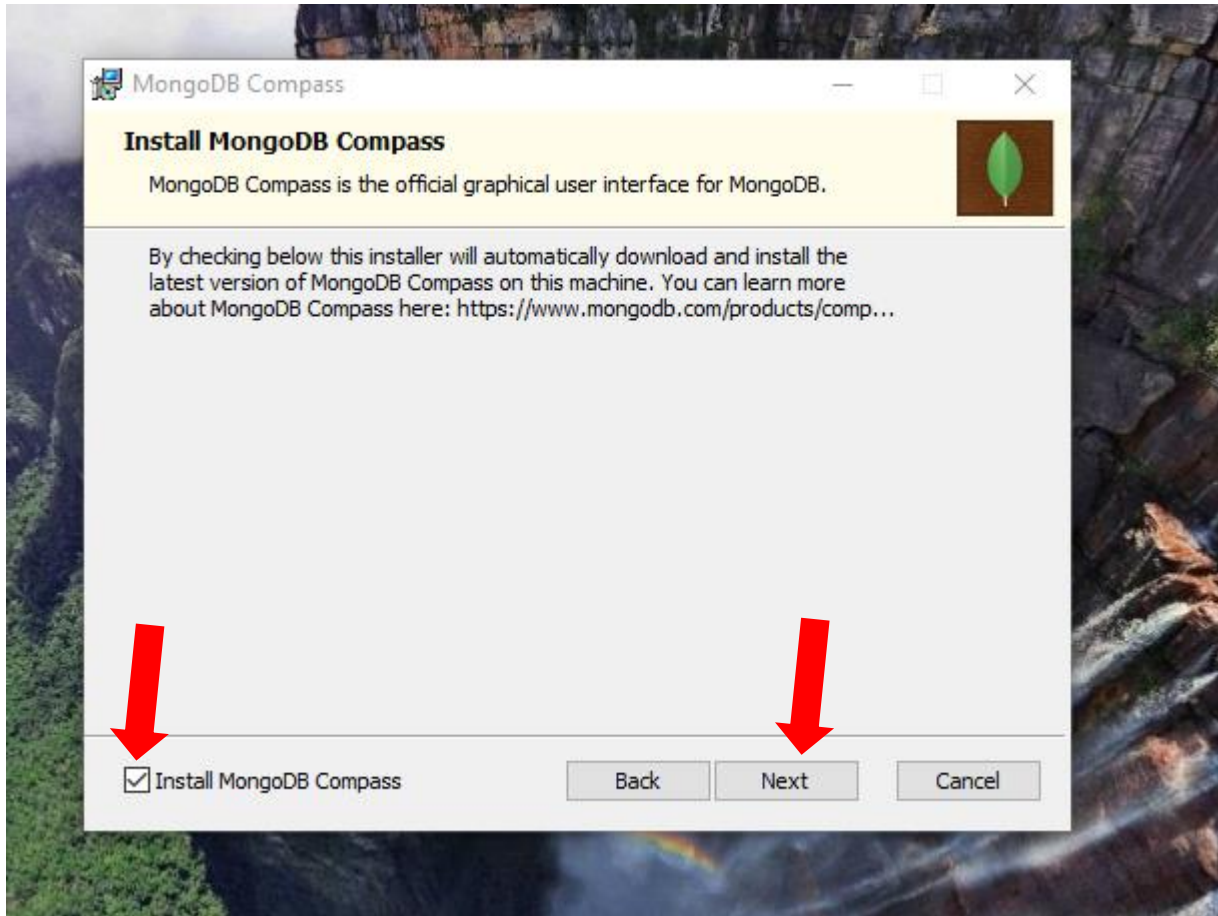
The screenshot shows the MongoDB website's 'Try MongoDB Community Edition' page. The main heading is 'Try MongoDB Community Edition' with the subtext 'The community version of our distributed document database provides powerful ways to query and analyze your data.' Below this, there's a sidebar with links to various MongoDB products and a main section titled 'MongoDB Community Server Download'. This section includes a description of the community version and a 'Download' button. Red arrows point from the 'Download' button to the right-hand screenshot.

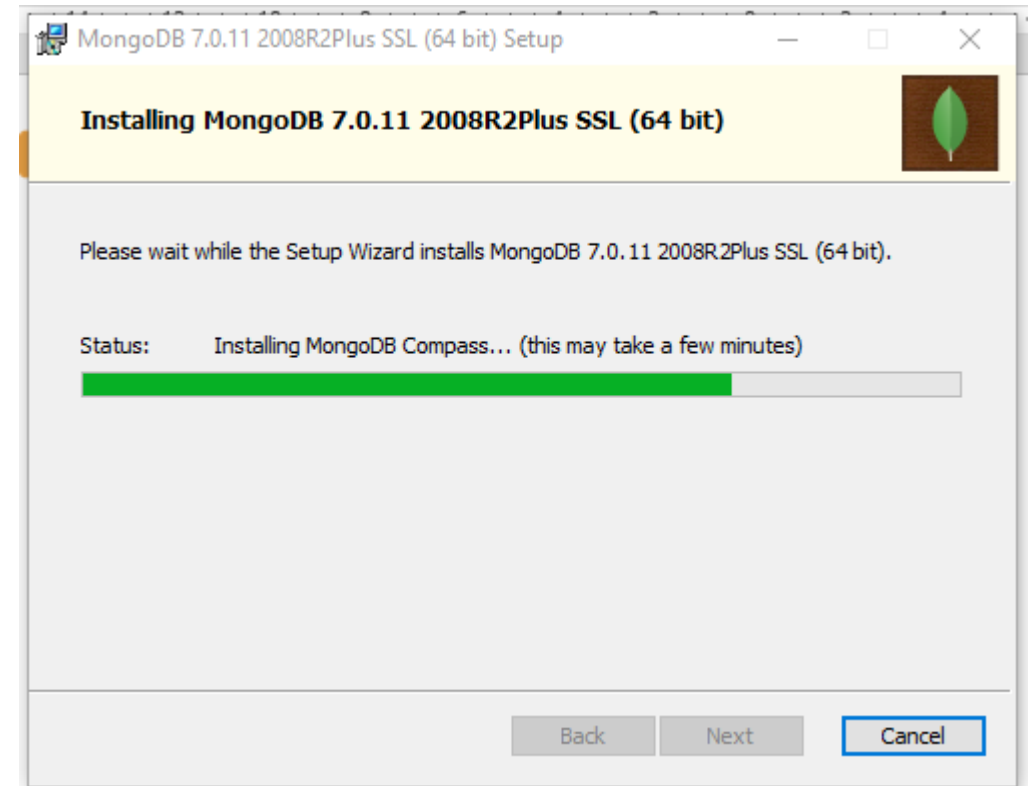
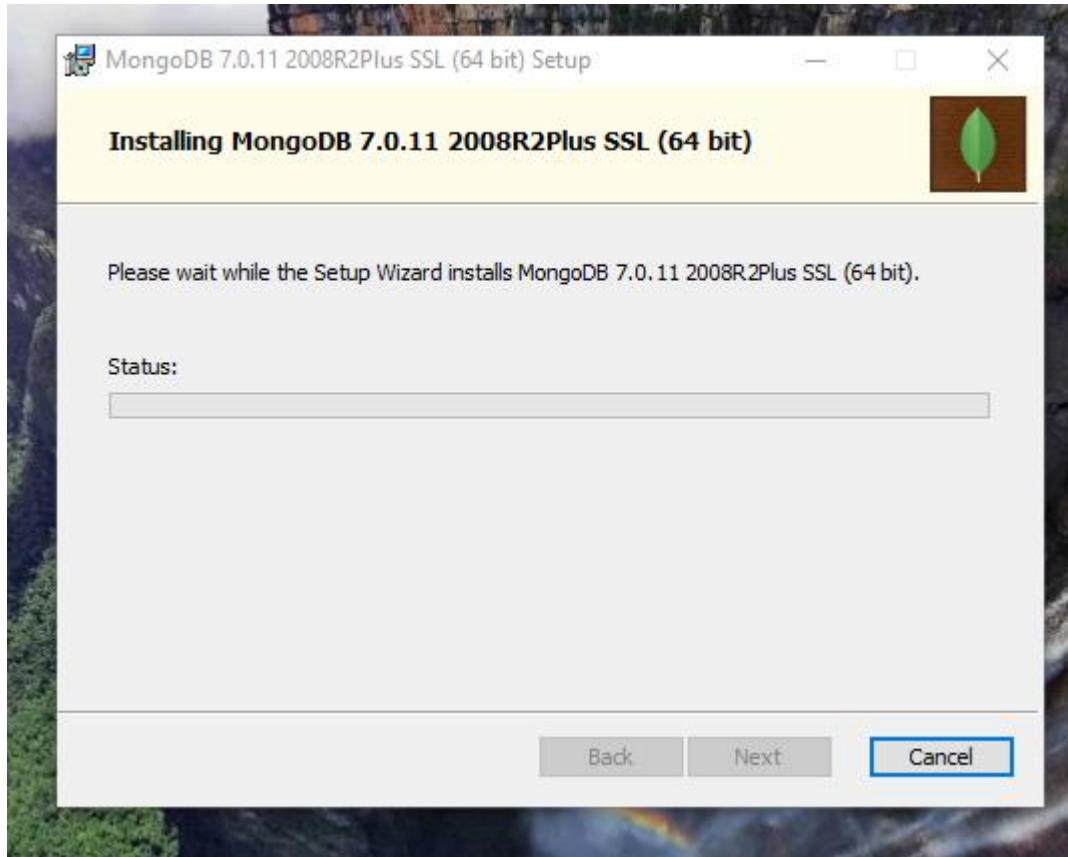


This screenshot shows the download options for MongoDB. It includes a terminal snippet with the commands: `$ brew install mongodb-atlas` and `$ atlas setup`. Below this, there are three dropdown menus for selecting the version, platform, and package. The selected options are: Version 7.0.11 (current), Platform Windows x64, and Package msi. At the bottom, there are buttons for 'Download', 'Copy link', and 'More Options'.





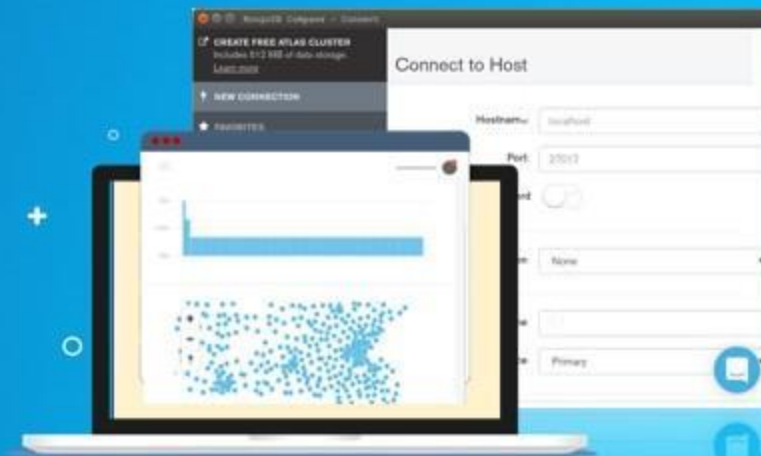




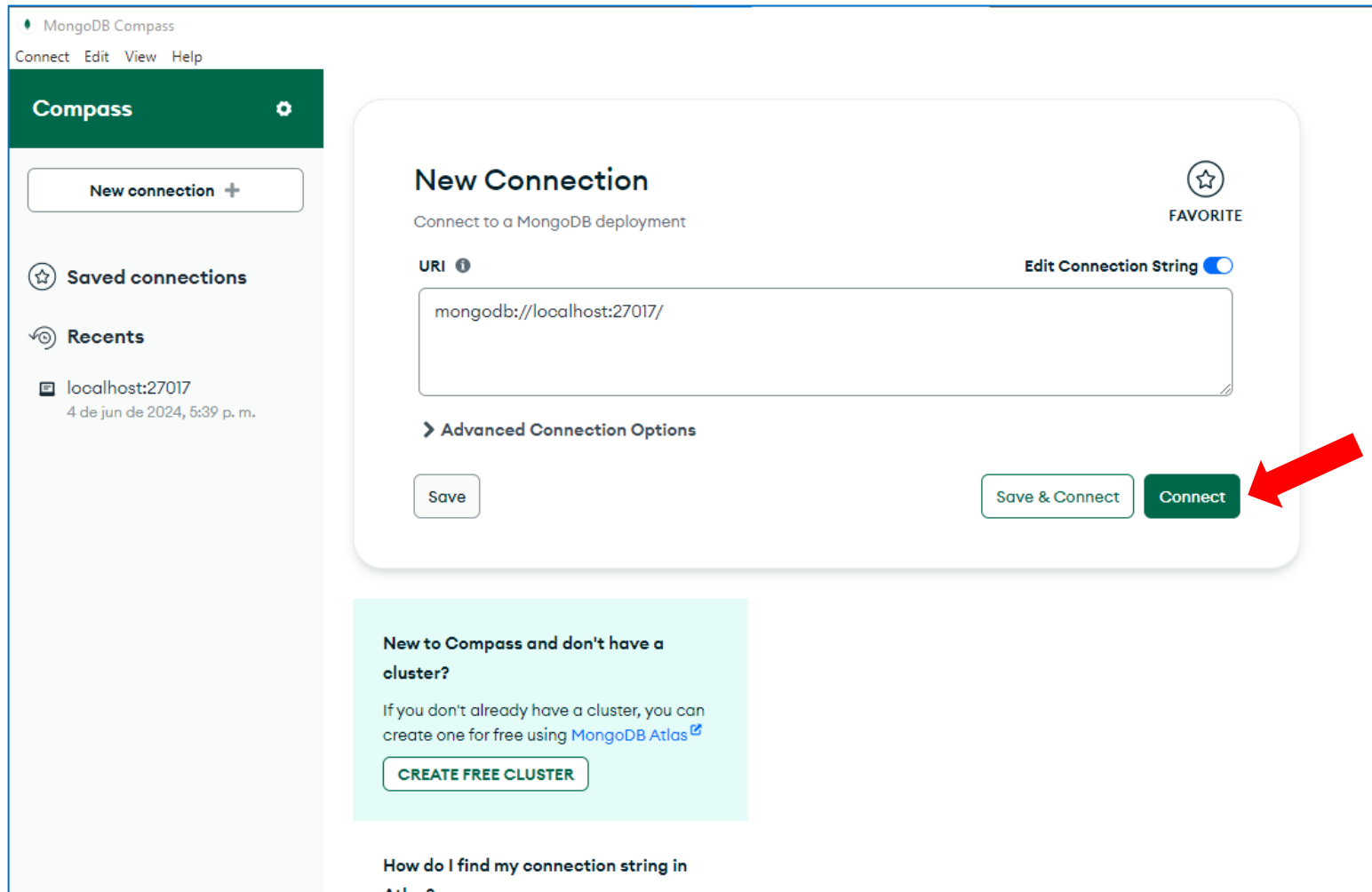
Es un poco demorado, cuando finaliza,
se abre Compass.

MongoDB Compass

MongoDB Compass



Configuración MongoDB Compass en Local



Interfaz MongoDB Compass

The screenshot shows the MongoDB Compass application window titled "MongoDB Compass - localhost:27017/My Queries". The interface includes a menu bar (Connect, Edit, View, Help), a connection bar showing "localhost:27017", and a sidebar with "My Queries", "Performance", and "Databases". The "Databases" section is expanded, showing a search bar and a list of databases: "admin", "config", and "local". A red bracket on the left groups these databases under the label "Bases de datos de Ejemplo". A red arrow points to the refresh icon in the "Databases" section, labeled "Actualizar o Refrescar las BD.". Another red arrow points to the "+" icon next to the refresh icon, labeled "Crear una nueva Base de Datos.". The bottom of the window features a dark blue console area labeled ">_MONGOSH" with a red arrow pointing to it from the label "consola de comandos de MongoDB".

MongoDB Compass - localhost:27017/My Queries

Connect Edit View Help

localhost:27017

My Queries

Performance

Databases

Search

admin

config

local

Start saving

N

>_MONGOSH

Bases de datos de Ejemplo

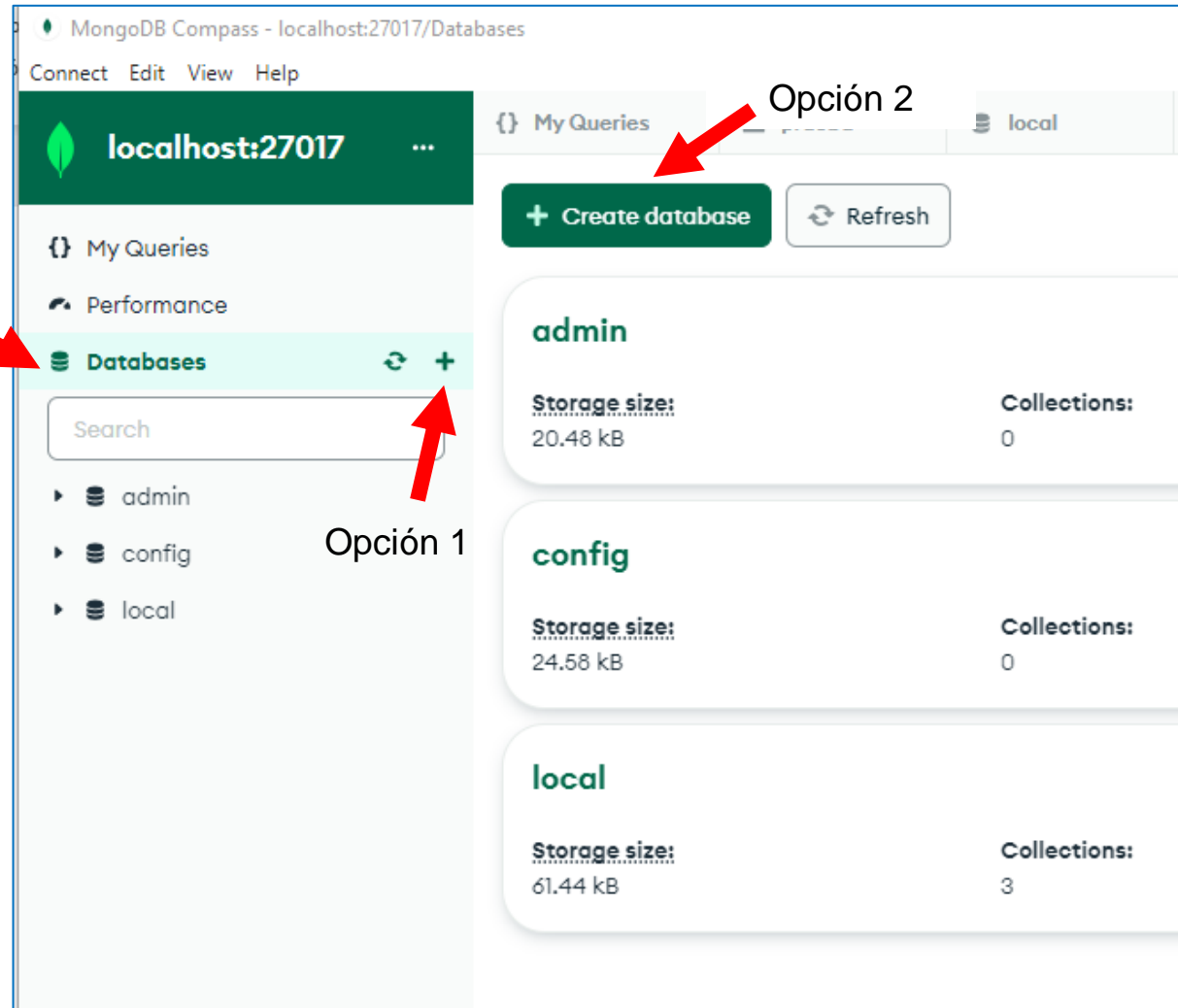
Actualizar o Refrescar las BD.

Crear una nueva Base de Datos.

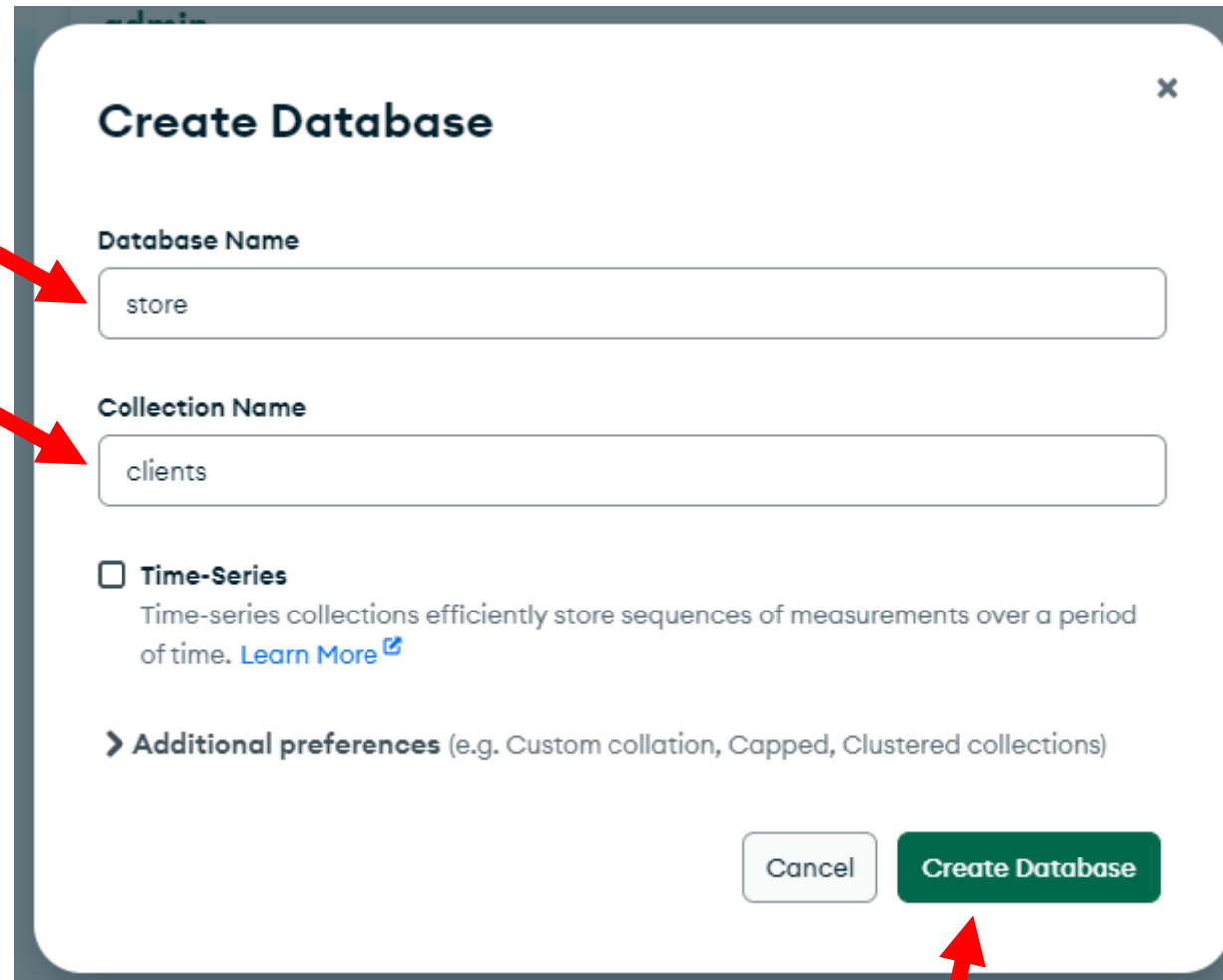
consola de comandos de MongoDB

Crear una Base de Datos desde la interfaz gráfica de Compass

Clic en Databases para que se muestren las dos opciones.



Crear una Base de Datos desde la interfaz gráfica de Compass



Create Database ×

Database Name

store

Collection Name

clients

☐ **Time-Series**
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

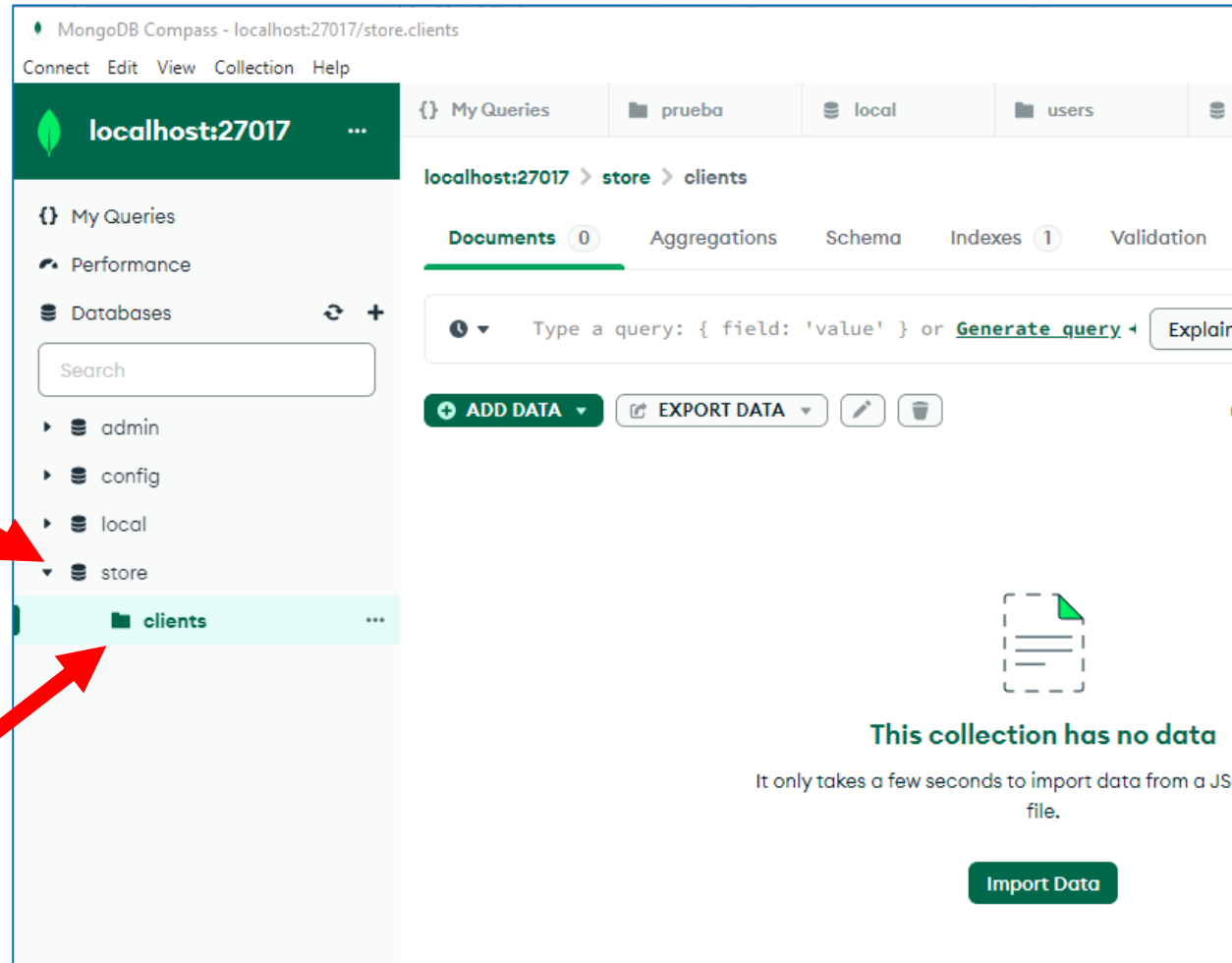
> Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Cancel Create Database

Crear una Base de Datos desde la interfaz gráfica de Compass

Nueva base de datos creada

Nueva colección creada, vacía, lista para insertarle los documentos.



Crear una Base de Datos desde la consola de comandos de MongoDB

Comando para
crear la base de
datos nueva

```
>_MONGOSH  
> use school  
< switched to db school  
school>
```

Nueva base de
datos creada

Comandos más usados en Lenguaje MQL (MongoDB Query Language)

>_MONGOSH

```
> db.prueba.insertOne({ name: "Alice", age: 30 })  
< {  
  acknowledged: true,  
  insertedId: ObjectId('666083f15fe3c002fcdbf4e9')  
}  
local>
```


Comandos más usado de MQL

Comando/Función	Descripción	Ejemplo de uso
use <database>	Selecciona una base de datos específica para su uso	use my_database
show collections	Lista todas las colecciones en la base de datos actual	show collections
db.createCollection()	Crea una nueva colección en la base de datos actual	db.createCollection("my_collection")
db.dropDatabase()	Elimina la base de datos actual y todos sus datos	db.dropDatabase()
db.collection.find()	Busca documentos en una colección específica	db.users.find({ age: { \$gt: 18 } })
db.collection.insertOne()	Inserta un nuevo documento en una colección específica	db.users.insertOne({ name: "Alice", age: 30 })
db.collection.updateOne()	Actualiza un único documento en una colección específica	db.users.updateOne({ name: "Alice" }, { \$set: { age: 31 } })
db.collection.insertMany()	Inserta múltiples documentos en una colección específica	db.users.insertMany([{ name: "Alice", age: 30 }, { name: "Bob", age: 35 }])
db.collection.deleteOne()	Elimina un único documento en una colección específica	db.users.deleteOne({ name: "Alice" })

Comandos más usado de MQL

Comando/Función	Descripción	Ejemplo de uso
db.collection.aggregate()	Realiza operaciones de agregación en una colección específica	db.sales.aggregate([{\$group: { _id: "\$item", total: { \$sum: "\$amount" } } }])
db.collection.createIndex()	Crea un índice en una colección específica	db.products.createIndex({ name: 1 })
db.collection.distinct()	Devuelve valores distintos para un campo específico en una colección	db.users.distinct("name")
db.collection.countDocuments()	Devuelve el número de documentos en una colección que coinciden con un criterio de filtro	db.users.countDocuments({ age: { \$gt: 18 } })

Crear Colección desde la interfaz gráfica de Compass

Clic en la Base de Datos que quieras trabajar

The image shows a sequence of three screenshots from the MongoDB Compass application, illustrating the process of creating a new collection. The first screenshot shows the main interface with the 'Databases' list on the left. A red arrow points to the 'local' database, which is highlighted. The second screenshot shows the 'local' database selected, with a red arrow pointing to the '+ Create collection' button. The third screenshot shows the 'Create Collection' dialog box, where the 'Collection Name' is set to 'users'. A red arrow points to the 'Create Collection' button at the bottom right of the dialog.

MongoDB Compass - localhost:27017/local

Connect Edit View Help

localhost:27017

My Queries

Performance

Databases

Search

admin

config

local

+ Create collection

prueba

Storage size: 4.10 kB

startup_log

Storage size: 20.48 kB

Collection Name

users

☐ Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

> Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Cancel Create Collection

Crear Colección desde la consola de comandos

Comando para crear la nueva colección

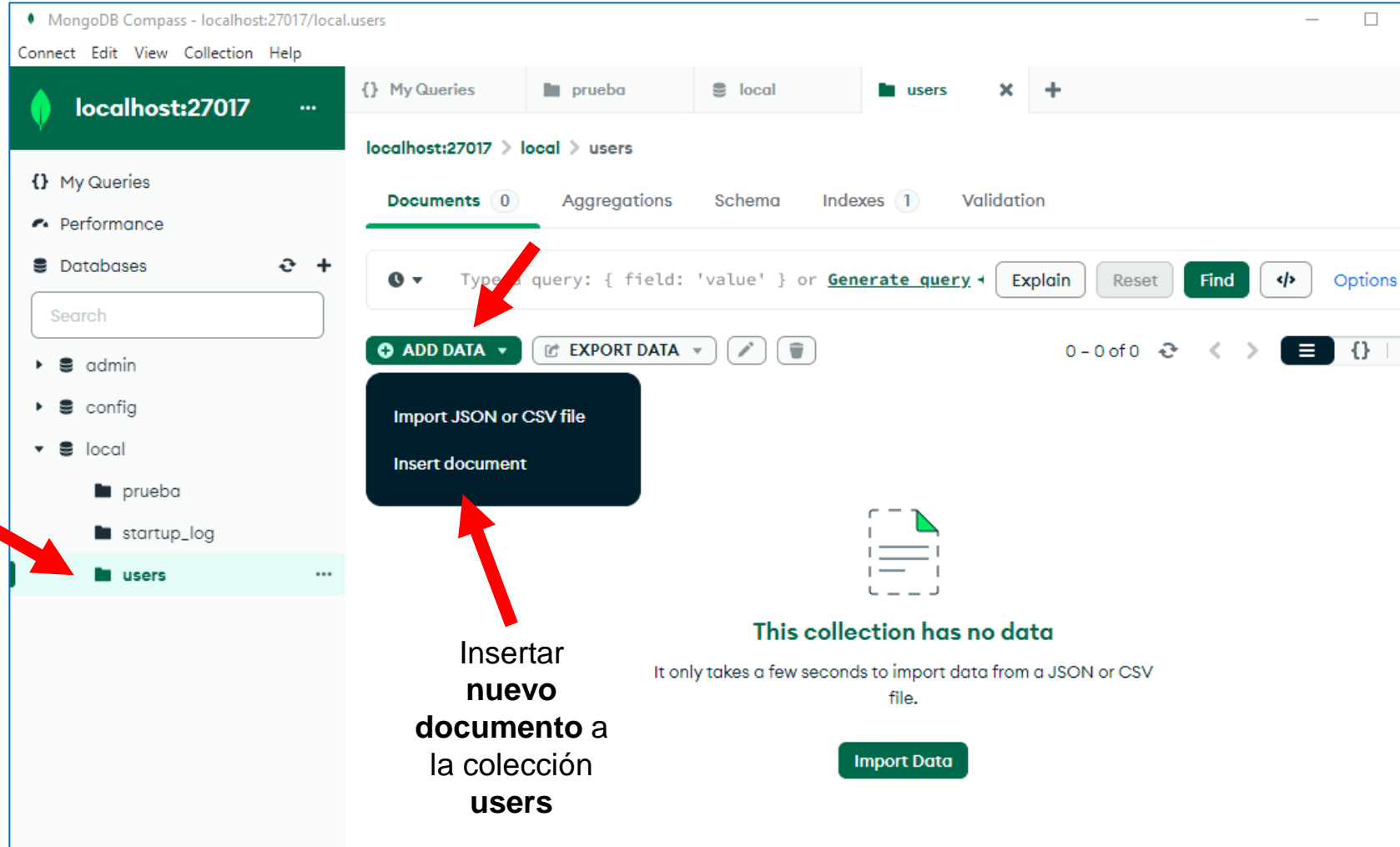
Comando para listar las colecciones de la base de datos

```
>_MONGOSH  
→ db.createCollection("grades")  
< { ok: 1 }  
→ show collections  
< grades  
school >
```

Antes de crear la nueva colección, se debe estar ubicado en la base de datos. **use base_datos** permite seleccionar la base de datos a usar.

Insertar Documento a Colección desde la interfaz gráfica de Compass

Nueva colección creada



MongoDB Compass - localhost:27017/local.users

Connect Edit View Collection Help

localhost:27017

My Queries

Performance

Databases

Search

admin

config

local

prueba

startup_log

users

My Queries

prueba

local

users

localhost:27017 > local > users

Documents 0 Aggregations Schema Indexes 1 Validation

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

ADD DATA EXPORT DATA

Import JSON or CSV file

Insert document

0 - 0 of 0

This collection has no data

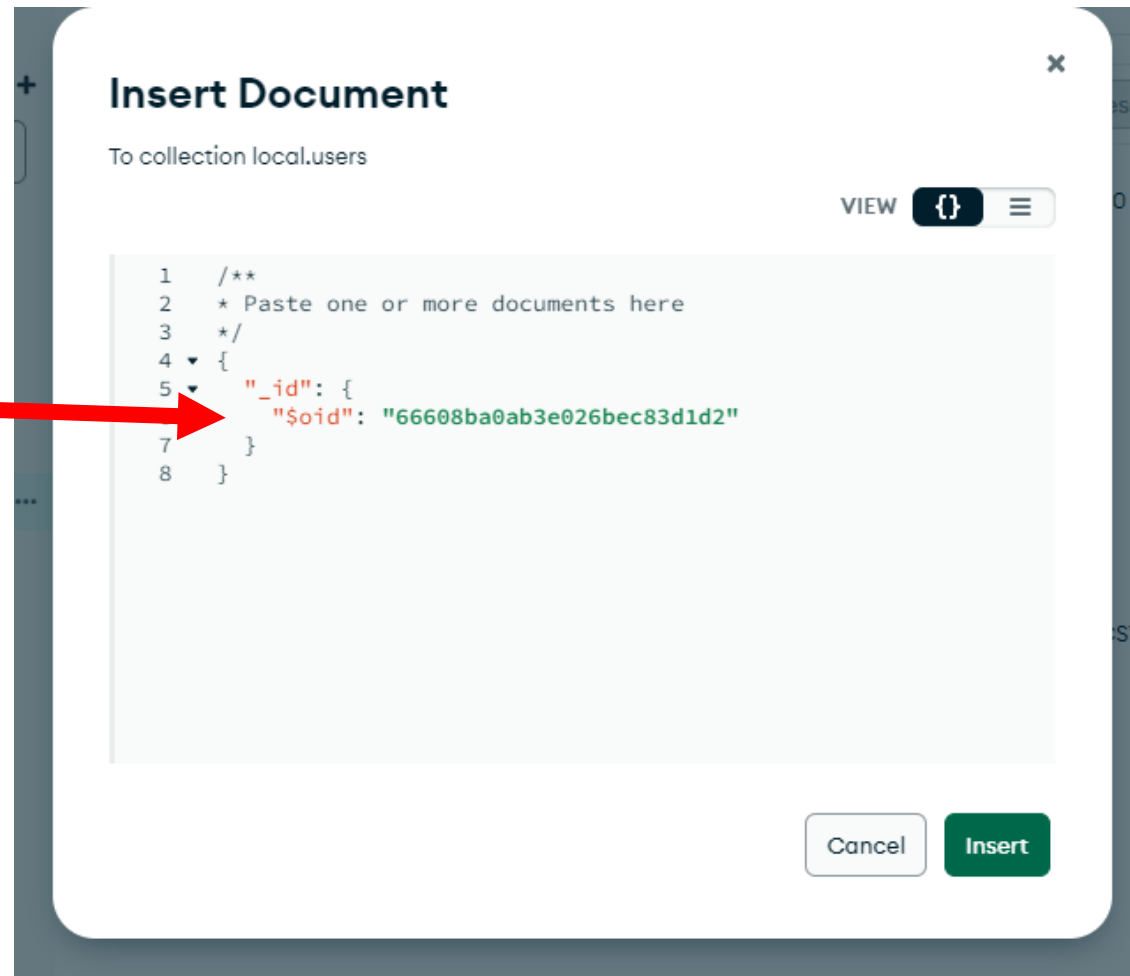
It only takes a few seconds to import data from a JSON or CSV file.

Import Data

Insertar nuevo documento a la colección users

Insertar Documento a Colección desde la interfaz gráfica de Compass

Eliminar el
contenido,
solo dejamos
las dos llaves
{ }



Insertar Documento a Colección desde la interfaz gráfica de Compass



Escribir los datos que se quieren guardar del documento en formato
“clave”: valor



Documents 0 Aggregations Schema Indexes 1 Validation

Insert Document

To collection local.users

VIEW  

```
1 {
2   {
3     "name": "Alice",
4     "age": 30,
5     "birthDate": "2000-01-01"
6   }

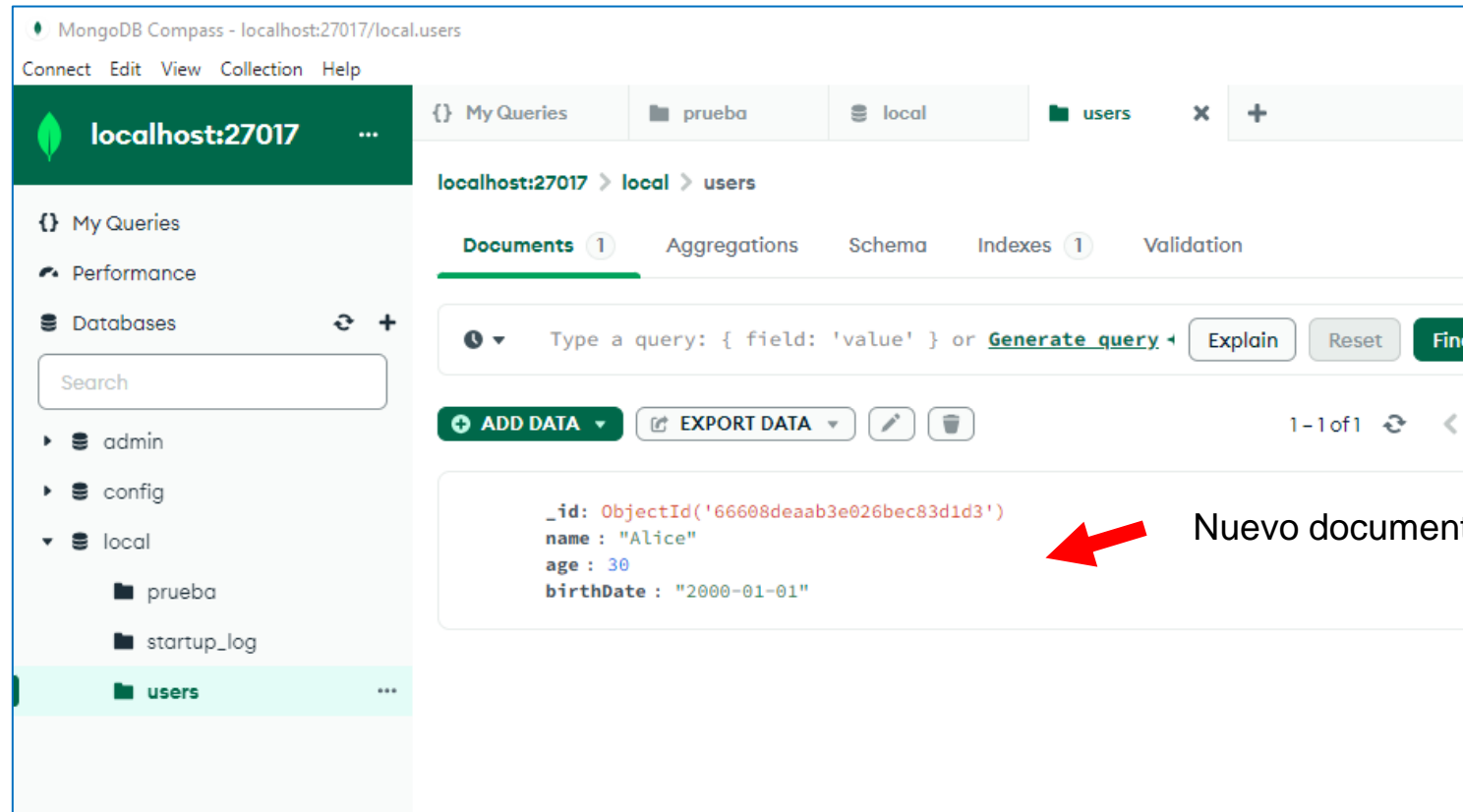
```

Cancel Insert

Ten en cuenta lo siguiente para el valor:

- **String:** se escribe en comillas, "Alice"
- **Numérico:** se escribe sin comillas, 30.
- **Fecha:** se escribe en comillas con formato "YYYY-MM-DD", "fechaNacimiento": "2000-01-01"

Insertar Documento a Colección desde la interfaz gráfica de Compass



Insertar Documento desde la consola de comandos

Comando para mostrar las colecciones en la base de datos actual

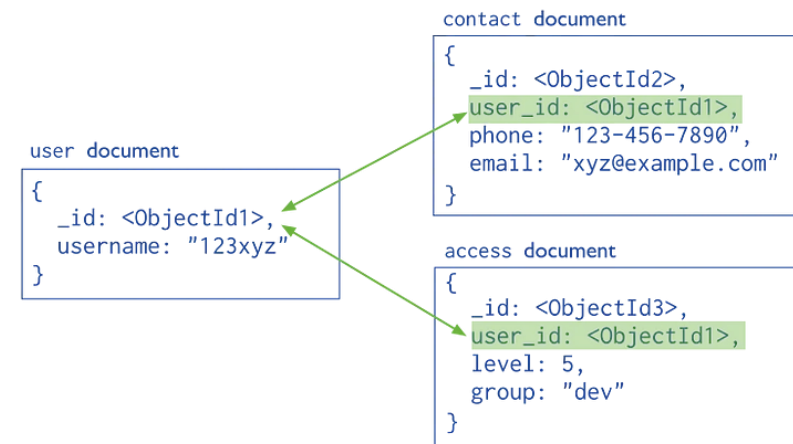
Insertar un **documento** a la colección **prueba**

Comando para mostrar todos los documentos de la colección **prueba**

Documentos retornados de la colección prueba

```
>_MONGOSH
> show collections
< prueba
  startup_log
  users
> db.prueba.insertOne({ name: "María", age: 35, city: "Bogotá" })
< {
  acknowledged: true,
  insertedId: ObjectId('66609a467a1a3c81a0b0ceee')
}
> db.prueba.find()
< {
  _id: ObjectId('666083f15fe3c002fcd9f4e9'),
  name: 'Alice',
  age: 30
}
{
  _id: ObjectId('66609a467a1a3c81a0b0ceee'),
  name: 'María',
  age: 35,
  city: 'Bogotá'
}
local>
```


Referencias en MongoDB

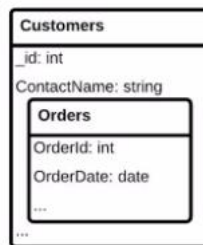


MongoDB permite implementar relaciones a través de un concepto llamado “**referencias**”. MongoDB no se apegas al modelo ER (Entidad Relación) y otras cualidades de las bases de datos relacionales.

Las relaciones en MongoDB se pueden modelar en 2 enfoques diferentes: la **relación incrustada** o **relación referenciada**. La elección de estos enfoques dependerá del tipo de caso y decisiones de modelamiento de datos.

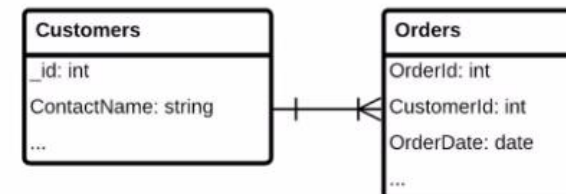
Relación Incrustada

Documentos dentro de Documentos (dentro de Documentos)



Relación referenciada

Un valor hace referencia a un par clave/valor en otro documento



Relación incrustada: relación que implica almacenar documentos secundarios incrustados dentro de un documento principal.

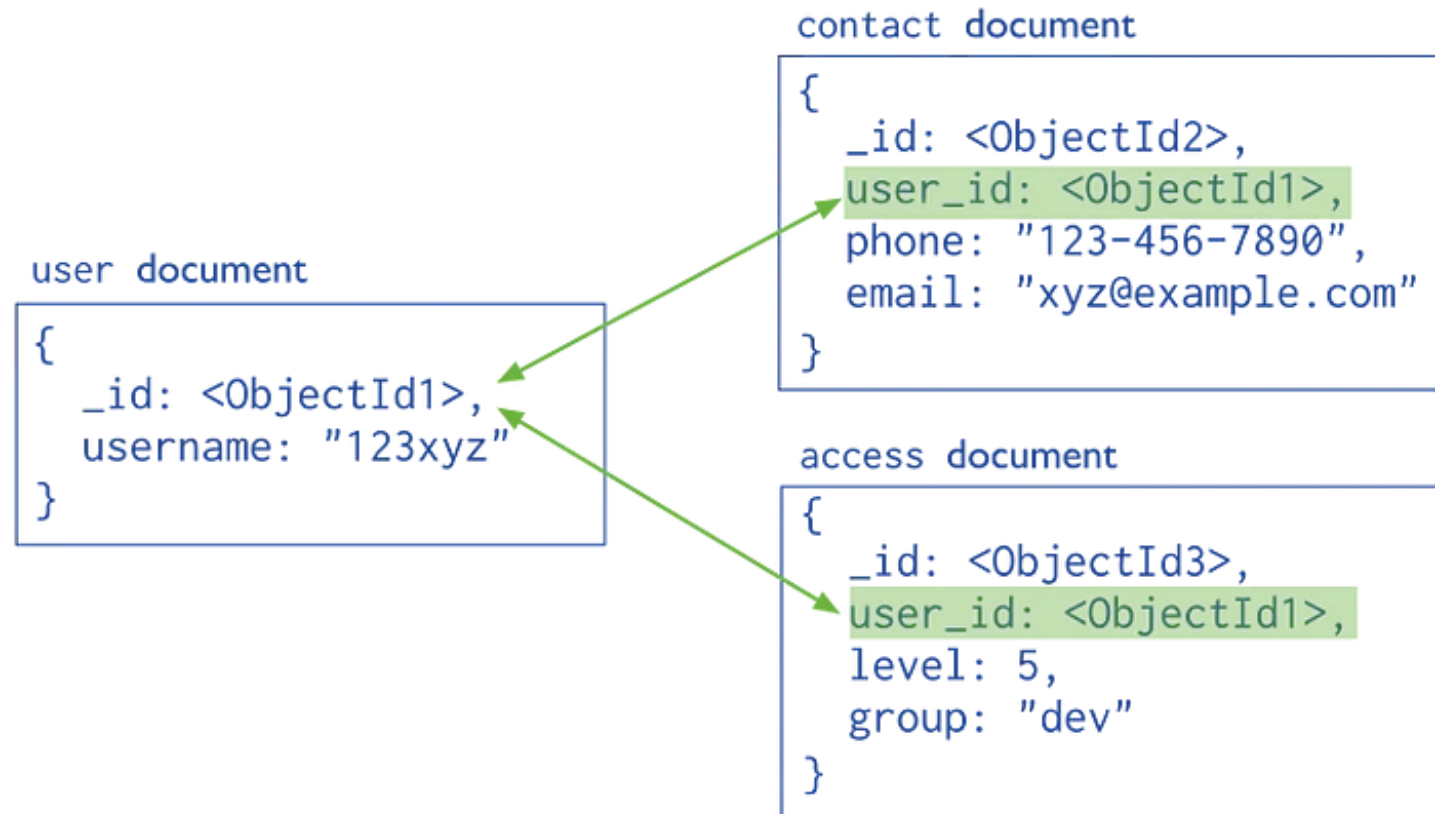
pedido		
 _id	objectId	NN
nombre	string	
direccion	string	
codigoPostal	string	
ciudad	string	
lineaDetalle [{}]	object	
id	objectId	
producto_id	objectId	
cantidad	int	
precio	decimal	

```
{
  _id: <ObjectId>,
  username: "123xyz",
  contact: {
    phone: "123-456-7890",
    email: "xyz@example.com"
  },
  access: {
    level: 5,
    group: "dev"
  }
}
```

Embedded sub-document

Embedded sub-document

Relación referenciada: relación que implica **conectar colecciones diferentes** por medio de **referencias**. La referencia se realiza a través de una **propiedad o clave** en común. Estas referencias pueden ser de dos tipos; referencias **manuales** o por **DBRefs**.



Referencias manuales:

Almacenamos el _id del documento relacionado en el documento principal. Este método es más común y eficiente en MongoDB porque evita la sobrecarga adicional que viene con las referencias de DBRefs.

Colección Profesores

```
{
  "_id": ObjectId("60d5ec49c0d9a4c8b0e7aef6"),
  "nombre": "Luisa",
  "apellidos": "García",
  "telefono": "123456789",
  "email": "luisa@example.com"
},
{
  "_id": ObjectId("60d5ec49c0d9a4c8b0e7aef7"),
  "nombre": "Javier",
  "apellidos": "Martínez",
  "telefono": "987654321",
  "email": "javier@example.com"
}
```

Colección Asignaturas

```
{
  "_id": ObjectId("60d5ec49c0d9a4c8b0e7aef4"),
  "nombre": "Matemáticas",
  "id_profesor": ObjectId("60d5ec49c0d9a4c8b0e7aef6")
},
{
  "_id": ObjectId("60d5ec49c0d9a4c8b0e7aef5"),
  "nombre": "Historia",
  "id_profesor": ObjectId("60d5ec49c0d9a4c8b0e7aef7")
}
```


Consultas con Referencias manuales:

Almacenamos el _id del documento relacionado en el documento principal. Este método es más común y eficiente en MongoDB porque evita la sobrecarga adicional que viene con las referencias de DBRefs.

Colección Profesores

```
{
  "_id": ObjectId("60d5ec49c0d9a4c8b0e7aef6"),
  "nombre": "Luisa",
  "apellidos": "García",
  "telefono": "123456789",
  "email": "luisa@example.com"
},
{
  "_id": ObjectId("60d5ec49c0d9a4c8b0e7aef7"),
  "nombre": "Javier",
  "apellidos": "Martínez",
  "telefono": "987654321",
  "email": "javier@example.com"
}
```

Colección Asignaturas

```
{
  "_id": ObjectId("60d5ec49c0d9a4c8b0e7aef4"),
  "nombre": "Matemáticas",
  "id_profesor": ObjectId("60d5ec49c0d9a4c8b0e7aef6")
},
{
  "_id": ObjectId("60d5ec49c0d9a4c8b0e7aef5"),
  "nombre": "Historia",
  "id_profesor": ObjectId("60d5ec49c0d9a4c8b0e7aef7")
}
```

Referencias por DBRefs:

DBRefs es una convención en MongoDB que proporciona una forma estándar de referenciar documentos de otras colecciones. Contiene la colección y el `_id` del documento objetivo, y opcionalmente la base de datos. Las DBRefs no son tan eficientes como las referencias manuales y rara vez se usan en aplicaciones modernas.

Ejemplo: Colección estudiantes con `ids_asignatura` usando DBRef

Este documento de ejemplo en la colección estudiantes contiene una lista de referencias a documentos en la colección asignaturas.

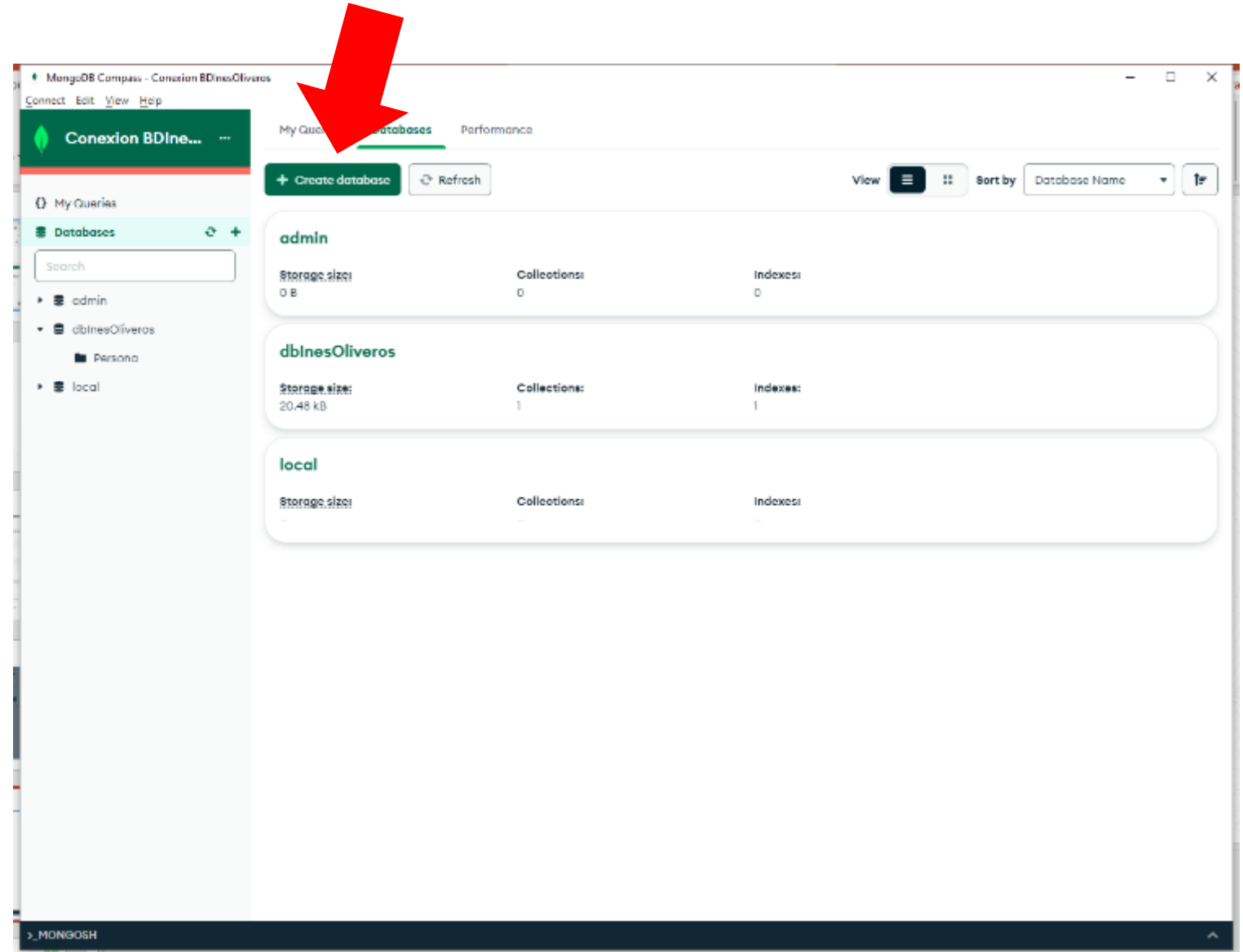
```
{
  "_id": ObjectId("60d5ec49c0d9a4c8b0e7aef3"),
  "nombre": "Juan",
  "apellidos": "Pérez",
  "edad": 20,
  "telefono": "123456789",
  "email": "juan@example.com",
  "ids_asignatura": [
    { "$ref": "asignaturas", "$id": ObjectId("60d5ec49c0d9a4c8b0e7aef4") },
    { "$ref": "asignaturas", "$id": ObjectId("60d5ec49c0d9a4c8b0e7aef5") }
  ]
}
```

Usar MongoDB Compass entorno gráfico

Usar MongoDB Compass

Crear una BD

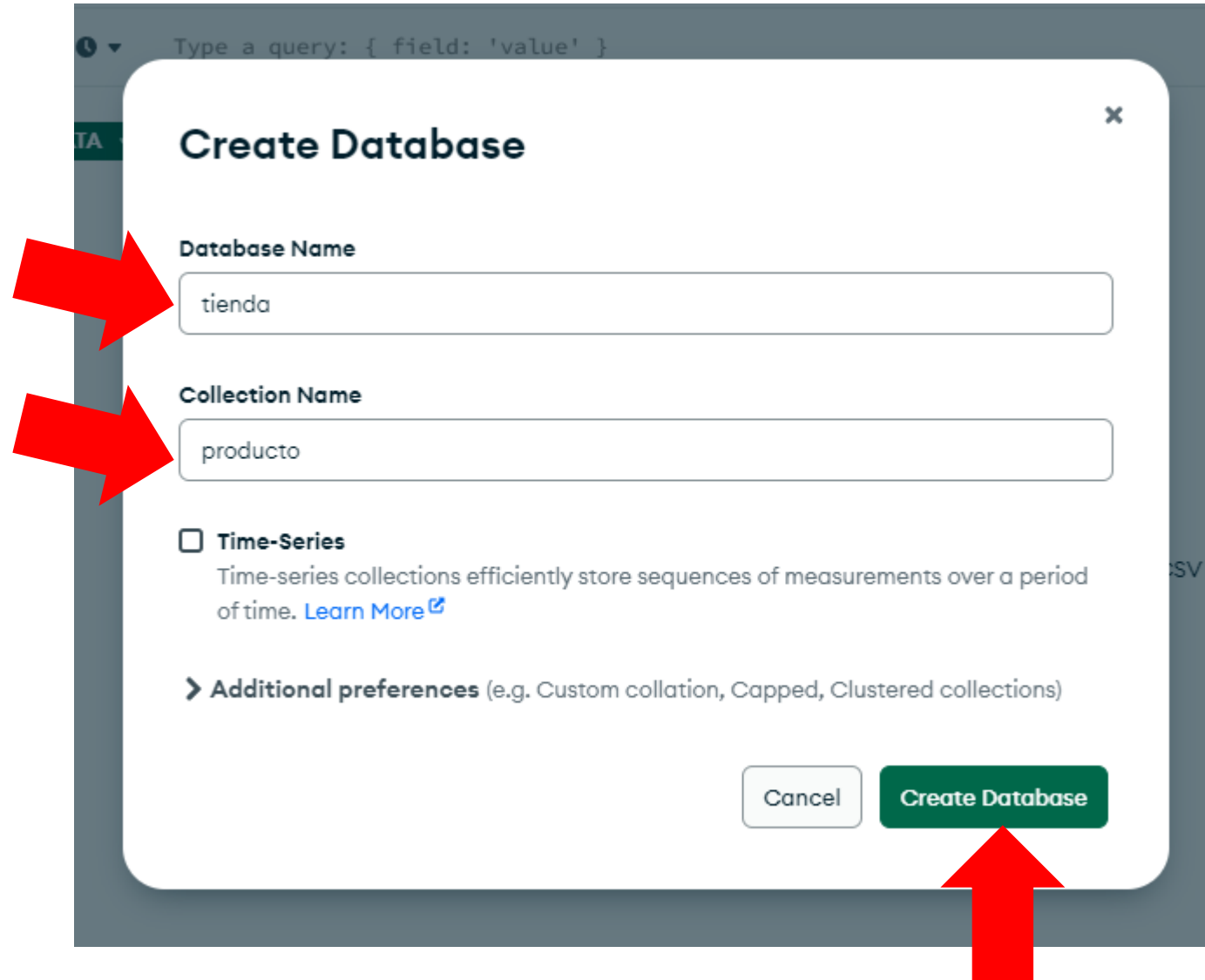
Para crear una nueva Base de datos, clic en + Create database.



Usar MongoDB Compass

Crear una BD

Se debe crear la base de datos junto con al menos una colección.



Type a query: { field: 'value' }

Create Database

Database Name

tienda

Collection Name

producto

☐ Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

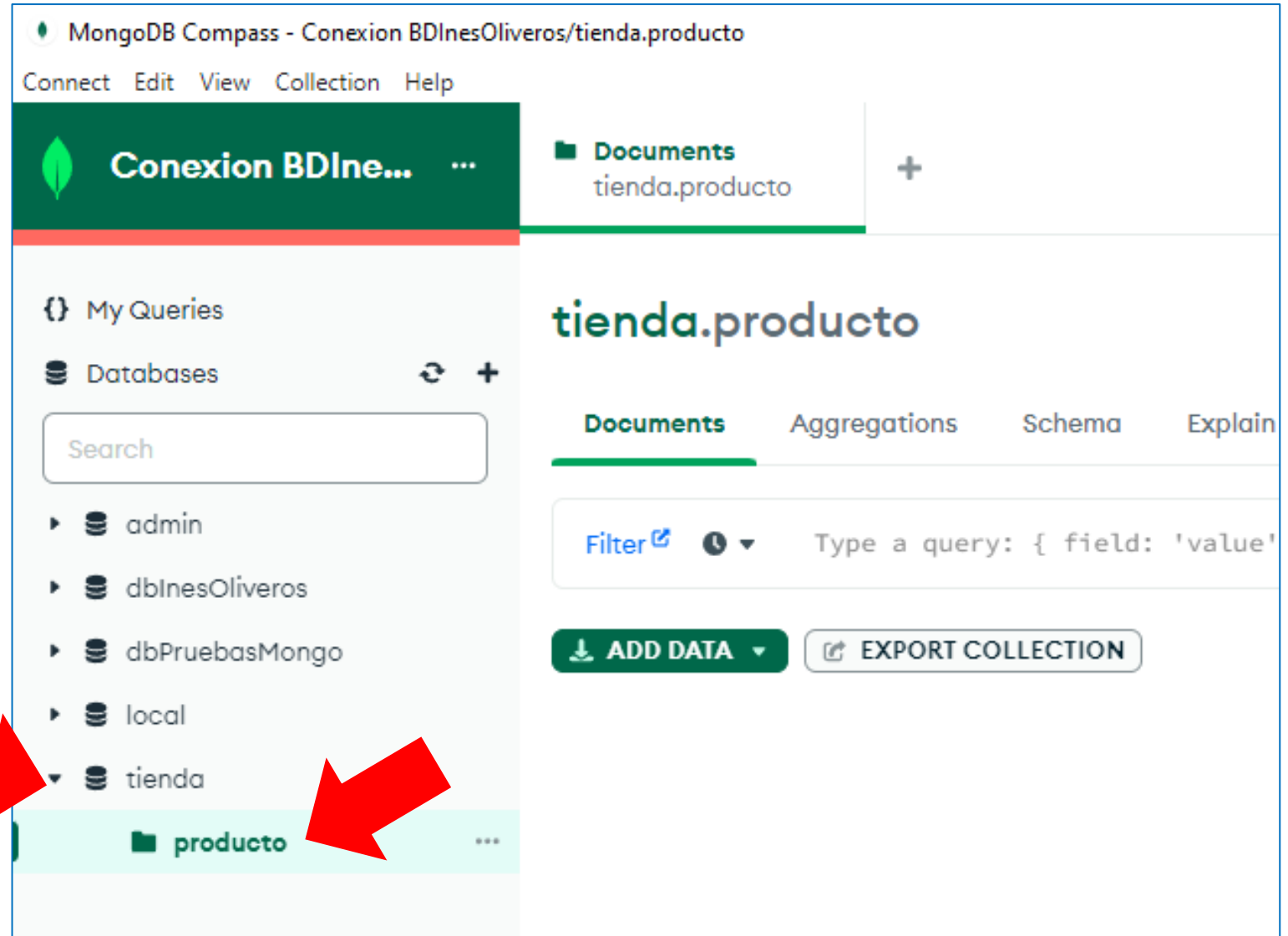
> Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Cancel Create Database

Usar MongoDB Compass

Crear una BD

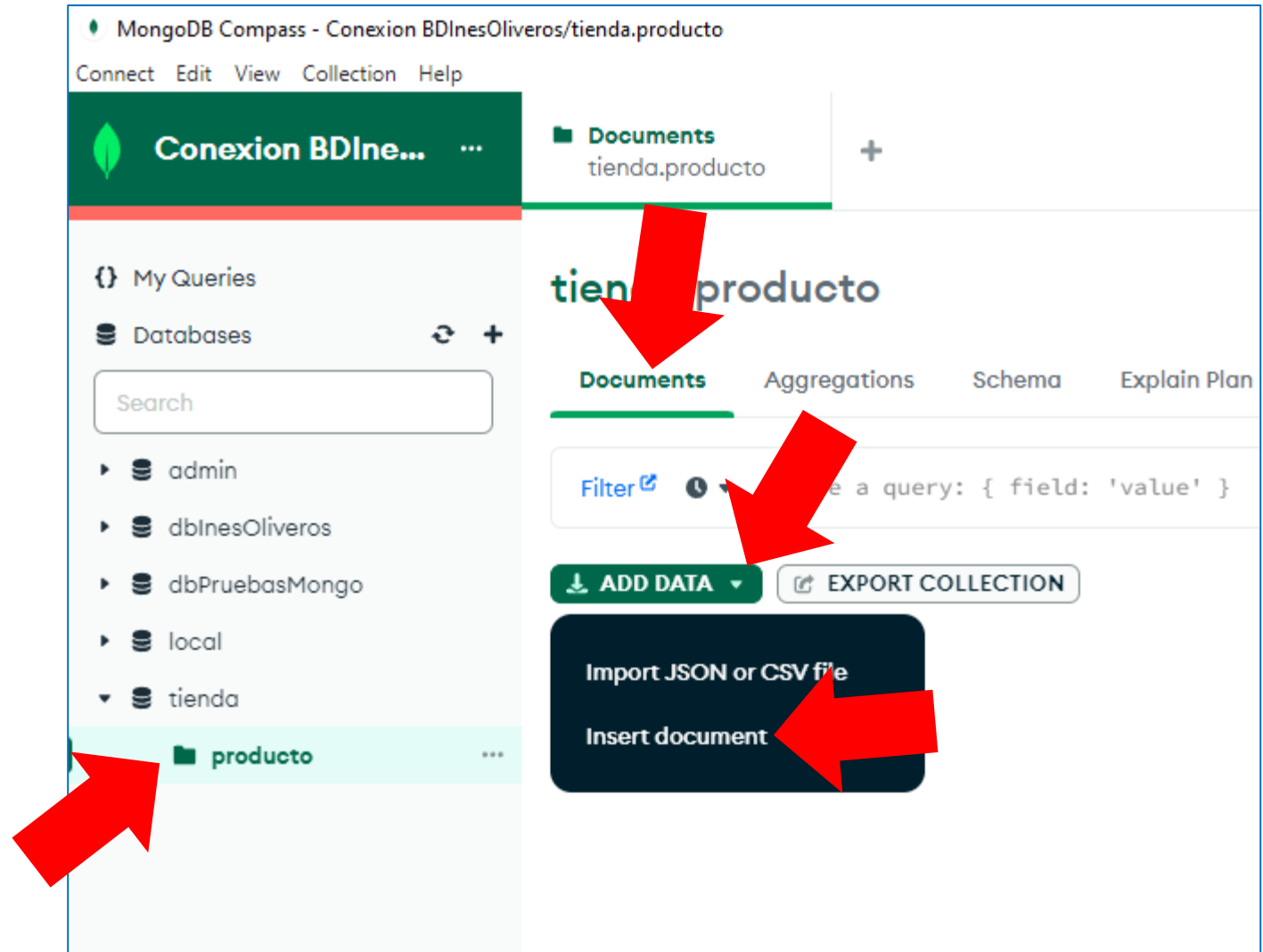
Nueva base de datos creada y su primer colección.



Usar MongoDB Compass

Insertar un documento en una colección

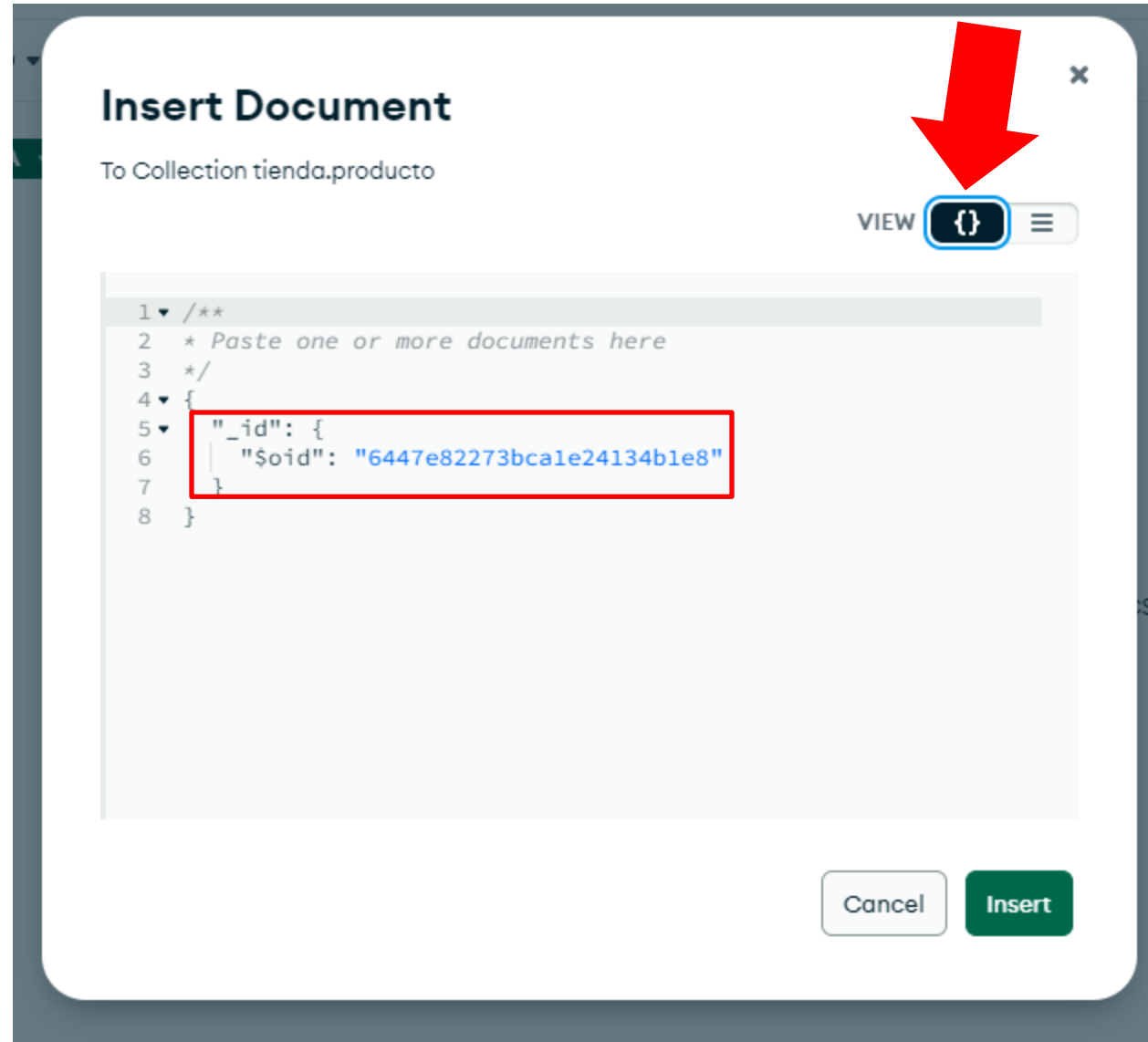
- Se selecciona la colección donde se creará el documento.
- Clic en **ADD DATA**
- Clic en **Insert document**.



Usar MongoDB Compass

Insertar un documento en una colección

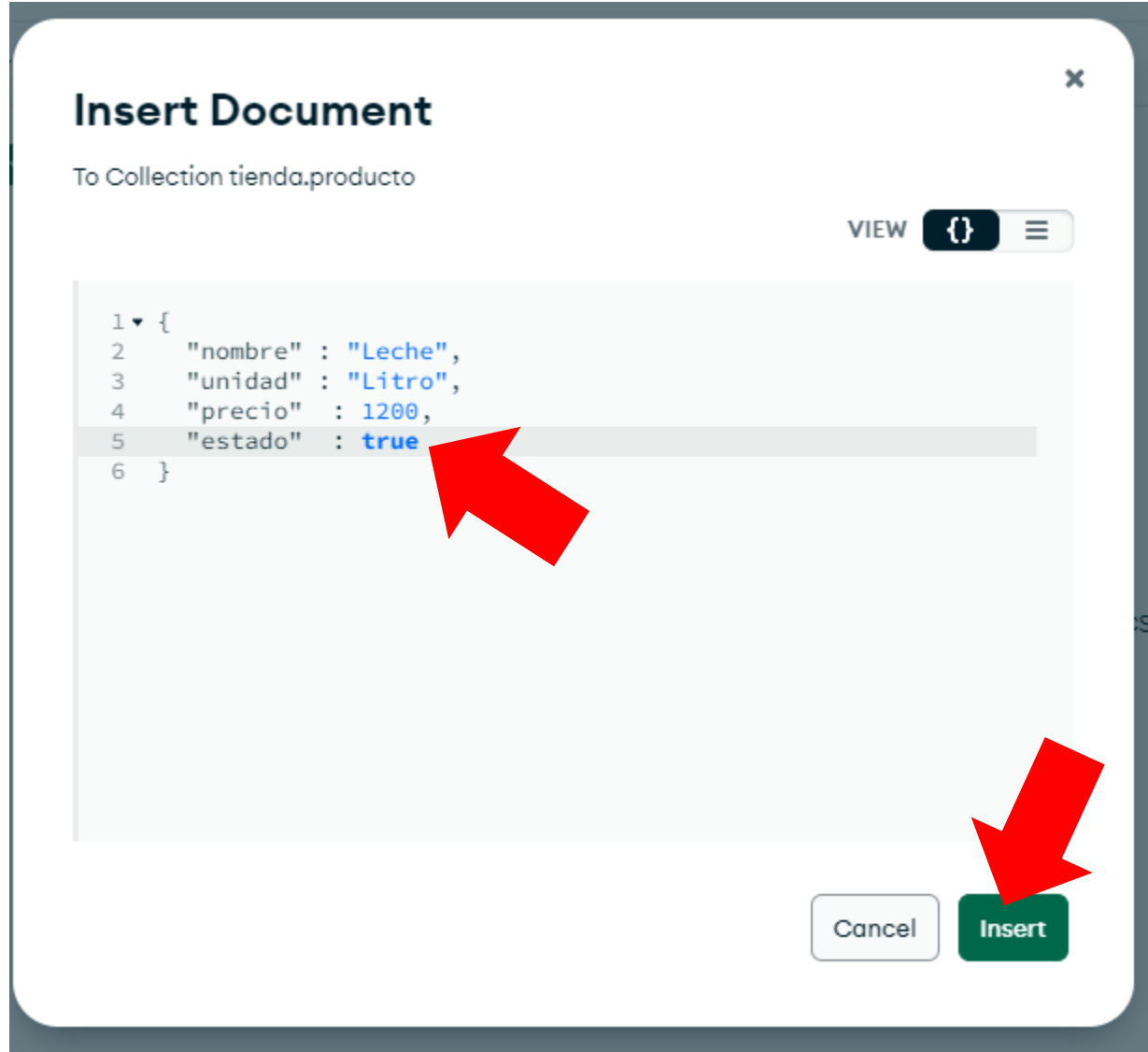
- Seleccionamos la forma que usaremos para crear el documento: JSON.
- Eliminamos el código de las líneas 7.



Usar MongoDB Compass



Insertar un documento en una colección

- Escribimos la **clave** y el **valor** separados por coma (,).
- A la última línea **no le colocamos coma.**
- Clic en Insert.





Insert Document

To Collection tienda.producto

VIEW  

```
1 {  
2   "nombre" : "Leche",  
3   "unidad" : "Litro",  
4   "precio" : 1200,  
5   "estado" : true  
6 }
```



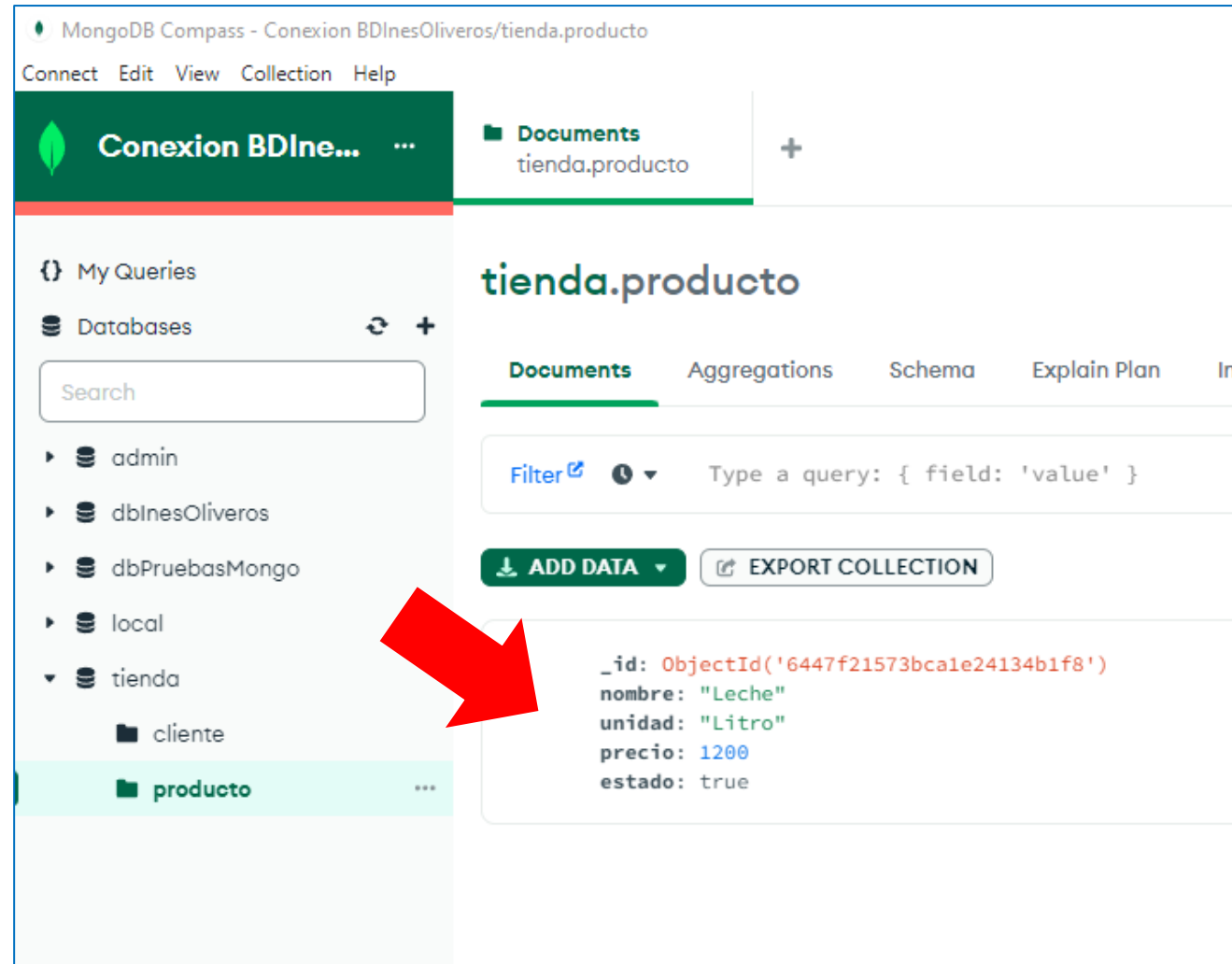


Cancel Insert

Usar MongoDB Compass

Insertar un documento en una colección

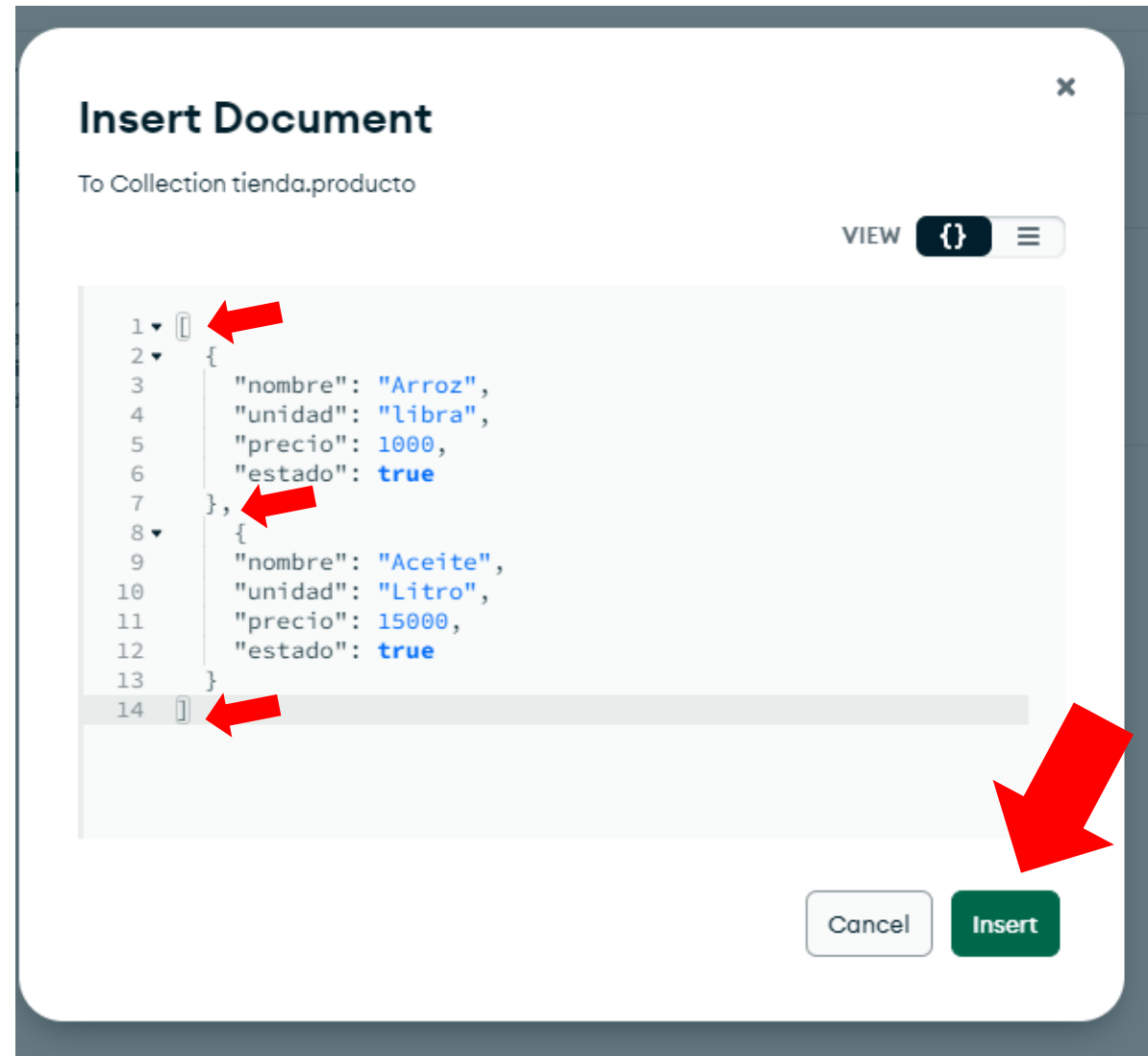
Así se ve el documento creado.



Usar MongoDB Compass

Insertar varios documentos al tiempo

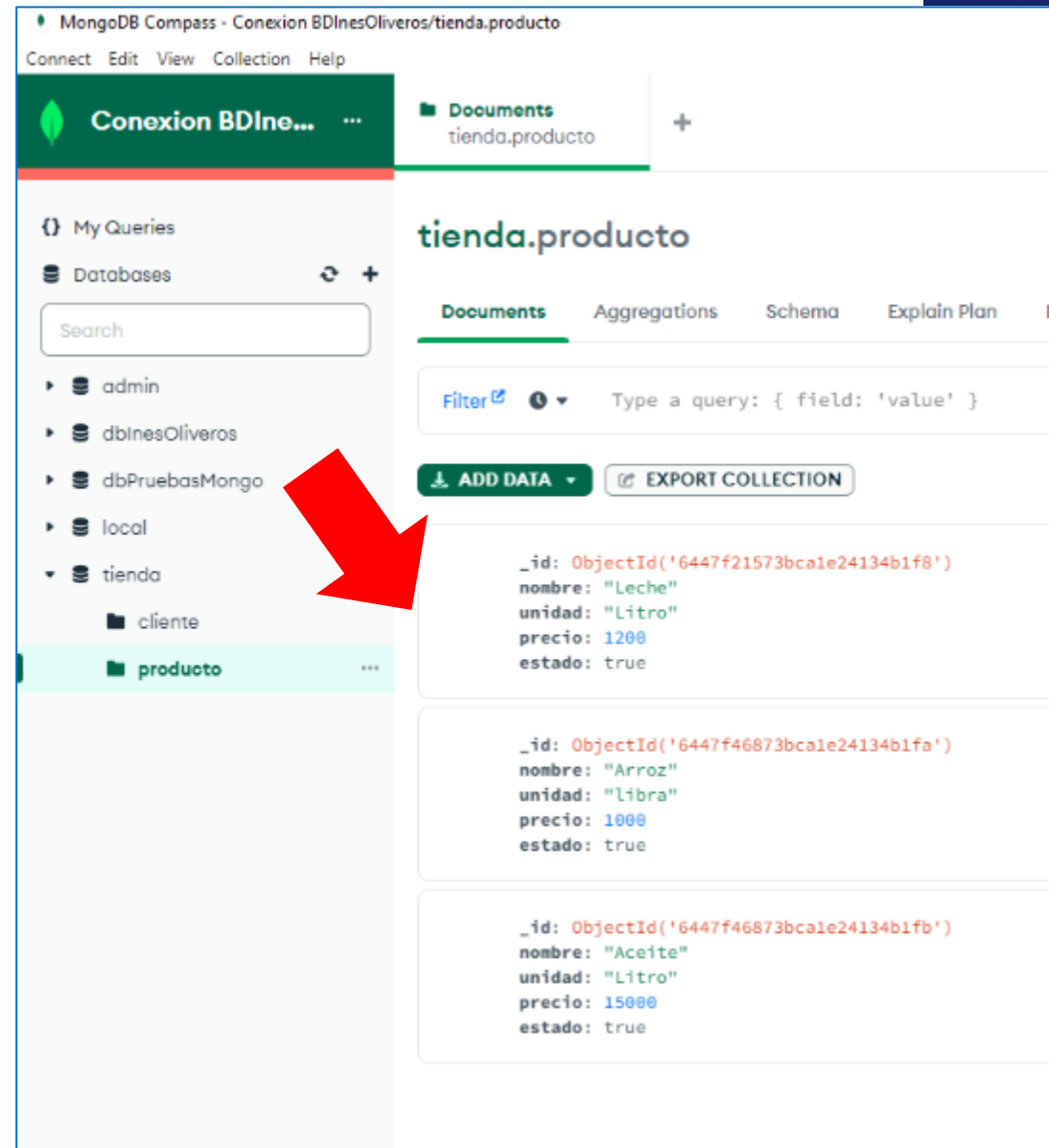
Colocamos la estructura de los documentos entre corchetes `[]` separados por coma `(,)`.



Usar MongoDB Compass

Insertar varios documentos al tiempo

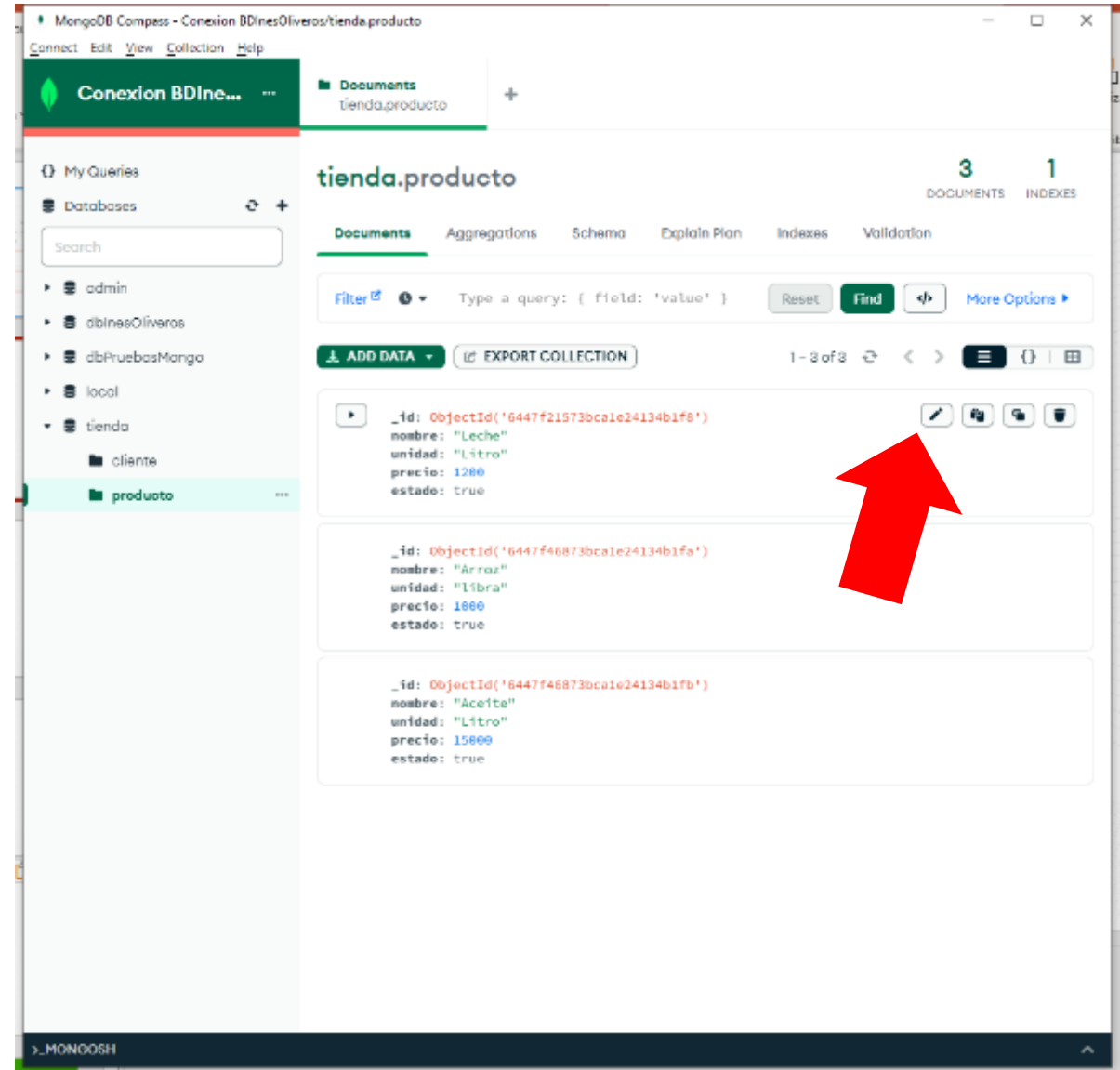
Así se visualizan los documentos creados dentro de la colección producto.



Usar MongoDB Compass

Editar documentos

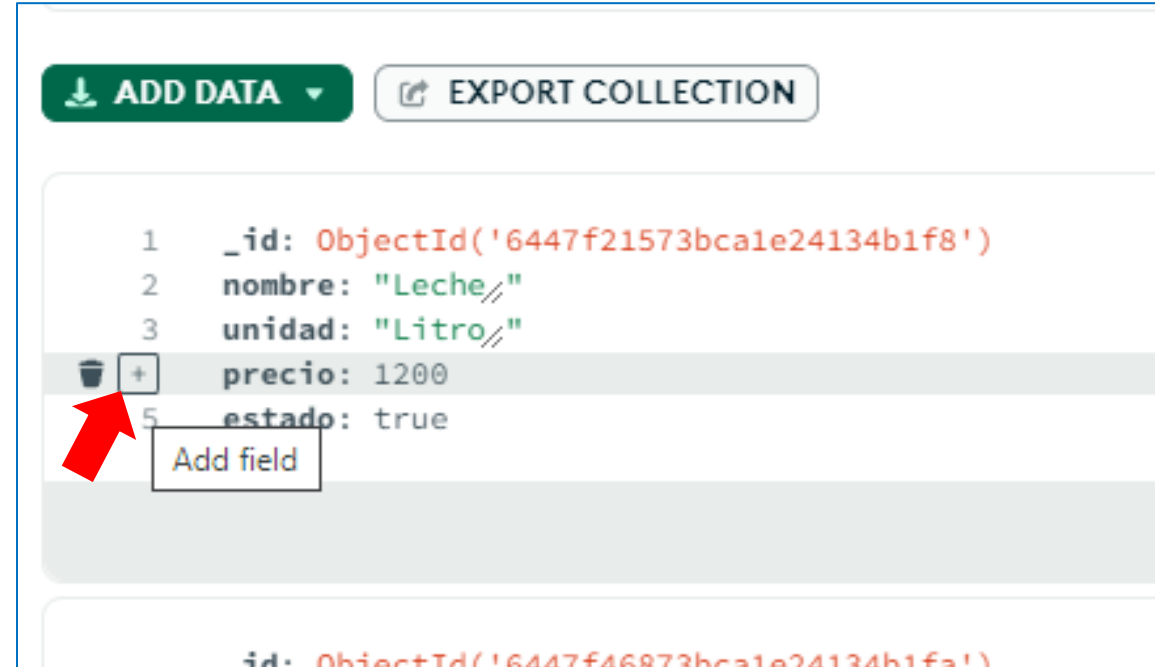
Se hace clic en el **ícono de lápiz** al frente del documento que se desea modificar.



Editar documentos

Por ejemplo si queremos **adicionar** un nuevo campo a un documento.

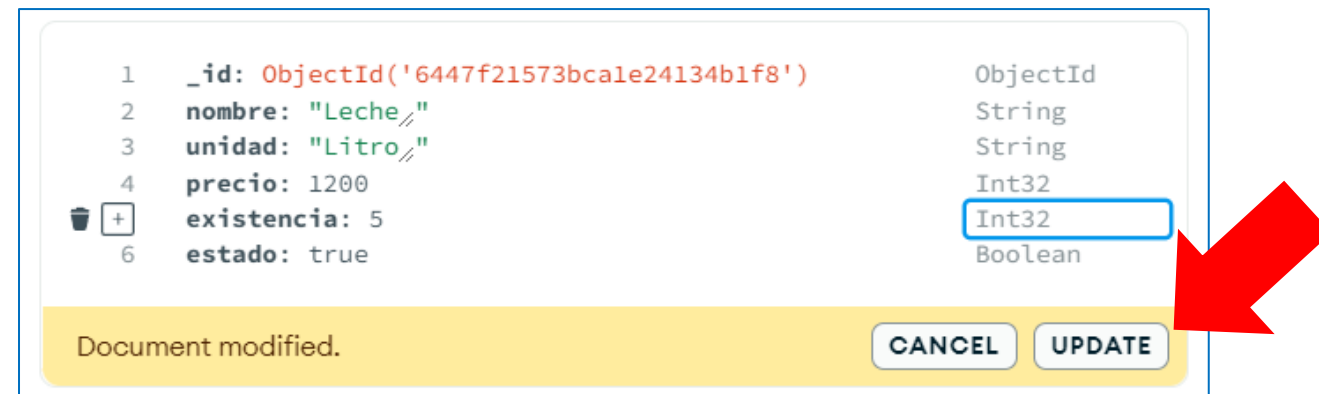
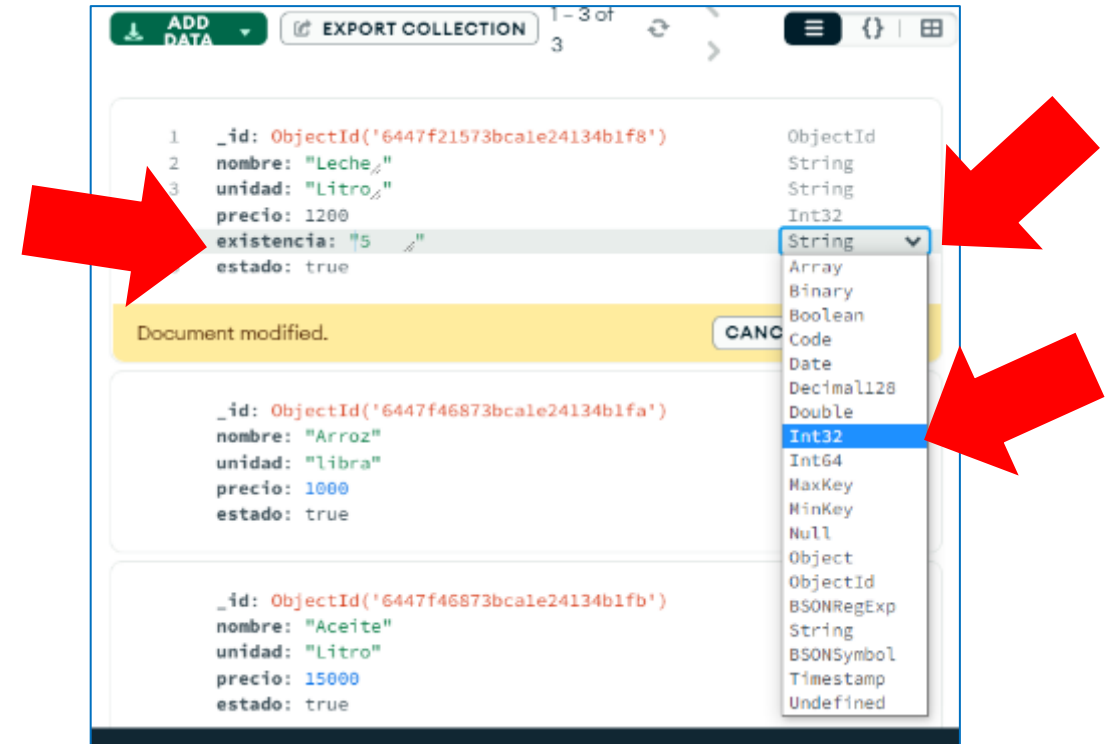
- Clic en el signo más (+)
- Clic en **Add field after...**



Usar MongoDB Compass

Editar documentos

- Se escribe la clave y el valor.
- Se escoge el tipo de dato.
- Se hace clic en **UPDATE**.

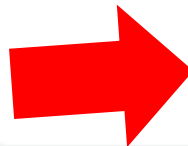


Usar MongoDB Compass

Editar documentos

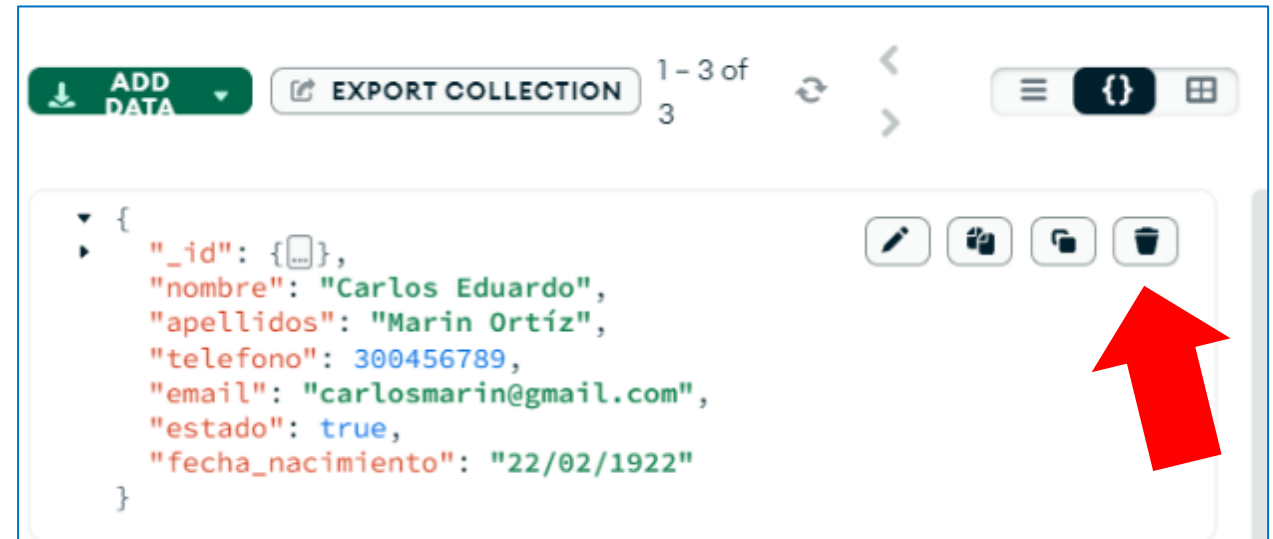
Documento actualizado.

```
_id: ObjectId('6447f21573bca1e24134b1f8')  
nombre: "Leche"  
unidad: "Litro"  
precio: 1200  
estado: true  
existencia: 5
```



Eliminar documentos

- Clic en el ícono de bote de basura.
- Clic en **DELETE**.



48 comandos y consultas de MongoDB para conocer como desarrollador y administrador de bases de datos

<https://geekflare.com/es/mongodb-queries-examples/>

Taller práctico grupal durante la sesión

Actividad Grupal

Crear una base de datos en MongoDB utilizando MongoDB Compass (interfaz gráfica) o la consola de comandos.

Los estudiantes deberán basarse en el siguiente diccionario de datos y modelo relacional, y deberán realizar consultas para validar el funcionamiento de la base de datos.

Al final, los estudiantes enviarán el script con las instrucciones necesarias para crear y poblar la base de datos.

Actividad Grupal

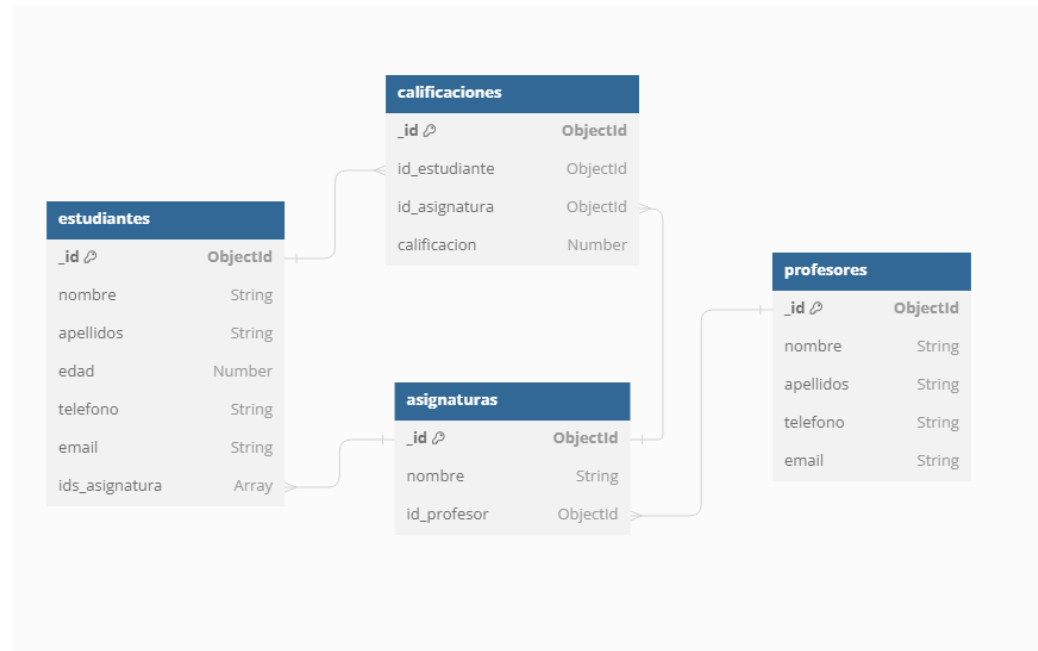
1. Crear la base de datos: Utiliza MongoDB Compass o la consola de comandos para crear la base de datos **colegio**. Agreguen al nombre el # de grupo en que están trabajando. Ejemplo: **colegio03**
2. Crea las **colecciones**: estudiantes, profesores, asignaturas y calificaciones.
3. Inserta al menos **5 documentos** en cada colección.
4. Realiza las siguientes **consultas** para validar los datos:
 - Consulta todos los estudiantes inscritos en una asignatura específica.
 - Consulta todas las calificaciones de un estudiante específico.
 - Consulta todas las asignaturas impartidas por un profesor específico.
5. Escribe un script que contenga todos los comandos necesarios para crear la base de datos, las colecciones y los documentos. Este script debe ser ejecutable en una nueva instancia de MongoDB.

Actividad Grupal

Referencias:

- Referencia 1: estudiantes.ids_asignatura > asignaturas._id
- Referencia 2: calificaciones.id_estudiante > estudiantes._id
- Referencia 3: calificaciones.id_asignatura > asignaturas._id
- Referencia 4: asignaturas.id_profesor > profesores._id

Modelo de Datos



Modelo de Datos



Colección: profesores

Campo	Tipo	Descripción	Ejemplo de Datos
_id	ObjectId	Identificador único del profesor	-
nombre	String	Nombre del profesor	"Luisa", "Javier", "Sofía"
apellidos	String	Apellidos del profesor	"García", "Martínez", "Díaz"
telefono	String	Número de teléfono del profesor	"123456789", "987654321"
email	String	Correo electrónico del profesor	"luisa@example.com", "javier@example.com"

Colección: asignaturas

Campo	Tipo	Descripción	Ejemplo de Datos
_id	ObjectId	Identificador único de la asignatura	-
nombre	String	Nombre de la asignatura	"Matemáticas", "Historia"
id_profesor	ObjectId	ID del profesor que imparte la asignatura	ObjectId("profesor_id_1"), ObjectId("profesor_id_2")

Diccionario de Datos

Colección: estudiantes

Campo	Tipo	Descripción	Ejemplo de Datos
<code>_id</code>	ObjectId	Identificador único del estudiante	-
<code>nombre</code>	String	Nombre del estudiante	"Juan", "María", "Carlos"
<code>apellidos</code>	String	Apellidos del estudiante	"Pérez", "Gómez", "López"
<code>edad</code>	Number	Edad del estudiante	20, 22, 21
<code>telefono</code>	String	Número de teléfono del estudiante	"123456789", "987654321"
<code>email</code>	String	Correo electrónico del estudiante	"juan@example.com", "maria@example.com"
<code>ids_asignatura</code>	Array	Lista de IDs de asignaturas inscritas	[ObjectId("asignatura_id_1"), ObjectId("asignatura_id_2")]

Diccionario de Datos

Colección: calificaciones

Campo	Tipo	Descripción	Ejemplo de Datos
_id	ObjectId	Identificador único de la calificación	-
id_estudiante	ObjectId	ID del estudiante	ObjectId("estudiante_id_1"), ObjectId("estudiante_id_2")
id_asignatura	ObjectId	ID de la asignatura	ObjectId("asignatura_id_1"), ObjectId("asignatura_id_2")
calificacion	Number	Calificación del estudiante en la asignatura	1, 2, 3, 4, 5

Recursos

1

<https://www.knowi.com/blog/the-best-introduction-to-mongodb-query-language-mql/>

2

<https://www.mongodb.com/docs/manual/crud/>

3

<https://www.mongodb.com/es>

4

<https://medium.com/@diego.coder/relaciones-en-mongodb-edf2107a94ad>

¡Gracias!