Preparación examen:

Ir a la carpeta setup:

* sh fichero.sh

Mongo:

**Comando (desde terminal):** mongosh  
 **Explicación:** Iniciar MongoDB shell

**Comando:** show dbs  
 **Explicación:** Muestra todas las bbdd disponibles

**Comando:** use <nombre\_base\_de\_datos>  
 **Explicación:** Cambia a una bbdd específica

**Comando:** show collections  
 **Explicación:** Lista todas las colecciones en la base de datos seleccionada

**Comando:** db.<colección>.find({})  
 **Explicación:** Devuelve todos los documentos de la colección.

**Comando:** db.<colección>.find({ campo: valor })  
 **Explicación:** Busca documentos donde el campo coincida con el valor.

**Comando:** db.<colección>.findOne()  
 **Explicación:** Muestra un solo documento, útil para ver cómo está estructurada la colección.

**Comando:** db.<colección>.findOne({ campo: valor })  
 **Explicación:** Devuelve solo el primer documento que coincida con el filtro.

**Comando:** db.<colección>.insertOne({ campo1: valor1, campo2: valor2 })  
 **Explicación:** Inserta un único documento en la colección.

**Comando:** db.<colección>.insertMany([{...}, {...}])  
 **Explicación:** Inserta múltiples documentos a la vez.

**Comando:** db.<colección>.updateOne({ filtro }, { $set: { campo: nuevo\_valor } })  
 **Explicación:** Actualiza el primer documento que cumpla con el filtro.

**Comando:** db.<colección>.updateMany({ filtro }, { $set: { campo: nuevo\_valor } })  
 **Explicación:** Actualiza todos los documentos que cumplan el filtro.

**Comando:** db.<colección>.replaceOne({ filtro }, { nuevo\_documento })  
 **Explicación:** Reemplaza un documento entero que cumpla con el filtro.

**Comando:** db.<colección>.deleteOne({ filtro })  
 **Explicación:** Elimina el primer documento que coincida con el filtro.

**Comando:** db.<colección>.deleteMany({ filtro })  
 **Explicación:** Elimina todos los documentos que coincidan con el filtro.

**Comando:** db.<colección>.find().sort({ campo: 1 })  
 **Explicación:** Ordena los resultados por un campo en orden ascendente (1) o descendente (-1)

**Comando:** db.<colección>.find().limit(5)  
 **Explicación:** Limita el número de resultados devueltos a 5.

**Comando:** db.<colección>.find().skip(5)  
 **Explicación:** Omite los primeros 5 documentos en los resultados.

**Comando:** db.<colección>.countDocuments({ filtro })  
 **Explicación:** Cuenta cuántos documentos cumplen con el filtro.

**Comando:** db.<colección>.distinct("campo")  
 **Explicación:** Devuelve los valores únicos del campo especificado.

**Comando:** db.<colección>.aggregate([...])  
 **Explicación:** Ejecuta una pipeline de agregación para procesamiento avanzado de datos.

**Comando:** db.<colección>.getIndexes()  
 **Explicación:** Muestra los índices definidos en una colección.

**Comando:** db.<colección>.stats()  
 **Explicación:** Muestra estadísticas como el tamaño, número de documentos, etc.

**Comando:** db.<colección>.drop  
 **Explicación:** Elimina completamente la colección (¡con cuidado!).

## Filtros:

Operadores para update:

* $set: Para cambiar un valor
* $unset: Para eliminar un campo
* $inc: Para incrementar un número
* $push: Para añadir un elemento a un array

Ej: db.zips.updateOne({"zip":"12534"},{"$set":{"pop":17630}})

Comparadores: (aprenderlo mamachotas)

* $eq: = (default)
* $ne: != (not equal)
* $gt: > (greater than)
* $lt: < (less than)
* $gte: >= (greater than equal)
* $lte: <=

Operadores lógicos:

* $and(default)
* $not
* $nor
* $or

Ejemplos:

Encontrar las 5 compañias más antiguas solo el nombre y año de fundación:  
db.companies.find(

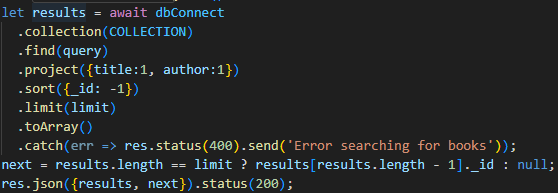
{ founded\_year: { $ne: null } }, // Filtra compañías con año fundado no nulo (especie de depuración)

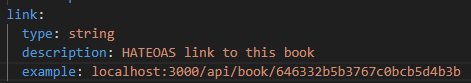
{ name: 1, founded\_year: 1, \_id: 0 } // Proyección: muestra solo nombre y año

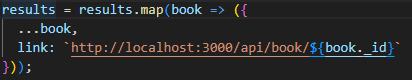
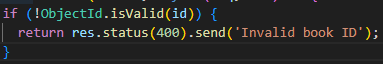
).sort({ founded\_year: 1 }) // Orden ascendente (más antiguas primero)

.limit(5) // Solo las 5 primeras

Simulacro:

* Problema 1:
  + Apartado 1:
    - Actualmente la API no se está ejecutando en la ruta que está especificada en el documento OpenAPI. Modifique el servidor para que coincidan.
    - Solución: Ir al .env → BASE\_URI=/api y poner =/api/v2 y en PORT = 3010 ---🡪 mirar en el .yaml servers (-url: localhost:3010/api/v2)
  + Apartado 2:
    - Actualmente la ruta GET /book está devolviendo la información completa de cada libro, pero eso no debería ser así. Modifique el servidor para que de cada libro se devuelva sólo la información especificada en el documento OpenAPI. 🡪 book.js
    - Solución:
    - 
  + Apartado 3:
    - Queremos hacer nuestra API restful y para eso nos falta una parte muy importante, HATEOAS. Vamos a empezar a implementarlo en alguna de las rutas, pero no queremos modificar los datos que tenemos en la base de datos. En GET /book añade a cada libro del array results un atributo link que enlace a la ruta completa de ese libro: /book/{id} De forma que por ejemplo se devuelva lo siguiente (por simplicidad sólo se muestra un libro en los resultados y puede ser que la ruta no sea correcta del todo): { "results": [ { "\_id": "646332b5b3767c0bcb5d4b3b", "title": "Speaking JavaScript", "author": "Axel Rauschmayer", "link": "localhost:3000/api/book/646332b5b3767c0bcb5d4b3b" } ], "next": null } Modifica el archivo OpenAPI para tener en cuenta esta modificación.
    - Solución: Poner en el OpenAPI



* + - Añadir en el código:
  + Apartado 4:
    - En la ruta DELETE /book/{id} no se están aplicando todas las respuestas definidas en la especificación OpenAPI. Modifique el servidor para que se tengan en cuenta todos los casos definidos.
    - Solución:
    - 
* Problema 2:
  + Apartado 1:
    - En la colección listingAndReviews indique el/los nombre(s) del alojamiento con más reviews.
    - db.listingsAndReviews.find({},{name:1, number\_of\_reviews:1}).sort({number\_of\_reviews:-1}).limit(1)
  + Apartado 2:
    - En la colección listingAndReviews indique el/los nombre(s) del alojamiento con más amenities.
  + Apartado 3:
    - En la colección listingAndReviews indique para cada tipo de property\_type el número de alojamientos de ese tipo
    - 
  + Apartado 4:
    - En la colección listingAndReviews indique el número de alojamientos que tienen 2, 3, 4 o 5 beds
    - 

Curls:

# GET simple

curl -X GET https://api.ejemplo.com/usuarios

# POST con JSON

curl -X POST https://api.ejemplo.com/usuarios \

-H "Content-Type: application/json" \

-d '{

"nombre": "Juan",

"email": "juan@correo.com"

}'

# PUT con JSON

curl -X PUT https://api.ejemplo.com/usuarios/123 \

-H "Content-Type: application/json" \

-d '{

"email": "nuevo@correo.com"

}'

# DELETE

curl -X DELETE <https://api.ejemplo.com/usuarios/123>

Más info: <https://github.com/PPerezdeMadrid/SistWebII/blob/main/apuntes/MongoDB/MongoDB.md>

<https://github.com/PPerezdeMadrid/f1-data-api/tree/main/src/f1-data-api>