Node & Mongo

Node & Mongo

1 Conexión

```
const { MongoClient, ObjectId } = require("mongodb");
const uri = "mongodb://127.0.0.1";
const client = new MongoClient(uri);
const connectToDatabase = async () => {
    try { await client.connect();
    } catch (e){  console.error(e);  }
}
const listDatabases = async (client) => {
    databasesList = await client.db().admin().listDatabases();
    console.log("Databases:");
    databasesList.databases.forEach(db => console.log(` - ${db.name}`));
}
const main = async () => {
    try {
        await connectToDatabase();
        await listDatabases(client);
    } catch (e){  console.error(e);
    } finally {  client.close();
    }
}
main();
```

client.connect()

listDatabases()

2 Insert

```
const { MongoClient, ObjectId } = require("mongodb");
const uri = "mongodb://127.0.0.1";
const client = new MongoClient(uri);
const connectToDatabase = async () => {
    try { await client.connect();
    } catch (e){  console.error(e);  }
}

const dbname = "bank";
const collection_name = "accounts";
const accountsCollection = client.db(dbname).collection(collection_name);
const sampleAccount = {
  account_holder: "Linus Torvalds",
  account_id: "MDB829001337",
  account_type: "checking",
  balance: 50352434,
}
const main = async () => {
  try {
    await connectToDatabase()
    let result = await accountsCollection.insertOne(sampleAccount);
    console.log(`Inserted document: ${result.insertedId}`);
  } catch (err) {  console.error(`Error inserting document: ${err}`);
  } finally {  await client.close();
  }
}
main();
```

Insert Many:

```
const sampleAccounts = [
  {
    account_id: "MDB011235813",
    account_holder: "Ada Lovelace",
    account_type: "checking",
    balance: 60210,
  },{
    account_id: "MDB823000001",
    account_holder: "Muhammad ibn Musa al-Khwarizmi",
    account_type: "savings",
    balance: 267914296,
  }];
const main = async () => {
  try {
    await connectToDatabase();
    let result = await accountsCollection.insertMany(sampleAccounts);
    console.log(`Inserted ${result.insertedCount} documents`);
    console.log(result);
  } catch (err) { console.error(`Error inserting documents: ${err}`);
  } finally { await client.close();
  }
}
main();
```

3 Find

```
const { MongoClient, ObjectId } = require("mongodb");
const uri = "mongodb://127.0.0.1";
const client = new MongoClient(uri);
const connectToDatabase = async () => {
    try { await client.connect();
    } catch (e){  console.error(e);  }
}

const dbname = "sample_mflix";
const collection_name = "movies";
const accountsCollection = client.db(dbname).collection(collection_name);
const documentToFind = { _id: new ObjectId("573a13adf29313caabd2b765") };

const main = async () => {
  try {
    await connectToDatabase();
    let result = await accountsCollection.findOne(documentToFind);
    console.log("Found the following document:\n", result);
  } catch (err) {
    console.error(`Error finding document: ${err}`);
  } finally {
    await client.close();
  }
}
main();
```

findOne (document)

Find (many):

```
const main = async () => {
  try {
    await connectToDatabase()
    let result = collection.find(documentsToFind).limit(5);
    let docCount = collection.countDocuments(documentsToFind);
    await result.forEach((doc) => console.log(doc));
    console.log(`Found ${await docCount} documents`);
  } catch (err) {
    console.error(`Error finding documents: ${err}`);
  } finally {
    await client.close();
  }
}
main();
```

.find (documents)

4 Update

```
const { MongoClient, ObjectId } = require("mongodb");
const uri = "mongodb://127.0.0.1";
const client = new MongoClient(uri);
const connectToDatabase = async () => {
    try { await client.connect();
    } catch (e){  console.error(e);  }
}

const dbname = "sample_mflix";
const collection_name = "movies";
const collection = client.db(dbname).collection(collection_name)
const documentToUpdate = { _id: new ObjectId("573a13adf29313caabd2b765") };
const update = { $inc: { metacritic: 1 } };
const main = async () => {
  try {
    await connectToDatabase()
    let result = await collection.updateOne(documentToUpdate, update);
    console.log(result);
    result.modifiedCount === 1
      ? console.log("Updated one document")
      : console.log("No documents updated");
  } catch (err) {
    console.error(`Error updating document: ${err}`);
  } finally {
    await client.close()
    ;
  }
}
main();
```

updateOne (doc, update)

Update many:

```
const update = { $inc: { metacritic: 1 } };
const main = async () => {
  try {
    await connectToDatabase()
    let result = await collection.updateMany(documentsToUpdate, update);
    result.modifiedCount > 0
      ? console.log(`Updated ${result.modifiedCount} documents`)
      : console.log("No documents updated");
  } catch (err) {
    console.error(`Error updating documents: ${err}`);
  } finally {
    await client.close();
  }
}
main();
```

updateMany (doc, update)

5 Delete

```
const { MongoClient, ObjectId } = require("mongodb");
const uri = "mongodb://127.0.0.1";
const client = new MongoClient(uri);
const connectToDatabase = async () => {
    try { await client.connect();
    } catch (e){  console.error(e);  }
}

const dbname = "bank";
const collection_name = "accounts";
const accountsCollection = client.db(dbname).collection(collection_name);
const documentToDelete = { _id: new ObjectId("6807ac86c53d41d592315dc1") };
const main = async () => {
  try {
    await connectToDatabase();
    let result = await accountsCollection.deleteOne(documentToDelete);
    console.log(result);
    result.deletedCount === 1
      ? console.log("Deleted one document")
      : console.log("No documents deleted");
  } catch (err) {
    console.error(`Error deleting documents: ${err}`);
  } finally {
    await client.close();
  }
}
main();
```

deleteOne (doc)

Delete many

```
const documentsToDelete = { balance: { $gt: 99000 } };
const main = async () => {
  try {
    await connectToDatabase()
    let result = await accountsCollection.deleteMany(documentsToDelete);
    console.log(result);
    result.deletedCount > 0
      ? console.log(`Deleted ${result.deletedCount} documents`)
      : console.log("No documents deleted");
  } catch (err) {
    console.error(`Error deleting documents: ${err}`);
  } finally {
    await client.close();
  }
}
main();
```

.delete Many (doc)

6 Aggregation

```
const { MongoClient, ObjectId } = require("mongodb");
const uri = "mongodb://127.0.0.1:27017";
const client = new MongoClient(uri);
const connectToDatabase = async () => {
  try { await client.connect();
  } catch (e){ console.error(e);
    }
}
const dbname = "bank";
const collection_name = "accounts";
const accounts = client.db(dbname).collection(collection_name);
const pipeline = [
  { $match: { balance: {$gt: 1000}}},
  { $group: {
    _id: "$account_type",
    total_balance: { $sum: "$balance"},
    avg_balance: { $avg: "$balance"}
  }}];
const main = async () => {
  try {
    await connectToDatabase();
    let result = await accounts.aggregate(pipeline);
    await result.forEach((doc) => console.log(doc));
  } catch (err) {
    console.error(`Error finding documents: ${err}`);
  } finally {
    await client.close();
  }
}; main();
```

Filtros → "Pipeline"

aggregate (pipeline)

1. **$match**: filtra los documentos para quedarnos solo con aquellos cuyo balance (saldo) sea mayor que 1000.
2. **$group**: agrupa esos documentos por account_type (tipo de cuenta) y calcula:
   - total_balance: la suma de los saldos ($sum: "$balance").
   - avg_balance: el promedio de los saldos ($avg: "$balance").

7 Index

```
const { MongoClient, ObjectId } = require("mongodb");
const uri = "mongodb://127.0.0.1:27017";
const client = new MongoClient(uri);
const connectToDatabase = async () => {
    try { await client.connect();
    } catch (e){ console.error(e);
    }
}
const dbname = "sample_mflix";
const collection_name = "movies";
const movies = client.db(dbname).collection(collection_name);
const main = async () => {
  try {
    await connectToDatabase();
    const result = await movies.createIndex({ title: 1 });
    console.log(`Index created: ${result}`)
  } catch (err) { console.error(`Error finding documents: ${err}`);
  } finally { await client.close();
  }
}
main();
```

createIndex

Index Search:

```
const movies = client.db(dbname).collection(collection_name);
const main = async () => {
  try {
    await connectToDatabase();
    const query = { title: /Batman.*/ };
    const sort = { year: 1 };
    const projection = { _id: 0, title: 1, year:1};
    let result = movies.find(query).sort(sort).project(projection);
    await result.forEach((doc) => console.log(doc));
  } catch (err) { console.error(`Error finding documents: ${err}`);
  } finally { await client.close();
  }
};
main();
```

(1) más rápida

8 Transactions

```
const { MongoClient} = require("mongodb");
const uri = "mongodb://127.0.0.1:27017";
const client = new MongoClient(uri);
const dbname = "bank";
const accounts = client.db(dbname).collection("accounts");
const transfers = client.db(dbname).collection("transfers");
let account_id_sender = "MDB829000001";
let account_id_receiver = "MDB829001337";
let transfer_id = "TR21072187";
let transaction_amount = 100;
const session = client.startSession();
const main = async () => {
  try {
    const transactionResults = await session.withTransaction(async () => { //Continuación
      const senderUpdate = await accounts.updateOne(
        { account_id: account_id_sender },
        { $inc: { balance: -transaction_amount } },
        { session }
      );
      const receiverUpdate = await accounts.updateOne(
        { account_id: account_id_receiver },
        { $inc: { balance: transaction_amount } },
        { session }
      );
      const transfer = {
        transfer_id: transfer_id,
        amount: transaction_amount,
        from_account: account_id_sender,
        to_account: account_id_receiver,
      };
      const insertTransferResults = await transfers.insertOne(transfer, { session });
      const updateSenderTransferResults = await accounts.updateOne(
        { account_id: account_id_sender },
        { $push: { transfers_complete: transfer.transfer_id } },
        { session }
      );
      const updateReceiverTransferResults = await accounts.updateOne(
        { account_id: account_id_receiver },
        { $push: { transfers_complete: transfer.transfer_id } },
        { session }
      );
    });
    if (transactionResults) {  console.log("Transaction completed successfully.");
    } else { console.log("Transaction failed.");}
  } catch (err) { console.error(`Transaction aborted: ${err}`)
    process.exit(1)
  } finally { await session.endSession()
    await client.close()
  }
}
main();
```

1º update sender (-$$)

2º update receiver (+$$)

3º Transfer info

4º update sender transfer info

5º update receiver transfer info

Node & Mongo

Proyecto micine

- bin
  - www
- db
  - conn.js
- public
  - stylesheets
    - style.css
- routes
  - index.js (default)
  - peliculas.js
- schema
  - cine.schema.yaml
- tests
  - app.test.js
- app.js

⚠ Añadir base de datos

yaml de OpenAPI

normal, como siempre y añadiendo las rutas /index y /peliculas
↓
(no la usaremos)

www

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

require('dotenv').config()
var app = require('../app');
var debug = require('debug')('micine:server');
var http = require('http');

require('../db/conn').connectToDatabase();

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);
```

Base de datos: conn.js

```
const { MongoClient } = require("mongodb");
const connectionString = process.env.MONGODB_URI;
console.log(connectionString);
const client = new MongoClient(connectionString);

let dbConnection;

module.exports = {
  connectToDatabase: async () => {
    try {
        await client.connect();
        dbConnection = client.db();
        console.log("Succesfully connected to database");
    } catch (e){
        console.error(e);
        process.exit();
    }
  },

  getDb: function () {
    return dbConnection;
  }
};
```

peliculas.js

```
const express = require('express');
const router = express.Router();
const dbo = require('../db/conn');
const ObjectId = require('mongodb').ObjectId;
const MAX_RESULTS = parseInt(process.env.MAX_RESULTS);
const COLLECTION = "peliculas";

router.get('/', async (req, res) => {
  let limit = MAX_RESULTS;
  if (req.query.limit){
    limit =  Math.min(parseInt(req.query.limit), MAX_RESULTS);
  }
  let next = req.query.next;
  let query = {}
  if (next){
    query = {_id: {$lt: new ObjectId(next)}}
  }
  const dbConnect = dbo.getDb();
  let results = await dbConnect
    .collection(COLLECTION)
    .find(query)
    .sort({_id: -1})
    .limit(limit)
    .project({titulo:1})
    .toArray()
    .catch(err => res.status(400).send('Error al buscar peliculas'));
  next = results.length == limit ? results[results.length - 1]._id : null;
  res.json({results, next}).status(200);
});

//getPeliculaById()
router.get('/:id', async (req, res) => {
  const dbConnect = dbo.getDb();
  let query = {_id: new ObjectId(req.params.id)};
  let result = await dbConnect
    .collection(COLLECTION)
    .findOne(query);
  if (!result){
    res.send("Not found").status(404);
  } else {
    res.status(200).send(result);
  }
});

//addPelicula()
router.post('/', async (req, res) => {
  const dbConnect = dbo.getDb();
  console.log(req.body);
  let result = await dbConnect
    .collection(COLLECTION)
    .insertOne(req.body);
  res.status(201).send(result);
});

//updatePeliculaById()
router.put('/:id', async (req, res) => {
  const query = {_id: new ObjectId(req.params.id)};
  const update = {$set:{
    titulo: req.body.titulo,
    descripcion: req.body.resumen,
    duracion: req.body.duracion
  }};
  const dbConnect = dbo.getDb();
  let result = await dbConnect
    .collection(COLLECTION)
    .updateOne(query, update);
  res.status(200).send(result);
});

//deletePeliculaById()
router.delete('/:id', async (req, res) => {
  const query = { _id: new ObjectId(req.params.id) };
  const dbConnect = dbo.getDb();
  let result = await dbConnect
    .collection(COLLECTION)
    .deleteOne(query);
  res.status(200).send(result);
});

module.exports = router;
```

pasas en el body el json con titulo descripcion duracion

*ejemplos Postman

Node & Mongo & OpenAPI

Estructura típica generada con nodejs-express-server

bash — Copiar

```
generated-user-api/
├── api/
│   └── openapi.yaml              # Tu especificación OpenAPI original
├── controllers/
│   ├── UserController.js         # Controlador generado para el endpoint /users
│   └── DefaultService.js         # Implementación base (puede estar vacía o con TOD
├── services/
│   ├── UserService.js            # Lógica de negocio (puede estar sin implementar)
│   └── ServiceFactory.js         # Fábrica para instanciar servicios
├── utils/
│   └── openapiRouter.js          # Configura las rutas desde OpenAPI
├── index.js                      # Archivo principal para iniciar el servidor
├── package.json
└── README.md
```

Ejemplo con openapi-generator-cli

```
openapi-generator-cli generate \
  -g nodejs-express-server \
  -i openapi.yaml \
  -o mi-api-generada
```

```
mi-api-generada/
├── controllers/
├── services/ → Aquí va la lógica.
├── apis/ ──────→ openapi.yml
├── models/
├── routes/
├── index.js
├── package.json
```

Ejemplo del Simulacro

bin
↳ www ⟶ require('../db/conn').connectToDatabase()

db
↳ conn.js ⟶ connectToDatabase

Public
↳ Stylesheet...

routes
↳ index.js } por defecto
↳ users.js }
↳ book.js ⟶ Aquí definimos las funciones

schema
↳ library.schema.yaml

views
↳ index.js

app.js

routes.get('/'... ⇒ GET /book getBooks()
routes.get('/:id'... ⇒ GET /book/id getBookId()
routes.post('/'... ⇒ POST /book addBook()
routen.delete('/:id'... ⇒ DELETE /book/id deleteBookId()

no tocar → controllers llama a los servicios
↳ Controllers.js → Punto central de los controles — Sirve como "registro" o "router" interno a conecta las rutas generadas con los servicios correspondientes
↳ DefaultControllers.js → Punto de entrada para la API

Services
↳ index.js
↳ Service.js
↳ DefaultServices.js ⟶ Aquí hemos del los funciones — bookGET, bookID POST ...

Ejemplo Conexión con Mongo

```
const driversGET = ({ offset = 0, limit = 10 }) => new Promise(
  async (resolve, reject) => {
    try {
      offset = parseInt(offset);
      limit = parseInt(limit);

      const drivers = await mongoose.connection.db.collection('drivers')
        .find({}, { projection: { _id: 0 } })
        .skip(offset)
        .limit(limit)
        .toArray();

      const total = await mongoose.connection.db.collection('drivers').countDocuments();

      return resolve(Service.successResponse({
        offset,
        limit,
        total,
        drivers
      }));
    } catch (e) {
      reject(Service.rejectResponse(e.message || 'Invalid input', e.status || 405));
    }
  },
);
```

uso de mongoose