Mongoose

Julian F. Latorre

Índice

1. Introducción

Este taller práctico te guiará a través del uso de Mongoose y MongoDB en Node.js. Aprenderás las diferencias clave entre ambos enfoques mediante ejercicios prácticos.

2. Configuración del Entorno

Ejercicio 1: Configuración

1. Instala Node.js si aún no lo tienes.
2. Crea un nuevo directorio para el proyecto e inicializa un nuevo proyecto Node.js:

```
mkdir taller-mongoose
cd taller-mongoose
npm init -y
```

3. Instala las dependencias necesarias:

```
npm install mongodb mongoose
```

3. Conexión a la Base de Datos

Ejercicio 2: Conexión con MongoDB

Crea un archivo llamado `mongodb-connect.js` y escribe el código para conectarte a MongoDB:

```
const MongoClient = require('mongodb').MongoClient;
const url = 'mongodb://localhost:27017/tallerDB';

MongoClient.connect(url, (err, client) => {
  if (err) throw err;
  console.log('Conectado a MongoDB');
  const db = client.db('tallerDB');
  // Aquí irían las operaciones con la base de datos
  client.close();
});
```

Ejecuta el script y verifica que puedes conectarte a MongoDB.

Ejercicio 3: Conexión con Mongoose

Crea un archivo llamado `mongoose-connect.js` y escribe el código para conectarte usando Mongoose:

```
const mongoose = require('mongoose');
const url = 'mongodb://localhost:27017/tallerDB';

mongoose.connect(url, { useNewUrlParser: true,
    useUnifiedTopology: true });

const db = mongoose.connection;
db.on('error', console.error.bind(console, 'error de conexión
    :'));
db.once('open', function() {
  console.log('Conectado a MongoDB con Mongoose');
});
```

Ejecuta el script y verifica la conexión.

4. Definición de Esquemas y Modelos

Ejercicio 4: Creación de un Esquema en Mongoose

En el archivo `mongoose-connect.js`, añade el siguiente código para definir un esquema y modelo de usuario:

```
const usuarioSchema = new mongoose.Schema({
  nombre: String,
  email: { type: String, required: true, unique: true },
  edad: Number
});

const Usuario = mongoose.model('Usuario', usuarioSchema);
```

5. Operaciones CRUD

Ejercicio 5: Crear (MongoDB vs Mongoose)

Crea dos archivos: `mongodb-create.js` y `mongoose-create.js`.
En `mongodb-create.js`:

```
// ... código de conexión ...
const usuario = { nombre: 'Ana', email: 'ana@ejemplo.com',
    edad: 28 };
db.collection('usuarios').insertOne(usuario, (err, result) =>
     {
  if (err) throw err;
  console.log('Usuario insertado con MongoDB');
  client.close();
});
```

En `mongoose-create.js`:

```
// ... código de conexión y esquema ...
const nuevoUsuario = new Usuario({ nombre: 'Ana', email: '
    ana@ejemplo.com', edad: 28 });
nuevoUsuario.save((err) => {
  if (err) throw err;
  console.log('Usuario guardado con Mongoose');
  mongoose.connection.close();
});
```

Ejecuta ambos scripts y compara los resultados.

Ejercicio 6: Leer (MongoDB vs Mongoose)

Crea `mongodb-read.js` y `mongoose-read.js`.
En `mongodb-read.js`:

```
// ... código de conexión ...
db.collection('usuarios').find({ nombre: 'Ana' }).toArray((
    err, docs) => {
  if (err) throw err;
  console.log('Usuarios encontrados con MongoDB:', docs);
  client.close();
});
```

En `mongoose-read.js`:

```
// ... código de conexión y esquema ...
Usuario.find({ nombre: 'Ana' }, (err, usuarios) => {
  if (err) throw err;
  console.log('Usuarios encontrados con Mongoose:', usuarios)
    ;
  mongoose.connection.close();
});
```

Ejecuta y compara los resultados.

Ejercicio 7: Actualizar (MongoDB vs Mongoose)

Crea `mongodb-update.js` y `mongoose-update.js`.
En `mongodb-update.js`:

```
// ... código de conexión ...
db.collection('usuarios').updateOne(
  { nombre: 'Ana' },
  { $set: { edad: 29 } },
  (err, result) => {
    if (err) throw err;
    console.log('Usuario actualizado con MongoDB');
    client.close();
  }
);
```

En `mongoose-update.js`:

```
// ... código de conexión y esquema ...
Usuario.updateOne({ nombre: 'Ana' }, { edad: 29 }, (err) => {
  if (err) throw err;
  console.log('Usuario actualizado con Mongoose');
  mongoose.connection.close();
});
```

Ejecuta ambos scripts y verifica los cambios.

Ejercicio 8: Eliminar (MongoDB vs Mongoose)

Crea `mongodb-delete.js` y `mongoose-delete.js`.
En `mongodb-delete.js`:

```
// ... código de conexión ...
db.collection('usuarios').deleteOne({ nombre: 'Ana' }, (err,
    result) => {
  if (err) throw err;
  console.log('Usuario eliminado con MongoDB');
  client.close();
});
```

En `mongoose-delete.js`:

```
// ... código de conexión y esquema ...
Usuario.deleteOne({ nombre: 'Ana' }, (err) => {
  if (err) throw err;
  console.log('Usuario eliminado con Mongoose');
  mongoose.connection.close();
});
```

Ejecuta los scripts y verifica los resultados.

6. Características Avanzadas de Mongoose

Ejercicio 9: Validación con Mongoose

Modifica el esquema de usuario en `mongoose-connect.js`:

```
const usuarioSchema = new mongoose.Schema({
  nombre: { type: String, required: true },
  email: {
    type: String,
    required: true,
    unique: true,
    validate: {
      validator: function(v) {
        return /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$
    /.test(v);
      },
      message: props => `${props.value} no es un email válido
    !`
    }
  },
  edad: { type: Number, min: 18, max: 100 }
});
```

Crea un nuevo archivo `mongoose-validation.js` e intenta guardar usuarios con datos inválidos.

Ejercicio 10: Middleware en Mongoose

Añade un middleware al esquema de usuario:

```
usuarioSchema.pre('save', function(next) {
  console.log('Guardando usuario:', this.nombre);
  next();
});

usuarioSchema.post('save', function(doc) {
  console.log('Usuario guardado:', doc.nombre);
});
```

Crea un nuevo usuario y observa los mensajes de los middlewares.

7. Conclusión y Reflexión

Ejercicio Final: Comparación y Reflexión

Después de completar todos los ejercicios, responde a las siguientes preguntas:

1. ¿Qué enfoque (MongoDB nativo o Mongoose) te pareció más fácil de usar? ¿Por qué?
2. ¿Cuáles son las principales ventajas que notaste al usar Mongoose?
3. ¿En qué situaciones crees que sería preferible usar MongoDB nativo en lugar de Mongoose?
4. ¿Cómo crees que las características de validación y middleware de Mongoose podrían beneficiar a un proyecto real (o al proyecto del curso)?