


Conectando con MongoDB

 gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/conectando-con-mongodb.html

Conectando MongoDB

Antes de continuar vamos a mezclar el *feature* anterior y crear una rama para trabajar:

```
git checkout master
git merge feature/08-handling-params-from-requests
git checkout -b feature/09-using-mongoose
```

Una vez MongoDB esté ejecutándose en nuestra máquina, podemos conectarlo con Node JS, para esto vamos a utilizar una librería llamada *mongoose*, ampliamente utilizada y provee muchas otras funcionalidades adicionales, procedemos a instalar nuestra librería como dependencia:

```
npm install -S mongoose
```

Vamos a necesitar un URL que nos indique a que base de datos debemos conectarnos, para esto vamos a añadirla en nuestro archivo de configuración (*server/config/index.js*):

```
require('dotenv').config();

const config = {
  server: {
    hostname: process.env.SERVER_HOSTNAME,
    port: process.env.SERVER_PORT,
  },
  database: {
    url: process.env.DATABASE_URL,
  },
};

module.exports = config;
```

Y añadimos la llave en el archivo *.env* :

```
SERVER_HOSTNAME=127.0.0.1
SERVER_PORT=3000
DATABASE_URL=mongodb://127.0.0.1/photobook
```

Como podemos observar el protocolo de conexión será *mongodb* en nuestra máquina local *localhost* y el nombre de la base de datos será *photobook* . Vale la pena nombrar que Mongo DB utilizar el puerto por defecto: 27017.

Creamos nuestro archivo que administrará todas las operaciones relacionadas con la conexión de la base de datos, creamos el archivo *server/database.js* con el siguiente contenido:

```

const mongoose = require('mongoose');
const logger = require('winston');

const config = require('./config');

exports.connect = () => {
  const {
    database,
  } = config;

  mongoose.connect(database.url);

  mongoose.connection.on('open', () => {
    logger.info('Database connected');
  });

  mongoose.connection.on('close', () => {
    logger.info('Database disconnected');
  });

  mongoose.connection.on('error', (err) => {
    logger.error(`Database connection error: ${err}`);
  });

  process.on('SIGINT', () => {
    mongoose.connection.close(() => {
      logger.info('Database connection disconnected through app termination');
      process.exit(0);
    });
  });
};

```

Explicuemos con mas detalle en el fragmento de código anterior:

1. Importamos las librerías necesarias para hacer la conexión y dejar una constancia de los eventos, también cargamos nuestra configuración.
2. Exportamos del módulo una función llamada connect que se encargará de realizar la conexión a la base de datos.
3. Dentro de la función tomamos solo la configuración correspondiente a la base de datos.
4. Invocamos la función de conexión a la base de datos
5. Creamos unos listeners para escuchar los eventos de la conexión, en este caso cuando la conexión fue exitosa, cuando se cerro la conexión o cuando hubo un error.
6. El último fragmento de código es un evento del sistema de Node JS que indica cuando se cerró la aplicación, puede ser debido a varias causas pero lo importante es que antes de cerrar la aplicación nos desconectamos de la base de datos para liberar esa conexión pues ya no se utilizará mas.

Ahora en nuestro archivo del servidor (`server/index.js`) añadimos nuestro paquete, la configuración y nos conectamos:

...

```
const bodyParser = require('body-parser');
```

```
const database = require('./database');
```

```
const api = require("./api/v1");
```

```
database.connect();
```

```
const app = express();
```

...

Si `nodemon` se está ejecutando, guardamos el archivo y no vemos ningún error, entonces quiere decir que todo el proceso fue satisfactorio, de lo contrario debemos revisar que el *daemon* de MongoDB se esté ejecutando y/o esté correctamente instalado.

Antes de continuar guardamos nuestro progreso:

```
git add .
```

```
git commit -m "Add mongoose and connect to database"
```

Más información:

[Mongoose](#)