Desarrollo de Aplicaciones Web

Tema Nº10:NODE JS – DATABASE II

Indicador de logro Nº10:Construye aplicaciones CRUD con conexión a base de datos a través de Node.JS en una arquitectura orientada a eventos y basado en el motor V8.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº10:**

NODE JS – DATABASE II

**Subtema 10.1:**

Comunicación con el gestor de base de datos MongoDB

En su mayoría, todas las aplicaciones web modernas tienen algún tipo de sistema de almacenamiento de datos en el backend. Por ejemplo, si toma el caso de una aplicación de compras web, datos como el precio de un artículo se almacenarán en la base de datos.

El framework Node js puede trabajar con bases de datos tanto relacionales (como Oracle y MS SQL Server) como no relacionales (como MongoDB). En este tutorial, vamos a ver cómo podemos utilizar bases de datos desde dentro de las aplicaciones Node js.

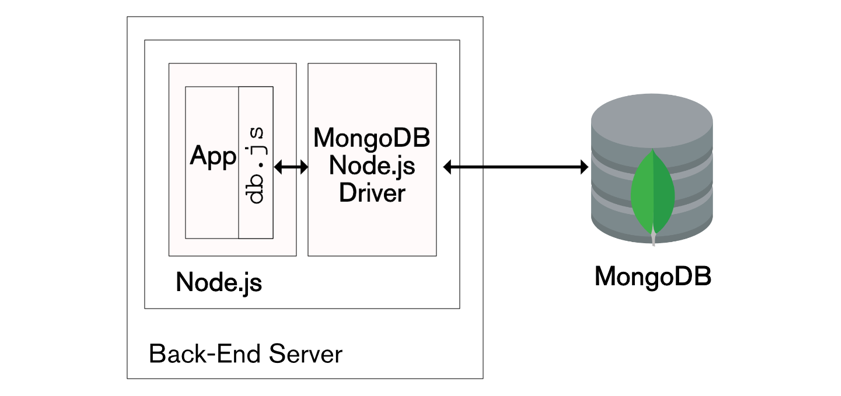
**Bases de datos Node.js y NoSQL**

Con los años, la base de datos NoSQL como MongoDB y MySQL se han vuelto bastante populares como bases de datos para almacenar datos. La capacidad de estas bases de datos para almacenar cualquier tipo de contenido y particularmente en cualquier tipo de formato es lo que hace que estas bases de datos sean tan famosas.

Node.js tiene la capacidad de trabajar con MySQL y MongoDB como bases de datos. Para utilizar cualquiera de estas bases de datos, es necesario descargar y utilizar los módulos necesarios mediante el administrador de paquetes de Node. Para MySQL, el módulo requerido se llama “mysql”y para usar MongoDB el módulo requerido para ser instalado es “Mongoose”

Con estos módulos, puede realizar las siguientes operaciones en Node.js

1. Administrar la agrupación de conexiones — Aquí es donde puede especificar el número de conexiones de base de datos MySQL que Node.js debe mantener y guardar.
2. Cree y cierre una conexión a una base de datos. En cualquier caso, puede proporcionar una función de devolución de llamada que se puede llamar siempre que se ejecuten los métodos de conexión “create”y “close”
3. Las consultas se pueden ejecutar para obtener datos de las respectivas bases de datos para recuperar datos.
4. La manipulación de datos, como la inserción de datos, la eliminación y la actualización de datos, también se puede lograr con estos módulos.



**Subtema 10.2:**

¿Qué es una base de datos NoSQL?

Tenemos que empezar por saber que es una base de dato NoSQL (Not only SQL), básicamente son bases de datos no relacionales, con pequeñas características que las definen como:

* No solo manejan SQL como lenguaje de consulta
* Los datos no se almacenan en tablas
* No necesita de un esquema

**Subtema 10.3:**

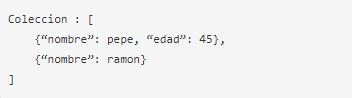
¿Qué es MongoDB?

Bueno ahora que tenemos más en claro que es una base de datos NoSQL podemos decir que MongoDB es un motor de base de datos NoSQL.

Fue desarrollado con el pensamiento de trabajar y manejar grandes cantidades de datos de una manera rápida y eficiente. Esto nos sirve mucho en aplicaciones a gran escala que necesitan consultar muchos datos en poco tiempo.

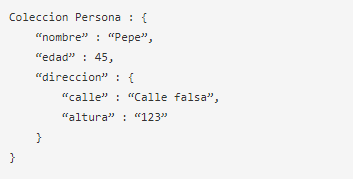
Una cosa importante a tener en cuenta es que MongoDB usa json para almacenar su información, por lo tanto, si estamos acostumbrados a estos nos será sencillo trabajar con esta base de datos.

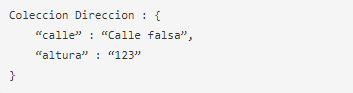
MongoDB poseen colecciones, que en el otro modelo (SQL) sería como las tablas, y dentro de estas tenemos documentos que son los objetos que queremos almacenar.Como es sin esquema nuestras colecciones puede tener documentos diferentes uno de otro, por ejemplo:



Muchos creen que en esta base de datos no existen las relaciones y eso no es del todo cierto, Mongo maneja un concepto llamado “nested-documents” que va a ser nuestra forma de relacionar 2 o más documentos, lo diferente con las bases de datos relacionales es que mientras ahí guardan un id de referencia, en esta vamos a guardar todo el objeto entero. Para verlo más claro, vamos con un ejemplo:

Imaginemos que tenemos 2 colecciones, una “Persona” y una “Direccion”, la relación seria que una persona tiene una dirección, ahora veamos cómo se manejaría en Mongo esto.



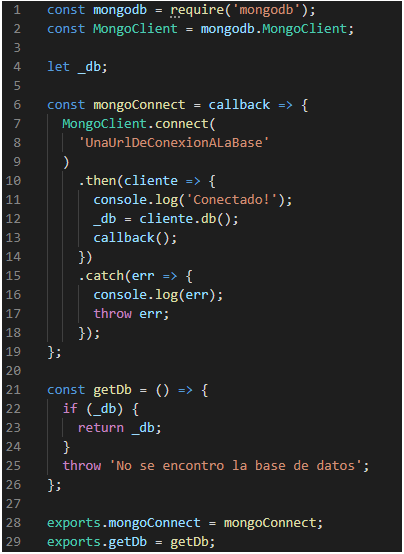


Como podemos ver el documento dirección está contenido dentro del documento de esta persona, es así como las relaciones se manejan, y esto hace que el buscar datos de este tipo sea tan rápido ya que no tengo que estar armando relaciones entre distintos objetos, solo con consultar a “Persona” puedo obtener todo lo que necesito.

**Subtema 10.4:**

MongoDB y Node

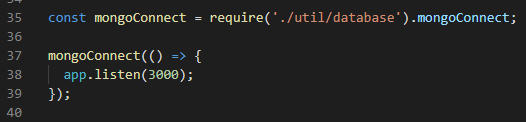
¿Como hago para usar mongo en mi proyecto con node? Esto es bastante simple gracias al módulo que nos brinda npm. Lo primero como siempre, es establecer la conexión a la base. Esto lo logramos de la siguiente forma:



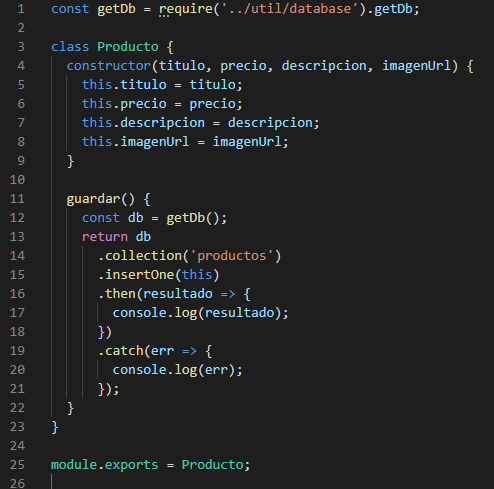
Los pasos son simples, importamos el módulo de MongoDB y creamos un cliente de mongo. También tendremos que crear una variable donde almacenaremos nuestra conexión a la base para que podamos recuperarla al momento de usarla.

Por último, pero no menos importante, definiremos una función constante que recibira un **callback**, esta lo único que ejecutará será el método “**connect**” del cliente de mongo, al cual deberemos informarle la url de conexión a la base.

El método **connect** nos devolverá una **promise** que en caso de que salga todo bien, nos devolverá un cliente conectado, a nosotros de este nos interesa el método db (que será lo que usemos para realizar peticiones la base. Y por último ejecutamos el **callback** y esto nos servirá para iniciar el servidor de **node** como se muestra a continuación.



Por último, pero no menos importante, veamos cómo interactuamos con nuestra conexión en node. En la siguiente imagen, hay un ejemplo de cómo se hace esto.



Lo primero que tenemos que hacer es importar nuestro método para obtener la conexión activa de Mongo, después en nuestra clase “Producto” vamos a tener un método guardar.

Y eso es todo lo que necesitamos saber para empezar a jugar con MongoDB y node, existen muchos métodos que nos servirán para realizar lo que queramos con la base, eso les queda a ustedes para seguir investigando y aprendiendo.

**1. MATERIALES**

Para la experiencia a realizar se requiere lo siguiente:

1. EQUIPO

Se requiere tener un equipo de cómputo (PC o laptop).

1. HERRAMIENTAS

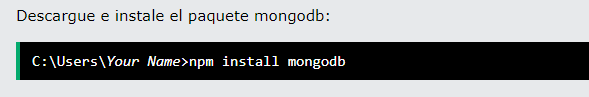
* Visual Stude Code
* Node.js
* Npm
* JavaScript
* MongoDB

**2. PROCEDIMIENTO**

Para poder experimentar con los ejemplos de código, necesitará acceso a una base de datos MongoDB. Puede descargar una base de datos gratuita de MongoDB en https://www.mongodb.com o comience de inmediato con un servicio en la nube de MongoDB en https://www.mongodb.com/cloud/atlas.

**Instalar el controlador MongoDB**

Intentemos acceder a una base de datos MongoDB con Node.js. Para descargar e instalar el controlador oficial de MongoDB, abra Command Terminal y ejecute lo siguiente:

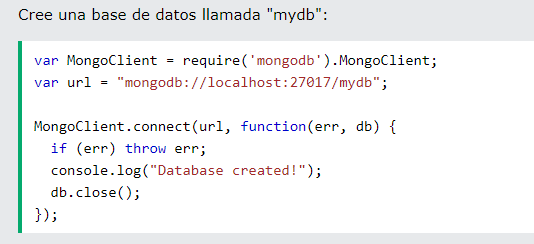


Ahora ha descargado e instalado un controlador de base de datos mongodb. Node.js puede usar este módulo para manipular bases de datos MongoDB:



**Node.js MongoDB Crear base de datos**

Para crear una base de datos en MongoDB, comience creando un objeto MongoClient, luego especifique una URL de conexión con la dirección IP correcta y el nombre de la base de datos que desea crear. MongoDB creará la base de datos si no existe y se conectará a ella.



Guarde el código anterior en un archivo llamado "demo\_create\_mongo\_db.js" y ejecute el archivo:



Importante: ¡En MongoDB, no se crea una base de datos hasta que obtiene contenido!

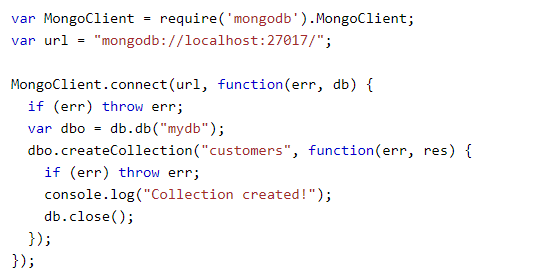
MongoDB espera hasta que haya creado una colección (tabla), con al menos un documento (registro) antes de que realmente cree la base de datos (y la colección).

**Node.js MongoDB Create Collection**

Una colección en MongoDB es lo mismo que una tabla en MySQL. Para crear una colección en MongoDB, use el **createCollection()** método:

Ejemplo

Crea una colección llamada "clientes":



Guarde el código anterior en un archivo llamado "demo\_mongodb\_createcollection.js" y ejecute el archivo:



Lo que te dará este resultado:



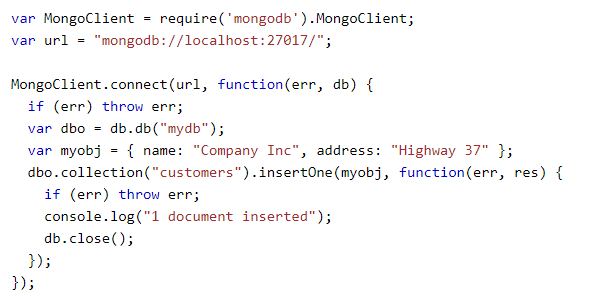
**Node.js MongoDB - INSERT**

Para insertar un registro, o documento como se llama en MongoDB, en una colección, usamos el **insertOne()** método.

El primer parámetro del **insertOne()** método es un objeto que contiene los nombres y valores de cada campo en el documento que desea insertar.

También necesita una función de devolución de llamada donde puede trabajar con cualquier error o el resultado de la inserción:

Ejemplo: Inserte un documento en la colección "clientes":



Guarde el código anterior en un archivo llamado "demo\_mongodb\_insert.js" y ejecute el archivo:



Lo que te dará este resultado:



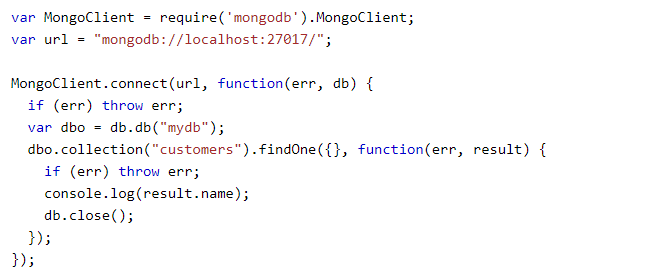
**Node.js MongoDB – FIND ONE**

En MongoDB usamos los métodos **find** y **findOne** para buscar datos en una colección. Al igual que la instrucción SELECT se usa para buscar datos en una tabla en una base de datos MySQL.

Para seleccionar datos de una colección en MongoDB, podemos usar el **findOne()** método. El **findOne()** método devuelve la primera aparición en la selección.

El primer parámetro del **findOne()** método es un objeto de consulta. En este ejemplo usamos un objeto de consulta vacío, que selecciona todos los documentos de una colección (pero devuelve solo el primer documento).

Ejemplo: Encuentre el primer documento en la colección de clientes:



Guarde el código anterior en un archivo llamado "demo\_mongodb\_findone.js" y ejecute el archivo:

Ejecute "demo\_mongodb\_findone.js"



Lo que te dará este resultado:

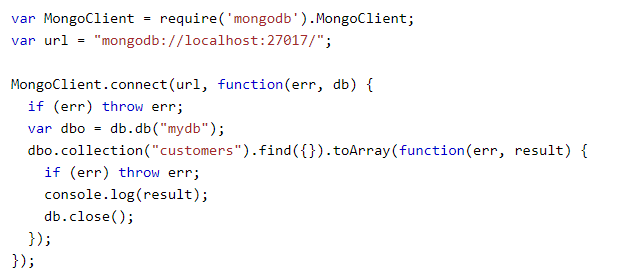


**Node.js MongoDB – FIND ALL**

Para seleccionar datos de una tabla en MongoDB, también podemos usar el **find()** método. El **find()** método devuelve todas las apariciones de la selección.

El primer parámetro del **find()** método es un objeto de consulta. En este ejemplo usamos un objeto de consulta vacío, que selecciona todos los documentos de la colección.

Ejemplo: Encuentre todos los documentos en la colección de clientes:



Guarde el código anterior en un archivo llamado "demo\_mongodb\_find.js" y ejecute el archivo:



Lo que te dará este resultado:



**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

Revisa y analiza el tema desarrollado en la presente sesión, luego responde las siguientes preguntas propuestas:

* ¿Qué es MongoDB y Node JS?
* ¿Cómo se conecta Node JS con MongoDB?
* ¿Qué ventajas y desventajas se tiene al usar una base de datos NoSQL? De una breve explicación
* Realice una App aplicando todo lo aprendido de esta guía

1. **ENLACES DE REFERENCIA**

* https://guru99.es/node-js-mongodb/
* https://somospnt.com/blog/163-mongodb-con-node-js
* https://www.w3schools.com/nodejs/nodejs\_mongodb.asp

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_