UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Programador Web Avanzado

Clase N° 4: Mongo DB

Profesor: Ing. Leandro Rodolfo Gil Carrano
Email: leangilutn@gmail.com

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Mongo DB

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Base de datos NO SQL

Difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales, en inglés, *RDBMS*) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad), y habitualmente escalan bien horizontalmente. Los sistemas NoSQL se denominan a veces "no sólo SQL" para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Base de datos NO SQL

Los sistemas de bases de datos NoSQL crecieron con las principales compañías de Internet, como Google, Amazon, Twitter y Facebook. Estas tenían que enfrentarse a desafíos con el tratamiento de datos que las tradicionales RDBMS no solucionaban

Con el crecimiento de la web en tiempo real existía una necesidad de proporcionar información procesada a partir de grandes volúmenes de datos que tenían unas estructuras horizontales más o menos similares.

No sql – Ventajas

- Estos sistemas responden a las necesidades de escalabilidad horizontal que tienen cada vez más empresas.[3]
- Pueden manejar enormes cantidades de datos.
- No generan cuellos de botella.
- Escalamiento sencillo.
- Diferentes DBs NoSQL para diferentes proyectos.
- Se ejecutan en clusters de máquinas baratas.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

No sql – Desventajas

- Los principales proveedores de RDBMS tales como Oracle, IBM o Microsoft ofrecen buenos soportes a pequeñas, medianas y grandes empresas y típicamente start-ups.
- No están lo suficientemente maduros para algunas empresas.- A pesar de sus puestas en práctica en algunas grandes empresas, las bases de datos NoSQL aún se enfrentan a un problema de credibilidad importante con muchas empresas.
- Limitaciones de Inteligencia de Negocios.- Hay una o dos cuestiones acerca de las capacidades de BI de las bases de datos NoSQL. ¿Pueden estas bases de datos proporcionar la clase de minería de datos rigurosos que las empresas se utilizan con las RDBMSes? ¿Cuántos conocimientos de programación se necesitan para hacer la consulta ad hoc y análisis?.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

No sql – Desventajas

- La falta de experiencia.- La novedad de NoSQL significa que no hay una gran cantidad de desarrolladores y administradores que conocen la tecnología -lo que hace difícil a las empresas encontrar personas con los conocimientos técnicos apropiados. Por el contrario, el mundo RDBMS tiene miles de personas muy cualificadas.
- Problemas de compatibilidad.- A diferencia de las bases de datos relacionales, que comparten ciertos estándares, las bases de datos NoSQL tienen pocas normas en común. Cada base de datos NoSQL tiene su propia API, las interfaces de consultas son únicas y tienen peculiaridades. Esta falta de normas significa que es imposible cambiar simplemente de un proveedor a otro, por si no quedara satisfecho con el servicio.

No sql – ¿Cuándo utilizarla?

- Velocidad. Si una aplicación necesita almacenar o acceder a mucha información en poco tiempo, se necesita una base de datos que aporte gran velocidad. Las bases de datos documentales son capaces de ser mucho más rápidas que las relacionales, pudiendo atender clientes que necesiten realizar muchas operaciones por segundo.

No sql – ¿Cuándo utilizarla?

- Volumen. En cuanto al tamaño de la base de datos, si tenemos una cantidad de información gigante o enorme, entonces tenemos unas necesidades importantes de volumen. Las bases de datos relacionales tienen tendencia a funcionar más lentamente cuando en una tabla se encuentran cantidades muy grandes de registros (del orden de un millón para arriba). Situaciones así obligan a los administradores a buscar soluciones, como dividir las tablas en diversos segmentos, produciendo un coste en el acceso a los datos y la operativa. Este no es un problema en las bases de datos NoSQL, que son capaces de administrar volúmenes gigantescos de datos en sus entidades.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

No sql – ¿Cuándo utilizarla?

•Variabilidad. Las necesidades enormes de velocidad y volumen suelen darse juntas y afectan a muchas aplicaciones actuales. Sin embargo, hay otra característica de la información que es todavía más representativa para decantarse por las NoSQL, como es la variabilidad. En bases de datos relacionales el esquema de la información está minuciosamente definido de antemano. Por ejemplo, no puedes inventarte campos en los registros sobre la marcha. En las bases de datos documentales, como MongoDB, no hay problema en que cada documento almacene campos distintos, pudiendo ser flexibles en cuanto al esquema de la información.

No sql – ¿Cuándo utilizarla?

Lo que a día de hoy no se encuentra disponible en las NoSQL son los mecanismos para hacer transacciones entre varios documentos. Debido a esto, pueden no ser la opción más adecuada para ciertas operativas de negocio, ya que se tendrá que profundizar mucho en los mecanismos para almacenar la información y puede no compensar el esfuerzo.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

No sql – Tipos

- Orientadas a documentos:

Son aquellas que gestionan datos semi estructurados. Es decir documentos. Estos datos son almacenados en algún formato estándar como puede ser XML, JSON o BSON.

```
{
  Name: "Genbeta Dev",
  Tipo: "Blogging",
  Categorias:
  [
    {
      Título: "Desarrollo",
      Articulos: 89
    },
    {
      Título: "Formación",
      Artículos: 45
    }
  ]
}
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

No sql – Tipos

Son las bases de datos NoSQL más versátiles. Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

No sql – Tipos

- Orientadas a columnas:

Este tipo de bases de datos están pensadas para realizar consultas y agregaciones sobre grandes cantidades de datos. Funcionan de forma parecida a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

No sql – Tipos

- Clave valor:

Estas son las más sencillas de entender. Simplemente guardan tuplas que contienen una clave y su valor. Cuándo se quiere recuperar un dato, simplemente se busca por su clave y se recupera el valor.

DynamoDB

amazon
web services™

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

No sql – Tipos

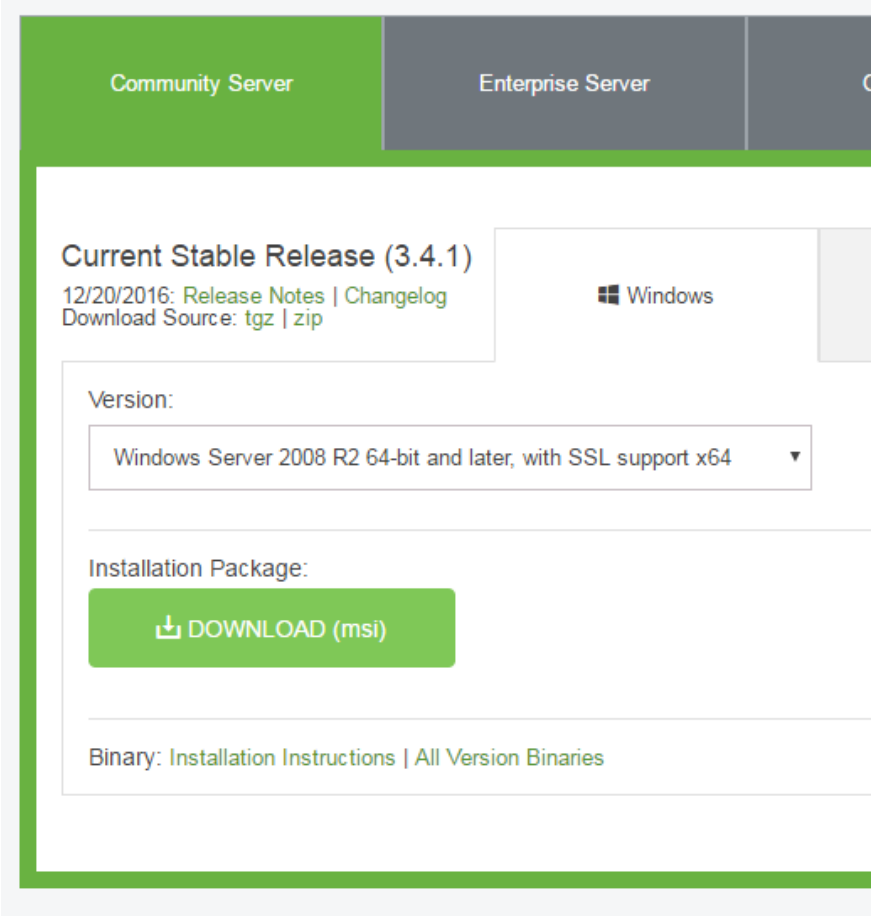
- Grafos:

Basadas en la teoría de grafos utilizan nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con muchas relaciones, como redes y conexiones sociales.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Instalación

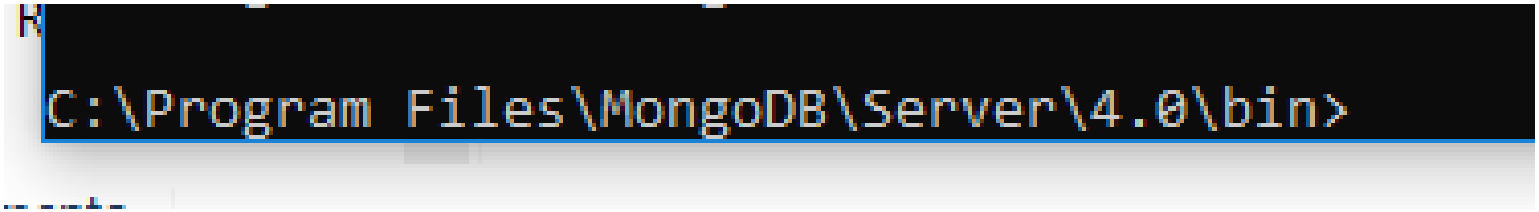
- Ingresar a http://www.mongodb.org/downloads

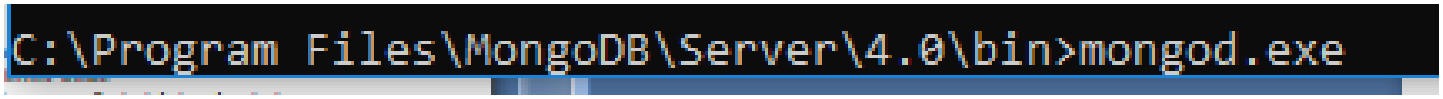


UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Instalación

- Crear la carpeta **C:\data\db**
- Abrir la consola (Ejecutar -> cmd)
- Ejecutar lo siguiente:





UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Instalación en Linux

- Ejecutar **sudo apt-get update**
- Ejecutar **sudo apt-get install mongodb**

Para corroborar si la instalacion se realizo correctamente ejecutar: **mongo –version**

Para iniciar, detener y reiniciar el servicio se ejecuta cualquiera de las siguientes líneas:
sudo service mongodb start | stop | restart

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Moongose

Para conectarnos con la base de datos de mongoDB debemos realizar lo siguientes pasos:

- Instalar el modulo de moongose

```
C:\Users\Leandro\node_app>npm install mongoose
```

- Luego debemos realizar la conexión de nuestra aplicación en node con mongo a través de moongose

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/primer_base', function(error){
    if(error){
        throw error;
    }else{
        console.log('Conectado a MongoDB');
    }
});
```

Ver archivo **mongodb.js**

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

CRUD

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Crud

La palabra CRUD significa Create(Crear), Read(Leer), Update(Modificar), Delete(Eliminar); y se refiere a las operaciones que podemos realizar en los registros de una base de datos, en el caso de MongoDB, en los documentos.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Crud

Crear archivo mongodb.js dentro de bin

```
var mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/startup', { useNewUrlParser: true }, function(error){
    if(error){
        throw error;
    }else{
        console.log('Conectado a MongoDB');
    }
});

module.exports = mongoose;
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Desde el modelo o router incluir mongodb

Debemos incluir el archivo definido en el directorio bin, que incluye el modulo mongoose

```
var mongoose = require("../bin/mongodb");
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Crear schema

Lo primero que debemos hacer para poder realizar operaciones en nuestra base de datos **mongoDB** es crear el schema. Llamaremos al método Schema del objeto mongoose y le pasaremos un json de configuración como parámetro.

```
//Crear schema
var UsuarioSchema = mongoose.Schema({
    nombre: String,
    apellido: String,
    usuario: String,
    password: String
});
```

Node JS – Crear modelo

Luego debemos crear el modelo de datos. Por ejemplo crearemos el modelo **Usuario**

```
//Crear schema
var UsuarioSchema = mongoose.Schema({
    nombre: String,
    apellido: String,
    usuario: String,
    password: String
});
//Crear modelo usuario
var Usuario = mongoose.model('Usuario', UsuarioSchema);
```

Al método **model** del objeto mongoose le debemos pasar por parámetro el nombre del modelo a crear y como segundo parámetro el schema.
Luego instanciamos al modelo creado.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Insertar datos

Debemos llamar al método **save** del modelo a insertar.
Como parámetro le pasaremos la función de callback

```
var usuario = new Usuario({
    nombre: data.nombre,
    apellido: data.apellido,
    usuario: data.usuario,
    password: data.password
});
//Insertar cliente
usuario.save(function(error, documento){
    if(error){
        return reject(error);
    }else{
        resolve(documento)
    }
});
```

Node JS – Insertar datos

La función de callback recibirá como parámetros:

- Variable error: Da información acerca del error producido
- Variable documento: Información del documento insertado

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Leer todos los datos

Debemos llamar al método **find** del modelo.
Como parámetro le pasaremos un json vacio y la función de callback

```
Usuario.find({}, function(error, documento){
    if(error){
        return reject(error);
    }else{

        resolve(documento);
    }
})
```

La función de callback recibirá como parámetros:

- Variable error: Da información acerca del error producido
- Variable clientes: Información del documento a leer

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Leer todos los datos

```
C:\Users\Leandro\node_crud>node leer.js
Conectado a MongoDB
[ { _id: 5862faec97f2b90bdcd04db4,
    nombre: 'leandro',
    apellido: 'Gil',
    domicilio: 'Domicilio',
    telefono: '123456',
    email: 'leangilutn@gmail.com',
    __v: 0 },
  { _id: 5862fb0f3e67142654a2f5f7,
    nombre: 'leandro',
    apellido: 'Gil',
    domicilio: 'Domicilio',
    telefono: '123456',
    email: 'leangilutn@gmail.com',
    __v: 0 } ]
```

Node JS – Leer por id

Debemos llamar al método **findById** del modelo.
Como parámetro le pasaremos el id y la función de callback

```
Usuario.findById(id, function(error, documento){
    if(error){
        return reject(error);
    }else{

        resolve(documento);
    }
})
```

La función de callback recibirá como parámetros:

- Variable error: Da información acerca del error producido
- Variable documento: Información del documento a leer

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Leer por id

```
{ _id: 5862faec97f2b90bdcd04db4,
  nombre: 'leandro',
  apellido: 'Gil',
  domicilio: 'Domicilio',
  telefono: '123456',
  email: 'leangilutn@gmail.com',
  __v: 0 }
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Leer por otra condición

Debemos llamar al método **find** del modelo.
Como parámetro le pasaremos un json con la condición buscada y la función de callback

```
Usuario.find({nombre:name}, function(error, documento){
    if(error){
        return reject(error);
    }else{
        console.log(documento)
        resolve(documento);
    }
})
```

La función de callback recibirá como parámetros:

- Variable error: Da información acerca del error producido
- Variable documento: Información del documento a leer

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Leer por otra condición

```
{ _id: 5862faec97f2b90bdcd04db4,
  nombre: 'leandro',
  apellido: 'Gil',
  domicilio: 'Domicilio',
  telefono: '123456',
  email: 'leangilutn@gmail.com',
  __v: 0 }
```

Node JS – Leer por otra condición (like)

```
Usuario.find({nombre:{ $regex: '.*' + name + '.*' }}, function(error, documento){
    if(error){
        return reject(error);
    }else{
        console.log(documento)
        resolve(documento);
    }
})
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Modificar

Debemos llamar al método **findById** y luego al metodo **save**del modelo.

```
Usuario.findById(id, function(error, documento){
    if(error){
        return reject(error);
    }else{
        var usuario = documento;
        usuario.nombre = data.nombre;
        usuario.apellido = data.apellido;
        usuario.usuario = data.usuario;
        usuario.save(function(error, documento){
            if(error){
                return reject(error);
            }else{
                resolve(documento)
            }
        });

    }
})
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Modificar

```
{ _id: 5862faec97f2b90bdcd04db4,
  nombre: 'leandro modificado',
  apellido: 'Gil',
  domicilio: 'Domicilio',
  telefono: '123456',
  email: 'leangilutn@gmail.com',
  __v: 0 }
```

Node JS – Eliminar

Debemos llamar al método **remove** del modelo.
Por parámetro pasaremos: un json con el criterio (condición) y en el segundo parámetro una función de callback

```
Usuario.deleteOne({_id:id}, function(error, documento){
    if(error){
        return reject(error);
    }else{
        resolve(documento);
    }
})
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Eliminar

```
^C
C:\Users\Leandro\node_crud>node eliminar.js
Conectado a MongoDB
Ok
^C
C:\Users\Leandro\node_crud>node leer.js
Conectado a MongoDB
[ { _id: 5862fb0f3e67142654a2f5f7,
    nombre: 'leandro',
    apellido: 'Gil',
    domicilio: 'Domicilio',
    telefono: '123456',
    email: 'leangilutn@gmail.com',
    __v: 0 } ]
null
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Aplicado a nuestro código

productsModels.js

```
const mongoose = require('../bin/mongodb');

//Define a schema
const Schema = mongoose.Schema;
const ProductSchema = new Schema({
 name: {
  type: String,
  trim: true,
  required: true,
 },
 sku: {
  type: String,
  trim: true,
  required: true
 },
 price: {
  type: Number,
  trim: true,
  required: true
 }
});
module.exports  =  mongoose.model('products', ProductSchema);
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Aplicado a nuestro código

Productos.js (controller)

```
var productModel = require("../models/productsModel")

module.exports = {
    getAll: function(req, res, next) {
        console.log(req.query)
        productModel.find({}, function(err, data){
            if (err) {
                next(err);
            } else {
                res.status(200).json({status: "success", message: "ok", data: data});
            }
        });
    },
    getById: function(req, res, next) {
        var id = req.params.productId;
        productModel.findById(id, function(err, data){
            if (err) {
                next(err);
            } else {
                res.status(200).json({status: "success", message: "ok", data: data});
            }
        });
    },
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Aplicado a nuestro código

Productos.js (controller)

```
    save: function(req, res, next) {
        productModel.create({
            name: req.body.name,
            sku: req.body.sku,
            price: req.body.price
        }, function (err, result) {
            if (err)
            next(err);
            //next('route');
            else
            res.status(200).json({status: "success", message: "Product added successfully!!!", data: result})

        });
    }
}
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Aplicado a nuestro código

Productos.js (routes)

```
var express = require('express');
var router = express.Router();
var users = require("../controllers/products")

/* GET home page. */
router.get('/', users.getAll);
router.get('/:productId', users.getById);
router.post('/', users.save);

module.exports = router;
```

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Robomongo

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Robomongo

- Descargar desde https://robomongo.org/download
- Ubicarse (por cmd) enel directorio de Descargas y ejecutar:

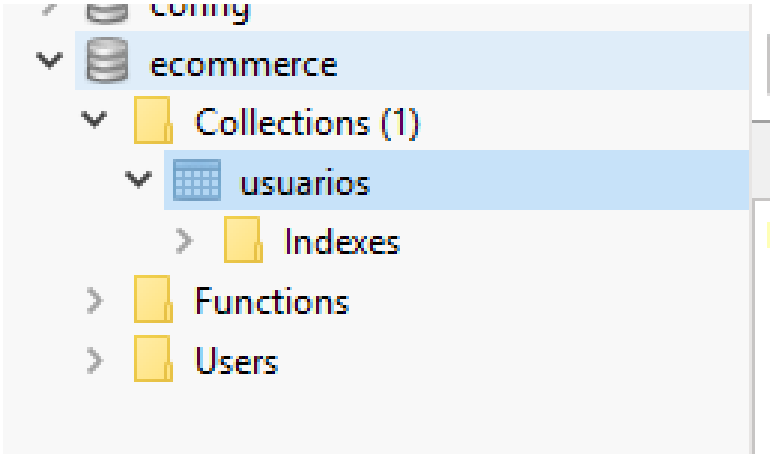
sudo dpkg -i robomongo-0.8.5-x86_64.deb

- Abrir robomongo desde la terminal ejecutando:

robomongo

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Node JS – Robomongo




```
db.getCollection('usuarios').find({})
```

usuarios 0.002 sec.

Key	Value
(1) ObjectId("5bce924bdbb61d5f10153be5")	{ 6 fields }
_id	ObjectId("5bce924bdbb61d5f10153be5")
nombre	Leandro10
apellido	Gil
usuario	leangil
password	123
__v	0