



Clase 19. Programación Backend

Mongoose



OBJETIVOS DE LA CLASE

- Conectarse a una base de datos MongoDB a través de Node.js.
- Utilizar mongoose para definir esquemas, modelos e interactuar con la base.
- Realizar un CRUD utilizando mongoose

CRONOGRAMA DEL CURSO

Clase 18



CRUD en MongoDB

Clase 19



Mongoose

Clase 20



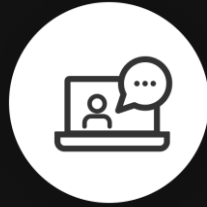
DBaaS & Firebase

Repasando...

Gestión de usuarios

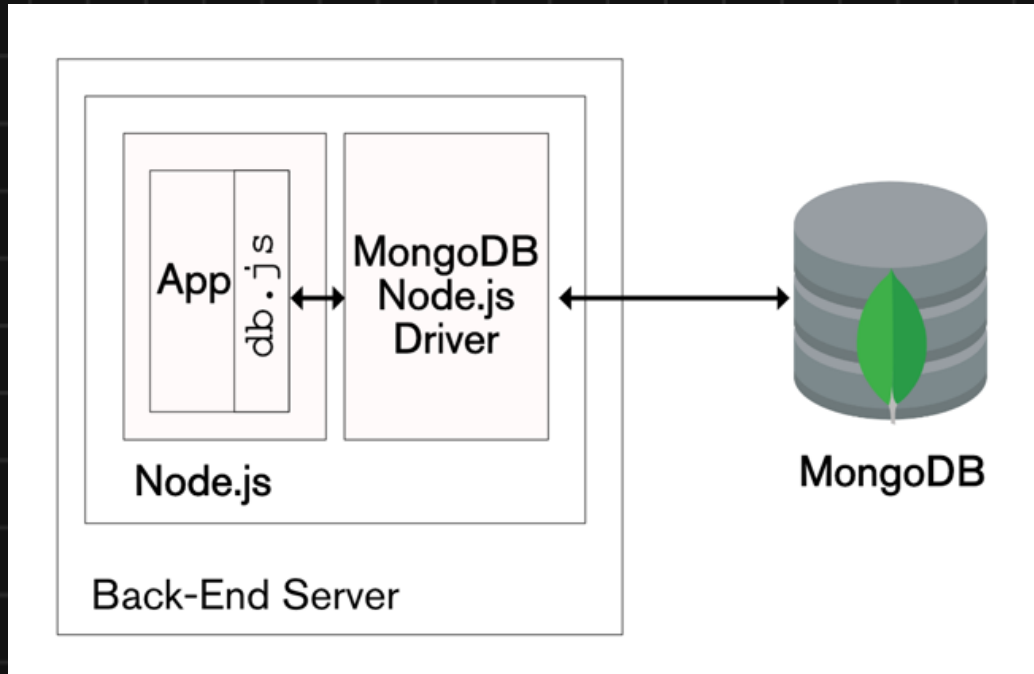


Vamos al Mongo Shell... 



***¿Alguna pregunta hasta
ahora?***

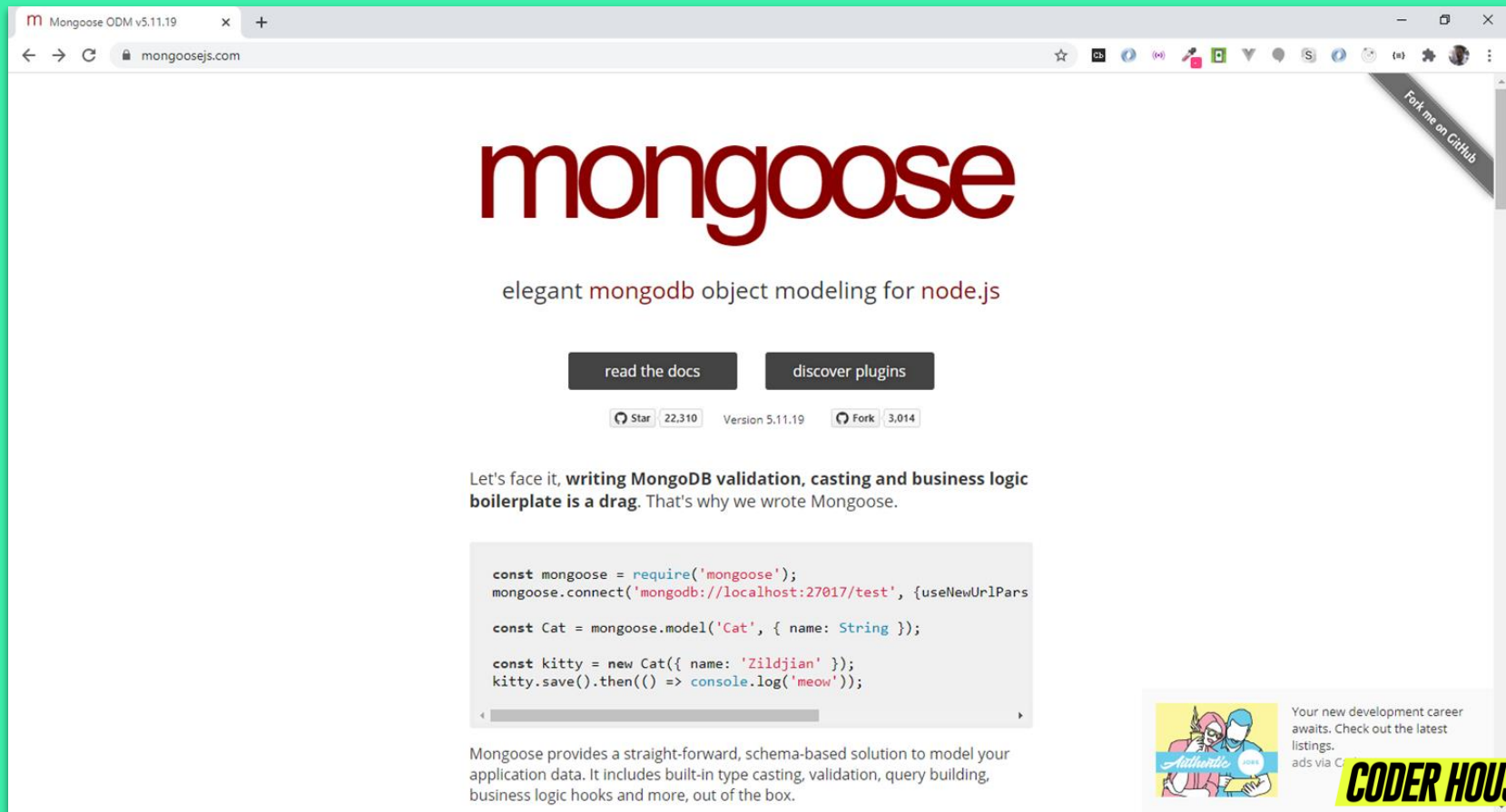
MongoDB con Node.js



- Mongoose es una **dependencia Javascript** que realiza la **conexión** a la **instancia** de MongoDB
- Pero la magia real del módulo Mongoose es la **habilidad** para **definir** un **esquema** del **documento**.
- *MongoDB* usa colecciones para almacenar múltiples documentos, los cuales no necesitan tener la misma estructura.
- Cuando tratamos con objetos es necesario que los documentos sean algo parecido. En este punto nos ayudan los esquemas y modelos de **Mongoose**.

***¿Qué es
Mongoose?***

Website oficial: <https://mongoosejs.com/>



The screenshot shows the official Mongoose website in a web browser. The browser's address bar displays 'mongoosejs.com'. The page features the 'mongoose' logo in a large, dark red font. Below the logo, the text 'elegant mongodb object modeling for node.js' is centered. Two dark buttons, 'read the docs' and 'discover plugins', are positioned below the text. Further down, GitHub statistics show 22,310 stars and 3,014 forks for version 5.11.19. A paragraph describes Mongoose as a solution for MongoDB validation, casting, and business logic boilerplate. A code block contains a JavaScript snippet demonstrating how to connect to a MongoDB instance and create a 'Cat' model. At the bottom, a brief description states that Mongoose provides a schema-based solution for modeling application data. In the bottom right corner, there is an advertisement for 'CODER HOUSE' featuring a cartoon illustration of two people and text about development career opportunities.

Mongoose ODM v5.11.19

mongoosejs.com

mongoose

elegant **mongodb** object modeling for node.js

[read the docs](#) [discover plugins](#)

Star 22,310 Version 5.11.19 Fork 3,014


Let's face it, **writing MongoDB validation, casting and business logic boilerplate is a drag.** That's why we wrote Mongoose.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/test', {useNewUrlParser: true});

const Cat = mongoose.model('Cat', { name: String });

const kitty = new Cat({ name: 'Zildjian' });
kitty.save().then(() => console.log('meow'));
```

Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

 Your new development career awaits. Check out the latest listings. ads via C

CODER HOUSE

- Mongoose usa un **objeto Schema** para definir una lista de **propiedades del documento**, cada una con su propio tipo y características para forzar la estructura del documento.
- Después de especificar un esquema deberemos definir un **Modelo constructor** para así poder crear instancias de los documentos de MongoDB

Mongoose: Schema y Model

- Mongoose es un ***Object Document Mapper (ODM)***. Esto significa que permite definir objetos con un **esquema fuertemente tipado** que se asigna a un documento MongoDB.
- Mongoose proporciona una amplia cantidad de funcionalidades para crear y trabajar con esquemas.

Schema y Model : Validaciones

Actualmente contiene ocho

SchemaTypes definidos para una propiedad

- *String* (Cadena)
- *Number* (Número)
- *Date* (Fecha)
- *Buffer*
- *Boolean* (Booleano)
- *Mixed* (Mixto)
- *ObjectId*
- *Array* (Matriz)

Schema y Model : Validaciones

Cada tipo de dato permite especificar:

- Un valor predeterminado
- Una función de validación personalizada
- La indicación de campo requerido
- Una función get que le permite manipular los datos antes de que se devuelva como un objeto
- Una función de conjunto que le permite manipular los datos antes de guardarlos en la base de datos
- Crear índices para permitir que los datos se obtengan más rápido

Schema y Model : Validaciones

Además de estas opciones comunes, ciertos tipos de datos permiten **personalizar** cómo se almacenan y recuperan los datos de la base de datos. Por ejemplo, un **String** especifica **opciones adicionales**:

- Convertir en minúsculas y a mayúsculas
- Recortar datos antes de guardar
- Una expresión regular que puede limitar los datos que se pueden guardar durante el proceso de validación
- Una enumeración que puede definir una lista de cadenas que son válidas

Schema y Model : Validaciones



***¿Alguna pregunta hasta
ahora?***

Integrando Mongoose en un proyecto Node.js



Vamos al código... 



***¿Alguna pregunta hasta
ahora?***



MongoDB con mongoose

Tiempo: 10 minutos



- 1) Realizar un proyecto en Node.js que se conecte a una base de datos MongoDB local llamada colegio. Utilizar mongoose importándolo en Módulo (import) y gestionar sus acciones a través de promesas.
- 2) Crear una colección llamada 'estudiantes' que incorporará 10 documentos con la siguiente estructura y datos que se detallan a continuación:
 - a) nombre: tipo string
 - b) apellido: tipo string
 - c) edad: tipo number
 - d) dni: tipo string (campo único)
 - e) curso: tipo string
 - f) nota: tipo number

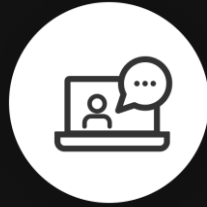
Todos los campos deben ser requeridos obligatoriamente (`{ required: true }`)



3) Tomar los valores del siguiente array de objetos

```
[
  { nombre: 'Pedro', apellido: 'Mei', edad: 21, dni: '31155898', curso: '1A', nota: 7 },
  { nombre: 'Ana', apellido: 'Gonzalez', edad: 32, dni: '27651878', curso: '1A', nota: 8 },
  { nombre: 'José', apellido: 'Picos', edad: 29, dni: '34554398', curso: '2A', nota: 6 },
  { nombre: 'Lucas', apellido: 'Blanco', edad: 22, dni: '30355874', curso: '3A', nota: 10 },
  { nombre: 'María', apellido: 'García', edad: 36, dni: '29575148', curso: '1A', nota: 9 },
  { nombre: 'Federico', apellido: 'Perez', edad: 41, dni: '320118321', curso: '2A', nota: 5 },
  { nombre: 'Tomas', apellido: 'Sierra', edad: 19, dni: '38654790', curso: '2B', nota: 4 },
  { nombre: 'Carlos', apellido: 'Fernández', edad: 33, dni: '26935670', curso: '3B', nota: 2 },
  { nombre: 'Fabio', apellido: 'Pieres', edad: 39, dni: '4315388', curso: '1B', nota: 9 },
  { nombre: 'Daniel', apellido: 'Gallo', edad: 25, dni: '37923460', curso: '3B', nota: 2 }
]
```

3) Verificar con el cliente Mongo Shell (CLI) que los datos estén almacenados en la base y colección que corresponda.



***¿Alguna pregunta hasta
ahora?***



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

Update and Delete



Vamos al código... 



***¿Alguna pregunta hasta
ahora?***



MongoDB: CRUD

Tiempo: 15 minutos



Realizar un proyecto en Node.js que sobre la base colegio realice las siguientes acciones:

- 1) Actualizar el dni del estudiante Lucas Blanco a 20355875
- 2) Agregar un campo 'ingreso' a todos los documentos con el valor false
- 3) Modificar el valor de 'ingreso' a true para todos los estudiantes que pertenezcan al curso 1A
- 4) Listar los estudiantes que aprobaron (hayan sacado de 4 en adelante) sin los campos de _id y __v
- 5) Listar los estudiantes que posean el campo 'ingreso' en true sin los campos de _id y __v
- 6) Borrar de la colección de estudiantes los documentos cuyo campo 'ingreso' esté en true



- 7) Listar el contenido de la colección estudiantes utilizando la consola, imprimiendo en cada caso los datos almacenados (sin el campo __v) junto a su fecha de creación obtenida del ObjectID en formato YYYY/MM/DD HH:mm:ss.

Por ejemplo:

```
{"_id":"604df61b5e39a84ba41313e4","nombre":"Fabio","apellido":"Pieres","edad":39,"dni":"4315388","curso":"1B","nota":9,"ingreso":false} -> Fecha creación:  
14/3/2021 08:40:11
```

- 7) Implementar estas funciones utilizando Promises en Mongoose con sintaxis async/await, utilizando la importación en formato ES Modules (import)
- 8) Verificar la información de la base 'colegio' a través de algún cliente (compass, etc).

¿PREGUNTAS?



¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- MongoDB con node.js
- Introducción a mongoose
- CRUD con mongoose



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDOLAEDUCACIÓN