

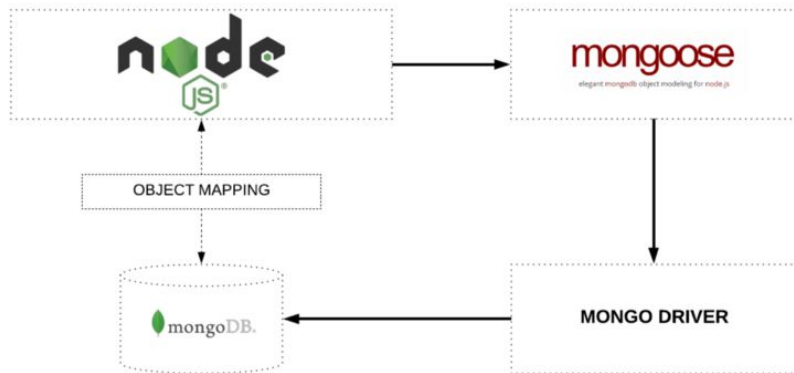
# Modelado de Datos con Mongoose

**DEV.FX**  
DESARROLLAMOS(PERSONAS);

Elaborado por César Guerra

[www.cesarquerra.mx](http://www.cesarquerra.mx)

dev



## ¿Qué es Mongoose?

Mongoose es una biblioteca de modelado de datos orientada a objetos (ODM) para MongoDB y Node.js. Administra las relaciones entre los datos, proporciona validación de esquemas y se utiliza para traducir entre objetos en el código y la representación de esos objetos en MongoDB.



mongoose

# Schemas con Mongoose

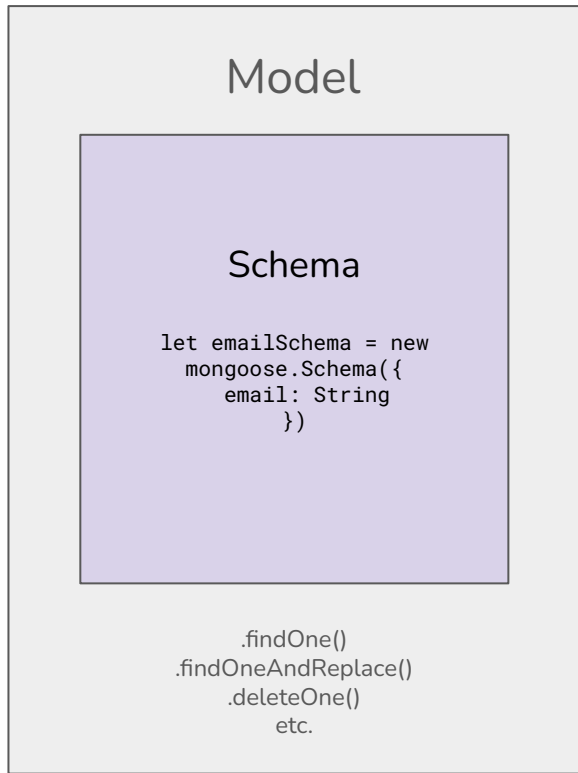
Gracias a **Mongoose**, ahora si podemos tener **Schemas** en nuestras bases de datos **NoSQL** de **MongoDB**.

```
const carSchema = new mongoose.Schema({
  // Campo: tipo de dato || Campo: { tipo de dato,
  // restricciones }
  plate: { type: String, required: true }, // No. de Placa
  year: { type: Number, required: true }, // Año
  model: { type: String, required: true }, // Modelo
  brand: { type: String, required: true }, // Marca
  version: String,
  color: {
    type: String,
    required: true,
    enum: ['Red', 'Blue', 'Black', 'White', 'Gray',
    'Silver', 'Golden', 'Green', 'Yellow']
  },
  carType: {
    type: String,
    required: true,
    enum: ['Sedan', 'Hatchback', 'SUV', 'Pickup',
    'Sport', 'Van', 'Coupe', 'Convertible', 'Wagon',
    'Minivan', 'Crossover', 'Compact', 'Subcompact',
    'Off-road', 'Luxury', 'Electric', 'Hybrid']
  },
  vin: String,
  newCar: { type: Boolean, required: true },
  isActive: { type: Boolean, default: true }
})
```

# Schemas

En Mongoose, un **Schema (Esquema)** es la estructura que indica cual es la forma en la que están estructurados los documentos que se almacenan en una **colección de MongoDB**. Cada schema está compuesto por campos y tipos de datos permitidos, además de opciones para llevar a cabo validación de dichos documentos.

En otras palabras, los esquemas sirven como guías de la estructura de los documentos. Estos son necesarios para la creación del modelo.



# Modelos

**Un modelo Mongoose es un contenedor en el esquema Mongoose.** Un esquema Mongoose define la estructura del documento, valores predeterminados, validadores, etc., mientras que **un modelo Mongoose proporciona una interfaz a la base de datos para crear, consultar, actualizar, eliminar registros, etc.**



```
// Estructura Basica de un Modelo de Mongoose en CommonJS
const mongoose = require('mongoose')

const emailSchema = new mongoose.Schema({
  email: String
})

module.exports = mongoose.model('Email', emailSchema)
```



```
// Estructura Basica de un Modelo de Mongoose en ESM
import mongoose from 'mongoose'

const emailSchema = new mongoose.Schema({
  email: String
})

const Email = mongoose.model('Email', emailSchema)

export default Email
```

# Estructura de un Modelo

La creación de un modelo Mongoose consta principalmente de tres partes:

- 1) **Importando la referencia de Mongoose:**  
Esta ya contiene la información de la conexión a la base de datos.
- 2) **Definir el esquema:** Definimos las propiedades del documento a través de un objeto donde el nombre de la clave corresponde al nombre de la propiedad en la colección.
- 3) **Crear y exportar el modelo:** Necesitamos llamar al constructor del modelo en la instancia de Mongoose y pasarle el nombre de la colección y una referencia a la definición del esquema.

# Schema Types

Los esquemas en Mongoose soportan diferentes tipos de datos.

Por lo que podemos especificarlos para cada valor al crear nuestro Schema.

Tipo	Descripción
String	Cadena de caracteres
Number	Número
Date	Fecha y hora
Boolean	Valor booleano
ObjectId	Identificador único de un documento
Mixed	Tipo genérico que puede contener cualquier tipo de dato
Buffer	Almacenamiento de datos binarios
Array	Colección de elementos con un tipo específico


Validador	Descripción
required	Requerido
min	Valor mínimo numérico
max	Valor máximo numérico
minlength	Longitud mínima de una cadena
maxlength	Longitud máxima de una cadena
enum	Valor permitido de un conjunto de valores
match	Expresión regular para validar una cadena
validate	Validador personalizado para un campo
unique	Indica que el valor de este campo debe ser único en la colección
sparse	Permite que el campo sea nulo o no exista para algunos documentos

<https://mongoosejs.com/docs/schematypes.html#schematype-options>

# Schema Options

Los **Schema Types** de **Mongoose** tienen varias opciones (**Options**) que puedes usar para personalizar el comportamiento de tu esquema. Algunas de las opciones disponibles para los tipos de esquema son que puede ser un campo requerido (**required**), asignar un valor por defecto (**default**), ser único (**unique**), o se requiere algún tipo de validación (**validate**) entre otros.





```
const userSchema = new Schema({
  email: {
    type: String,
    set: v => v.toLowerCase()
  }
})
```

<https://mongoosejs.com/docs/tutorials/getters-setters.html>

## Setters

Un **setter** es una función que te permite ejecutar lógica personalizada al establecer una propiedad en un documento de Mongoose.

*Los setters te permiten transformar los datos del usuario antes de que lleguen a MongoDB.*

En este ejemplo, el setter transforma el valor del correo electrónico a minúsculas antes de almacenarlo en la base de datos. Mongoose también ejecuta setters en operaciones de actualización, como **updateOne()**.

```
function obfuscate (cc) {  
  return '****-****-****-' + cc.slice(cc.length-4, cc.length);  
}  
  
const AccountSchema = new Schema({  
  creditCardNumber: { type: String, get: obfuscate }  
});  
  
const Account = mongoose.model('Account', AccountSchema);  
  
Account.findById( someId, function (err, found)  
  console.log(found.creditCardNumber); // '****-****-****-1234'  
});
```

**Nota:** Es posible funciones flecha para declarar setters y getters en Mongoose. Sin embargo, debes tener en cuenta que las funciones flecha no tienen su propio this, por lo que si necesitas acceder al contexto del documento de Mongoose (por ejemplo, usar this para referirte a otras propiedades del documento), deberías usar una función regular (function) en lugar de una función flecha.

<https://mongoosejs.com/docs/tutorials/getters-setters.html>

# Getters

Un **getter** es una función que te permite ejecutar lógica personalizada al obtener una propiedad en un documento de Mongoose.

*Los getters permiten transformar los datos que se recuperan de MongoDB en una forma más conveniente.*

Por ejemplo, al recuperar los datos almacenados de números de tarjetas de crédito, necesitamos ocultar todo excepto los últimos 4 dígitos para el usuario de mongoose.

## Tamaño de Documentos

*En MongoDB, **un documento tiene un tamaño máximo de 16 MB**, por lo que es importante tomar en cuenta esta restricción a la hora de modelar nuestros documentos.*



# ¡GRACIAS!



César Guerra



[www.cesarguerra.mx](http://www.cesarguerra.mx)