[Creando APIs con Node JS, Express y MongoDB](https://legacy.gitbook.com/book/gmoralesc/creando-apis-con-node-js-express-y-mongodb)

Portada

[Introducción](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/)

Configurando el proyecto con express

[Crear un simple Web Server](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/configurando-el-proyecto-con-express/crear-un-simple-web-server.html)

[Utilizando Express](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/configurando-el-proyecto-con-express/utilizando-express.html)

[Herramientas de desarrollo](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/configurando-el-proyecto-con-express/configurando-el-ambiente-de-desarrollo.html)

[Configuración y variables de entorno](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/configurando-el-proyecto-con-express/configuracion-y-variables-de-entorno.html)

[Middleware, manejo de errores y logs](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/configurando-el-proyecto-con-express/middleware-manejo-de-errores-y-logs.html)

Router y Routes en Express

[Utilizando el Router y Routes de Express](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/router-y-routes-en-express/utilizando-el-router-y-routes-de-express.html)

[Creando el layout del API](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/router-y-routes-en-express/creando-el-layout-del-api.html)

[Capturando y procesando parámetros de las peticiones](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/router-y-routes-en-express/capturando-y-procesando-parametros-de-las-peticiones.html)

Persistencia de datos con MongoDB

[Instalando y configurando MongoDB](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/instalando-mongo-db.html)

[Conectando con MongoDB](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/conectando-con-mongodb.html)

[Mongoose Models](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/mongoose-models.html)

[Procesando parámetros comunes con middleware](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/procesando-parametros-comunes-con-middleware.html)

[Estandarización de la respuesta](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/estandarizacion-de-la-respuesta.html)

[Paginación](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/paginacion.html)

[Mongoose Schemas](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/mongoose-schemas.html)

[Ordenamiento](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/ordenamiento.html)

[Creando recursos](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/creando-recursos.html)

[Relaciones entre recursos](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/relaciones-entre-colecciones.html)

[Consultar recursos anidados](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/consultar-recursos-anidados.html)

[Añadir recursos anidados](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/anadir-recursos-anidados.html)

Asegurando el API

[Añadir y remover campos de un documento](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/asegurando-el-api/anadir-y-remover-campos-de-un-documento.html)

[Encriptando información sensible](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/asegurando-el-api/encriptando-informacion-sensible.html)

[Publicado con GitBook](https://www.gitbook.com/)

[Mongoose Models](https://gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/)

**Mongoose Models**

*Mongoose* es un ORM (Object-relational mapping) que permite la definición de estructuras de datos ( *Models* ) para interactuar con la base de datos *MongoDB* a nivel de aplicación mediante muchas funciones de su API, una vez definidos los modelos son almacenados en la base de datos como **colecciones** y los registros de esa colección son almacenados como **documentos** . Una de las particularidades de las base de datos NoSQL es que no necesariamente todos los documentos deben ser iguales así si el modelo cambia en el futuro no habrá que actualizar todos los documentos existentes en la base de datos.

Procedemos a crear nuestro primer modelo para el recurso de *posts* :

touch server/api/v1/posts/model.js

Colocamos el siguiente contenido dentro del archivo creado:

const mongoose = require ( "mongoose" ); const post = {
location: String ,
photo\_url: String ,
description: String ,
}; module .exports = mongoose.model( 'post' , post);

Hemos creado un nuevo objeto con las propiedades necesarias, definiendo a su vez el tipo y las propiedades. Finalmente exportamos el modelo nombrado post basado en el objeto creado.

Por el momento no estamos realizando ningún tipo de validación, entraremos en detalle mas adelante.

Antes de continuar guardamos nuestro progreso:

git add .
git commit -m "Create Model for Posts"

**Creando objetos en la base de datos**

Una vez definido el modelo procedemos a requerirlo en nuestro controlador ( server/api/v1/posts/controller.js ) y modificamos nuestra acción de create :

const logger = require ( "winston" ); const Model = require ( "./model" );
...
exports.create = (req, res, next) => { const { body } = req; const document = new Model(body); document .save()
.then((doc) => {
res.json(doc);
})
.catch((err) => {
next( new Error (err));
});
};
...

En el código anterior tenemos los siguientes pasos:

Creamos nuestro nuevo objeto llamado document a partir de los datos que recibimos en el objeto body de la petición ( req ).

Invocamos la función save para guardar finalmente la instancia del modelo *Post* en la base de datos, esta función retorna una promesa ( *Promise* ) la cual es un patrón asincrónico muy utilizado y soportado por *Mongoose* .

Si es una operación exitosa ingresa en la función then donde devolvemos un objeto JSON con la información del documento guardado en la base de datos, pues a diferencia de los datos enviados este tiene campos creados en la base de datos como el identificador del documento ( \_id ).

Si hay algún error ingresa en la función catch donde guardamos en el *log* el error y procedemos a llamar a la función next con el error recibido para que sea procesado por nuestro *middleware* de errores definido anteriormente.

**Leyendo objetos de la base de datos**

Modificamos ahora la función de all en nuestro controlador para obtener todos los documentos de la colección *posts* guardados en la base de datos:

exports.all = (req, res, next) => {
Model.find().exec()
.then((docs) => {
res.json(docs);
})
.catch((err) => {
next( new Error (err));
});
};

Esta vez utilizamos los *Queries* de *Mongoose* , la primera función es el find que sin ningún parámetro encuentra todos los documentos guardados en la colección de p *osts* ., finalmente para ejecutar el *Query* utilizamos la función *exec* que al igual que las otras funciones devuelve una promesa y aplicamos el mismo patrón para la promesa de respuesta, existe una sutil diferencia en la respuesta y es que es un Array de documento no un simple objeto como devolvimos cuando guardamos un documento.

Antes de continuar guardamos nuestro progreso:

git add .
git commit -m "Create and Read all for posts"

Mas información:

[Mongoose Models](http://mongoosejs.com/docs/models.html)

[Promises/A+ conformant promises](https://promisesaplus.com/)

[Mongoose Promises](http://mongoosejs.com/docs/promises.html)

[Mongoose Queries](http://mongoosejs.com/docs/queries.html)