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

**Mongoose Schemas**

Los *Schemas* de *Mongoose* brindan una funcionalidad y consistencia adicional a los modelos, hasta el momento no he validado con rigurosidad los datos enviados por el usuario y es aquí donde los *Schemas* nos pueden ayudar mucho, ademas agregan una gran cantidad de funcionalidades que veremos con detalle en las siguientes secciones.

Comenzamos por establecer las diferentes propiedades de cada uno de los atributos de nuestro modelo ( /server/api/v1/posts/model.js ):

const mongoose = require ( 'mongoose' ); const { Schema } = mongoose; const fields = {
location: {
type: String , default : '' ,
trim: true },
photo\_url: {
type: String ,
required: true ,
},
description: {
type: String , default : '' ,
trim: true },
}; const post = new Schema(fields, {
timestamps: true ,
}); module .exports = mongoose.model( 'post' , post);

Revisemos cada uno de los cambios realizados en nuestro modelo:

Extraemos el constructor del esquema ( Schema ) del objeto mongoose .

Renombramos el antiguo objeto modelo como fields , ahora ademas de especificar el tipo de atributo, podemos asignarle un valor por defecto ( default ), efectuarle operaciones con trim para quitar los espacios en blanco y establecer si el atributo es requerido o no.

Utilizamos los fields y las opciones para construir el *Schema* y almacenarlo en la variable post

Agregamos la opción timestamps al *Schema* para que automáticamente nos añada los atributos de createdAt y updatedAt , el primero se establece una vez se guarda satisfactoriamente el documento y el segundo una vez se actualice satisfactoriamente el documento.

Como estamos añadiendo validaciones básicas al modelo si el documento no puede guardarse porque le falta un atributo requerido después de la función save ejecutara la función catch en vez la función then , lo cual llegará a nuestro *middleware* de errores, pero no podemos decir que es un error *500 Internal Server Error* , en este caso es un *422 Unprocessable Entity* , la cual significa que no se pudo procesar la entidad debido a un error en este caso que le faltan campos requeridos, por lo tanto procedemos a modificar nuestro *middleware* de errores ( /server/index.js ):

...
app.use((err, req, res, next) => { let {
statusCode = 500 ,
} = err; const {
message,
} = err; // Validation Errors if (err.message.startsWith( 'ValidationError' )) {
statusCode = 422 ;
}
logger.error( `Error: ${message} ` );
res.status(statusCode);
res.json({
error: true ,
statusCode,
message,
});
}); module .exports = app;

Más información:

[Mongoose Schema](http://mongoosejs.com/docs/guide.html)

[Mongoose Schema Types](http://mongoosejs.com/docs/schematypes.html)