[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/)

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

**Estandarización de la respuesta**

Como habíamos mencionado antes el contrato por defecto de REST API sugiere con que verbos y rutas se deben hacer las operaciones del CRUD con los recursos, esto no es una camisa de fuerza y podemos modificar este contrato ya se extendiéndolo o adecuándolo a las necesidades particulares de nuestra aplicación, pero este contrato no especifica como debe ser la respuesta, esta depende de cada aplicación.

Teniendo en cuenta lo anterior vamos a realizar los siguientes cambios:

Aunque el código de la respuesta HTTP indica si fue una operación exitosa con algún código de la familia del 200 o hubo en error del servidor (código 500) vamos a añadir campos adicionales en la respuesta para brindar mas información acerca del resultado de la operación. Por ejemplo al buscar un *post* si no es encontrado no podríamos decir que es un error del servidor o que la ruta no fue encontrada, fue una operación exitosa en cuanto a su funcionamiento pero no a su respuesta por ende este campo nos indicará el resultado de la operación ( success ).

A parte de enviar los datos también colocaremos un campo que contenga el mensaje legible para la respuesta. Por ejemplo: Si el *post* no fue encontrado o no se cumplió alguna regla de negocio. ( message )

Encapsularemos los documentos que vienen de la base de datos en un campo estándar ya sea cuando en un solo documento ( item ) o son varios documentos ( items ).

Cuando la respuesta son varios documentos también añadiremos información adicional ( meta ) que indique por ejemplo el numero de registros totales y cuantos se han enviado de respuesta.

Cuando ocurra un error a parte del código HTTP correspondiente lo indicaremos explicitamente ( error ) y de igual forma enviaremos un mensaje.

Editamos las respuestas de cada una de las acciones nuestro controlador /server/api/v1/posts/controller.js :

A continuación **SOLO** se muestra el la firma de la función para identificarla y el cambio en el objeto de respuesta, los ... significan que allí esta el resto del código de cada función:

exports.id = (req, res, next, id) => {
...
res.json({
success: false ,
message,
});
...
};
exports.all = (req, res, next) => {
...
res.json({
success: true ,
items: docs,
});
...
};
exports.create = (req, res, next) => {
...
res.status( 201 );
res.json({
success: true ,
item: doc,
});
...
};
exports.read = (req, res, next) => {
...
res.json({
success: true ,
item: doc,
});
};
exports.update = (req, res, next) => {
...
res.json({
success: true ,
item: updated,
});
...
};
exports.delete = (req, res, next) => {
...
res.json({
success: true ,
item: removed,
});
...
};

Finalmente editamos el archivo principal de nuestro servidor ( /server/index.js ) y modificamos de la misma forma la respuesta de los *middlewares* que capturan los errores:

// Handle middleware errors app.use((req, res, next) => { const message = 'Resource not found' ;
logger.info(message);
res.status( 404 );
res.json({
error: true ,
message,
});
});
app.use((err, req, res, next) => { const {
statusCode = 500 ,
message,
} = err;
logger.error( `Error: ${message} ` );
res.status(statusCode);
res.json({
error: true ,
statusCode,
message,
});
});