


Estandarización de la respuesta

 gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/estandarizacion-de-la-respuesta.html

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:

1. 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**).
2. 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**)
3. 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**).
4. 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.
5. Cuando ocurra un error a parte del código HTTP correspondiente lo indicaremos explícitamente (**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:

```
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,
  });
});
```