# Carga masiva A+

Todo sistema necesita datos para poder funcionar, y uno de los principales problemas es que mientras se desarrolla no se dispone de dichos datos. Para solventar este problema se deben generar datos, cuanto más realistas sean esos datos más errores de código se van a poder detectar.

En este caso para generar datos hemos optado por una librería javascript llamada [dummy-json](https://www.npmjs.com/package/dummy-json), esta librería permite generar datos realistas a partir de una plantilla donde se especifica el formato, tipo de los datos y el número de repeticiones.



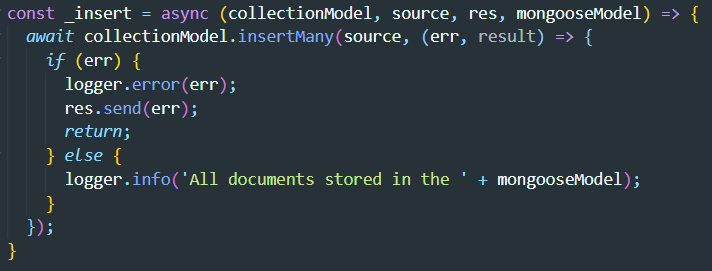
Como se puede ver en la imagen anterior se está usando una plantilla para generar un json en el que la clave para los datos es el nombre de la colección de mongo, esto se usará en la función insertMany de moongose que se comentará más adelante. Usando la nomenclatura anterior se generan 10000 objetos gracias al *#repeat*, aunque hemos llegado a introducir 20000 objetos en la base de datos usando esta técnica.

Esta librería dispone de diferentes tipos como puede ser *loren* que genera un texto loren ipsum del tamaño especificado, tipo date, boolean y muchos más. Además de los tipos por defecto también se pueden usar funciones propias para generar tipos de datos, para trip usamos funciones propias para el ticker o mangedBy. A continuación, se pueden ver las funciones propias que hemos usado para generar los datos



Una vez generado el json con los datos, solo es necesario recorrerlo y cargarlos en la colección usando la función insertMany.

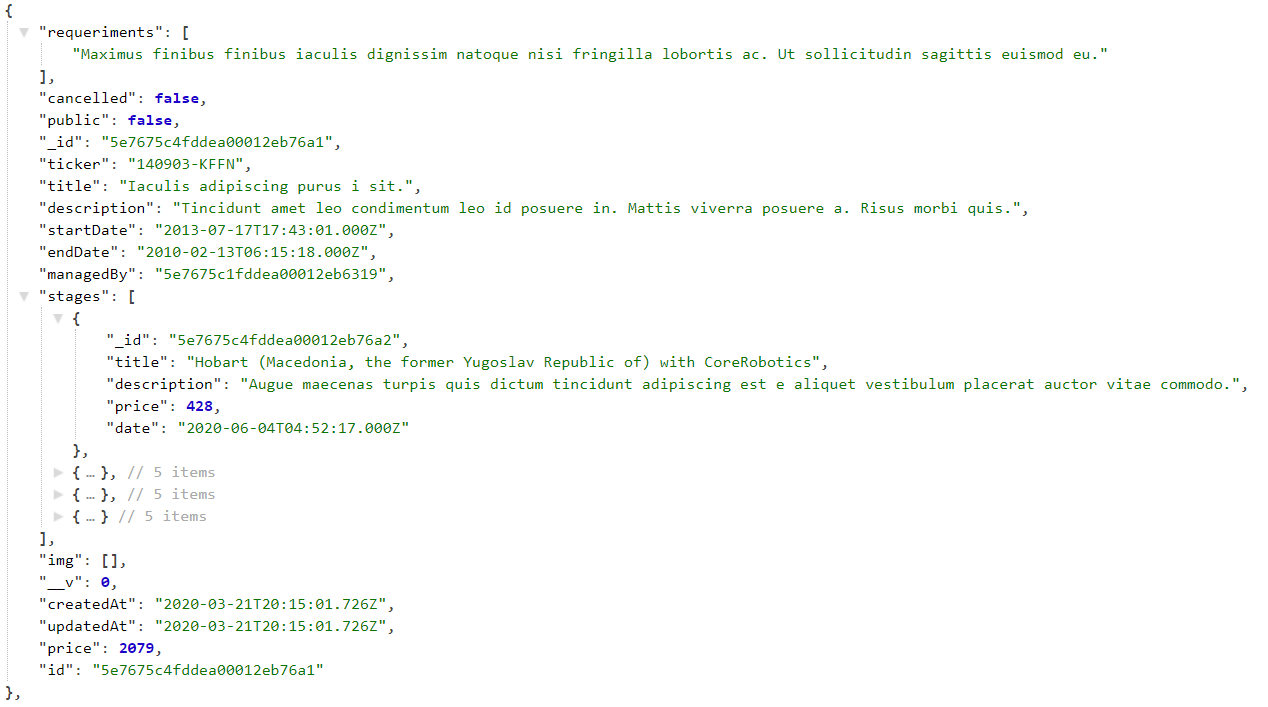




Para poder cargar los datos solo debemos hacer un POST a la ruta:

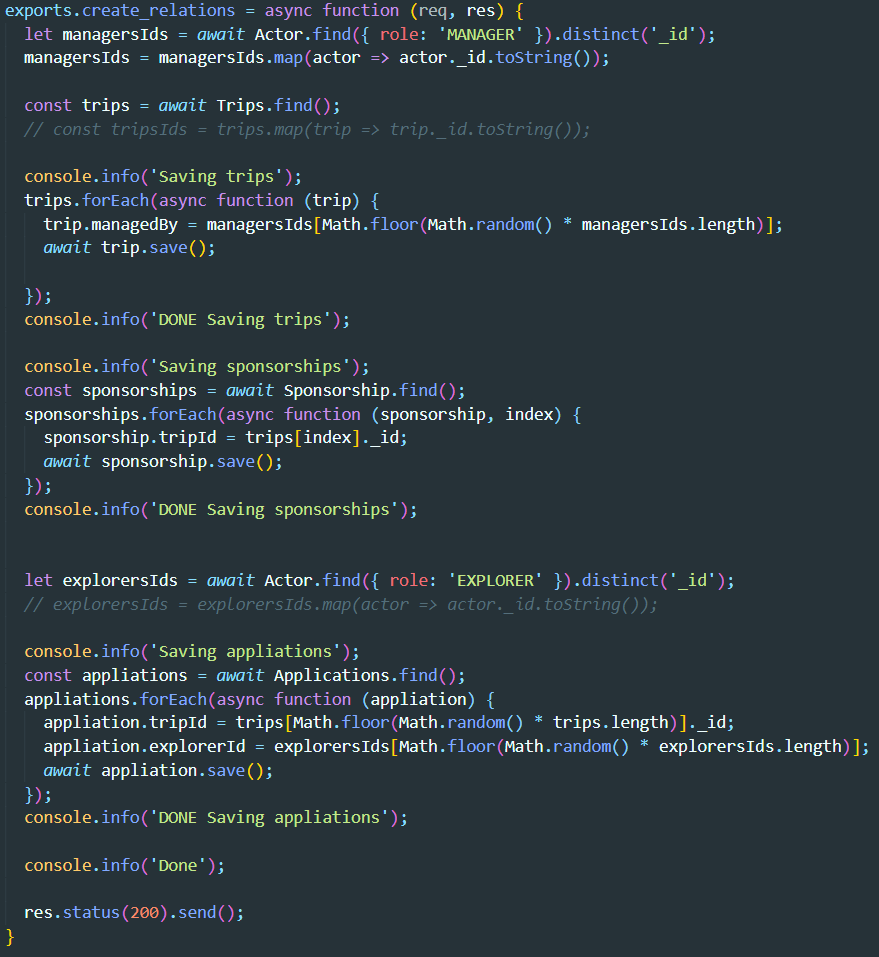
*localhost:3000/api/v1/storage/insertMany*

Podemos ver a continuación una muestra de un Trip insertado en base de datos.



Los datos se introducen bien, pero ahora surge otra problemática, relacionar los documentos entre sí.

Para resolver este problema hemos optado por ir recuperando todos los documentos de las colecciones para ir asignado ids al azar de otras colecciones, por ejemplo, para asignar un Manager a un Trip primero recuperamos la lista de todos los manager para seguidamente obtener uno aleatorio de la lista y asignárselo al Trip. Este funcionamiento se puede ver a continuación.



Para poder crear las relaciones entre los datos debemos hacer un POST a la ruta:

*localhost:3000/api/v1/storage/createRelations*