¿Qué es NoSQL y qué diferencias hay con SQL? ¿Qué es MongoDB?

NoSQL son sistemas de gestión de bases de datos no relacionales. Estos sistemas no utilizan como lenguaje principal SQL y no requieren de estructuras fijas como tablas para ir almacenando la información. SQL está estructurado de forma que todos la forma de los datos necesita estar predefinida de forma que una tabla definirá qué datos almacenará el objeto que contenga y estos datos han de estar presentes si o si de forma que todos los datos en una tabla tienen la misma estructura. Por otra parte, SQL requiere de un ID único y autoincremental para poder identificar inequívocamente estos elementos y el cual se va formando a medida que se agregan los elementos a la base de datos. SQL al tratarse de un sistema de gestión de bases relacionales relaciona los datos entre sí mediante el uso de tablas intermedias y soporta operaciones de tipo JOIN. Este tipo de bases se utiliza sobre todo cuando el rendimiento o tiempo de reacción no es tan importante en comparación a la estructura de los datos, como puede ser guardar información de carácter sensible como los datos bancarios de una persona. En cambio, las bases de datos NoSQL al no tener una estructura tan fija ni estar ligados los datos entre sí de forma tan directa, permite un acceso a la información mucho más rápido. Por ello es ampliamente utilizado a la hora de gestionar datos que necesiten ser actualizados en tiempo real como puede ser en redes sociales o aplicaciones en web.

MongoDB es un sistema de gestión de base de datos de tipo NoSQL de código abierto. La gran ventaja de MongoDB es que guarda las estructuras de datos (documentos) en BSON, el cual es muy similar a JSON. Esto hace que su integración con Javascript sea rápida y sencilla lo que lo hace ideal para utilizarlo en todo tipo de desarrollo ya sea web o de aplicaciones. A diferencia de SQL, MongoDB utiliza colecciones en lugar de tablas y documentos en lugar de objetos. Por otra parte, la estructura de los documentos en una misma colección puede variar.

¿Por qué crees que utiliza Javascript y retorna los datos en JSON?

MongoDB al ser de tipo NoSQL está pensado para ser veloz y eficiente y, al igual que Javascript, para ser usado, sobre todo, en la web. Como bien se ha comentado, MongoDB utiliza una estructura de datos muy similar a JSON por lo que la conversión a este es muy rápida y relativamente sencilla. Se podría decir que MongoDB está pensado para ser utilizado con Javascript de una forma más o menos nativa ya que la filosofía de ambas tecnologías es muy similar. Al retornar los datos en JSON permite que Javascript pueda procesarlos de forma inmediata ya que Javascript está más que preparado para procesar este tipo de datos debido a que es el estándar para uso en la web debido.

¿Por qué crees que puede ser interesante utilizar los "ids" de sus réplicas MySQL?

MongoDB utiliza un tipo de id que viene dado en formato hexadecimal y depende de varios factores por lo que este no es del todo intuitivo. Al utilizar un id equivalente al de MySQL se obtiene un mayor control sobre los valores de este y una mayor simplicidad a la hora de acceder y evitarnos así la necesidad de tener que construir objetos del tipo ObjectId en MongoDB y simplemente filtrar por enteros.

Matías Ezequiel Seguer

Código

Index.js

```
#!/usr/bin/node
let http = require("http");
let fs = require("fs");
const { debug } = require("console");

let mongo_client = require("mongodb").MongoClient;
let ObjectId = require("mongodb").ObjectID;

let url = "mongodb://localhost/";

console.log("Iniciando script mongo-http");

let database;

mongo_client.connect(url, function(error, connection){

   console.log("Dentro de MongoDB.");
   console.log(error);

   if(error){
       console.log("ERROR!!!");
       return;
   }

   database = connection.db("tffhd");

});
```

```
function SendDataList(database, response, peticion){


   let testing;


   if(peticion == "characters"){

         testing  = database.collection(peticion).find({}, { projection:
{ name: 1} })

   }

   else if(peticion == "items"){

         testing  = database.collection(peticion).find({}, { projection:
{ item: 1} })

   }


   testing.toArray(function(error,data){

       if(data.length != 0){

           dataToSend  = JSON.stringify(data);

           response.end(dataToSend);

           return;

           }

           else{

           response.end("DATA NOT FOUND");

           return;

           }

   })


}
```

```
http.createServer(function(request, response){
   response.writeHead(200);
   let saludo = "<h1>Muy buenos dias companero!</h1>";
   let array = [];
   let i = 0;
   let dataToSend;
   var peticion = request.url;
   if(peticion == "/"){
       fs.readFile("index.html", function (err, data){
           response.writeHead(200, {"Content-Type" : "text/html"});
           response.end(data);
           return;
       })
       return;
   }
   peticion = request.url.split("/");
     if(peticion.length == 2){
       SendDataList(database, response, peticion[1]);
   }
   else
   {
       if(peticion[2].length != 24){
           response.end();
           return ;
       }
       if(peticion[1] == "characters"){
           let obj_id = new ObjectId(peticion[2]);
                                                    let     colData     =
database.collection("characters").find({"_id":obj_id},     {projection:{
"name":1 } });
```

```
            colData.toArray(function(err, data){
                let string = JSON.stringify(data);
                response.end(string);
            });
        }
        else if(peticion[1] == "items"){
            let obj_id = new ObjectId(peticion[2]);
let    colData    =    database.collection("items").find({"_id":obj_id},
{projection:{ "item":1 } });
            colData.toArray(function(err, data){
                let string = JSON.stringify(data);
                response.end(string);
            });
        }
        else if(peticion[1] == "remove"){
            let obj_id = new ObjectId(peticion[2]);
            collectionName = peticion[3];
          database.collection(collectionName).deleteOne({"_id":obj_id});
            response.end("DELETED");
        }
    }
}).listen(1095);
```

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Too Fast FamilyHD</title>
    </head>
```

```
   <body>
        <h1>Patata</h1>
        <main>
            <section id="characters">
                <h2>Characters</h2>
                    <ul id="characters-list">
                    </ul>
            </section>
            <section id="items">
                <h2>Items</h2>
                    <ul id="items-list">
                    </ul>
            </section>
        </main>
<script>
LoadCharactersData();
LoadItemsData();
function WriteCharactersList(data){
   let text = "";
   for(let i = 0; i < data.length; i++){
      text += "<li> <a href=\"characters/" + data[i]._id + "\">" +
data[i].name + "</a><a href=\"remove/" + data[i]._id + "/characters" +
"\" >X</a></li>";
   }
   document.getElementById("characters-list").innerHTML = text;
};

function WriteItemsList(data){
   let text = "";
```

```
   for(let i = 0; i < data.length; i++){

            text += "<li> <a href=\"items/" + data[i]._id + "\">" +
data[i].item + "</a><a href=\"remove/" + data[i]._id + "/items" + "\"
>X</a></li>";

   }

   document.getElementById("items-list").innerHTML = text;

};

function LoadCharactersData()

{

   fetch("/characters")

        .then( response => response.json())

        .then( data => WriteCharactersList(data));

}

function LoadItemsData()

{

   fetch("/items")

        .then( response => response.json())

        .then( data => WriteItemsList(data));

}

let                          CharactersInterval                          =
setInterval(LoadCharactersData("characters"),2000);

let ItemsInterval = setInterval(LoadItemsData("items"), 2000);

</script>

   </body>

</html>
```

```
130 screight@screight:~/DAMVIOD/Rafa/2122_M02UF4$ ./index.js
Iniciando script mongo-http
Dentro de MongoDB.
undefined
```

Primero activamos el script principal index.js que se encarga de procesar las peticiones al servidor, en este caso se trata del puerto 10095.

Patata

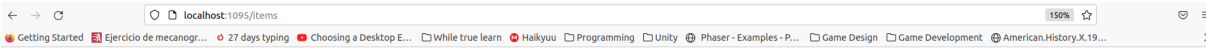
Characters

- ObamaX
- Vin DieselX
- Sailor MoonX
- ArsenioX

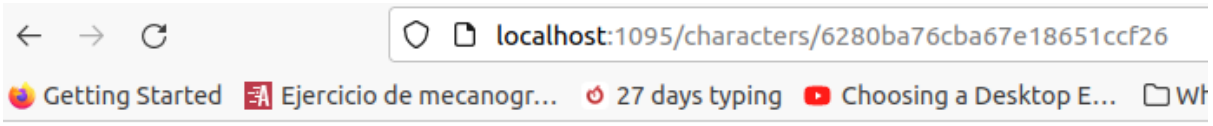
Items

- Medallon de oroX
- EspadaX
- PocionX
- Mochila desgarradaX

Al acceder al puerto en cuestión, 1095, se obtiene un listado de los documentos guardados en la base de datos de Mongo en la colección de characters e items.

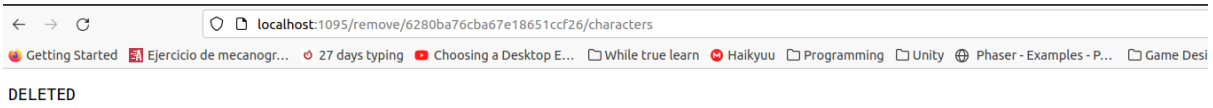


Se puede comprobar como al diferentes peticiones al servidor, este devuelve un listado con los documentos en cada colección. Luego se procesan los datos y se obtiene las listas anteriores.

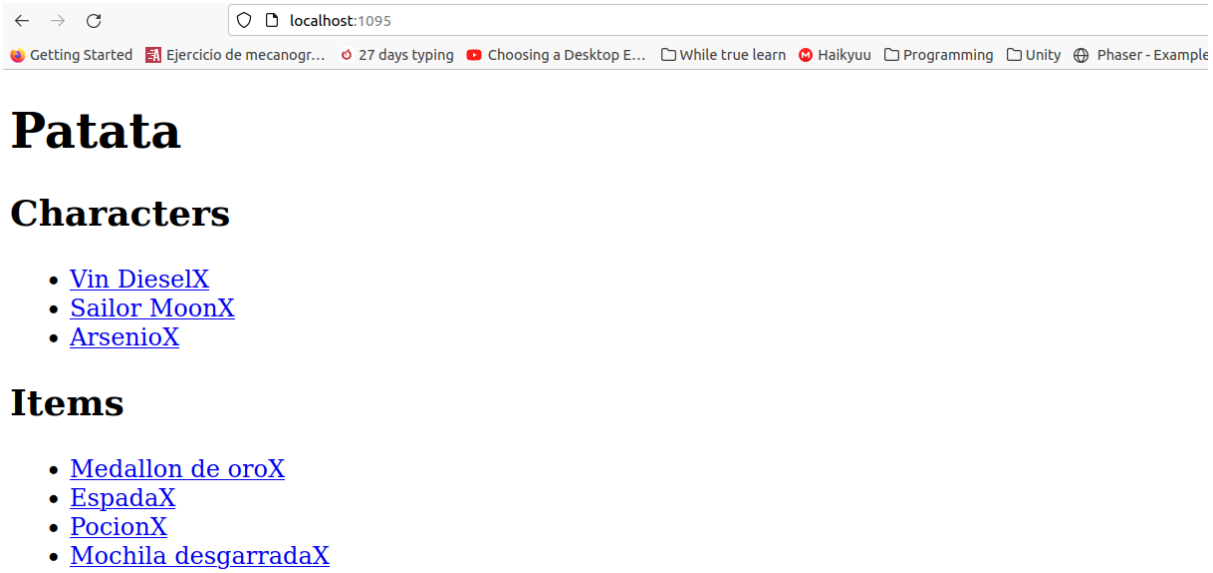


[{"_id":"6280ba76cba67e18651ccf26","name":"Obama"}]

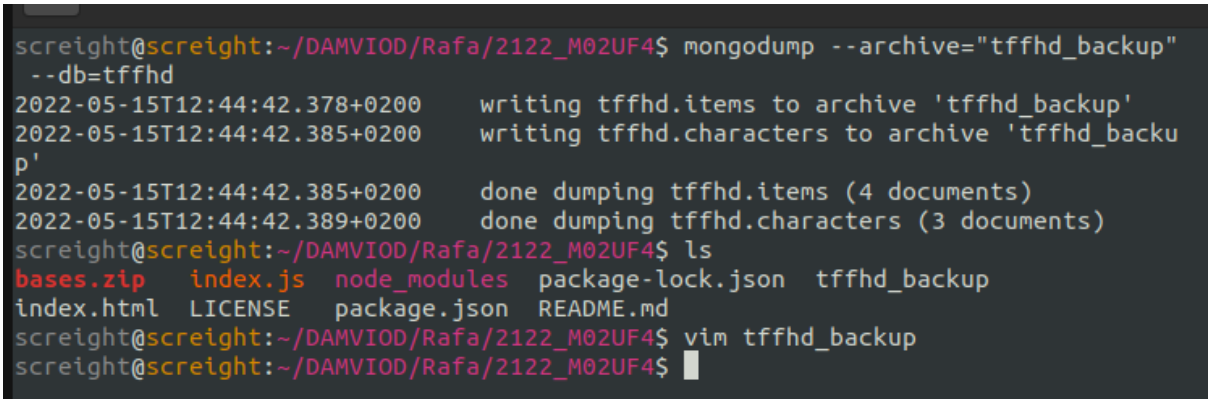
Al hacer click en uno de los links que representan cada uno de los documentos, este nos manda a un url que contiene tanto la coleccion a la que pertenece como el id del documento. Esto será útil a la hora de hacer peticiones al servidor y obtener información de este.



Al hacer click en una de las X que acompaña al link de los documentos, nos manda a un url que contiene la palabra clave remove, el id y la colección. En index.js se procesa esta información de forma que se determina que lo que se quiere hacer es eliminar un documento que pertenece a la colección characters y tiene el siguiente ID. De forma que el servidor accede a la base de datos y elimina, en caso de existir, el documento de la colección correspondiente.



Al volver a cargar la página, se hace la misma petición anterior que pide todos los documentos de ambas colecciones pero al haber eliminado uno de ellos, este ya no se encuentra en la base de datos y, en consecuencia, el servidor ya no lo devuelve y no se muestra en pantalla.



Hacer copia de seguridad de la base de datos.