

Reto de Framework

Backend

Express js

Elaboró:

Allan Jair Escamilla Hernández

**Arquitectura de Información
web**

Índice

Arranque de la aplicación	3
Datos de prueba	4
Archivo gitignore	5
Creación de base de datos con MongoDB	5
Bibliotecas utilizadas y bibliotecas más usadas en node	10
Haciendo debugging	11
Considerando las mejores prácticas para el despliegue	11
Remover contraseñas quemadas.	11
Encapsular código spaghetti.	11
Revisar la estructura del proyecto.	11
Configurar los scripts de build.	11
Agregar soporte de caché.	11
Añadir HTTPS y CORS.	11
Json Web Tokens	11
JSON Web Tokens Auth y ventajas de JWT	12
Buenas prácticas con JSON Web token	12
¿Dónde guardar los tokens?	13
Conclusiones	13

Arranque de la aplicación

Para ejecutar la aplicación, si estás en windows debes dirigirte hacia el archivo package.json y escribir la siguiente línea en los scripts:

```
"dev": "set DEBUG=app*&& nodemon index"
```

Si estás en un sistema operativo basado en Unix, deberás de escribir la siguiente línea en los scripts de archivo package.json:

```
"dev": "DEBUG=app:* nodemon index"
```

Una vez que hayas establecido estos scripts, puedes arrancar el proyecto corriendo el siguiente comando (estando posicionado con la terminal dentro de la carpeta del proyecto):

```
npm run dev
```

Datos de prueba

Para poder desarrollar la API sin la necesidad de estar inventando datos al azar de películas, se ha utilizado Mockaroo. Mockaroo* es un servicio que nos permite crear datos simulados a partir de una estructura, por ejemplo para generar la estructura de una película:

Field Name	Type	Options
id	GUID	blank: 0 % fx ×
title	Movie Title	blank: 0 % fx ×
year	Car Model Year	blank: 0 % fx ×
cover	Dummy Image URL	size: 100 × 100 to 250 × 250 format: random fx blank: 0 % fx ×
description	Paragraphs	at least 1 but no more than 3 blank: 0 % fx ×
duration	Number	min: 1888 max: 2070 decimals: 0 blank: 0 % fx ×
contentRating	Custom List	G, PG, PG-13, R, NC-17 random fx blank: 0 % fx ×
source	URL	include: <input checked="" type="checkbox"/> protocol <input checked="" type="checkbox"/> host <input checked="" type="checkbox"/> path <input type="checkbox"/> query string blank: 0 % fx ×
tags[1-5]	Movie Genres	blank: 0 % fx ×

[Add another field](#)

Rows: Format: [JSON](#) array include null values

Hint: Use "." in column names to generate nested json objects, brackets to generate arrays. [More information...](#)

Archivo gitignore

Git tiene una herramienta imprescindible casi en cualquier proyecto, el archivo "gitignore", que sirve para decirle a Git qué archivos o directorios completos debe ignorar y no subir al repositorio de código. Para no tener que estar agregando uno por uno los archivos que se van a ignorar con git, se ha utilizado la aplicación gitignore.io que nos permite hacer un gitignore efectivo para cada una de las tecnologías que usemos.

Creación de base de datos con MongoDB

Para el desarrollo de este proyecto se ha utilizado como base de datos a MongoDB, haciendo uso de MongoDB Atlas que nos provee de clusters distribuidos alrededor del mundo para tener una mayor accesibilidad a nuestros datos. A continuación, muestra el proceso seguido para la creación de la base de datos con MongoDB.

Para la conexión con una base de datos de MongoDB, es necesaria una URI que te entrega MongoDB y esta tiene la siguiente estructura:

```
mongodb+srv://DB_USER:DB_
PASSWORD@DB_HOST/DB_NAME
```

Los pasos a seguir para la conexión con MongoDB son los siguientes:

1) Crear cuenta en mongodb atlas

2) Crear un cluster

The screenshot shows the MongoDB Atlas web interface. On the left, there's a sidebar with 'Project 0' selected. Under 'DATA STORAGE', 'Clusters' is highlighted. The main area shows a 'Clusters' table with one entry:

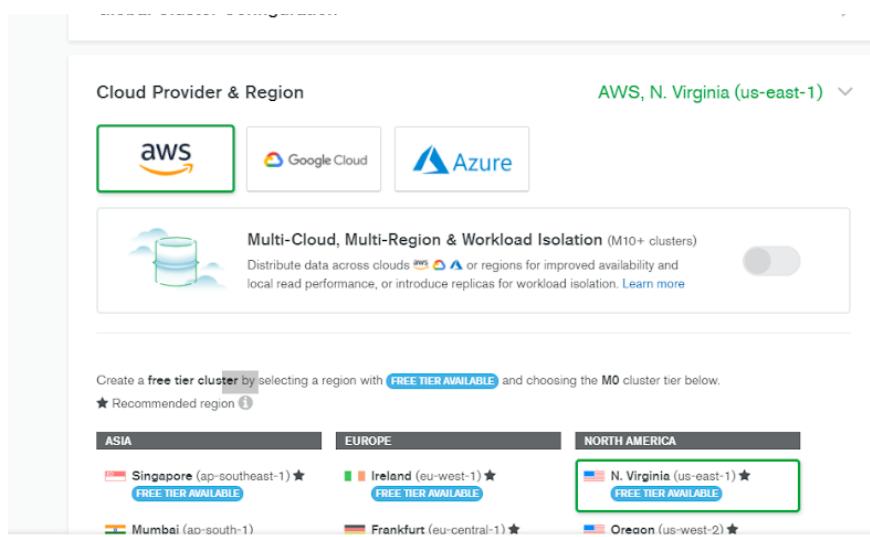
Cluster	Tier	Type	Region	Status
Cluster0	Sandbox	Replica Set - 3 nodes	AWS / N. Virginia (us-east-1)	Shutting down

Details for 'Cluster0':

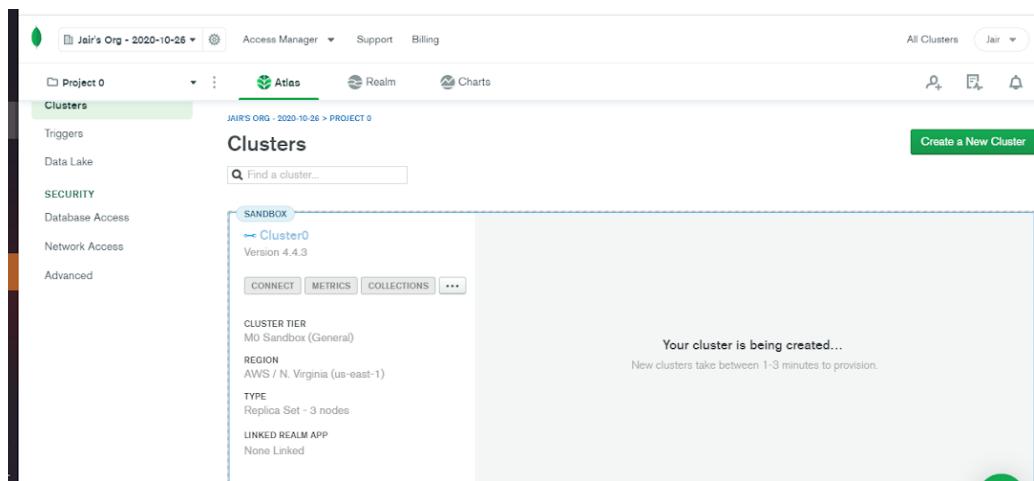
- CONNECT** button
- METRICS** tab (disabled)
- COLLECTIONS** tab
- CLUSTER TIER**: MO Sandbox (General)
- REGION**: AWS / N. Virginia (us-east-1)
- TYPE**: Replica Set - 3 nodes
- LINKED REALM APP**: None Linked

A message at the top says: "We are deploying your changes. 0 of 3 servers complete (current action: configuring MongoDB)". A green button on the right says "Create a New Cluster".

3) Seleccionar un proveedor de servicios, seleccionar un servidor y finalmente crear el cluster.



4) Una vez que el cluster haya sido creado, parecerá una imagen como la siguiente:



- 5) Para crear un usuario para la base de datos, ir al apartado Security y en Network Access dar click en Add new Database user.

The screenshot shows the MongoDB Atlas interface under the 'Database Access' section. On the left, there's a sidebar with 'Project 0' and tabs for 'Clusters', 'Triggers', 'Data Lake', 'SECURITY' (which is selected), 'Database Access' (which is also selected), 'Network Access', and 'Advanced'. The main area is titled 'Database Access' and has tabs for 'Database Users' (selected) and 'Custom Roles'. A table lists a single user: 'User Name' is 'jair', 'Authentication Method' is 'SCRAM', 'MongoDB Roles' is 'readWriteAnyDatabase@admin', 'Resources' is 'All Resources', and 'Actions' includes 'EDIT' and 'DELETE'. At the top right, there's a green button labeled '+ ADD NEW DATABASE USER'.

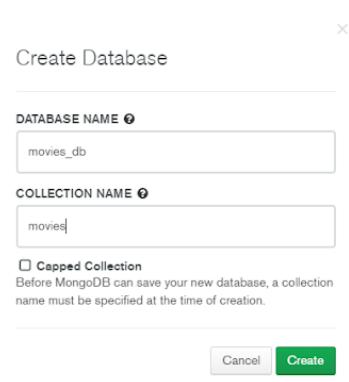
- 6) Creamos un nuevo usuario escribiendo nombre de usuario y contraseña.

- 7) Para obtener los datos de conexión a la base de datos, basta con dirigirse hacia el apartado clusters para después dar click en connect, en el cluster que acabamos de crear.

The screenshot shows the MongoDB Compass connection setup wizard. Step 1: 'Download the latest version of Compass (1.18.0)'. It offers two options: 'Ubuntu 64-bit (14.04+)' (selected) and 'Fetch via URL'. Below this, it says '② Copy the connection string below, and open Compass:' followed by a connection string: 'mongodb+srv://db_user_platzivideos:<password>@cluster0-higgd.mongodb.net/'. There is a 'Copy' button next to the string. A note below says: 'Replace <password> with the password for the db_user_platzivideos user. When entering your password, make sure that any special characters are URL encoded.' At the bottom, it says 'Having trouble connecting? View our troubleshooting documentation' and has 'Go Back' and 'Close' buttons.

8) Finalmente, necesitamos crear una colección, que es donde se almacenarán todos nuestros datos, basta con dar click en collections en el cluster que creamos, para posteriormente dar click en “add my own data”. Posteriormente le daremos un nombre a la base de datos y a la colección donde almacenaremos los datos.

9) Y finalmente, ya tenemos una base de datos con MongoDB



MongoDB Atlas crea réplicas, que están compuestas de 3 instancias de MongoDB, por lo que tendremos una alta disponibilidad de acceso a nuestra base de datos

Bibliotecas utilizadas y bibliotecas más usadas en node

Joi(Object Schema Validation). Sirve para validar que los datos que recibimos vengan en el formato adecuado.

Boom(HTTP-friendly error objects). Nos ayuda a imprimir errores comunes en peticiones HTTP de forma sencilla.

Body parser. Analiza los cuerpos de las solicitudes entrantes en un *middleware* antes que los manejadores de ruta disponibles bajo la propiedad `req.body`

CORS. *Middleware* para habilitar CORS (Cross-origin resource sharing) en nuestras rutas o aplicación.

Morgan. Un *logger* de solicitudes HTTP para Node.js.

Helmet. Helmet nos ayuda a proteger nuestras aplicaciones Express configurando varios encabezados HTTP.

Express Debug. Nos permite hacer *debugging* de nuestras aplicaciones en Express mediante el uso de un *toolbar* en la página cuando las estamos desarrollando.

Express Slash. Este *middleware* nos permite evitar preocuparnos por escribir las rutas con o sin *slash* al final de ellas.

Passport. Passport es un *middleware* que nos permite establecer diferentes estrategias de autenticación a nuestras aplicaciones.

Mocha: nos ayuda a correr los test.

Supertest: levanta un servidor temporal.

Sinon: crea mocks para tests.

Proxyquire: inyecta los mocks cuando se requieren los paquetes.

Haciendo *debugging*

Para aprovechar por completo la funcionalidad de *debugging* que implementa Express, se recomienda cambiar todos los console.log por debug haciendo uso de un *namespace* de la siguiente forma:

```
const debug = require("debug")("app:server");
debug("Hello debug");
```

De esta manera si ejecutamos nuestra aplicación con el comando `DEBUG=app:*` `node index.js` nos mostrará los diferentes logs. Express.js por defecto ya trae unos *logs* de *debugging* por defecto los podemos activar mediante la variable de entorno `DEBUG=express:*`.

Considerando las mejores prácticas para el despliegue

1. Remover contraseñas quemadas.
2. Encapsular código spaghetti.
3. Revisar la estructura del proyecto.
4. Configurar los scripts de build.
5. Agregar soporte de caché.
6. Añadir HTTPS y CORS.

Json Web Tokens

Un JWT consta de tres partes generalmente divididas por punto, ejemplo:

```
eyJdudsjfjhjdfdf.yudfndjfjnjdfnjsjfsfhsbfjsbd.nSDFsersfsdfsfsf.
```

Las partes del JWT son las siguientes:

Header. Tiene 2 atributos:

- El tipo que en este caso siempre debe ser JWT
- El algoritmo de encriptación de la firma

Payload. Guarda toda la información de:

- Usuarios
- Scopes de autorización

Signature. Se compone por:

- Header codificado
- Payload codificado
- Se emplea un Secret

JSON Web Tokens Auth y ventajas de JWT

En el proceso de Autenticación el server firma un token. A partir de ese momento el cliente almacena el token en memoria y en una cookie. Todos los request de ahí en adelante llevan el token.

Ventajas

1. No requiere del backend para saber si está autenticado porque lleva una firma (post autenticación).
2. El backend puede recibir múltiples request de múltiples clientes (sólo necesita saber si el token está bien firmado)
3. El cliente conoce los permisos que tiene, por lo que no los tiene que bajar de base de datos

Buenas prácticas con JSON Web token

En los últimos años se ha criticado fuertemente el uso de JSON Web Tokens como buena práctica de seguridad. La realidad es que muchas compañías hoy en día los usan sin ningún problema siguiendo unas buenas prácticas de seguridad, que aseguran su uso sin ningún inconveniente. Consejos que se deben tener en cuenta:

- Evitar almacenar información sensible.
- Mantener su peso lo más liviano posible
- Establecer un tiempo de expiración corto.

¿Dónde guardar los tokens?

Cuando estamos trabajando con SPA (Single Page apps) debemos evitar almacenar los tokens en Local Storage o Session Storage. Estos deben ser almacenados en memoria o en una Cookie, pero solo de manera segura y con el flag httpOnly, esto quiere decir que la cookie debe venir del lado del servidor con el token almacenado.

Conclusiones

A lo largo del desarrollo de este reto he podido comprender y aplicar muchos de los términos de Javascript en una aplicación de backend que puede ser consumida por un cliente a través de una Single Page Application. Así mismo, he aprendido sobre algunas herramientas que facilitan el proceso de desarrollo y que ayudan a escribir un mejor código, además de testearlo.