

MANUAL DE USUARIO “PRÁCTICA MICROSERVICIOS Y CONTENEDORES”

API REST DE CHISTES PARA CONTANAR EN UNA REUNIÓN



Ajitzi Ricardo Quintana Ruiz
8 de septiembre de 2020

Contenido

Lenguaje de programación.....	3
Estructura de mi API REST	3
Desarrollo de la práctica	4
Contenerizar la aplicación	16

Lenguaje de programación

El lenguaje de programación que elegir para hacer esta práctica es JavaScript, me parece un lenguaje muy versátil y amplia, con mucha documentación que te permite hacer cualquier cosa, es uno de mis favoritos y tengo la verificación de evaluaciones de LinkedIn en mi [perfil](#). Para hacerlo más eficaz utilizare un framework para Node, Express, que facilita mucho el código para las API's y para el uso del protocolo HTTP.

Como base de datos usare Mongo DB porque es una base de datos NoSQL que es basado en documentos y para aplicaciones sencillas orientadas a objetos es más que suficiente.



Estructura de mi API REST

Entrada

```
{
  "clave" : "maestro",
  "tipo" : "blanco",
  "longitud": "corto",
}
```

La entrada va a ser en formato JSON donde: clave, son las palabras que contenga el chiste; tipo, es tipo string con 2 opciones blanco o negro dependiendo de las personas en la reunión es el tipo de humor; longitud, son 2 opciones cortos (máximo 255 caracteres) y largos (más de 255 caracteres).

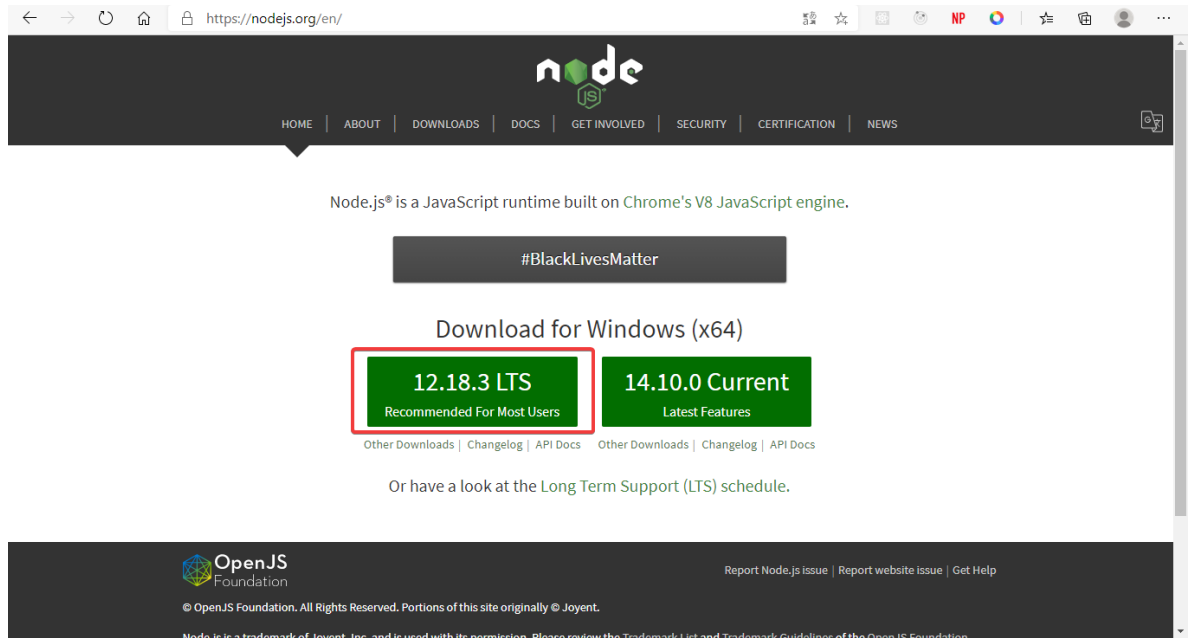
Salida

```
{
  "encontrado": true,
  "chistes": [
    {
      "chiste" : "- Andresito, ¿qué planeta va después de Marte? - Miércoles, maestro.",
      "tipo" : "blanco",
      "longitud": "corto",
      "gracia" : 5
    },
    {
      "chiste" : "- Mama, mama!!! En clase soy el mas alto y el que mas sabe!! - Claro cariño... eres el maestro...",
      "tipo" : "blanco",
      "longitud": "corto",
      "gracia" : 8
    }
  ]
}
```

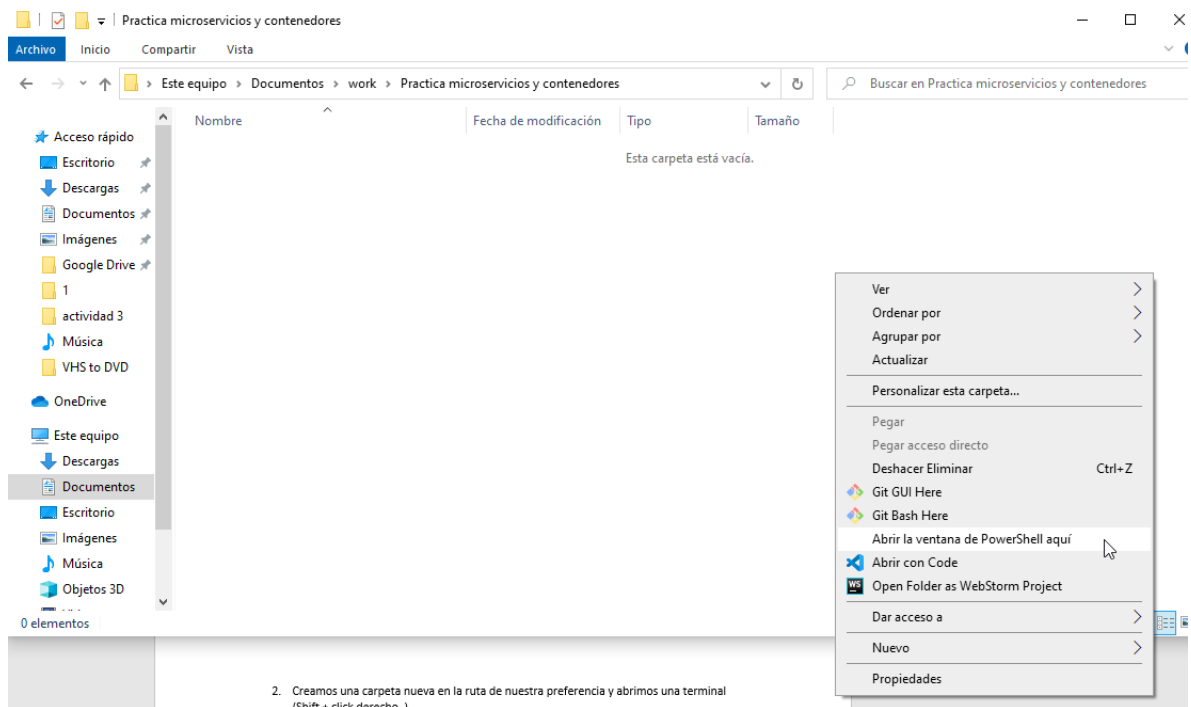
La salida va a ser en formato JSON con una propiedad, encontrado, que es un booleano que avisa si se encontraron chistes dependiendo de la entrada. Un arreglo de objetos donde vendrán los chistes encontrados donde: chiste, es un string con el chiste; tipo y longitud son igual que en la entrada y gracia que es un entero de 0 a 10 dependiendo de cuanta gracia le causo al que escribió el chiste en la base de datos, en este caso yo.

Desarrollo de la práctica

1. Descargamos e instalamos Node de la página [oficial](#). Es recomendable la versión LTS porque es más estable.



2. Creamos una carpeta en nuestra ruta de preferencia (el nombre de la carpeta debe ser sin espacios, estos se pueden sustituir con guiones). Abrimos una terminal (Shift + click derecho, abrir la ventana de PowerShell aquí).



3. Escribimos en la terminal el siguiente comando. Este lo que hará será crearnos el archivo package.json que contiene información de nuestra aplicación, nombre, descripción, dependencias, scripts, etc.

```
Windows PowerShell
PS C:\Users\ajitz\Documents\work\Practica_microservicios> npm init -y
Wrote to C:\Users\ajitz\Documents\work\Practica_microservicios\package.json:

{
  "name": "Practica_microservicios",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PS C:\Users\ajitz\Documents\work\Practica_microservicios>
```

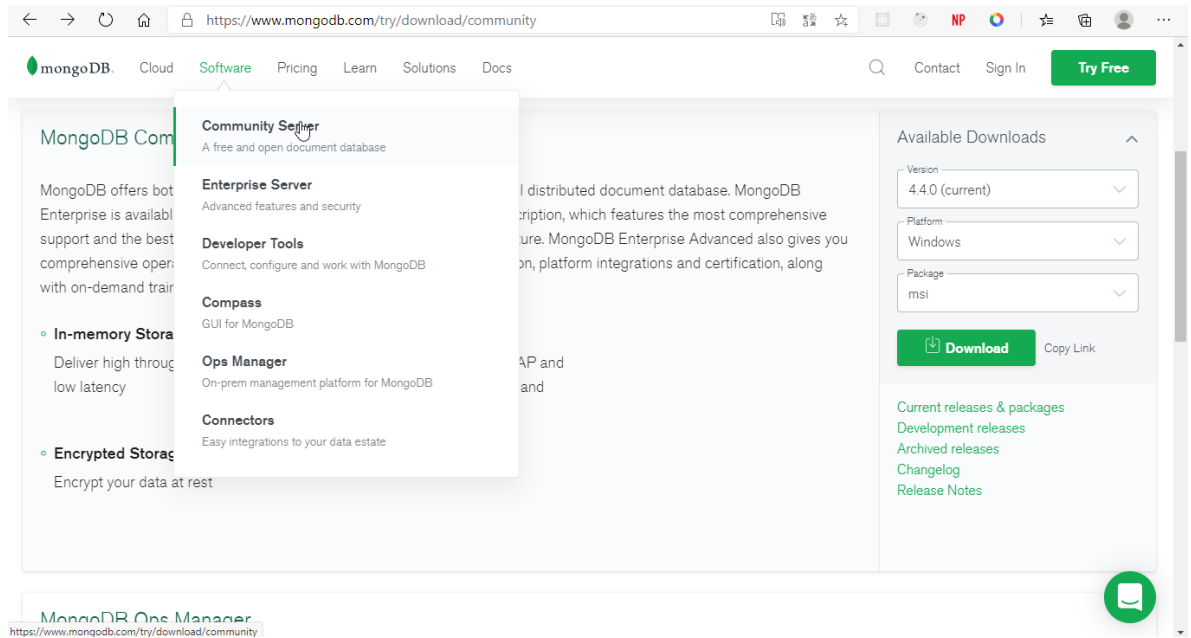
4. Instalamos nuestro framework Express con el siguiente comando, este comando se obtuvo de la página principal de [Express](#). Esto creará un archivo llamado package-lock.json que contiene el orden en que se crearon las dependencias (**este archivo no debe ser editado manualmente**). Y una carpeta node_modules que contiene las dependencias para que nuestra aplicación se ejecute correctamente.

```
PS C:\Users\ajitz\Documents\work\Practica_microservicios> npm install express --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN Practica_microservicios@1.0.0 No description
npm WARN Practica_microservicios@1.0.0 No repository field.

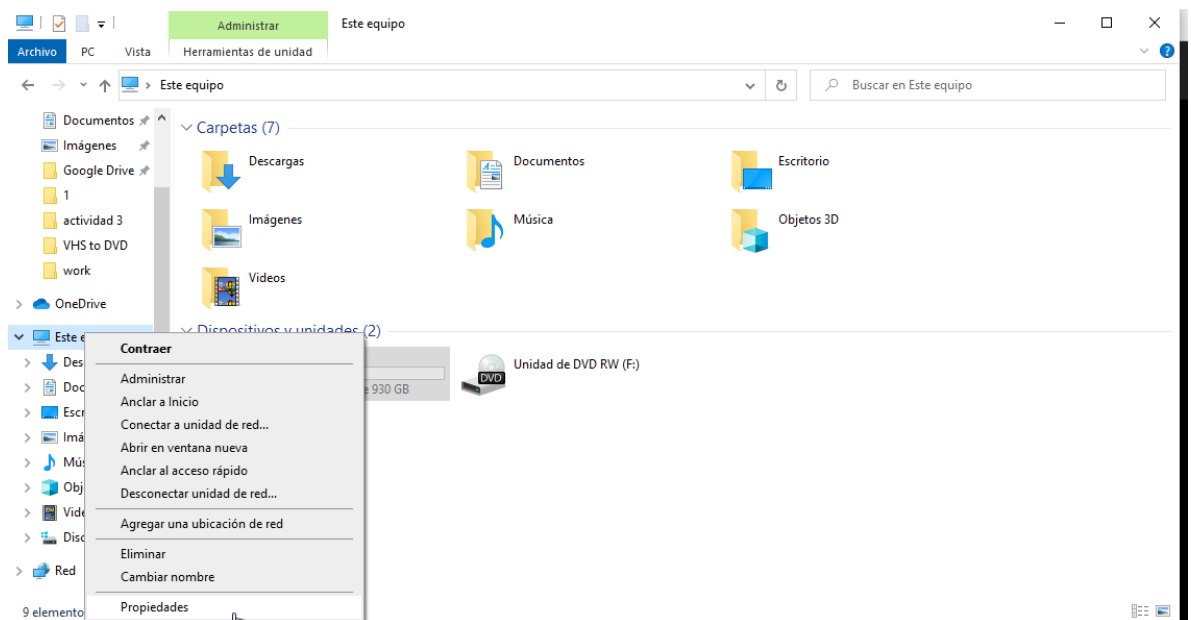
+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 10.14s
found 0 vulnerabilities

PS C:\Users\ajitz\Documents\work\Practica_microservicios>
```

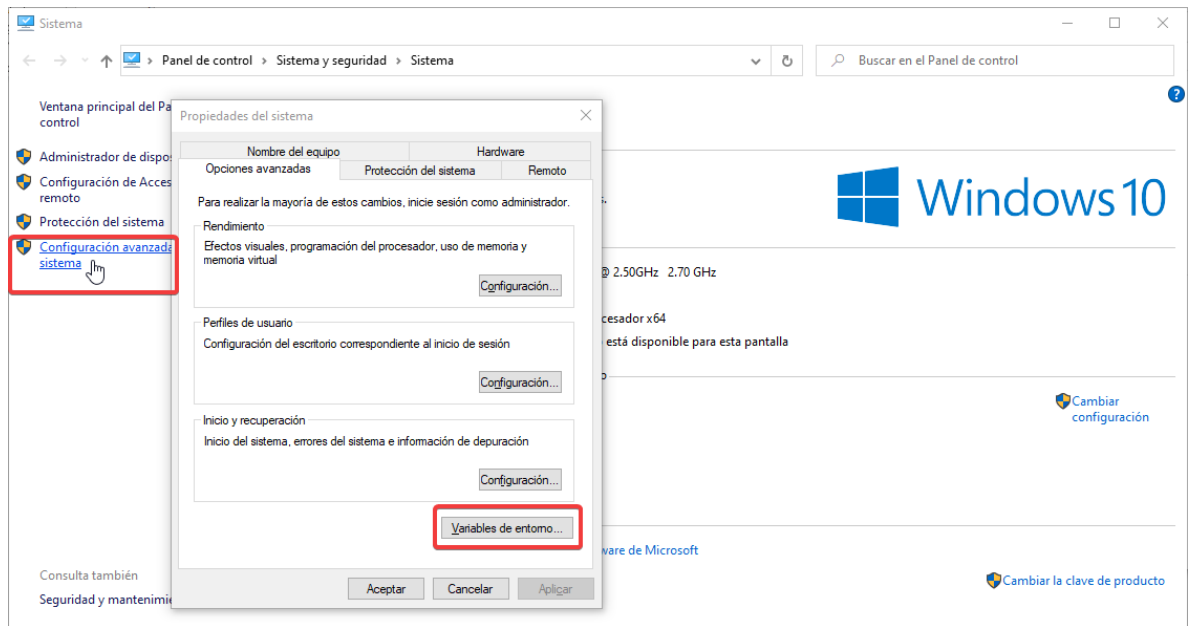
5. Descargamos Mongo DB Server de su [página oficial](#), instalamos la versión completa y Mongo Compass que es una interfaz para nuestra base de datos.



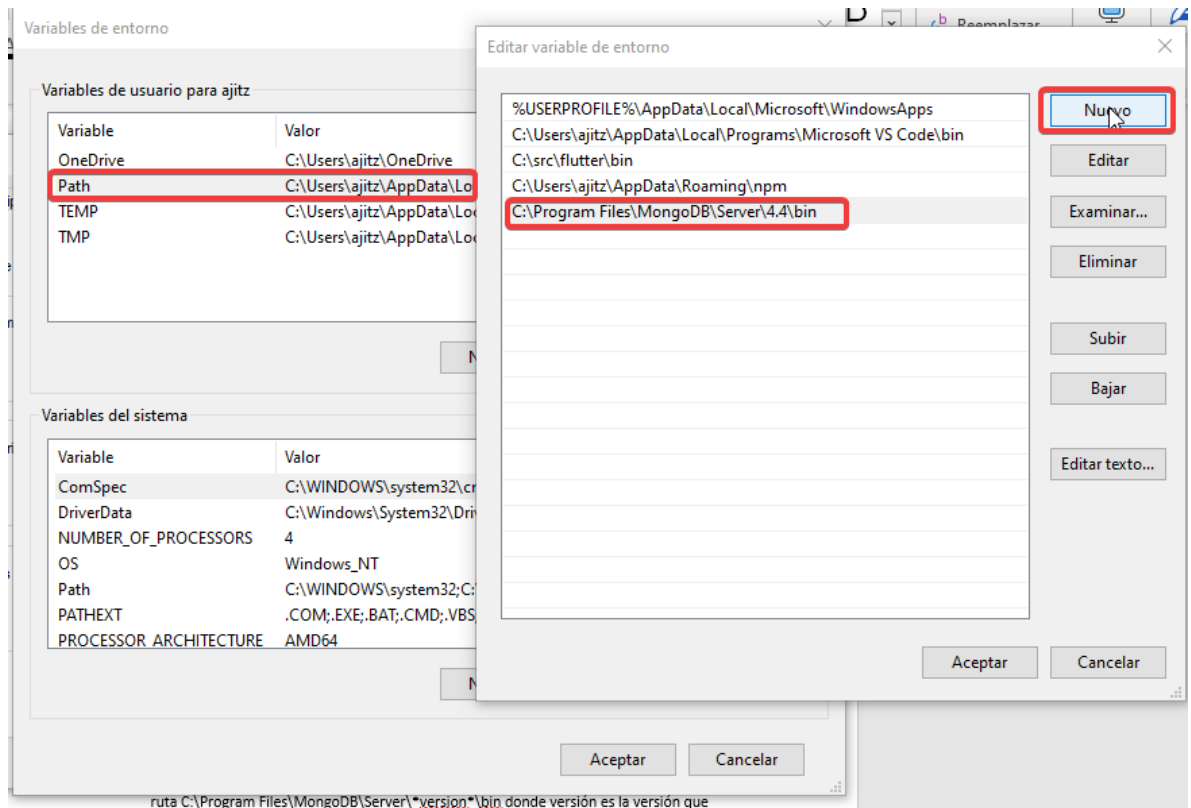
- Ahora agregaremos la variable de entorno a nuestra computadora. Navegamos hasta la ruta `C:\Program Files\MongoDB\Server*versión*\bin` donde versión es la versión que descargaron. Y copiamos dicha ruta, abrimos el explorador de archivos hacemos click derecho en este equipo y seleccionamos propiedades



Despues nos vamos a configuración avanzada del sistema -> variables de entorno

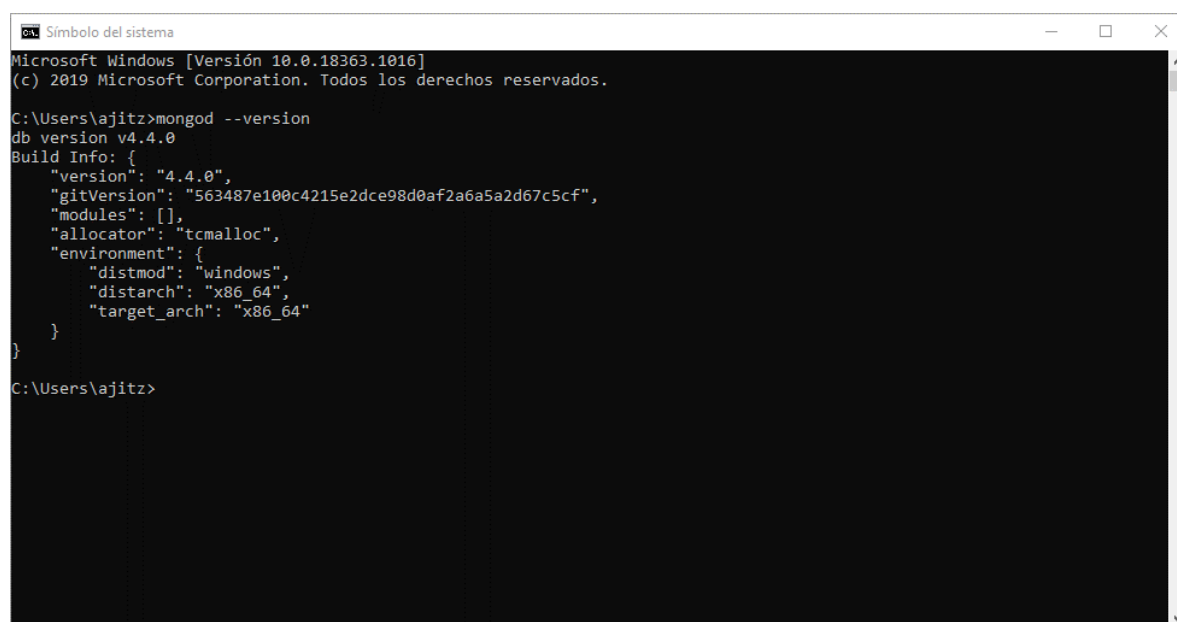
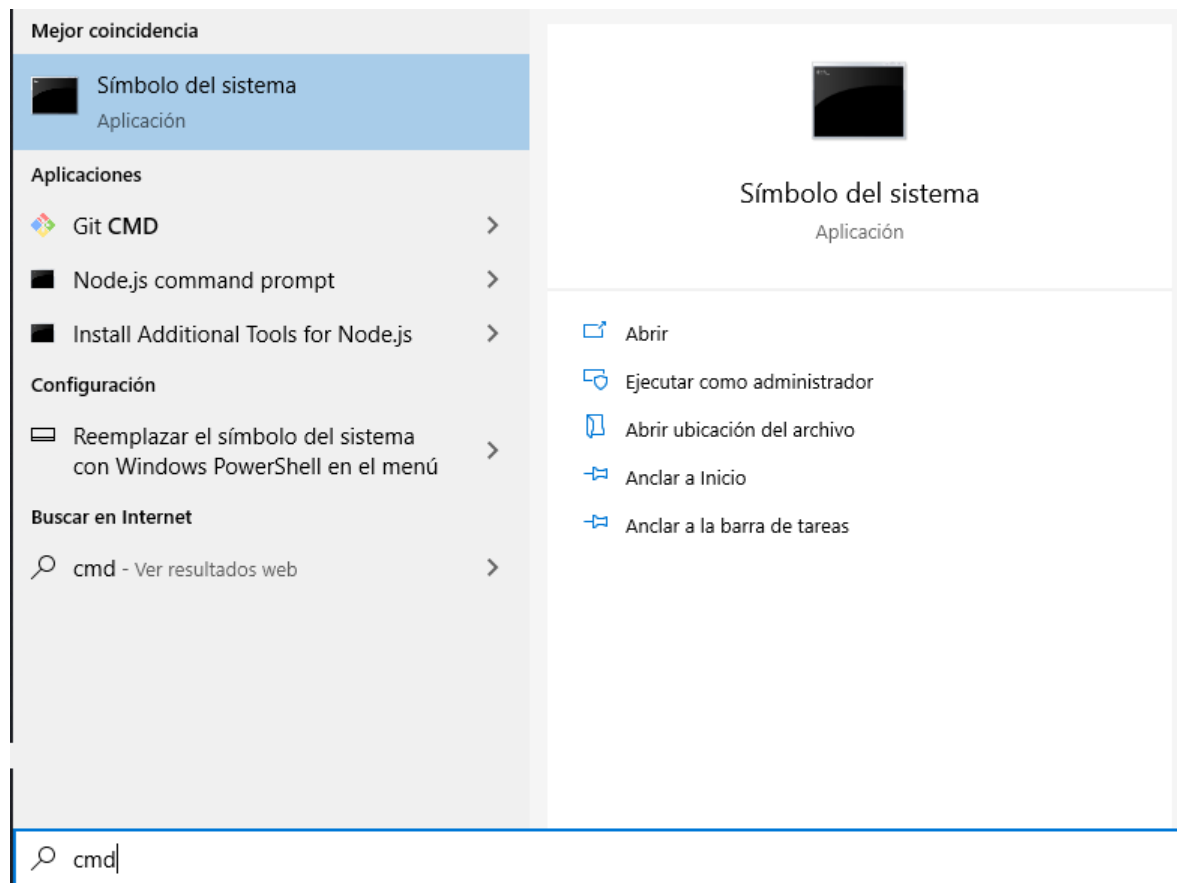


Hacemos click en path -> nuevo, pegamos la ruta y le damos en aceptar a todas las ventanas anteriores. Esto nos permitira levantar el servidor de Mongo DB en la terminal sin importar en la carpeta que estemos.

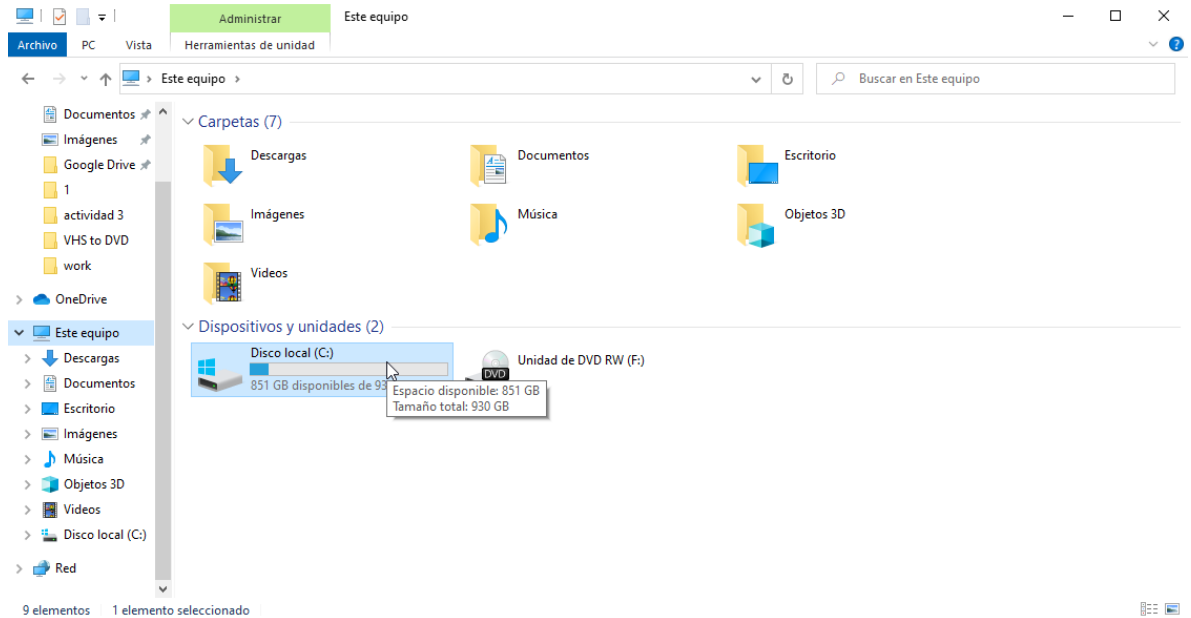


Para comprobarlo abrimos una terminal buscando cmd en el inicio y corremos el siguiente

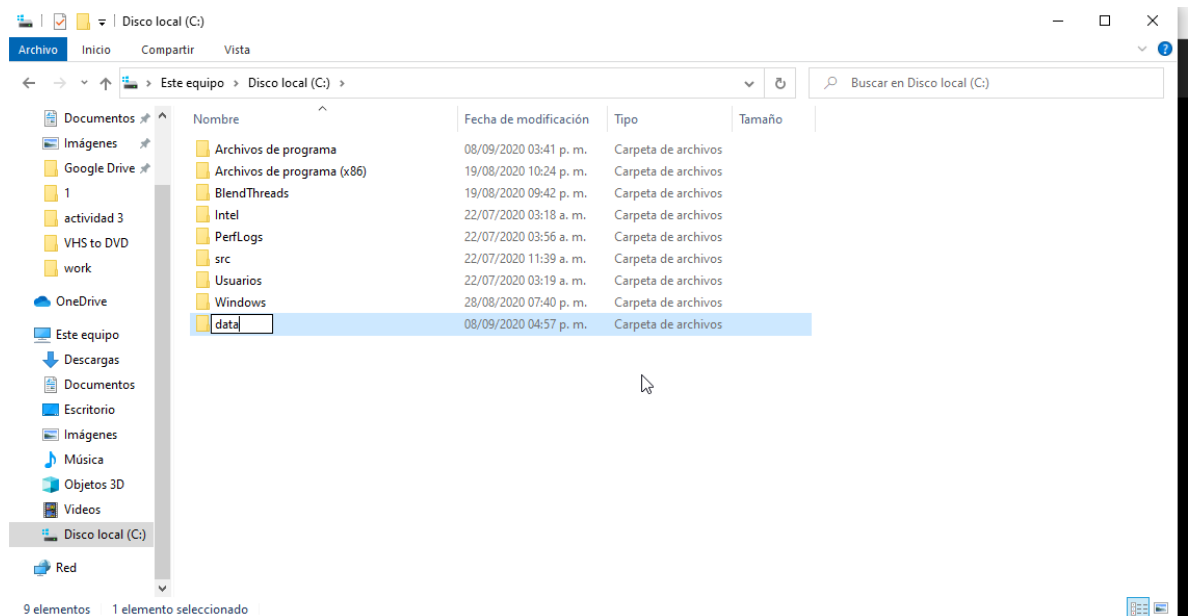
comando, si nos aparece esto quiere decir que creamos la variable de entorno correctamente.

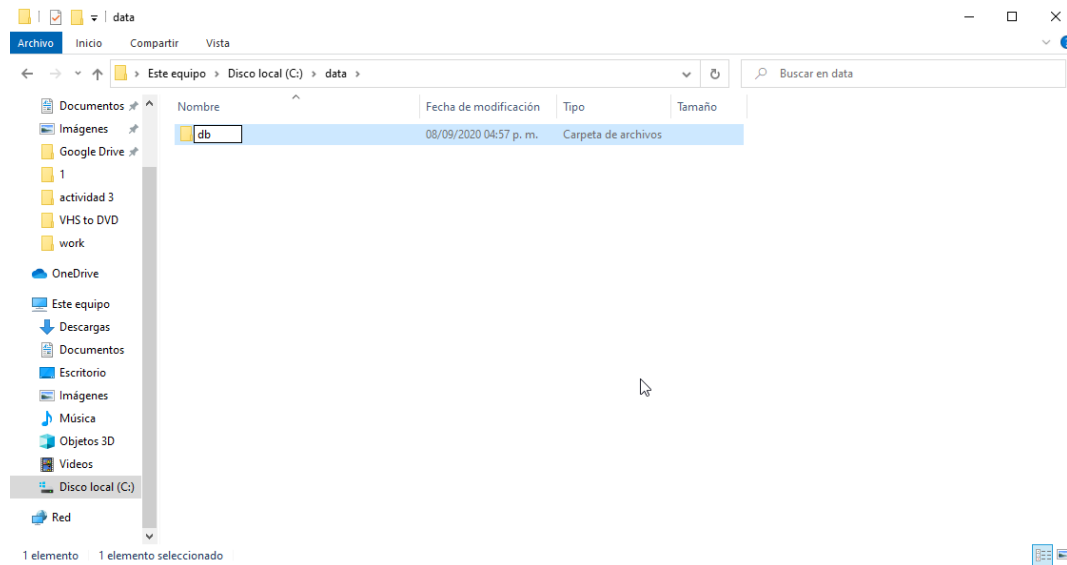


7. Abrimos un explorador de archivos -> este equipo -> disco local C:

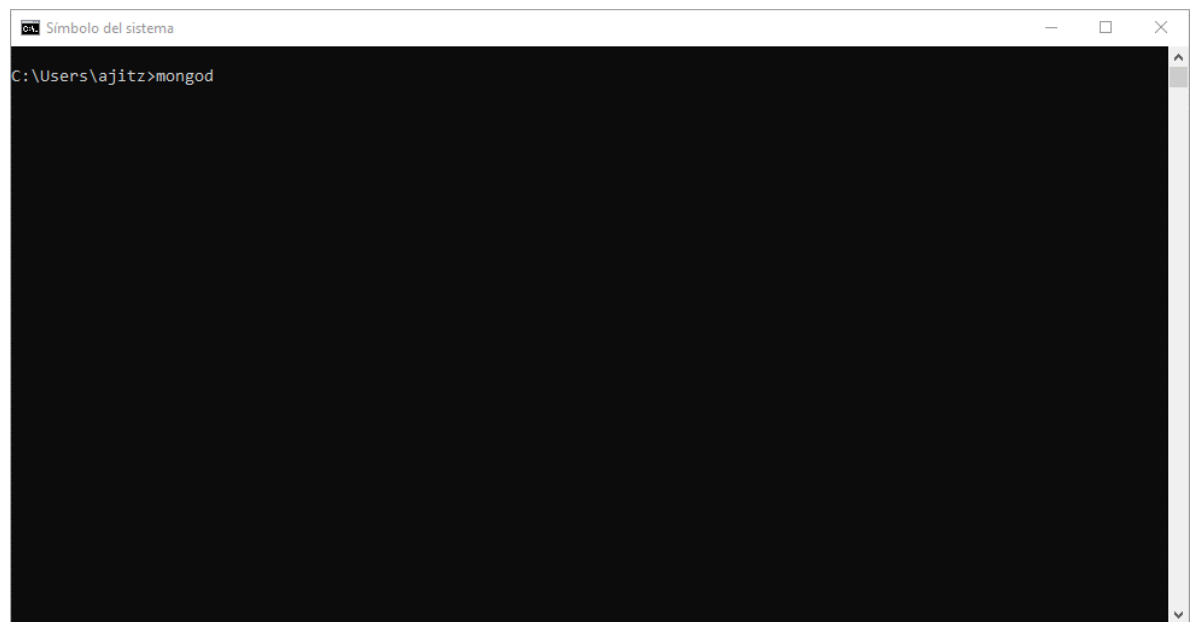


creamos una carpeta con nombre data, abrimos esta carpeta y creamos una nueva carpeta con nombre db





8. Abrimos otra terminal cmd como anteriormente lo realizamos y ejecutamos el siguiente comando, esto levantara el servicio de Mongo DB Server



9. Abrimos otra ventana cmd y ejecutamos el siguiente comando, damos enter y nos aparecerá lo siguiente

```
Símbolo del sistema - mongo
C:\Users\ajitz>mongo
MongoDB shell version v4.4.0
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("5896ddac-2921-4cd6-98f7-d7611b7bfa09") }
MongoDB server version: 4.4.0
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
  https://community.mongodb.com
---
The server generated these startup warnings when booting:
  2020-09-08T15:45:48.755-05:00: ***** SERVER RESTARTED *****
  2020-09-08T15:45:53.374-05:00: Access control is not enabled for the database. Read and write access to data and
  configuration is unrestricted
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
>
```

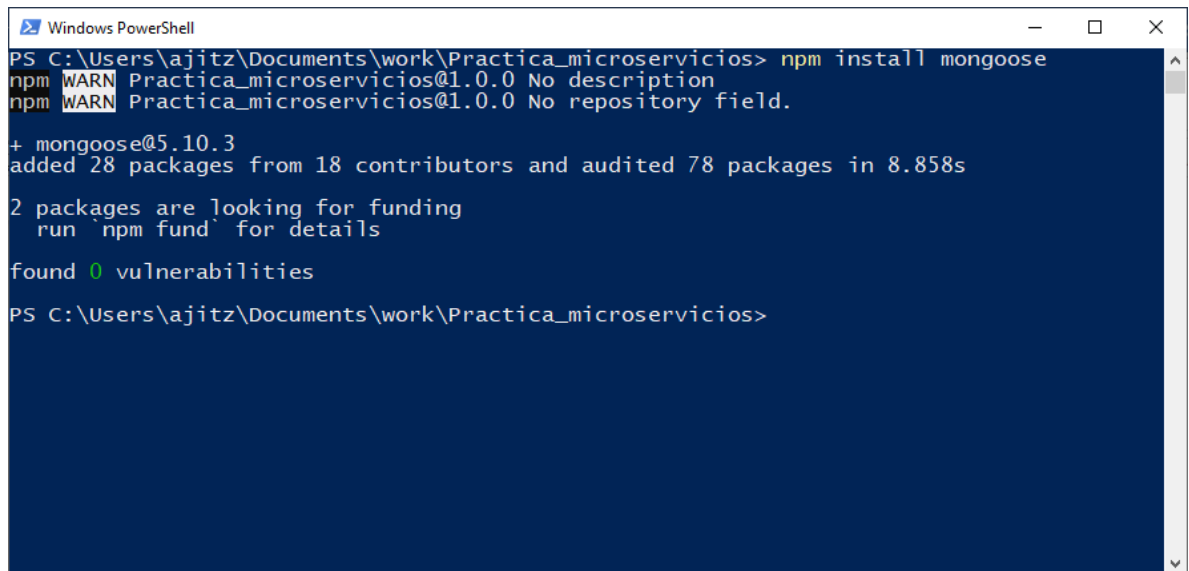
10. Ejecutamos los siguientes comandos en esta consola, use es para crear nuestra base de datos y createCollection para crear una nueva colección donde estarán nuestros chistes

```
Símbolo del sistema - mongo
> use microservicio
switched to db microservicio
> db.createCollection("chistes")
{ "ok" : 1 }
>
```

11. Creamos un JSON con un arreglo con los chistes que queremos insertar en nuestra base de datos, yo voy a usar [Visual Studio Code](#) para hacerlo más sencillo y más visible. escribiré pocos chistes porque es una prueba, todos serán sacados de internet. (El archivo chistes.json estará en el repositorio también).



13. Abrimos la terminal dentro de nuestro proyecto y ejecutamos el comando “npm install mongoose” este es un paquete que necesitamos para comunicar nuestra base de datos con nuestra aplicación



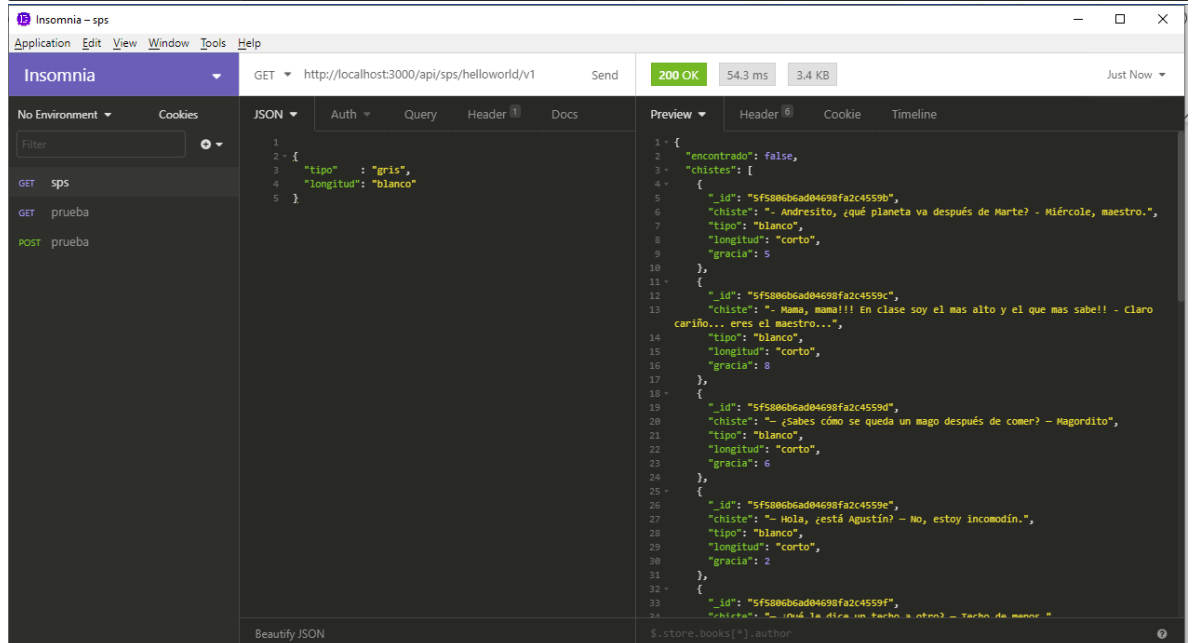
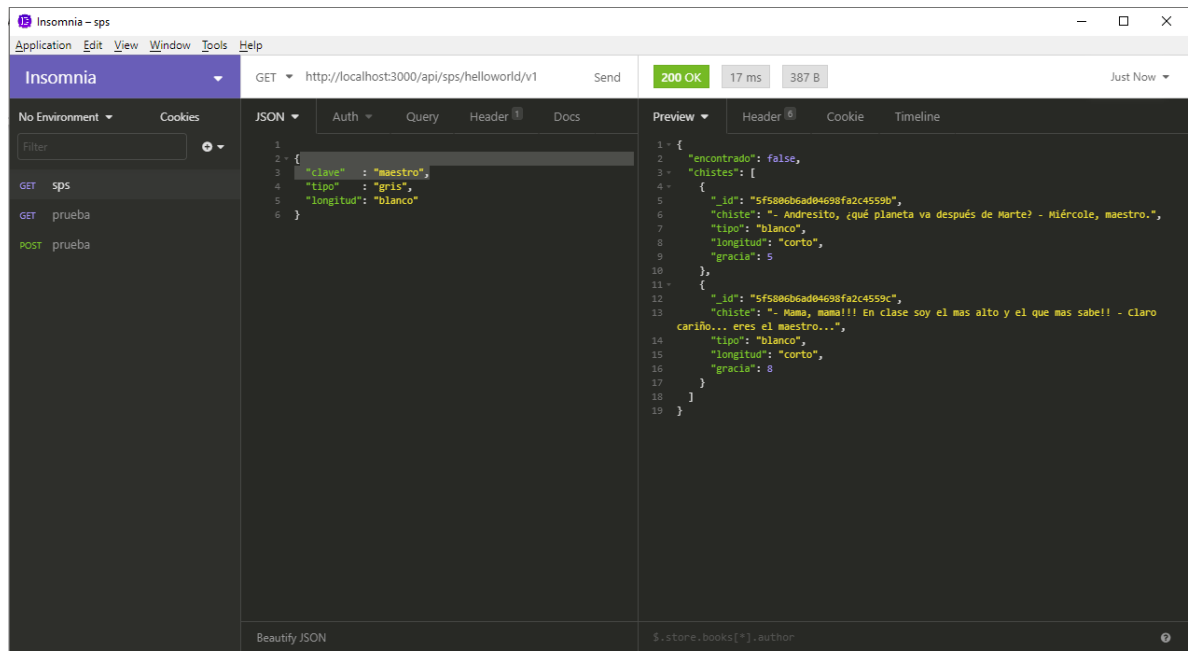
```
Windows PowerShell
PS C:\Users\ajitz\Documents\work\Practica_microservicios> npm install mongoose
npm WARN Practica_microservicios@1.0.0 No description
npm WARN Practica_microservicios@1.0.0 No repository field.
+ mongoose@5.10.3
added 28 packages from 18 contributors and audited 78 packages in 8.858s

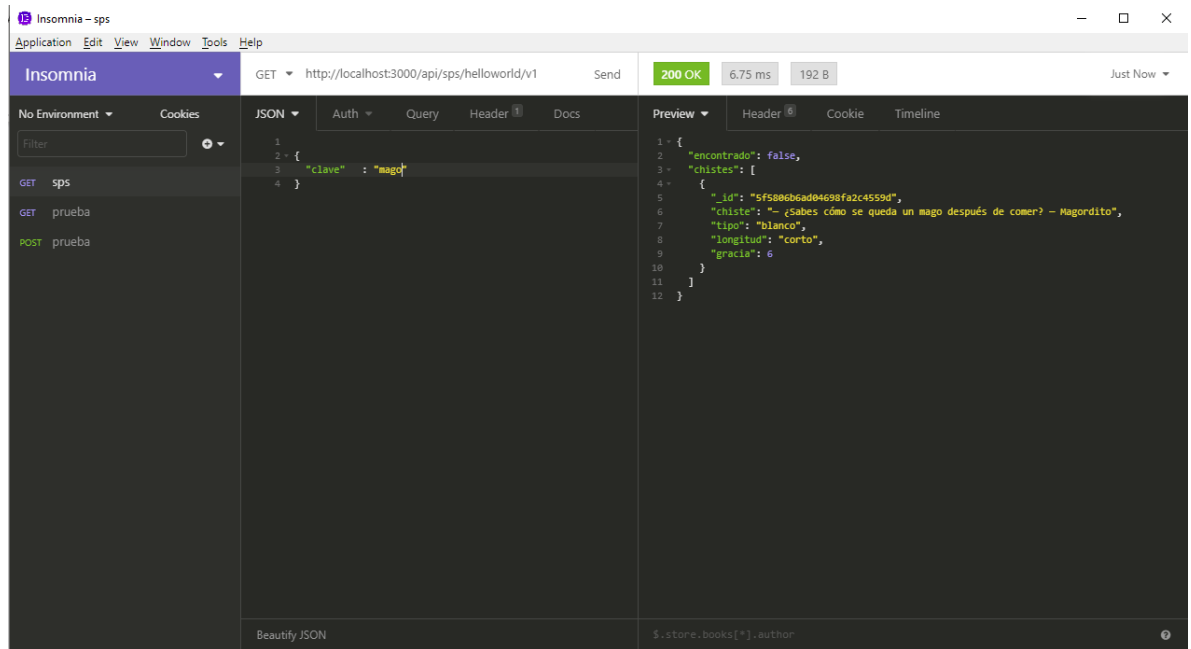
2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

PS C:\Users\ajitz\Documents\work\Practica_microservicios>
```

14. Creare el código para que se ejecute nuestra aplicación y la documentare dentro del mismo.
15. Instalamos cualquier programa que nos permita hacer peticiones http, a mí me gusta más insomnia que otros programas porque me parece más minimalista pero cualquiera funcionaria. Hacemos una petición get a nuestro localhost con el url requerido, en el cuerpo de la petición ponemos JSON llenamos los datos y le damos enviar. Podemos ver que nuestra búsqueda si cambia dependiendo los valores de entrada.



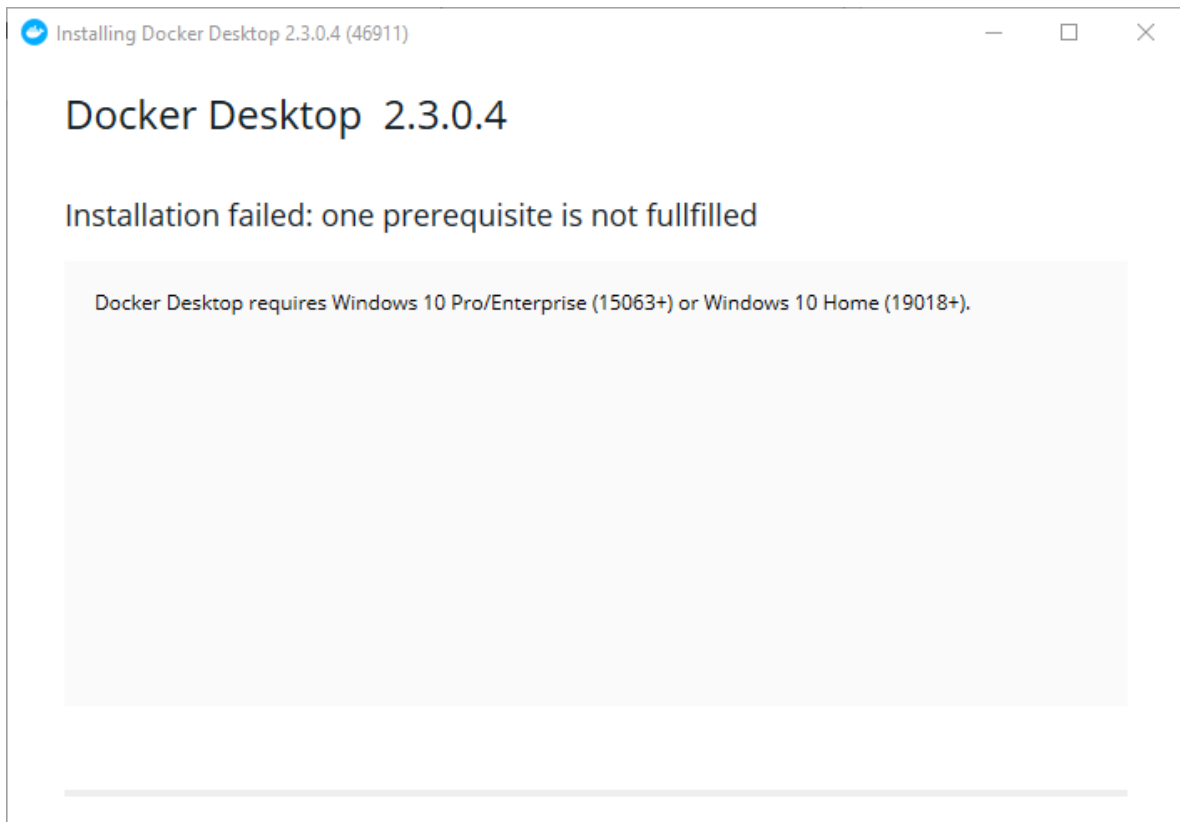


Contenerizar la aplicación

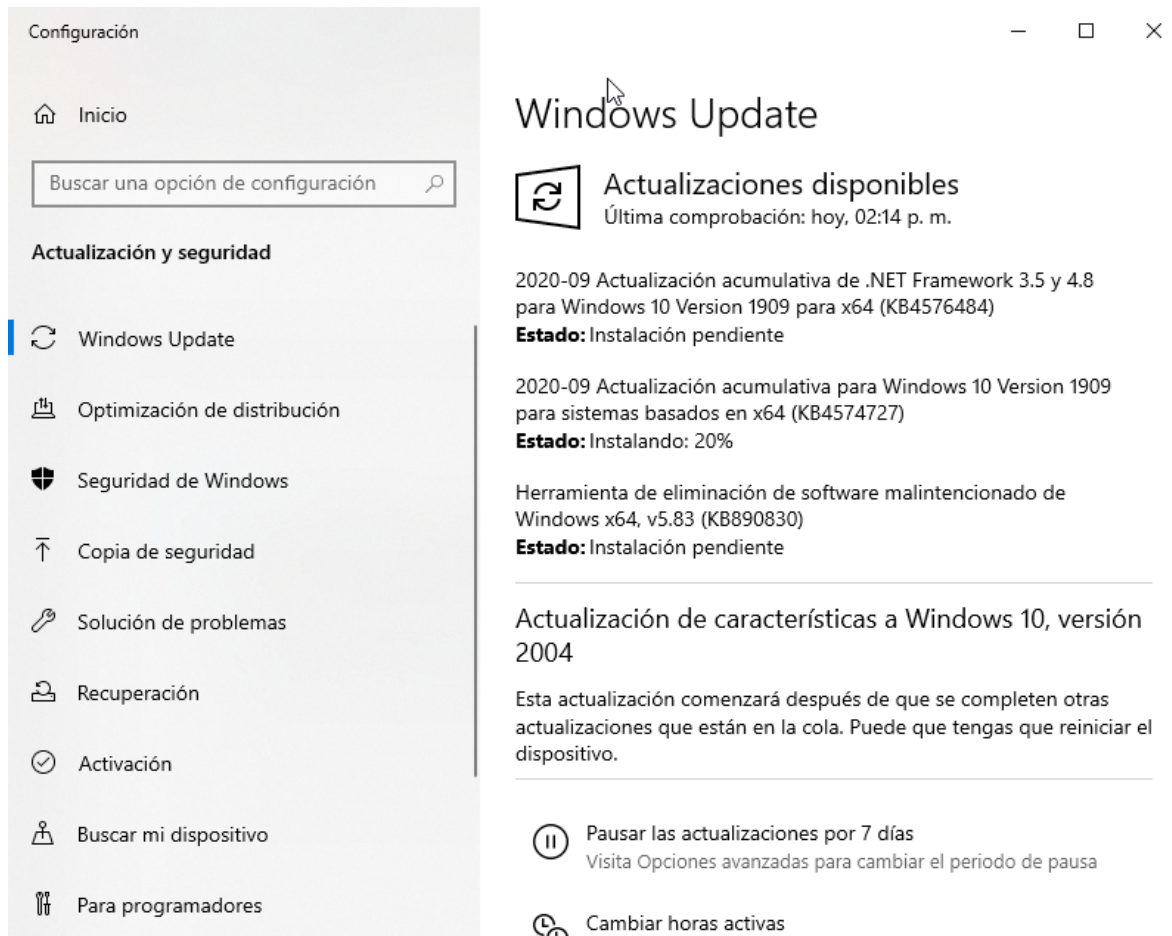
Hasta este punto todo lo anterior yo ya lo había realizado en mi experiencia laboral y en proyectos escolares, hasta este punto lo que me llevo más tiempo fue la documentación.

De los contenedores yo tenía una idea muy vaga de que eran como máquinas virtuales para correr una aplicación sin el problema de no tener la misma versión y demás. Pero nunca la había utilizado.

Mi primer problema fue que mi Windows no estaba actualizado.



Y si es verdad que mi Windows no estaba actualizado



Me puse a actualizar mi Windows y esto me quito algo de tiempo, mientras tanto estuve leyendo la [documentación oficial](#) y leyendo [tutoriales](#) de como contenerizar una aplicación, entendí que tenemos que hacer una imagen de nuestra aplicación y después usar Docker-compose para juntarlas en un solo contenedor.

Mientras se actualizaba Windows fue creando la imagen de mi aplicación con Express

```
Dockerfile X  docker-compose.yml  config.js  index.js  chistes.js
Practica_microservicios > Dockerfile > ...
1  # version oficial de Node de Docker
2  FROM node:14
3  # mkdir es el comando para crear carpetas
4  RUN mkdir -p /usr/src/app
5  # WORKDIR es para cambiar la ruta donde se estan ejecutando los comandos
6  WORKDIR /usr/src/app
7  # se copian las dependencias necesarias
8  COPY package*.json ./
9  # se instalan las dependencias de package.json
10 RUN npm install
11 # se copia todo el codigo en la carpeta nueva
12 COPY . .
13 # el puerto en donde se va a correr el contenedor
14 EXPOSE 8090
15 # el comando para ejecutar la API
16 CMD ["npm","start"]
```

No hice una imagen de mongo porque leí en la [documentación oficial](#) de Docker que ya tiene una imagen para eso.

Después fue creando el contenedor principal donde estarán mis dos servicios, la aplicación con express y la base de datos de mongo.

```
Dockerfile  docker-compose.yml X  config.js  index.js  chistes.js
docker-compose.yml
1  version: '3' # La version de docker-compose
2
3  services:
4    express:
5      build: Practica_microservicios # La carpeta donde esta el dockerfile
6      ports:
7        - "8090:8090" # Los puertos donde se levantara la aplicacion
8      links: # depende de otro servicio
9        - mongo
10     volumes: # mantener la informacion
11       - ./usr/src/app
12     mongo:
13       image: mongo # La imagen que docker tiene de MongoDB
14       ports:
15         - "27017:27017" # Los puertos donde se levantara la base de datos
16       logging:
17         driver: none
```

Ejecutamos los siguientes comandos en consola.

```
MINGW64/c:/Users/ajitz/Documents/work
ajitz@DESKTOP-8SOAL: MINGW64 ~/Documents/work (master)
$ docker-compose build
mongo uses an image, skipping
Building express
Step 1/8 : FROM node:14
14: Pulling from library/node
419e7ae5bb1e: Pull complete
848839e0cd3b: Pull complete
de30e8b35015: Pull complete
258fdea6ea48: Pull complete
ca1b0e608d7b: Pull complete
dd8cac1f0c02: Pull complete
ea929affc906: Pull complete
a247faa805ec: Pull complete
e6f246c690fd: Pull complete
Digest: sha256:77a2dd1c6b9f21a8dc3fe0392338adc56355f0252e22918487cd48c4e4f55217
Status: Downloaded newer image for node:14
---> c2b4a3b480bf
Step 2/8 : RUN mkdir -p /usr/src/app
---> Running in 07181f79b5f5
Removing intermediate container 07181f79b5f5
---> d0f6d1a7ea48
Step 3/8 : WORKDIR /usr/src/app
---> Running in 2952baa8d596
Removing intermediate container 2952baa8d596
---> ea2b29b7d443
Step 4/8 : COPY package*.json ./
---> 549b45d9a4df
Step 5/8 : RUN npm install
---> Running in 4a0a43e157f8
```

```
ajitz@DESKTOP-8SOAL: MINGW64 ~/Documents/work (master)
$ docker-compose up
Pulling mongo (mongo)...
latest: Pulling from library/mongo
f08d8e2a3ba1: Pull complete
3baa9cb2483b: Pull complete
94e5ff4c0b15: Pull complete
1860925334f9: Pull complete
9d42806c06e6: Pull complete
31a9fd218257: Pull complete
5bd6e3f73ab9: Pull complete
f6ae7a64936b: Pull complete
80fde2cb25c5: Pull complete
1bec62fe62fc: Pull complete
2cf4970a1653: Pull complete
39fac3226e16: Pull complete
86bca9c64faf: Pull complete
Digest: sha256:77f3d623d475c56ab43d61864bc3aa610953869724dbbea25989a0718741d1c
Status: Downloaded newer image for mongo:latest
Creating work_mongo_1 ... done
Creating work_express_1 ... done
Attaching to work_mongo_1, work_express_1
mongo_1 | {"t":{"$date":"2020-09-10T04:33:57.004+00:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"
mongo_1 | {"t":{"$date":"2020-09-10T04:33:57.017+00:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main","msg":"
No TransportLayer configured during NetworkInterface startup"}
mongo_1 | {"t":{"$date":"2020-09-10T04:33:57.018+00:00"},"s":"I", "c":"NETWORK", "id":4648601, "ctx":"main","msg":"
Implicit TCP FastOpen unavailable. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
```

Comprobamos que funciona en Insomnia

GET http://localhost:8090/api/sps/helloworld/v1

Send

200 OK613 ms192 BJust Now

JSONAuthQueryHeader1Docs

PreviewHeader6CookieTimeline

```
1
2 {
3   "clave" : "mago"
4 }
```

```
1 {
2   "encontrado": false,
3   "chistes": [
4     {
5       "id": "5f5086b6ad04698fa2c4559d",
6       "chiste": "- ¿Sabes cómo se queda un mago después de comer? - Magordito",
7       "tipo": "blanco",
8       "longitud": "corto",
9       "gracia": 6
10    }
11  ]
12 }
```