NODE.JS EXPRESS Y MONGO

# INSTALAR NODE.JS

Se recomienda siempre utilizar nvm (node version maanger) para instalar node.

NVM se puede conseguir a través de repositorio de GIT.

## NODEMON

Para instalar nodemon, la forma más recomendada es instalar **en el proyecto** nodemon con:

npm init -y (-y para que no pregunte)

npm install nodemon

Para usarlo, tendremos que llamar a:

./node\_modules/.bin/nodemon

Para usarlo más fácil, utilizar npx después de instalar nodemon

npx nodemon

NODEMON es una utilidad de desarrollo. Podemos instalarla sólo para el mismo y que no se cargue en los proyectos:

npm install -D nodemon

## UTILIDADES DE NPM

npm repo *librería* → lleva al repositorio GIT de la librería.

[http://npmjs.com](http://npmjs.com/) → Página oficial de npm, con las librerías registradas.

## UTILIDADES DE JAVASCRIPT

Librería async (npm repo async)

## OBJETO PROCESS DE NODE.JS

Es un objeto global que nos da el estado de la aplicación.

**process.platform →** devuelve la plataforma en la que corre.

**process.exit(int)** → termina el proceso y devuelve un código de salida.

Se puede ver la documentación en la página oficial de node.

**process.nextTic(callback)** → Coloca nuestra función callback al principio de la cola del event loop.

**process.stdin.on(‘data’, function(data) {file.write(data)}** → recoge y envía los datos recibidos de la entrada estándar.

## EVENTOS

Node trabaja en un solo hilo, con un event loop, y en cada vuelta atiende los eventos recibidos, pasándolos al manejador correspondiente.

## MÓDULOS

En node, por defecto, son módulos CommonJS.

Son **singleton** (no puede haber dos instancias en memoria).

# CREACIÓN DE APLICACIONES CON NODE.JS

## CREACIÓN DE UNA APLICACIÓN CON EXPRESS Y GENERADOR DE VISTAS EJS

npx express-generator *nombreaplicación* –ejs

cd *nombreaplicación*

npm install

DEBUG=nodeapp:\* npm start → lanzar la aplicación con desarrollo y debug.

Para lanzarlo automáticamente con npm run dev, en linux y otros sistemas diferentes de windows, se puede modificar en package.json, en el apartado scripts

"scripts": {

"start": "node ./bin/www",

"dev":"DEBUG=nodeapp:\* npm start"

}

Para windows, hay que instalar la librería cross-env, y poner

"dev":"cross-env DEBUG=nodeapp:\* npm start"

Y podemos utilizar nodemon como

"dev":"DEBUG=nodeapp:\* nodemon ./bin/www"

Express siempre debe estar instalado en la aplicación para que funcione.

## ESTRUCTURAR APLICACIONES

Patrón MVC (Modelo – Vista – Controlador)

Vista → lo que se ve

Controlador → indica qué página mostrar y le pasa los datos para renderizar según el modelo.

Modelo → Es el core de la aplicación.

## VALIDACIONES

Es responsabilidad del backend la validación final de los datos.

Se pueden utilizar librerías, como por ejemplo express-validator.

## API

Los APIS devuelven datos (generalmente JSON) en vez de un html.

Para ello, generar un módulo con los datos o el modelo a responder, y la respuesta validada, exportando el módulo.

Asegurar en la app que el manejador de errores captura los errores del API para responder con JSON.

# PROMESAS, BASES DE DATOS, API

## PROMESAS

Los callbacks no son la mejor manera de trabajar con asincronía.

Para evitar la anidación problemática de callbacks (callback hell) se crearon las promesas.

Tiene una sintaxis más fácil.

Una promesa es un objeto que representa una operación que todavía no se ha completado.

Tres estados posibles en una promesa:

- Pending

- Fullfilled (value(

- Rejected

### ASYNC / AWAIT

Async hace que una función devuelva una promesa. **async** function

Await consume una promesa. Await se usa siempre dentro de una función async

Asíncrono en paralelo:

## BASES DE DATOS

Node se conecta a bases de datos a través de módulos de terceros.

### SQL

Recomendado el uso de Docker → gestor de control de bases de datos. Como nvm pero para bases de datos.

Para bases de datos MySql, usar librería mysql2, que trabaja con promesas.



Recomendado Prisma. Traducen y dan soporte para convertir los resultados en objetos y los objetos en elementos para hacer persistencia en la base de datos.

## mongoDB

### Instalar

Descargar e instalar Community Server. Arrancar el servicio *./bin/mongod –dbpath=”./data*

Después descargar e instalar el shell de la sección tools.

Otra tool importante es https://nosqlbooster.com/

### Comandos

show databases o show dbs→ muestra las bases de datos

use *database* → utiliza el contexto indicado, exista o no. Si no existe, no lo crea hasta que se persiste algún dato.

Show collections → muestra las tablas

db.*tabla*.insert({name: ‘Brown’, age: 28}) → inserta datos (formato objetos)

db.*tabla*.find() → devuelve datos

db.*tabla*.find().pretty() → formatea salida más legible.

db.agentes.remove({"\_id" : ObjectId("62729a134c9dc509aff86de5")}) → elimina el registro con id…

db.agentes.update(*filtro* { "\_id" : ObjectId("631796c0203517dd9f0c5f14"),{$set: {age:48}}) **en versiones anteriores a la 6.**

db.agentes.updateOne(*filtro* { "\_id" : ObjectId("631796c0203517dd9f0c5f14"),{$set: {age:48}}) **en versiones actuales.**

Índices

db.agentes.getIndexes() → Mostrar los indices actuales.

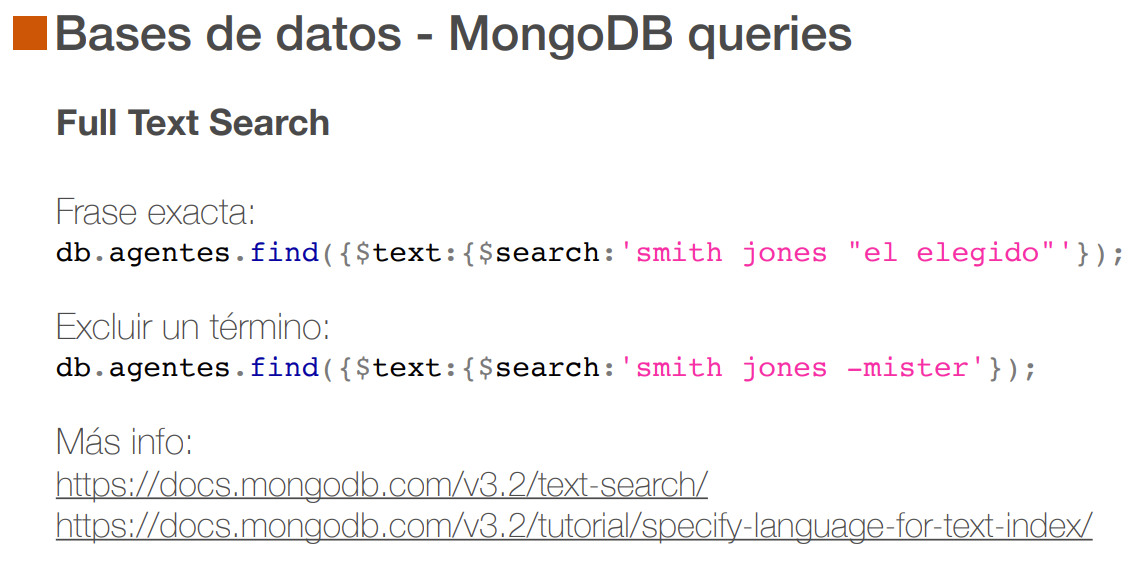
db.agentes.createIndex({name: 1}) → Crear un índice por nombre ascendente (para descendente poner -1) → Se puede crear dos indices sobre el mismo campo. Uno ascendente y uno descendente.

**Crear índices asegura la velocidad de todas las consultas y evita problemas de rendimiento en todo tipo de bases de datos.**

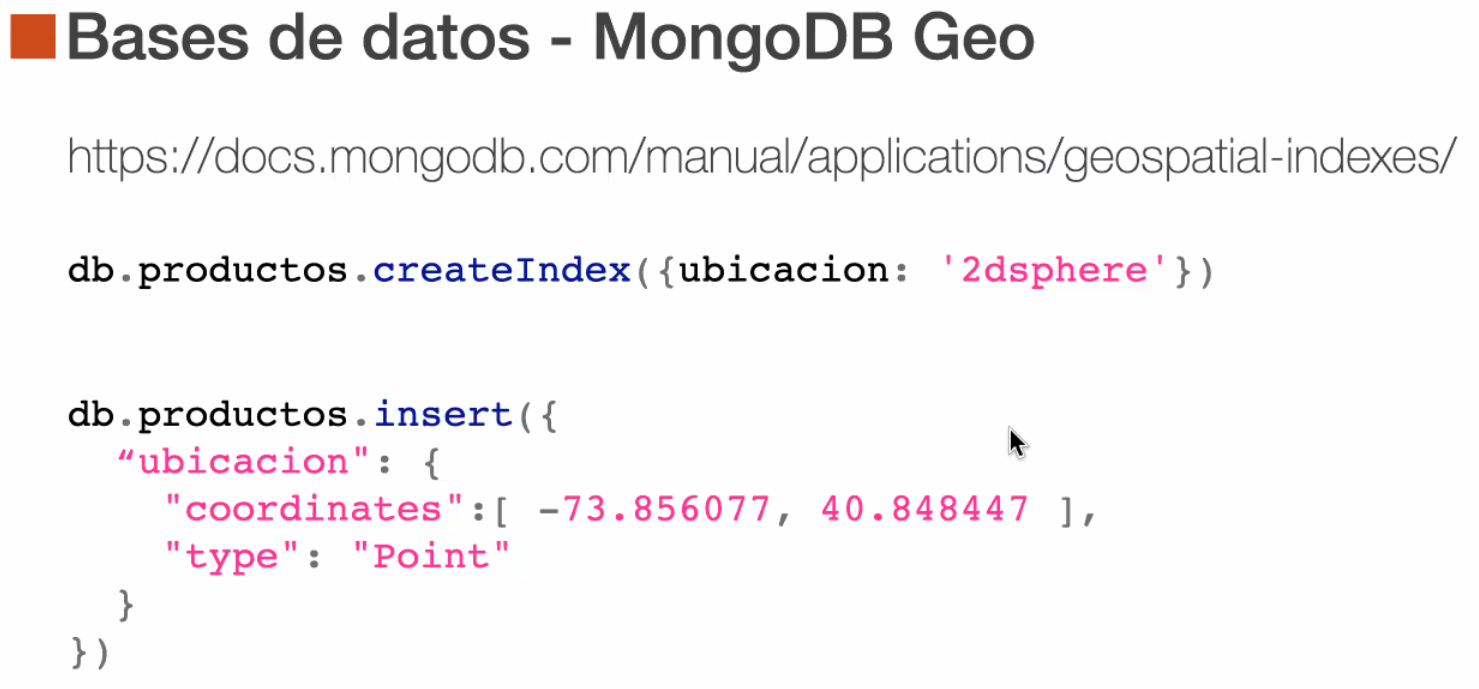
**FTS**

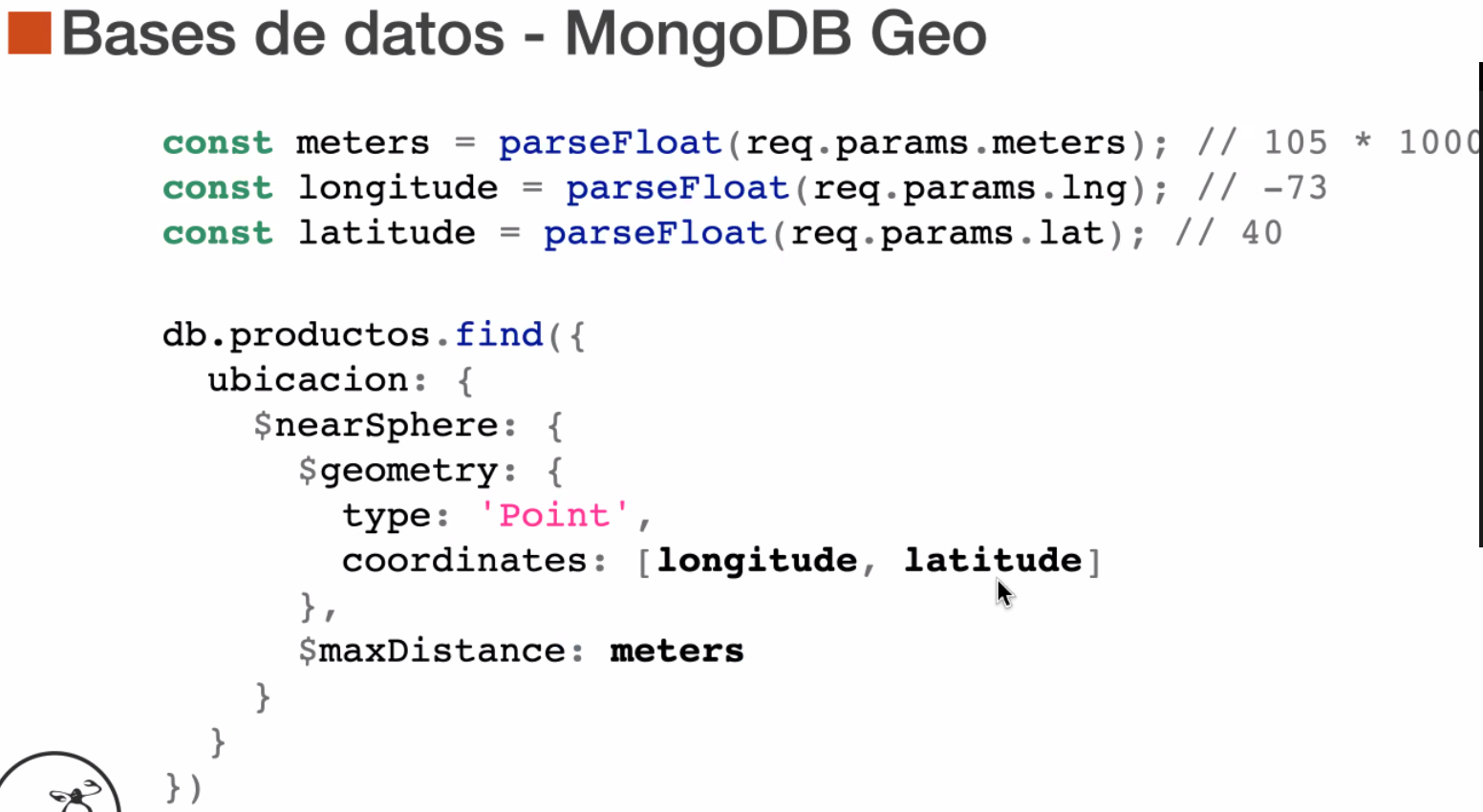
Para crear búsquedas de texto en los campos, por orden de relevancia.





Búsquedas geoespaciales





~