

# Manual mongoDB:



**Nombre: Pablo Jesús**

**Apellidos: Ramos Fernández**

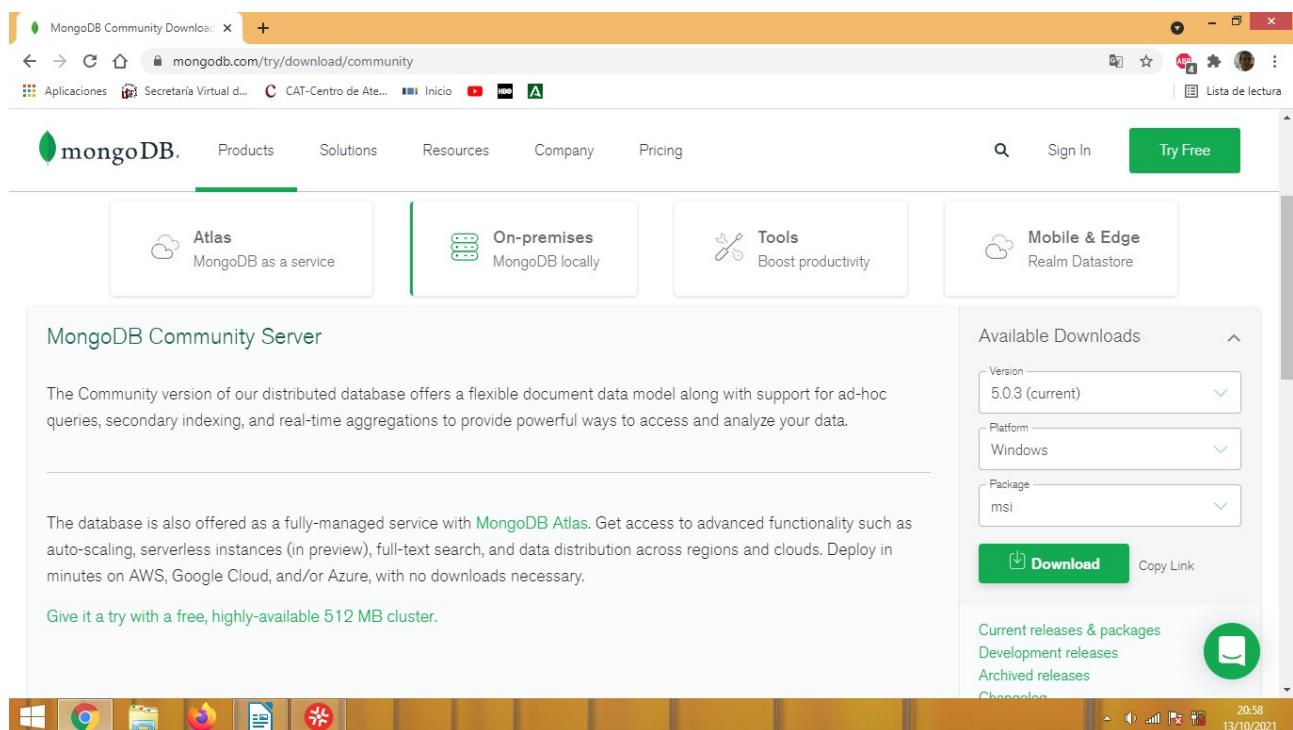
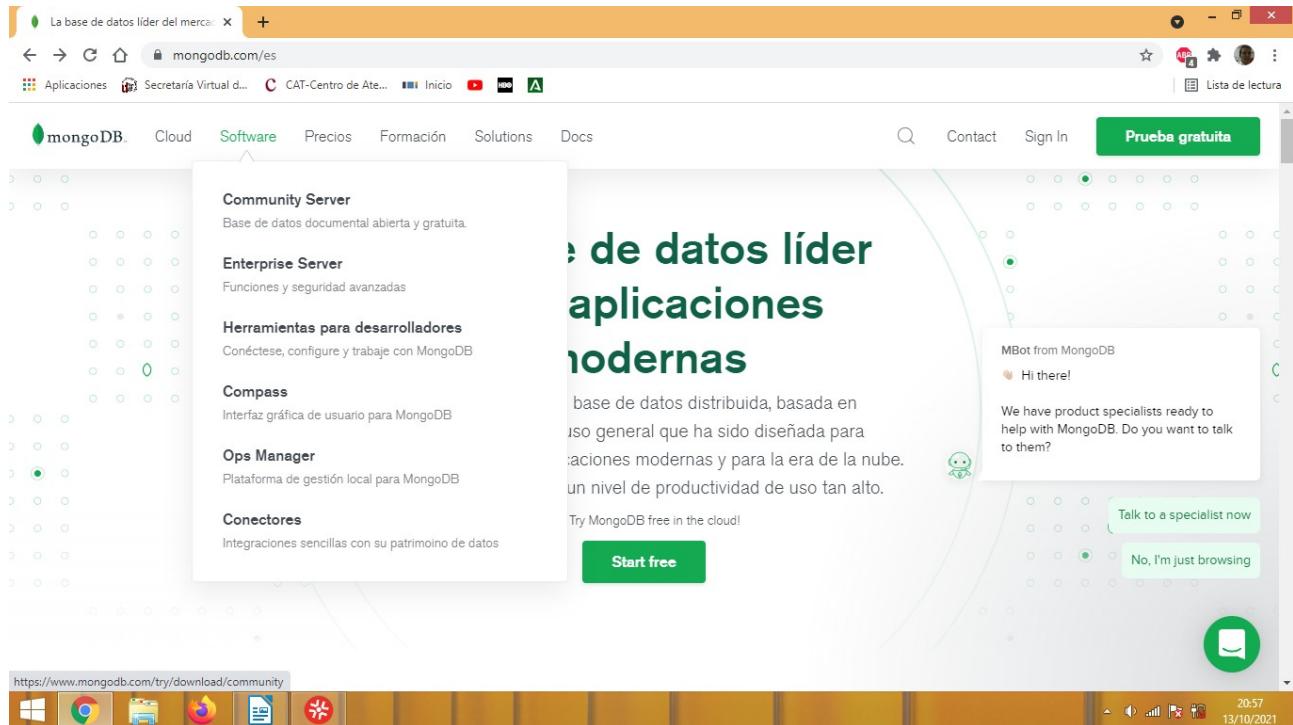
**Asignatura: Gestión de Base de Datos**

# Índice:

Descarga e instalación de MongoDB y MongoDB Compass.....	3
Ruta de la carpeta de MongoDB.....	6
Edición de las variables de entorno.....	7
Manual MongoDB.....	9
Servicios.....	9
Lista de comando en el terminal PowerShell.....	11
Familiarizándonos con el entorno de trabajo de Visual Studio Code.....	15
Aplicación de comando en PowerShell II.....	18

## **Descarga e instalación de MongoDB y MongoDB Compass:**

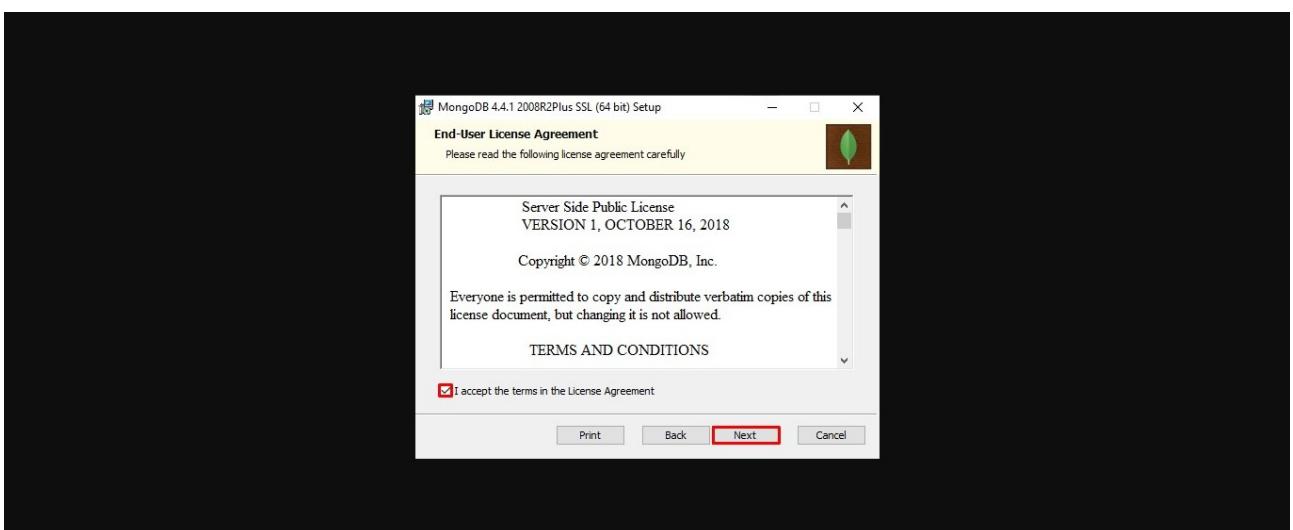
Para descargar MongoDB nos iremos a la página oficial de Mongo <http://www.mongodb.com/es> . Y para descargarnos la herramienta de trabajo seleccionaremos en la pestaña superior Software/Community Server y lo siguiente Download para Windows la versión 5.0.3.



Una vez descargado, vamos a proceder a su instalación.

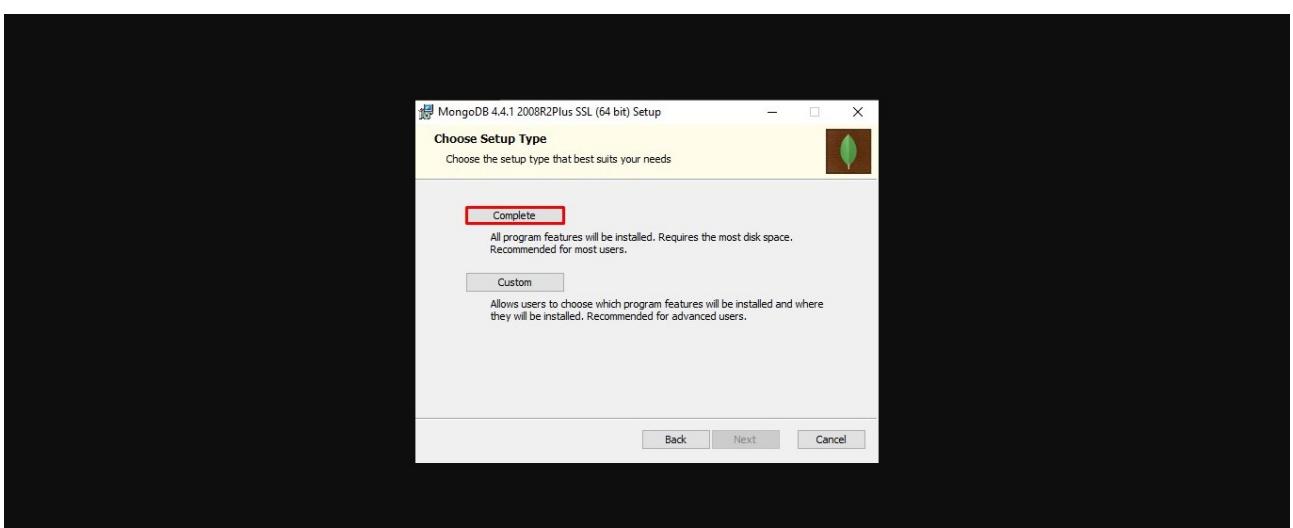


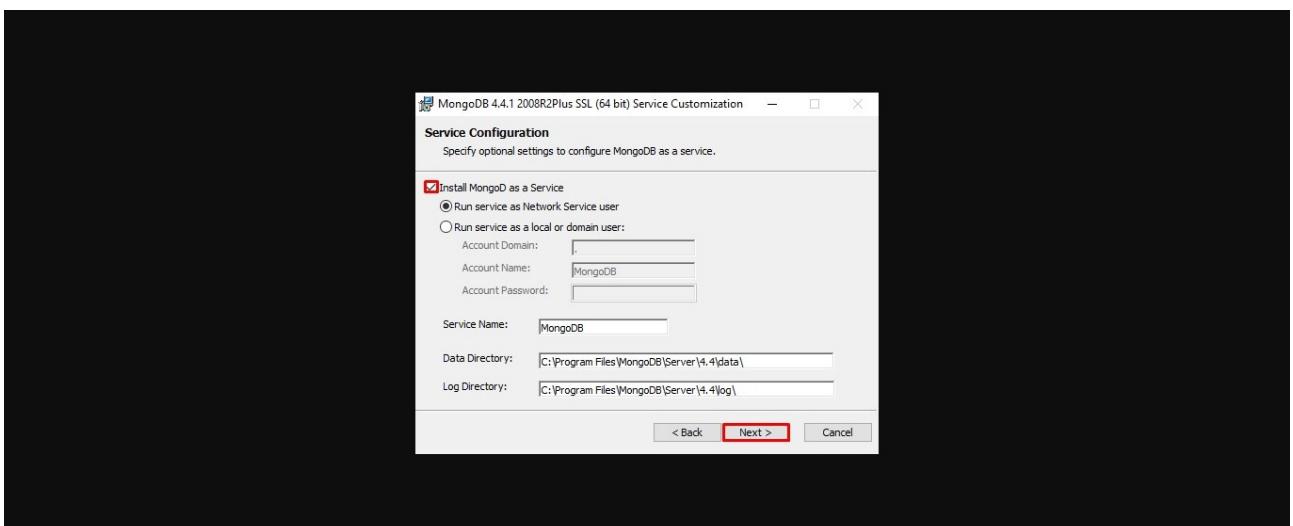
Iniciamos la instalación pulsando “Next”.



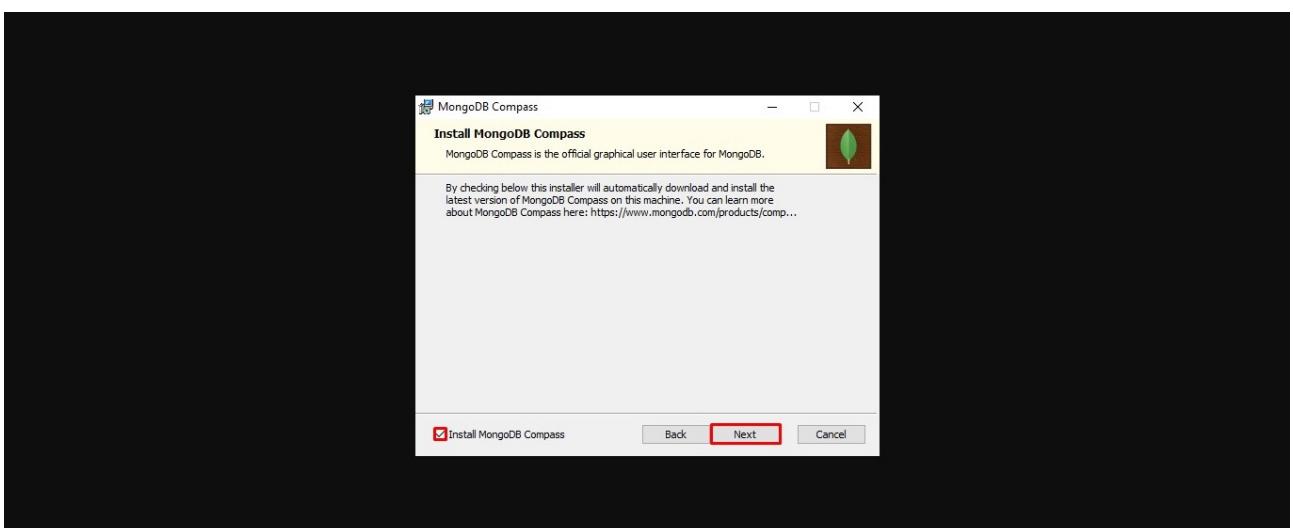
Aceptamos la licencia y “Next” de nuevo.

A continuación escogeremos la opción Completa.

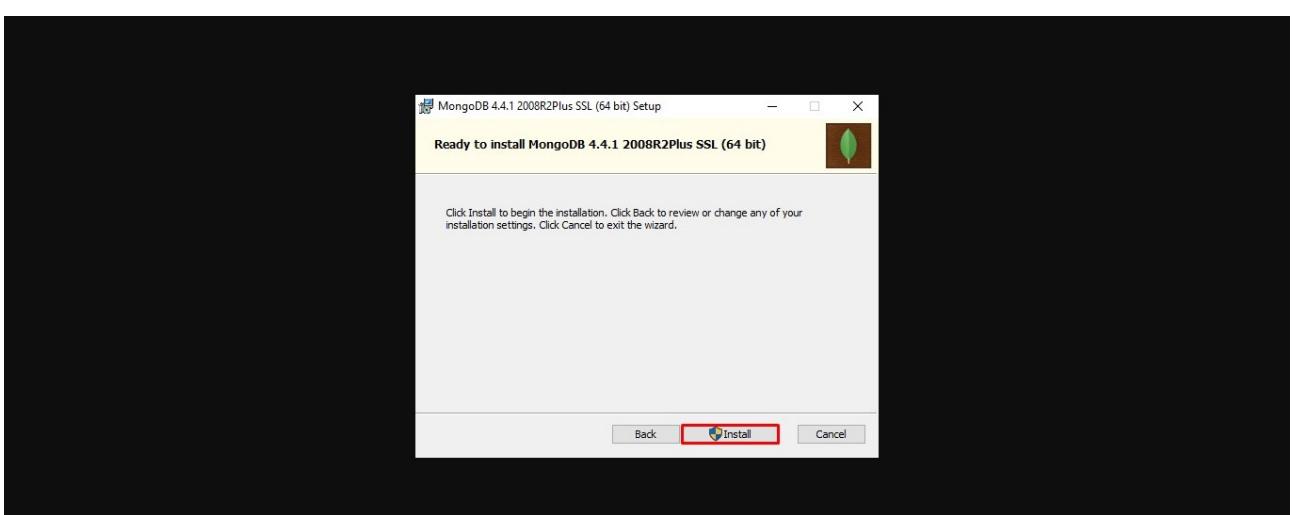




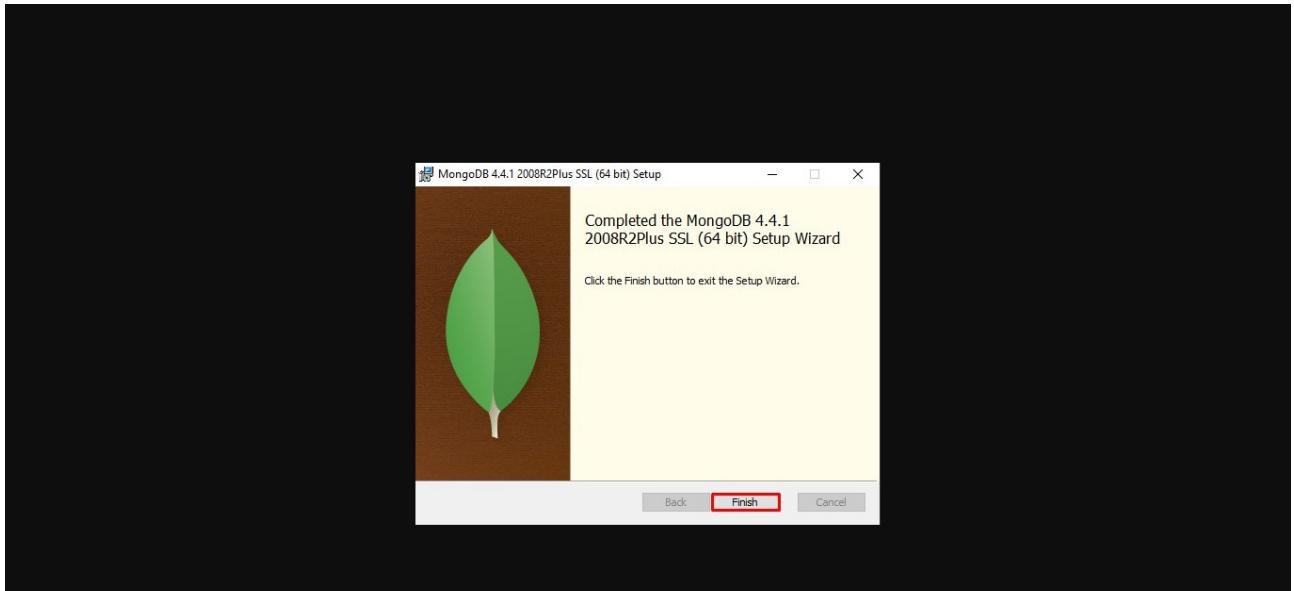
Dejaremos todo por defecto y si nos fijamos en la parte inferior nos especificará la ruta en que se va a guardar la carpeta de MongoDB.



Instalaremos la opción de MongoDB Compass, que es otra shell desde la que podremos trabajar, y pulsamos "Next".

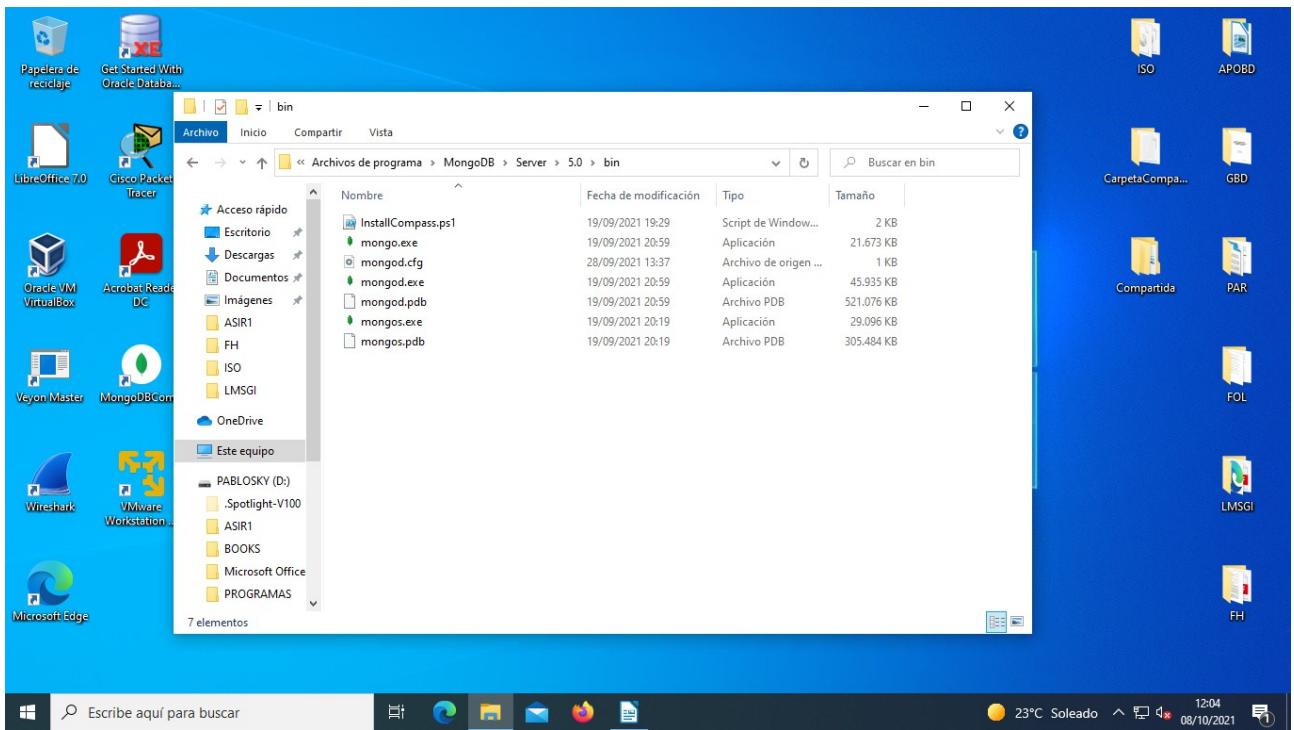


Y ya por último, instalamos el software.



Y finalizamos la instalación de MongoDB y MongoDB Compass.

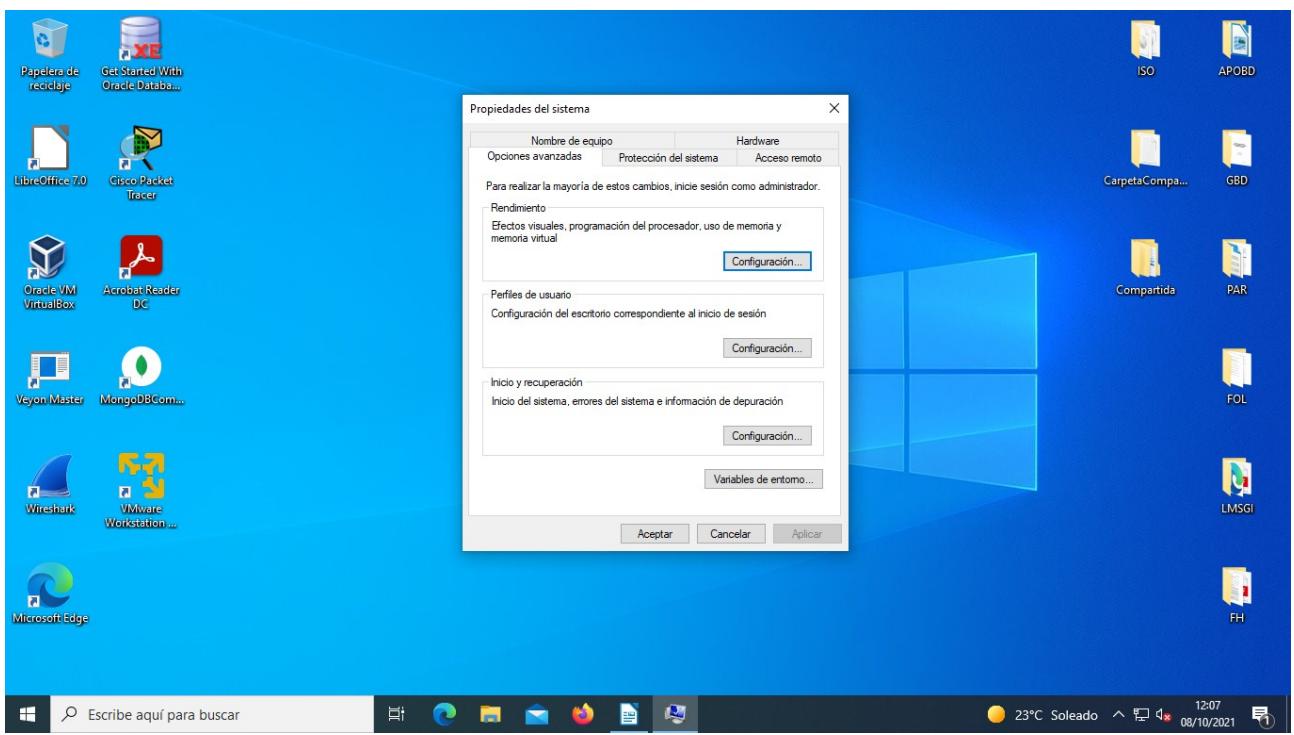
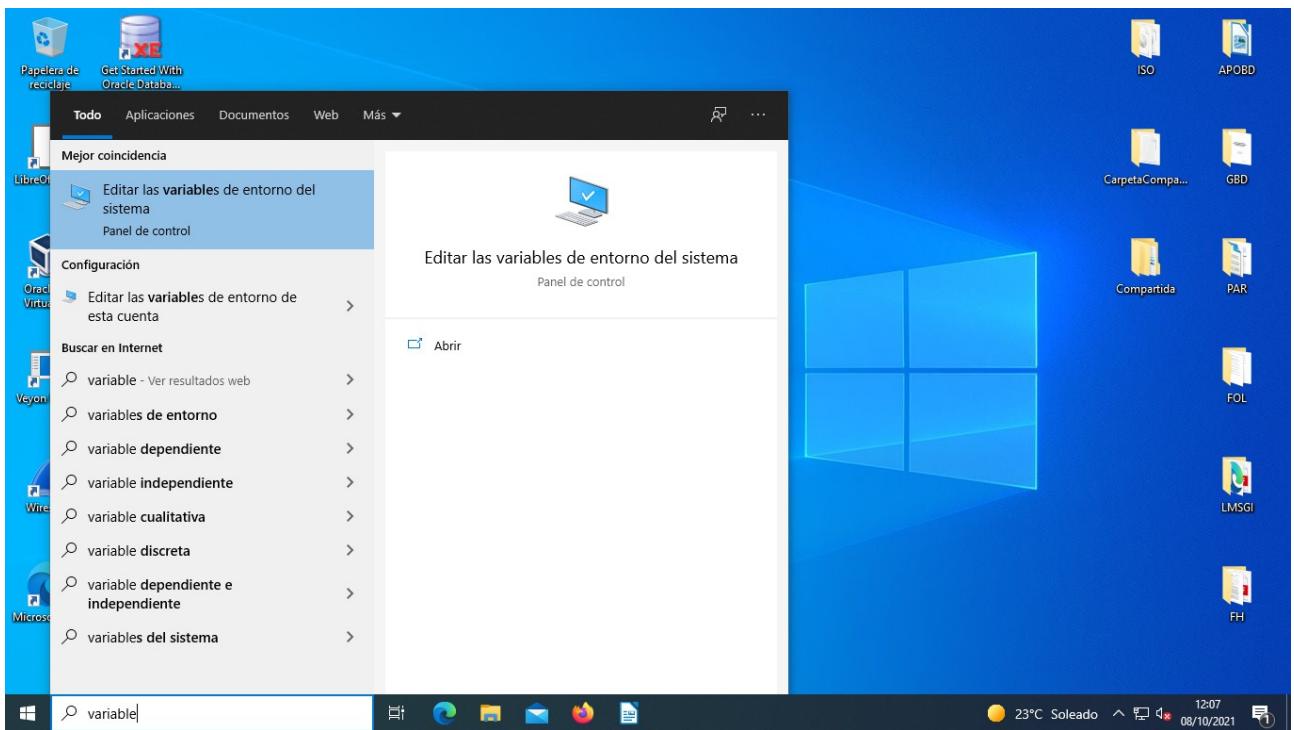
### **Ruta de la carpeta MongoDB:**



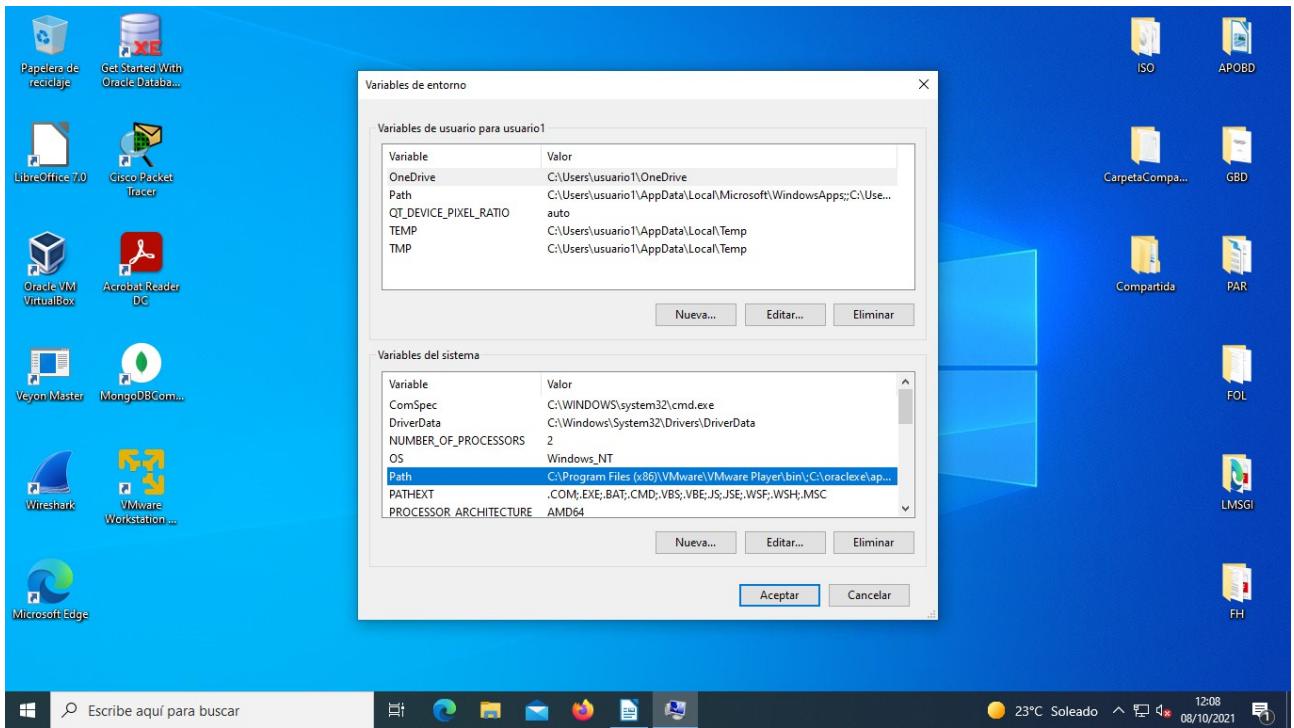
Nombre	Fecha de modificación	Tipo	Tamaño
InstallCompass.ps1	19/09/2021 19:29	Script de Windows...	2 KB
mongo.exe	19/09/2021 20:59	Aplicación	21.673 KB
mongod.cfg	28/09/2021 13:37	Archivo de origen ...	1 KB
mongod.exe	19/09/2021 20:59	Aplicación	45.935 KB
mongod.pdb	19/09/2021 20:59	Archivo PDB	521.076 KB
ASIR1	19/09/2021 20:19	Aplicación	29.096 KB
ISO	19/09/2021 20:19	Archivo PDB	305.484 KB
LMSGI			

Ruta en la que se encuentras nuestro software MongoDB, en mi caso sería (C:\Program Files\ MongoDB\Server\5.0\bin)

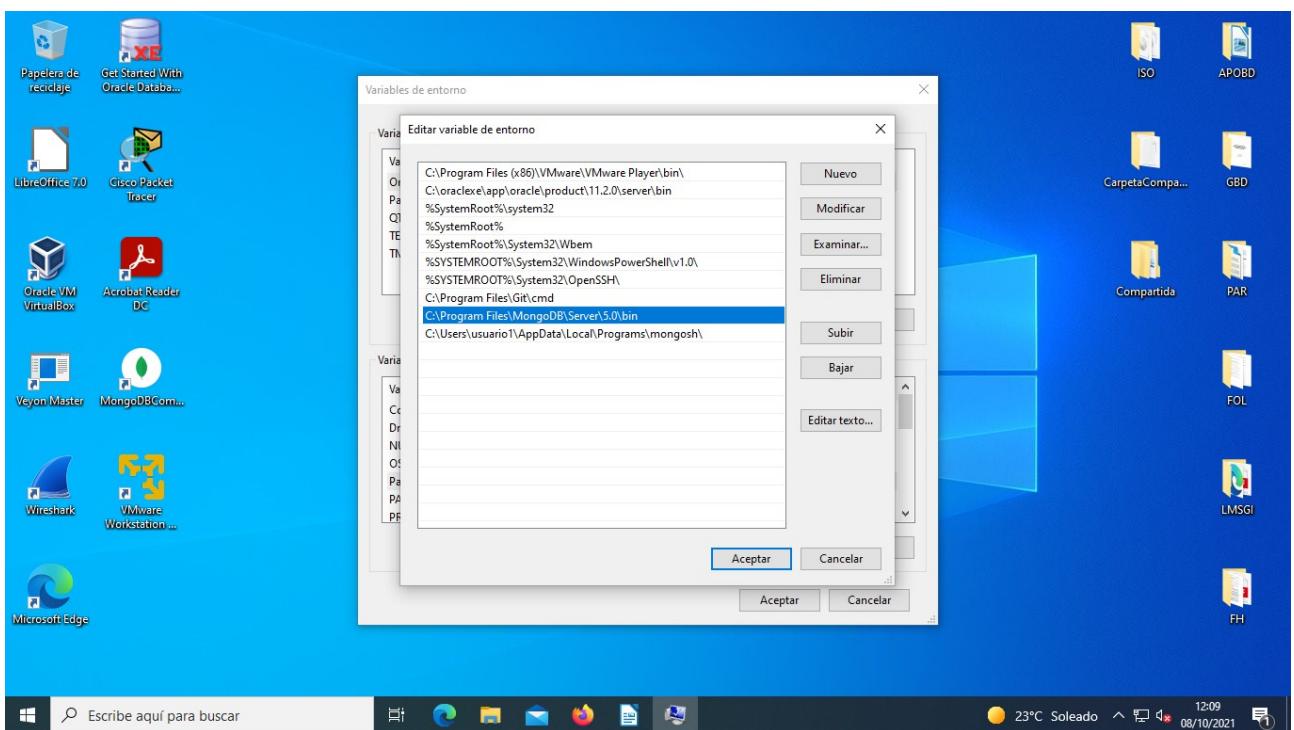
## Edición de las variables de entorno:



En esta pestaña, nos iremos a la opción de “Variables de entorno”. Donde editaremos el PATH, para añadir la nueva ruta en la que se encuentra MongoDB.



Ahora seleccionamos en la parte de variables de entorno “OneDrive” y en la parte de variable del sistema “PATH”.



Una vez dentro del PATH, en el caso de que hubiera una ruta antigua de MongoDB, la vamos a eliminar y añadiremos la nueva ruta opción “Nuevo” y la pegamos y guardaremos los cambios “Aceptar”.

## Aprendemos conceptos de MongoDB:

The screenshot shows a Microsoft Edge browser window displaying the MongoDB 5.0 Manual. The URL is https://docs.mongodb.com/manual/. The page title is "The MongoDB 5.0 Manual". On the left, there is a sidebar titled "MongoDB Manual" with a dropdown menu showing "Version 5.0". Below it are links to "Introduction", "Installation", "MongoDB Shell (mongosh)", "MongoDB CRUD Operations", "Aggregation", "Data Models", "Transactions", "Indexes", and "Security". At the bottom of the sidebar is a search bar with placeholder text "Escribe aquí para buscar". The main content area has a "NOTE" section stating "MongoDB 5.0 Released Jul 13, 2021". It mentions new features in MongoDB 5.0, the Release Notes for MongoDB 5.0, and the MongoDB Download Center. Below this, there is a welcome message about MongoDB 5.0 being a document database designed for ease of development and scaling, introducing key concepts, query language, operational and administrative considerations, and a comprehensive reference section. It also notes local and cloud-hosted deployment options, mentioning MongoDB Community as a free edition. The bottom right corner of the page has a "Give Feedback" button. The browser's taskbar at the bottom shows the Windows Start button, a search bar with "ser", and several pinned icons.

Dentro de la multitud de actividades que podemos realizar con el MongoDB, éste nos ofrece un completo manual en el que nos explica paso a paso con detalle cada acción que vamos a realizar tanto en Visual Studio como con el PowerShell.

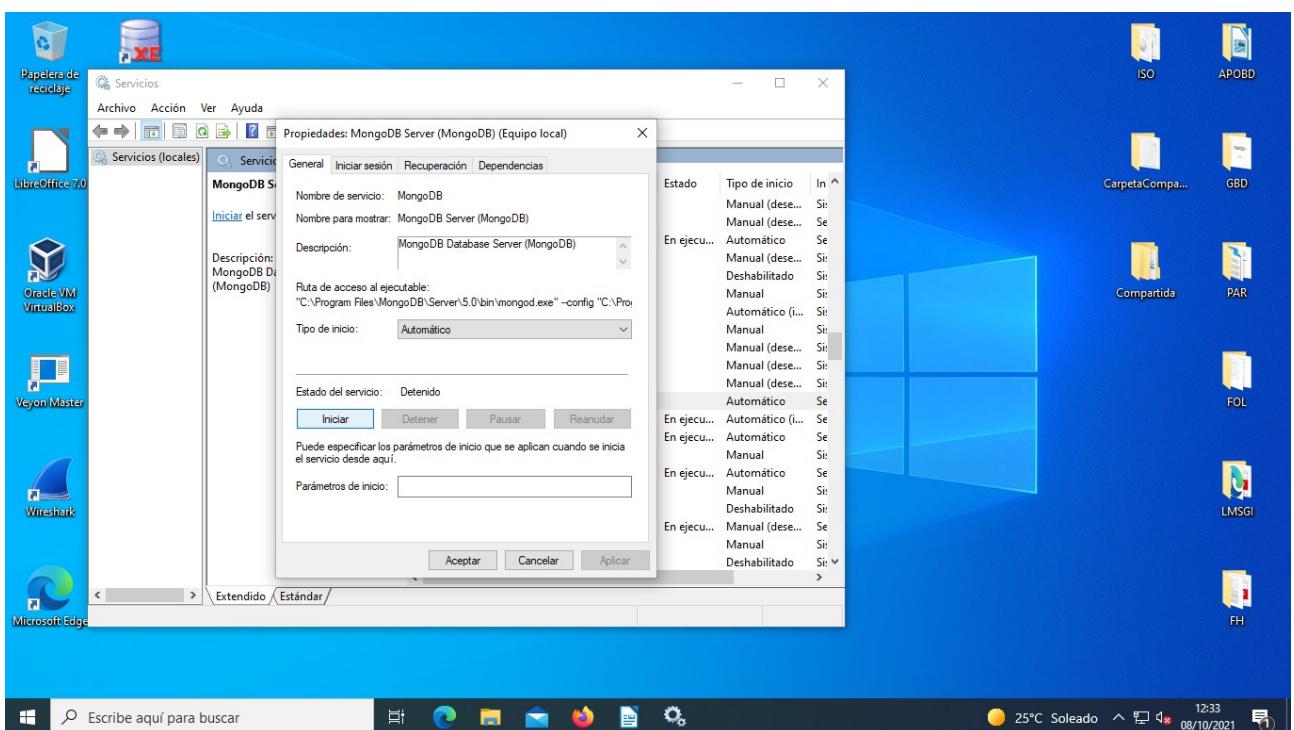
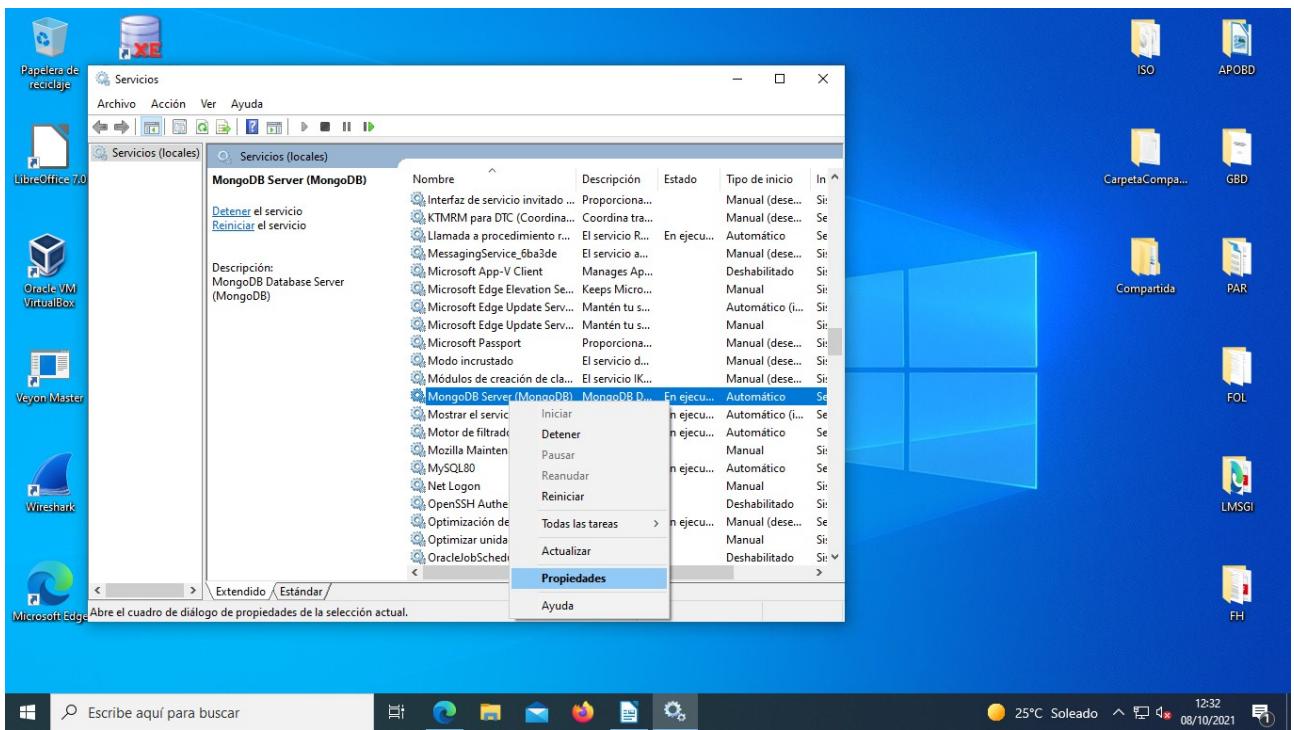
## Servicios:

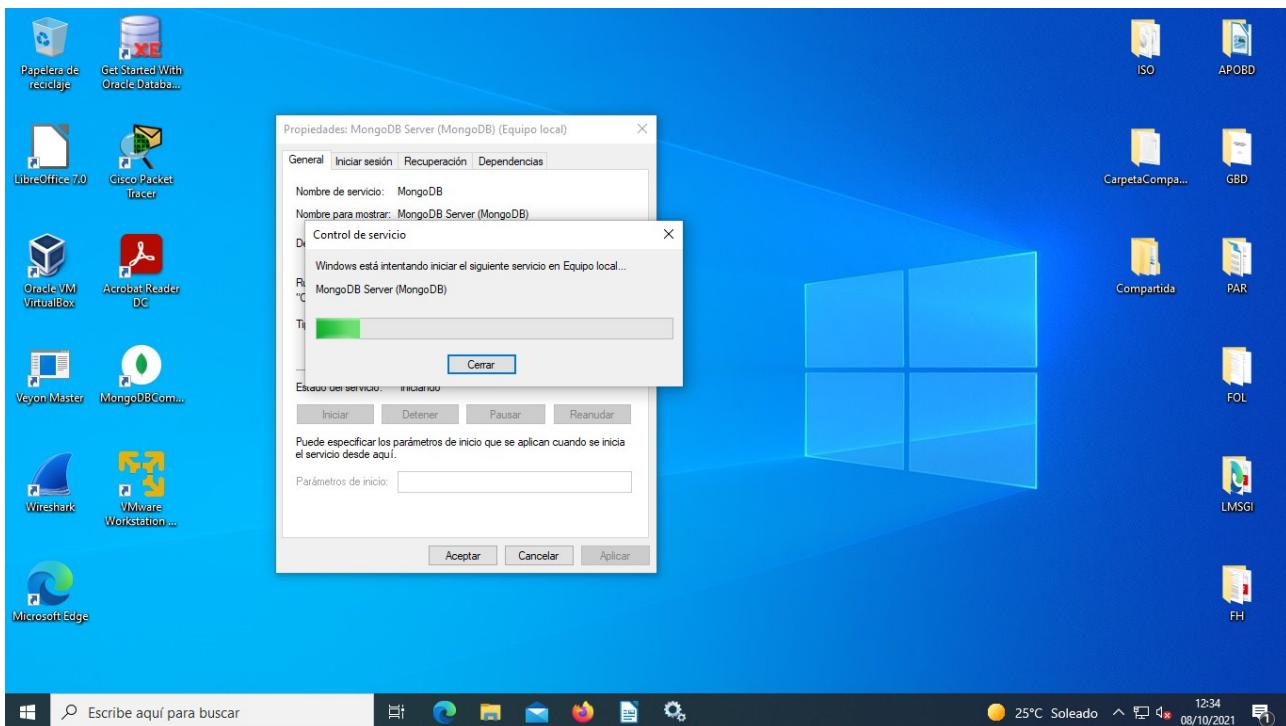
Un pequeño detalle a tener en cuenta es la habilitación de los servicios de MongoDB en el gestor de "Servicios". En el caso, de que estuviera detenido, hay que ponerlo en ejecución.

The screenshot shows the Windows Services application window. The title bar says "Servicios" and "Aplicación". The main pane displays a list of services with the following details:

- Icon: Gears
- Name: Servicios
- Type: Aplicación
- Status: Detenida
- Actions:
  - Abrir
  - Ejecutar como administrador
  - Abrir ubicación de archivo
  - Anclar a Inicio
  - Anclar a la barra de tareas

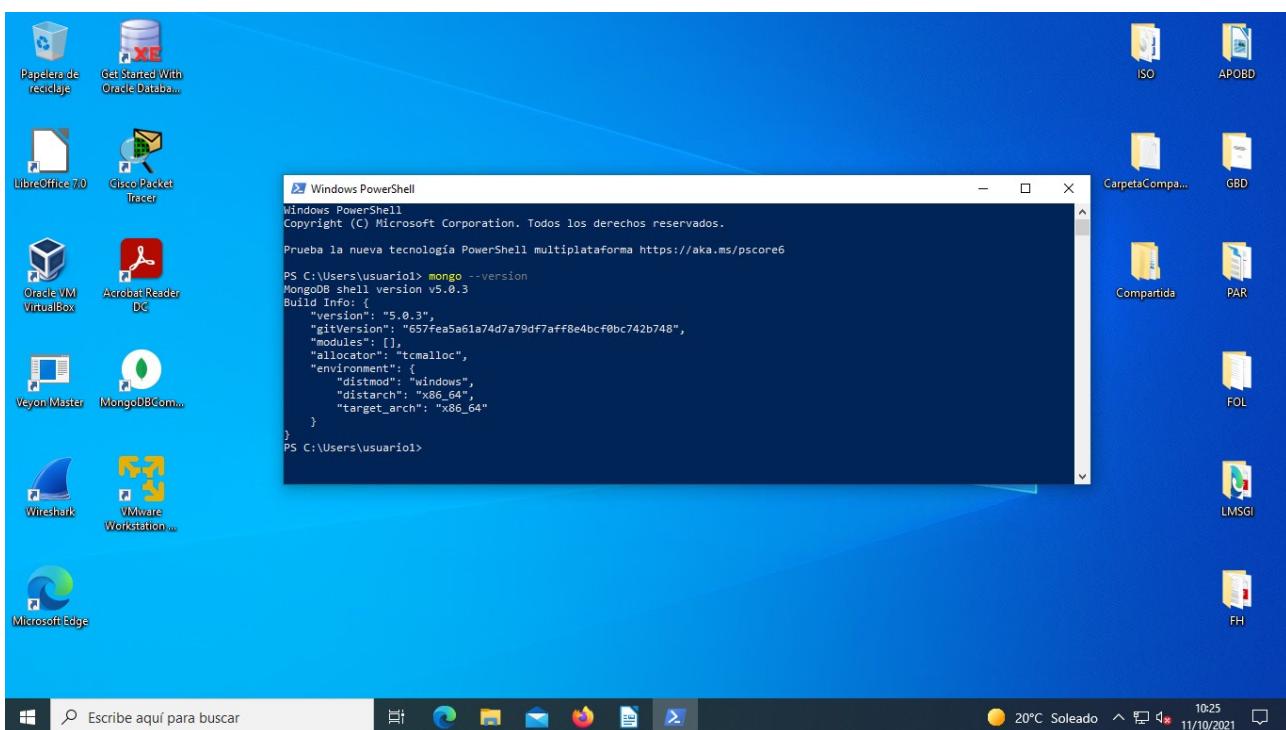
On the left, there is a sidebar with a search bar containing "ser". Other sections include "Mejor coincidencia" (Servicios), "Aplicaciones" (Servicios de componentes), and "Configuración (4+)". The desktop background is visible on the right, showing various file icons like ISO, APOBD, CarpetaCompa..., GBD, Compartida, PAR, FOL, LMSGI, and FH. The taskbar at the bottom shows the Windows Start button, a search bar with "ser", and several pinned icons.



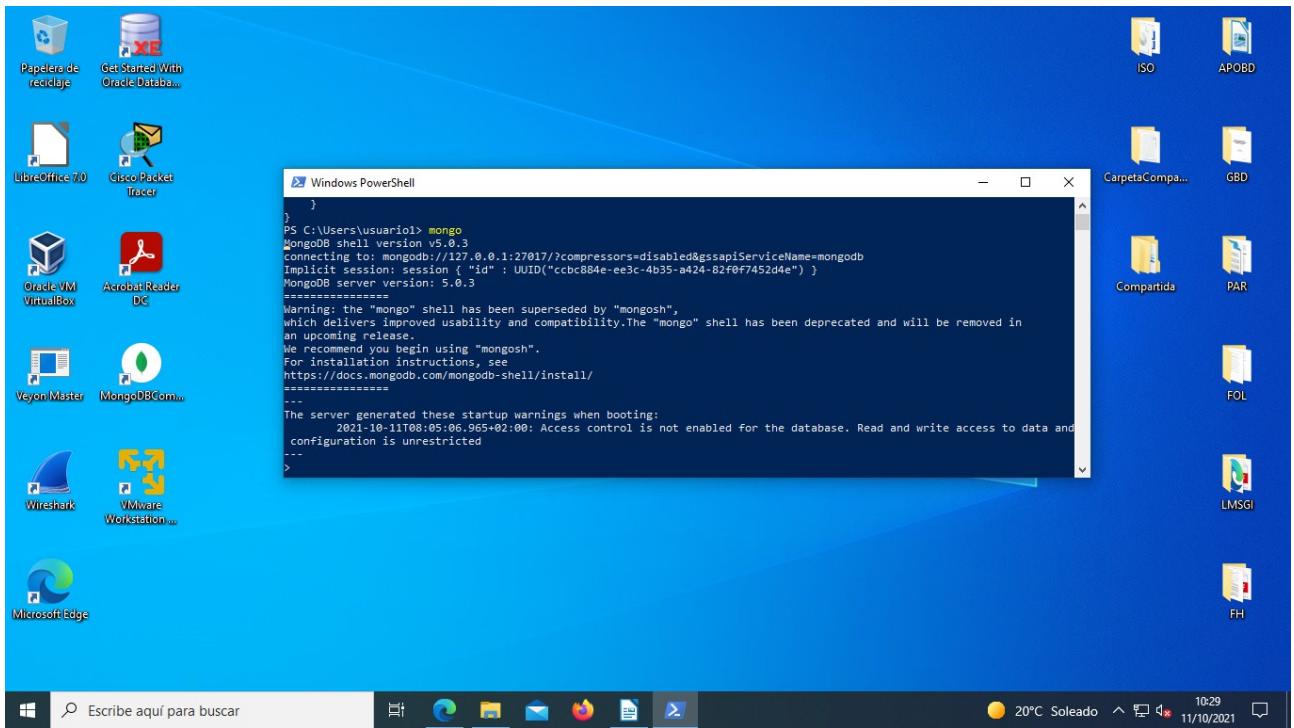


## Familiarizándonos con el entorno PowerShell

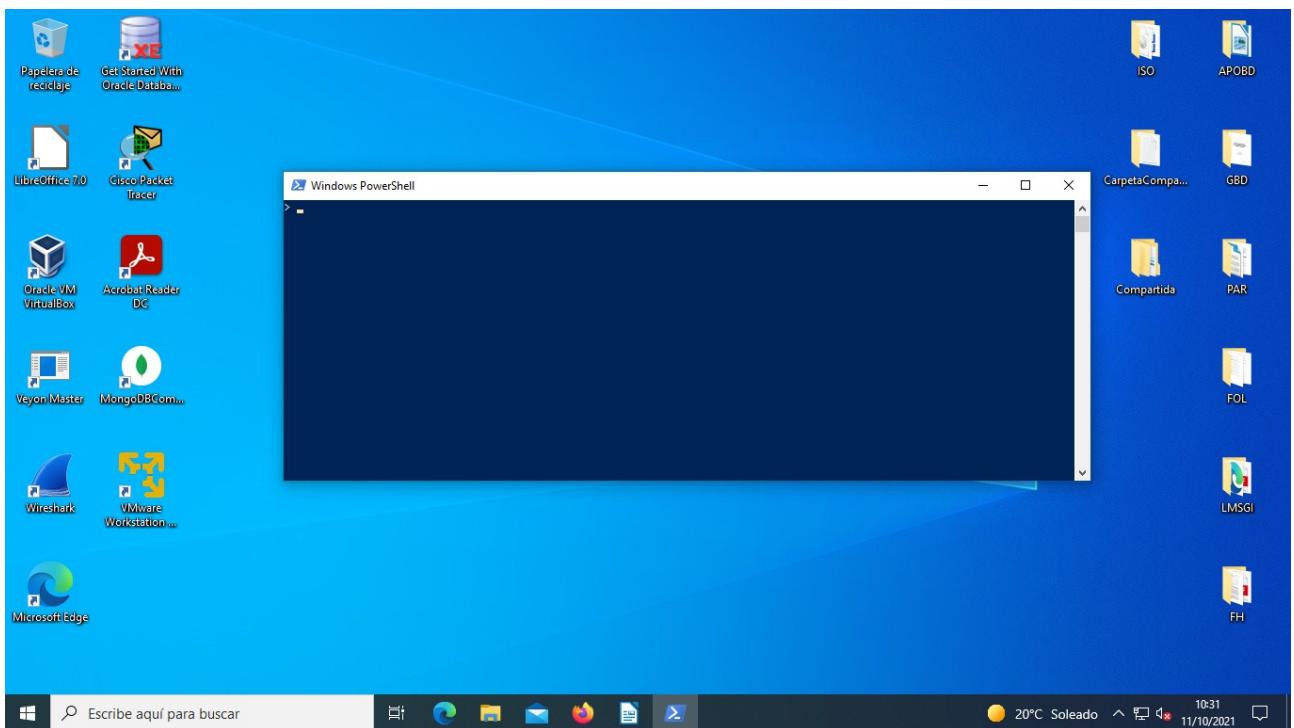
El primer comando sería mongo **--version**, el cual, nos muestra cual es la versión que se encuentra en nuestro equipo de MongoDB.



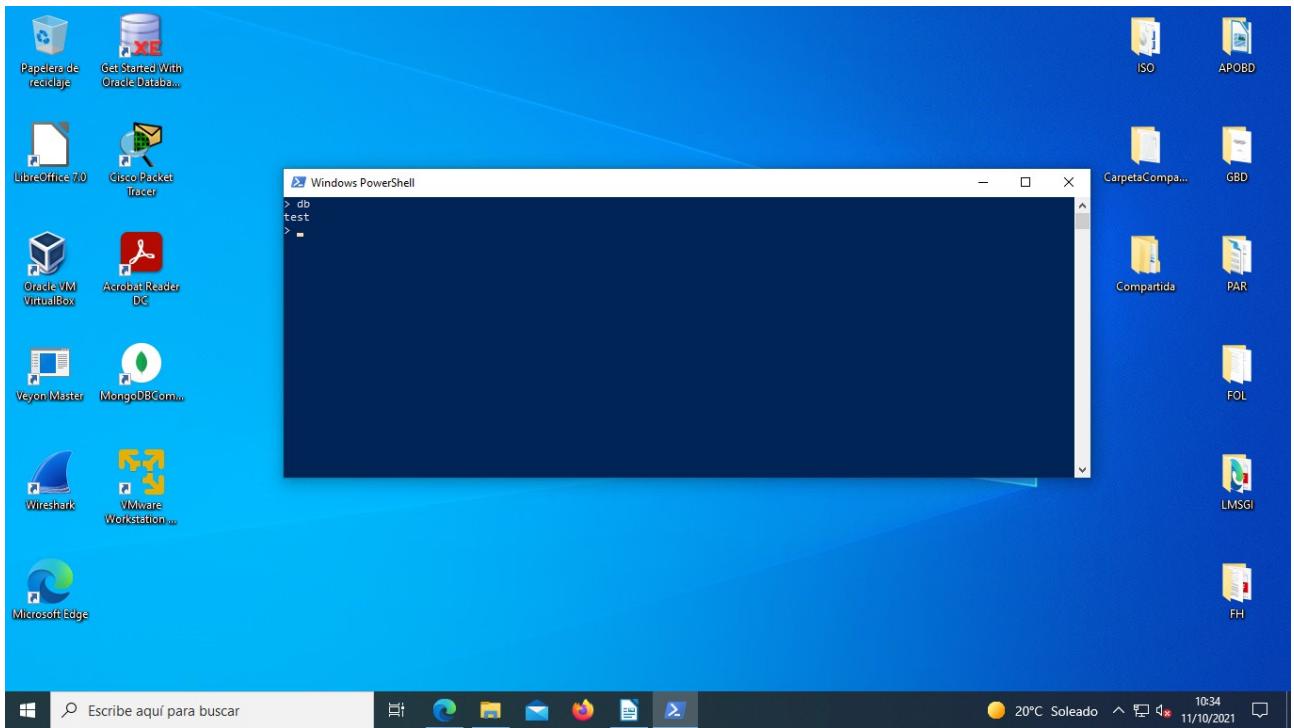
Ahora nos vamos a meter dentro del servidor de Mongo, través del comando "**mongo**".



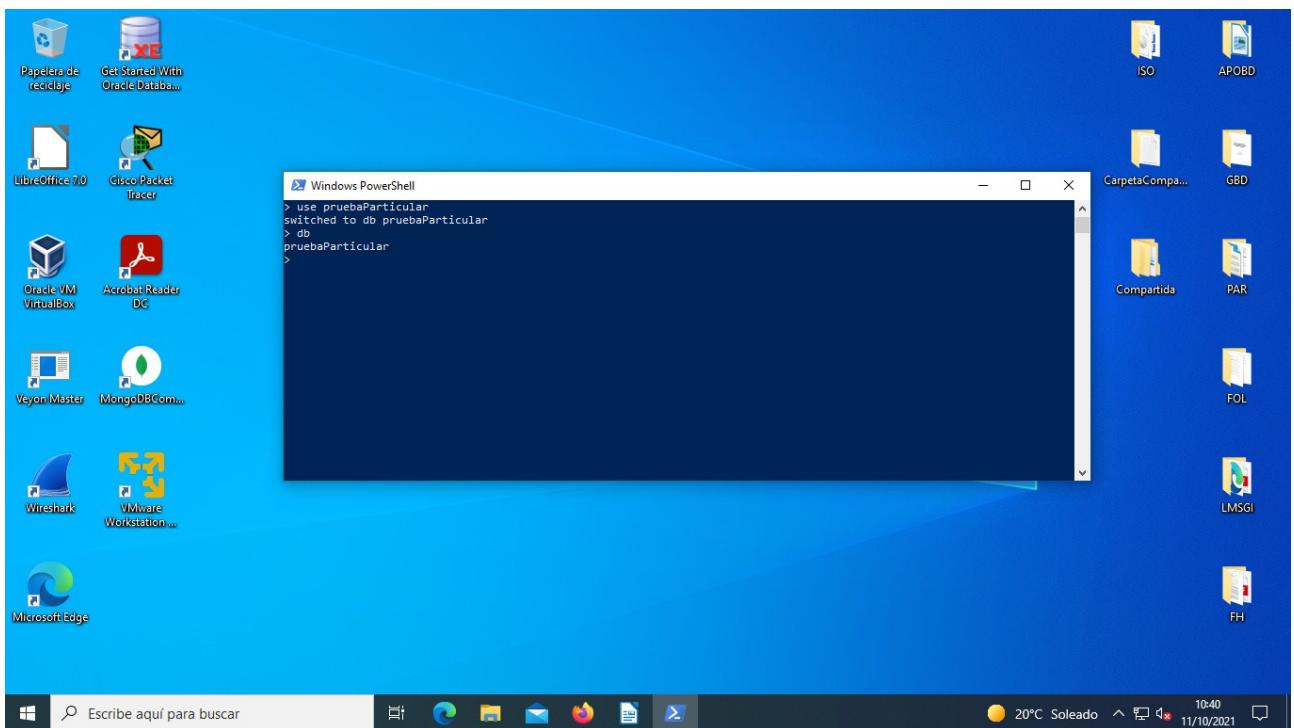
Con el comando **cls** borra todos los comando ejecutados y te limpia el terminal para una claridad mayor.



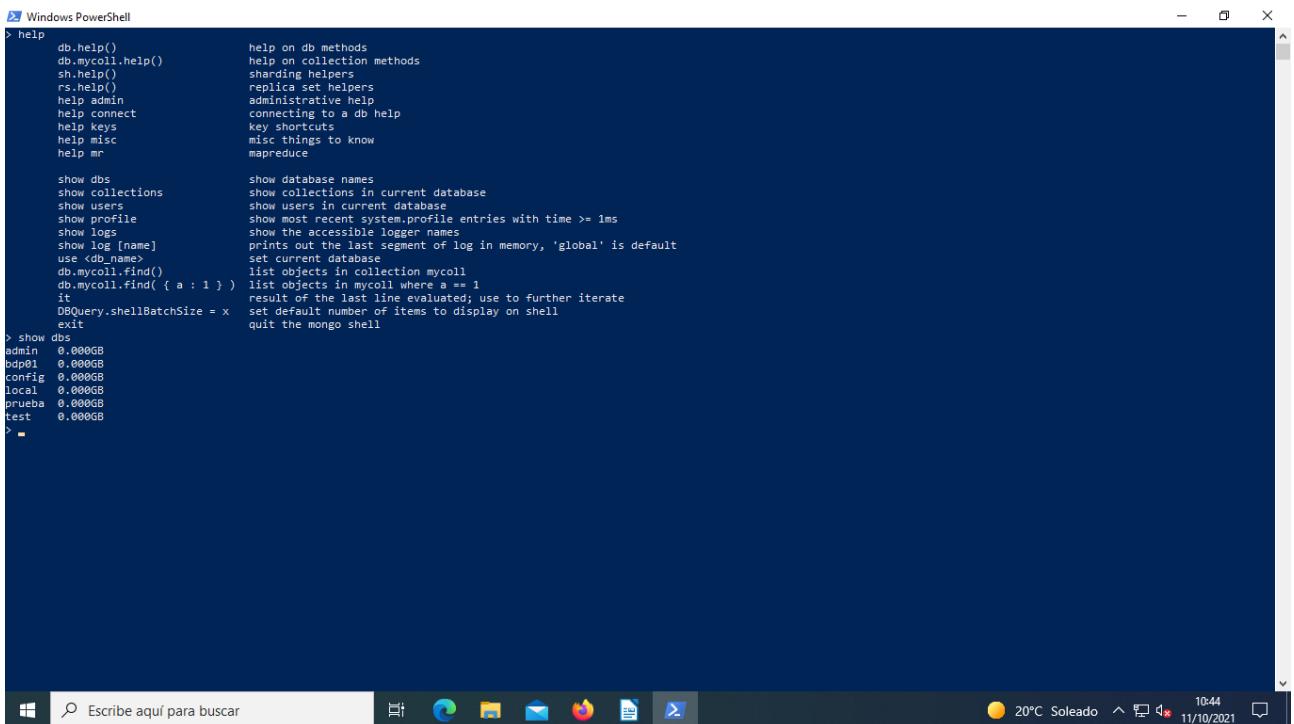
Con el siguiente comando, sabremos la base de datos en la que me encuentro conectado. Sería con “**db**”. En mi caso, la base de datos existente es test.



Acto seguido, voy a agregar una nueva base de datos con (use nombre base datos). En mi caso, “**use pruebaParticular**” y la muestro con **db**.



Con los siguientes dos comandos nos saldrá una ayuda de PowerShell y el otro nos muestra los nombre de las diferentes bases de datos. “**help**” (nos lanza una serie de ayudas) y “**show dbs**”, va a mostrar las diferentes bases de datos que tenemos.

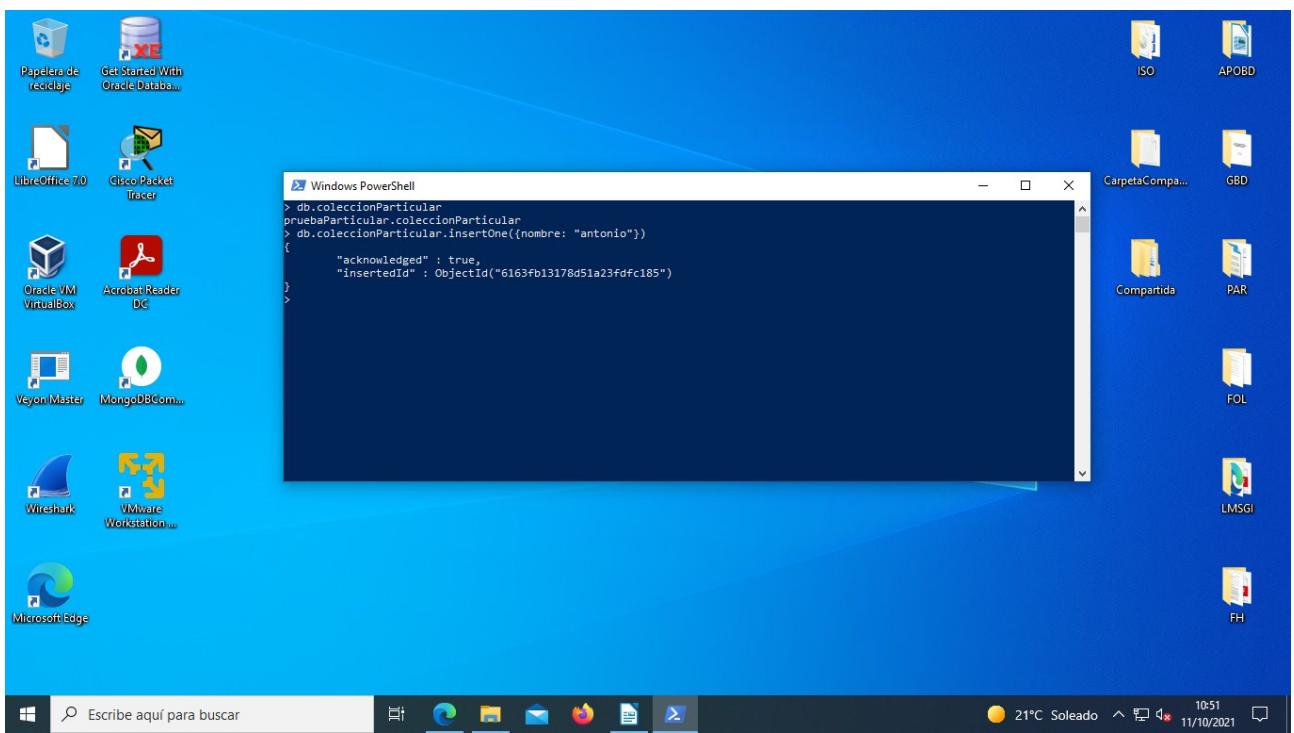


```
> Windows PowerShell
> help
db.help()           help on db methods
db.mycoll.help()   help on collection methods
sh.help()           sharding helpers
rs.help()           replica set helpers
help admin         administrative help
help connect      connecting to a db help
help keys          key shortcuts
help misc          misc things to know
help mr            mapreduce

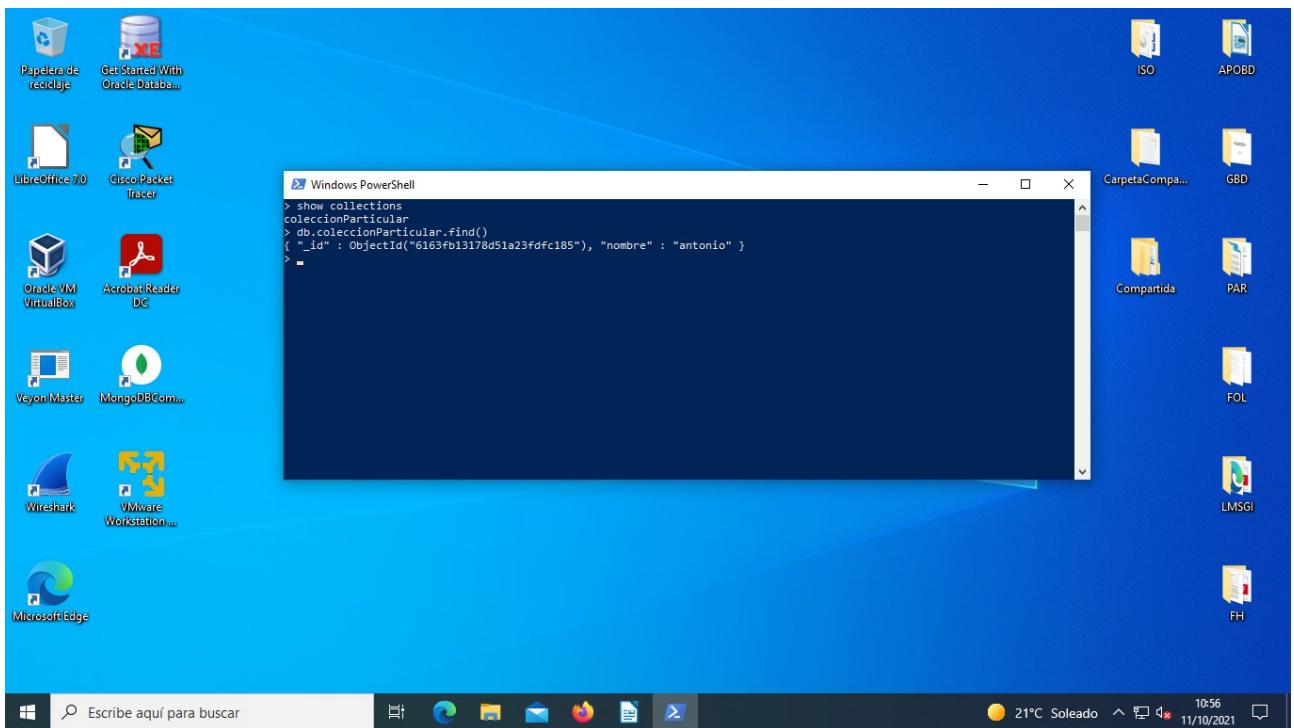
show dbs           show database names
show collections  show collections in current database
show users         show users in current database
show profile       show most recent system.profile entries with time >= 1ms
show logs          show the accessible logger names
show log [name]    prints out the last segment of log in memory, 'global' is default
use <db_name>     set current database
db.mycoll.find()   list objects in collection mycoll
db.mycoll.find( { a : 1 } )  list objects in mycoll where a == 1
it                result of the last line evaluated; use to further iterate
DBQuery.shellBatchSize = x  set default number of items to display on shell
exit              quit the mongo shell

> show dbs
admin  0.000GB
bdp01  0.000GB
config 0.000GB
local  0.000GB
prueba 0.000GB
test   0.000GB
>
```

Ahora vamos a ver como se inserta un documento en una colección con la línea de comando: **db.coleccionParticular.insertOne({nombre: "antonio"})**. Ahora bien, para añadir algo a la colección deberemos crearla. Para ello usaremos el comando **"db.coleccionParticular"**.

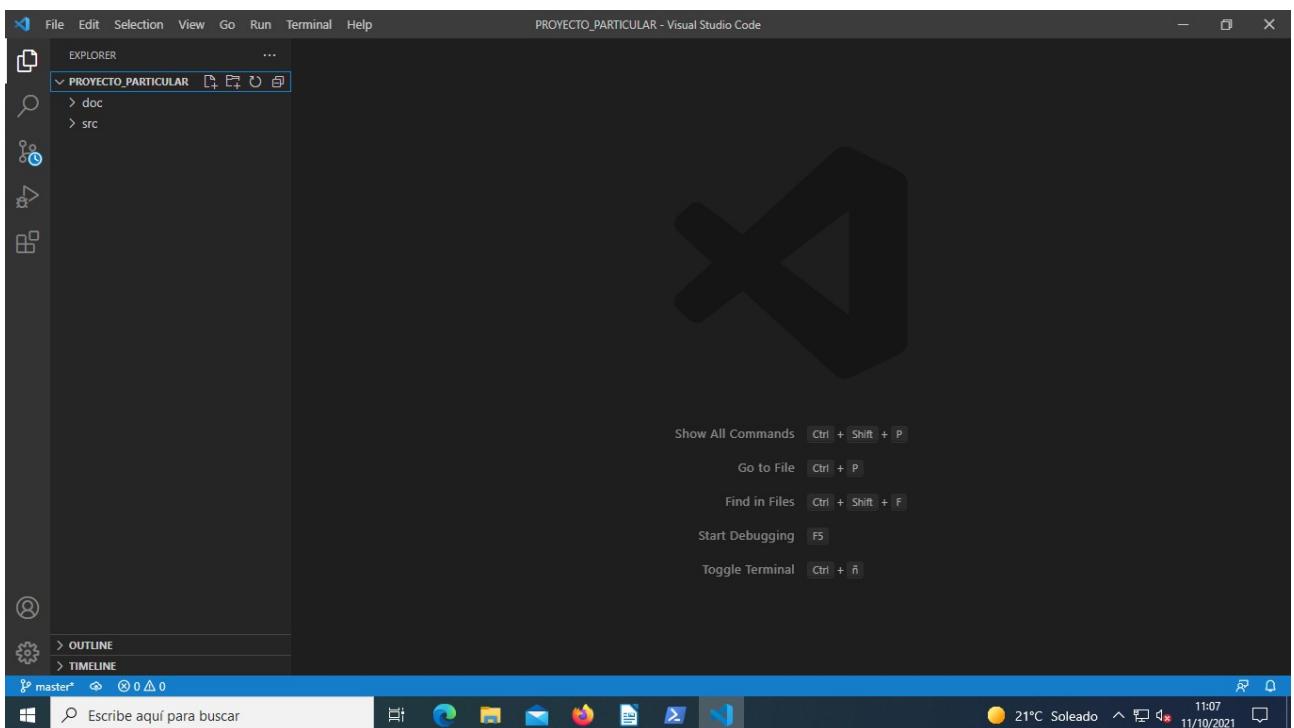


Ya insertado el documento, nos muestra un mensaje de validación como de que ha sido añadido dicho documento. Si hemos añadido algo, que menos, que nos lo muestre con el comando **show collections** y con este otro muestra los archivos dentro de una colección **db.coleccionParticular.find()**.

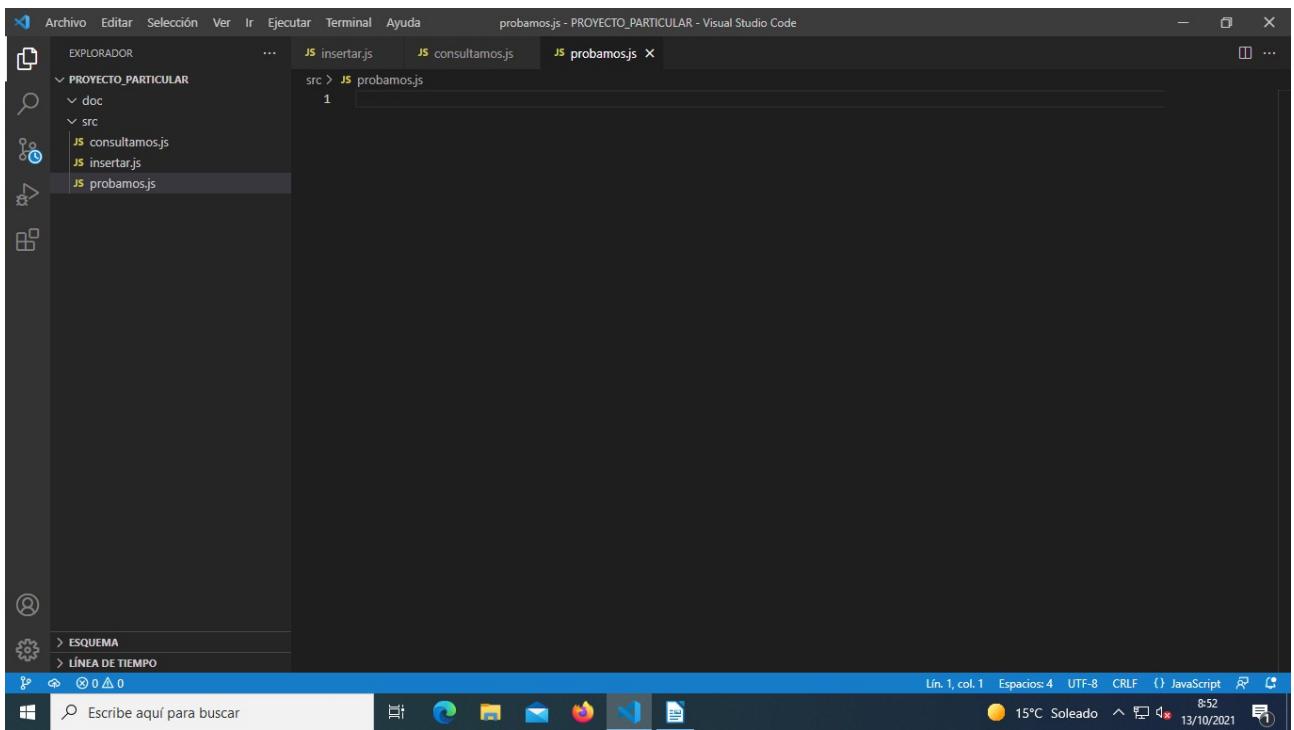


## Entorno Visual Studio Code:

Vamos a empezar con la creación de una nueva carpeta dentro de mi ruta de trabajo (C:\Users\usuario1\Documents\1ASIR\_A\PROYECTO PARTICULAR). Dentro de ella, crearemos dos carpetas más llamadas “src” y otra denominada “doc”. Para añadir este entorno al Visual Studio Code, tan sólo arrastramos la carpeta al programa y ya nos quedaría empezar a trabajar sobre él.



A continuación añadiremos tres ficheros a dentro de la carpeta src. Botón derecho sobre la carpeta y “Add file”, al que llamaremos “insertar.js”, “consultamos.js” y “probamos.js”.



En nuestro primer fichero llamado “insertar.js”, voy a usar una serie de funciones, como son:

**insertOne:** Añade un documento a una colección, en concreto a mi colecciónParticular.

**insertMany:** Añade una serie de documentos a una colección, la diferencia con el insertOne, es que esta función añade muchos documentos en un mismo instante.

**deleteMany:** Esta función elimina todo lo ejecutado anteriormente a ella.

The screenshot shows the Visual Studio Code interface with the title bar 'insertar.js - Sin título (área de trabajo) - Visual Studio Code'. The left sidebar shows a folder 'SIN TÍTULO (ÁREA DE TRABAJO)' containing 'PROYECTO PARTICULAR' with files 'insertar.js', 'probamos.js', and 'consultamos.js'. The main editor tab is 'insertar.js'. The code in 'insertar.js' is:

```
/* La función deleteMany elimina todos los documentos insertados anteriormente*/
db.coleccion01.deleteMany({})

/* La función insertOne inserta un nuevo documento a la colección*/
db.coleccionParticular.insertOne({nombre:"Pepe", apellidos:"López García", edad: 20, dni:"12345678-0"})
db.coleccionParticular.insertOne({nombre:"Lolo", apellidos:"Reyes Madrid", edad: 30, dni:"87654321-P"})
db.coleccionParticular.insertOne({nombre:"Tomás", apellidos:"Ramiro Leal", edad: 42, dni:"12349876-M"})
db.coleccionParticular.insertOne({nombre:"Juna", apellidos:"Corbacho Moya", edad: 18, dni:"09128734-L"})
db.coleccionParticular.insertOne({nombre:"Andrés", apellidos:"González Sánchez", edad: 43, dni:"56761209-H"})
db.coleccionParticular.insertOne({nombre:"Rodolfo", apellidos:"Román Macho", edad: 55, dni:"12345678-J"})

/* La función insertMany añade muchos valores a la colección*/
db.coleccionParticular.insertMany(
[
    [
        { _id: 1, item: { name: "ab", code: "123" }, qty: 15, tags: [ "A", "B", "C" ] },
        { _id: 2, item: { name: "cd", code: "123" }, qty: 20, tags: [ "B" ] },
        { _id: 3, item: { name: "ij", code: "456" }, qty: 25, tags: [ "A", "B" ] },
        { _id: 4, item: { name: "xy", code: "456" }, qty: 30, tags: [ "B", "A" ] },
        { _id: 5, item: { name: "mn", code: "000" }, qty: 20, tags: [ [ "A", "B" ], "C" ] }
    ]
])

/* Compara los valores de ITEM que sean igual a "123"
*/
db.coleccionParticular.find({{"item.code":{$eq:"123"}}})
```

```

1  /*Todos los documentos con todos sus campos*/
2  db.colecciónParticular.find()
3  {
4      [
5          { "_id" : ObjectId("6156cea90a77a2d853785f76") , "nombre" : "Pepe" , "edad" : 20 } ,
6          { "_id" : ObjectId("6156cea90a77a2d853785f77") , "nombre" : "Lolo" , "edad" : 40 } ,
7          { "_id" : ObjectId("6156cea90a77a2d853785f78") , "nombre" : "Tomás" , "edad" : 37 } ,
8          { "_id" : ObjectId("6156cea90a77a2d853785f79") , "nombre" : "Juna" , "edad" : 21 } ,
9          { "_id" : ObjectId("6156cea90a77a2d853785f7a") , "nombre" : "Andrés" , "edad" : 15 } ,
10         { "_id" : ObjectId("6156cea90a77a2d853785f7b") , "nombre" : "Rodolfo" , "edad" : 63 }
11     ]
12 }
13
14 /*Todos los documentos con edad igual a 20*/
15 db.colecciónParticular.find({edad: 20})
16
17 /*Todos los documentos de una colección*/
18 db.colecciónParticular.find({})
19
20 /*Encuentra a traves del campo status los valores iguales a D*/
21 db.colecciónParticular.find({status: "D"})
22
23 /*Encuentra a traves del campo status los valores iguales a D, pero usando el método equals*/
24 db.colecciónParticular.find({ status: { $eq: "D" } })

```

En nuestro fichero “consultamos.js” vamos a usar otra serie de funciones que os explicaré a continuación:

**find():** Muestra todos los documentos que están en nuestra base de datos colecciónParticular.

**find(edad: 20):** Muestra a aquellas personas que tienen una edad igual a 20 años. Para ello le pasamos el filtro “edad: 20”. Este tipo de filtro se puede para cualquier campo del documento, como por ejemplo he hecho también con (status: “D”). Que mostrará aquellos documentos que presenten como campo status = D.

**find({status:{\$eq: "D"}}):** Esta es otra forma de comparar, la diferencia es que usamos el operador equals.

```

22 [
23     { item: "journal" , qty: 25 , size: { h: 14 , w: 21 , uom: "cm" } , status: "A" } ,
24     { item: "notebook" , qty: 50 , size: { h: 8.5 , w: 11 , uom: "in" } , status: "A" } ,
25     { item: "paper" , qty: 100 , size: { h: 8.5 , w: 11 , uom: "in" } , status: "D" } ,
26     { item: "planner" , qty: 75 , size: { h: 22.85 , w: 30 , uom: "cm" } , status: "D" } ,
27     { item: "postcard" , qty: 45 , size: { h: 10 , w: 15.25 , uom: "cm" } , status: "A" }
28 ]
29 )
30
31 db.colecciónParticular.deleteMany({})
32
33 db.colecciónParticular.insertMany(
34 (
35     [
36         { _id: 1 , item: { name: "ab" , code: "123" } , qty: 15 , tags: [ "A" , "B" , "C" ] } ,
37         { _id: 2 , item: { name: "cd" , code: "123" } , qty: 20 , tags: [ "B" ] } ,
38         { _id: 3 , item: { name: "ij" , code: "456" } , qty: 25 , tags: [ "A" , "B" ] } ,
39         { _id: 4 , item: { name: "xy" , code: "456" } , qty: 30 , tags: [ "B" , "A" ] } ,
40         { _id: 5 , item: { name: "mn" , code: "000" } , qty: 20 , tags: [ [ "A" , "B" ] , "C" ] }
41     ]
42 )

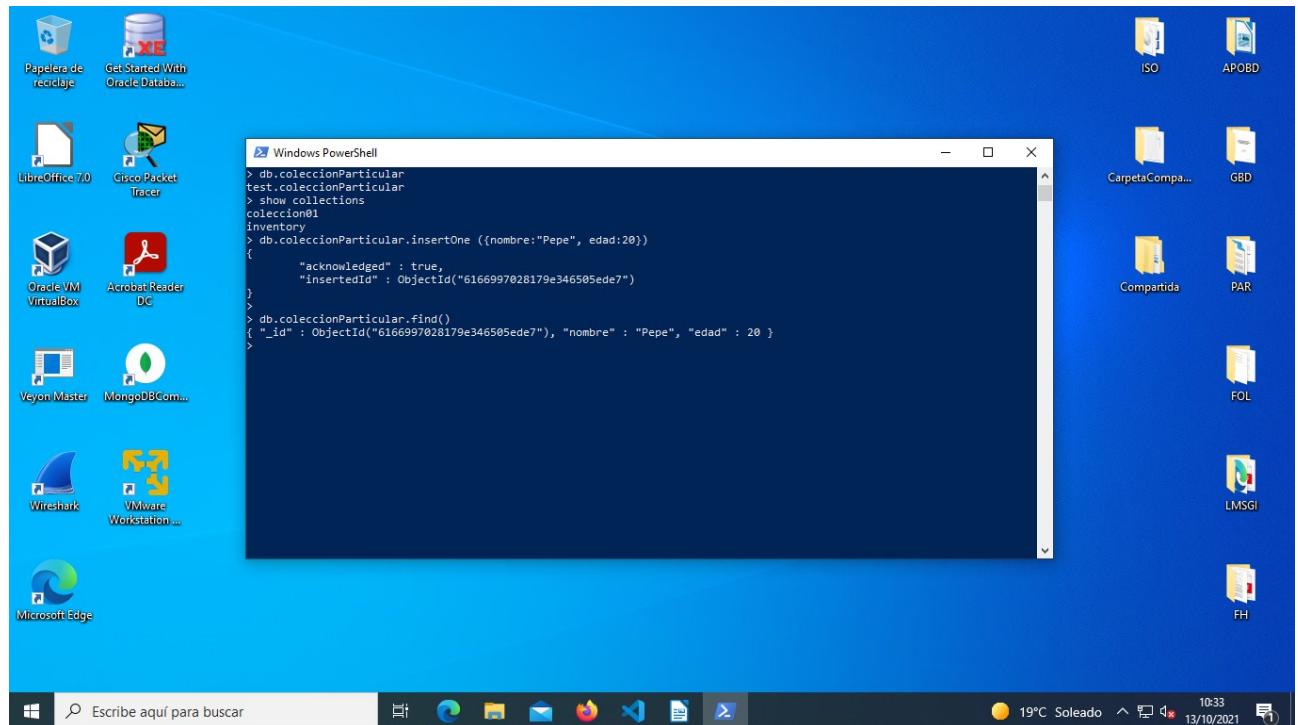
```

En mi último fichero llamado “probamos.js”, como bien dice, vamos a realizar todo tipo de pruebas, como insertando varios documentos a la vez (**insertMany**) como eliminando o comentando aquellas pruebas que ya no nos hagan falta (**deleteMany** y /\*aquí se añade un comentario\*/).

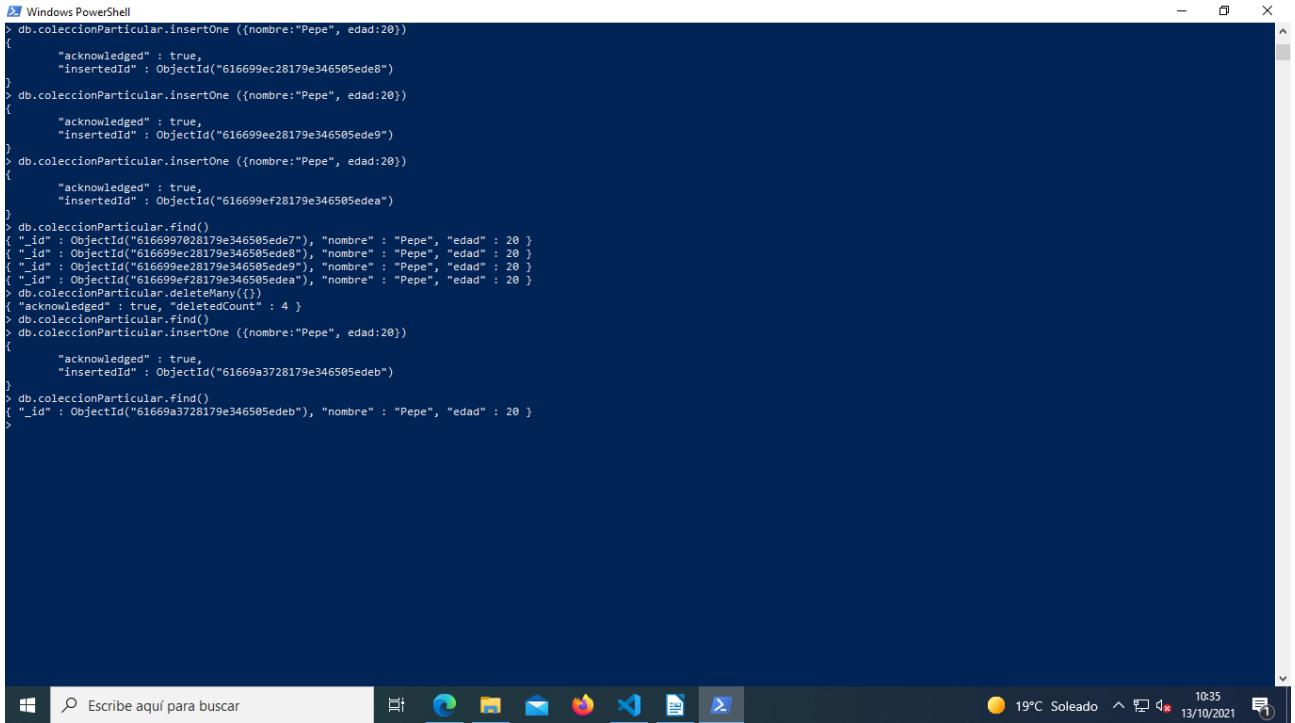
Ahora nos pasaremos a la ventana de PowerShell de nuevo para probar eso mismo comandos que hemos añadido en el Visual Studio Code.

Empezaremos creando una nueva colección llamada en mi caso “colecciónParticular” con el comando **db.colecciónParticular**, acto seguido aplico el comando **show collections**, al no tener nada dentro de dicha colección, por eso tan solo nos muestra la colección inventory, para ello añadiremos un documento con el comando **db.colecciónParticular.insertOne ({campo:"valor"})**.

### Aplicación de comando en PowerShell:

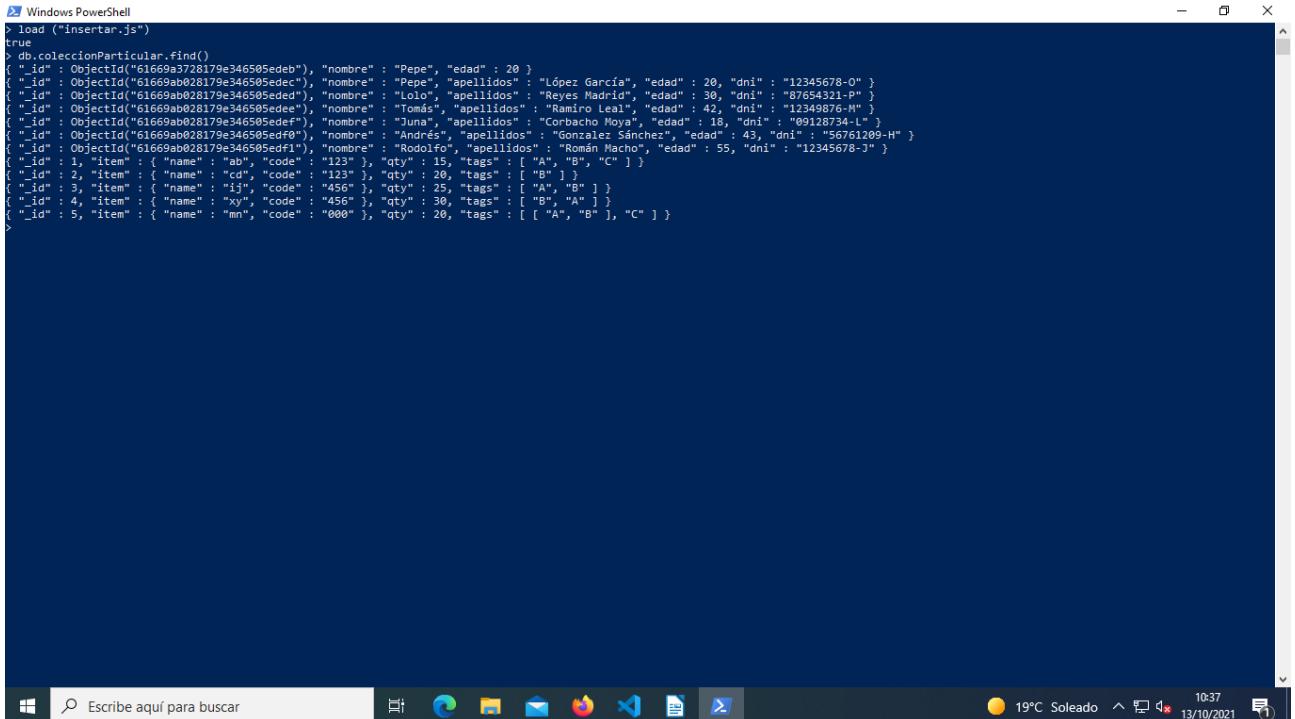


Ahora para que veamos la función que desempeña el campo “**ID**” de cada documento que nos proporciona el propio programa automáticamente, he insertado (insertOne) una serie de documentos con los mismo valores. El cuál aplica un ID único exclusivamente para diferenciar cada documento en el caso de que sus valores sean iguales.



```
> db.coleccionParticular.insertOne ({nombre:"Pepe", edad:20})
{
    "acknowledged" : true,
    "insertedId" : ObjectId("616699ec28179e346505ede8")
}
> db.coleccionParticular.insertOne ({nombre:"Pepe", edad:20})
{
    "acknowledged" : true,
    "insertedId" : ObjectId("616699ee28179e346505ede9")
}
> db.coleccionParticular.insertOne ({nombre:"Pepe", edad:20})
{
    "acknowledged" : true,
    "insertedId" : ObjectId("616699ef28179e346505edea")
}
> db.coleccionParticular.find()
{
    "_id" : ObjectId("6166997028179e346505ede7"),
    "nombre" : "Pepe",
    "edad" : 20
}
{
    "_id" : ObjectId("616699ec28179e346505ede8"),
    "nombre" : "Pepe",
    "edad" : 20
}
{
    "_id" : ObjectId("616699ee28179e346505ede9"),
    "nombre" : "Pepe",
    "edad" : 20
}
{
    "_id" : ObjectId("616699ef28179e346505edea"),
    "nombre" : "Pepe",
    "edad" : 20
}
> db.coleccionParticular.deleteMany({})
{
    "acknowledged" : true,
    "deletedCount" : 4
}
> db.coleccionParticular.find()
{
    "_id" : ObjectId("61669a3728179e346505edb"),
    "nombre" : "Pepe",
    "edad" : 20
}
```

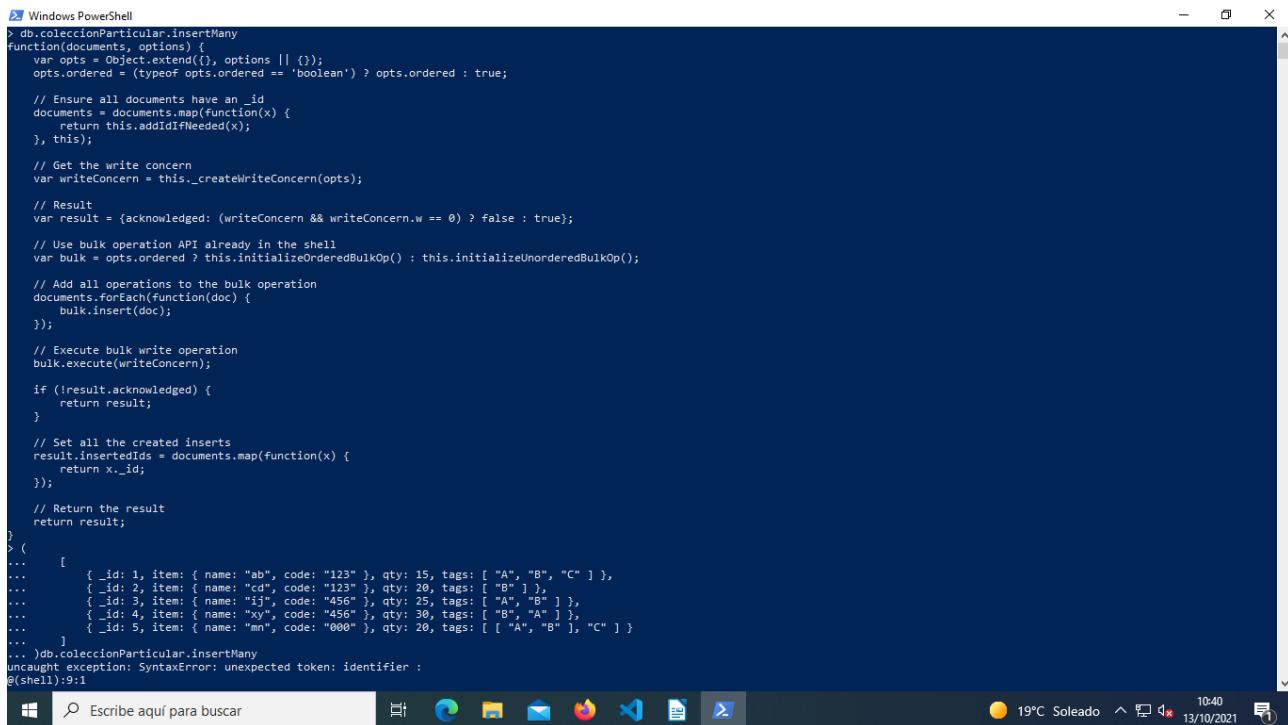
Acto seguido vamos a eliminar todos los documentos añadidos anteriormente porque al final tan sólo queremos que una persona se llame Pepe y que tenga 20 años. Para ello haremos uso del comando **db.coleccionParticular.deleteMany({})**. Como respuesta nos muestra un mensaje como de que ha borrado 4 documentos (**"acknowledged" : true, "deleteCount": 4**), como podemos ver en la captura anterior.



```
> load ("insertar.js")
true
> db.coleccionParticular.find()
{
    "_id" : ObjectId("61669a3728179e346505edb"),
    "nombre" : "Pepe",
    "edad" : 20
}
{
    "_id" : ObjectId("61669ab028179e346505edec"),
    "nombre" : "Pepe",
    "apellidos" : "López García",
    "edad" : 20,
    "dni" : "12345678-0"
}
{
    "_id" : ObjectId("61669ab028179e346505eded"),
    "nombre" : "Lolo",
    "apellidos" : "Reyes Madrid",
    "edad" : 30,
    "dni" : "87654321-P"
}
{
    "_id" : ObjectId("61669ab028179e346505edee"),
    "nombre" : "Tomás",
    "apellidos" : "Ramiro Leal",
    "edad" : 42,
    "dni" : "12349876-M"
}
{
    "_id" : ObjectId("61669ab028179e346505edef"),
    "nombre" : "Juana",
    "apellidos" : "Corbacho Moya",
    "edad" : 18,
    "dni" : "09128734-L"
}
{
    "_id" : ObjectId("61669ab028179e346505edf0"),
    "nombre" : "Andrés",
    "apellidos" : "González Sánchez",
    "edad" : 43,
    "dni" : "56761289-H"
}
{
    "_id" : ObjectId("61669ab028179e346505edf1"),
    "nombre" : "Rodolfo",
    "apellidos" : "Román Macho",
    "edad" : 55,
    "dni" : "12345678-J"
}
{
    "_id" : 1,
    "item" : {
        "name" : "abi",
        "code" : "123",
        "qty" : 15,
        "tags" : [ "A", "B", "C" ]
    }
}
{
    "_id" : 2,
    "item" : {
        "name" : "cd",
        "code" : "123",
        "qty" : 20,
        "tags" : [ "B" ]
    }
}
{
    "_id" : 3,
    "item" : {
        "name" : "ij",
        "code" : "456",
        "qty" : 25,
        "tags" : [ "A", "B" ]
    }
}
{
    "_id" : 4,
    "item" : {
        "name" : "xy",
        "code" : "456",
        "qty" : 30,
        "tags" : [ "B", "A" ]
    }
}
{
    "_id" : 5,
    "item" : {
        "name" : "mn",
        "code" : "000",
        "qty" : 20,
        "tags" : [ [ "A", "B" ], "C" ]
    }
}
```

Ahora vamos a usar otro comando parecido al insertMany que es el comando load, que lo que hace es cargar un fichero completo con todos los documentos que éste contenga. En mi caso, he añadido el fichero insertar.js de la forma load("insertar.js").

Nos devuelve un valor booleano, en nuestro caso, TRUE. Y ahora para visualizar los documentos que constan en ese fichero usaremos el comando **db.coleccionParticular.find()**.



```
> db.coleccionParticular.insertMany
function(documents, options) {
  var opts = Object.extend({}, options || {});
  opts.ordered = (typeof opts.ordered == 'boolean') ? opts.ordered : true;

  // Ensure all documents have an _id
  documents = documents.map(function(x) {
    return this._addIfNeeded(x);
  }, this);

  // Get the write concern
  var writeConcern = this._createWriteConcern(opts);

  // Result
  var result = {acknowledged: (writeConcern && writeConcern.w == 0) ? false : true};

  // Use bulk operation API already in the shell
  var bulk = opts.ordered ? this.initializeOrderedBulkOp() : this.initializeUnorderedBulkOp();

  // Add all operations to the bulk operation
  documents.forEach(function(doc) {
    bulk.insert(doc);
  });

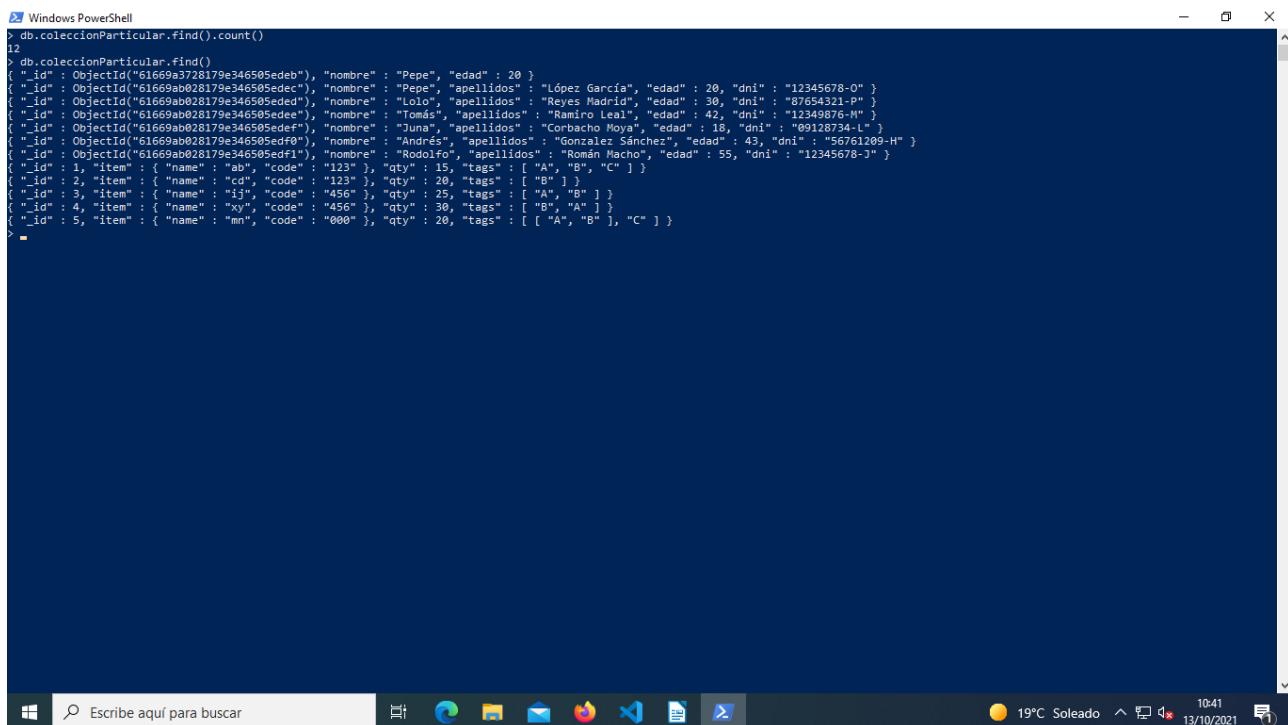
  // Execute bulk write operation
  bulk.execute(writeConcern);

  if (!result.acknowledged) {
    return result;
  }

  // Set all the created inserts
  result.insertedIds = documents.map(function(x) {
    return x._id;
  });

  // Return the result
  return result;
}
> [
...   {
...     "_id": 1, "item": { "name": "ab", "code": "123" }, "qty": 15, "tags": [ "A", "B", "C" ] },
...     "_id": 2, "item": { "name": "cd", "code": "123" }, "qty": 20, "tags": [ "B", "C" ] },
...     "_id": 3, "item": { "name": "ij", "code": "456" }, "qty": 25, "tags": [ "A", "B" ] },
...     "_id": 4, "item": { "name": "xy", "code": "456" }, "qty": 30, "tags": [ "B", "A" ] },
...     "_id": 5, "item": { "name": "mn", "code": "000" }, "qty": 20, "tags": [ [ "A", "B" ], "C" ] }
... ]
... )db.coleccionParticular.insertMany
uncaught exception: SyntaxError: unexpected token: identifier
@(shell):9:1
```

Aquí aplicaremos el comando **insertMany**, que es muy parecido al **load**, es decir, añade una serie de documentos a la vez a una colección. En mi caso, **db.coleccionParticular.insertMany**, y seguido de ello, los documentos que queramos añadir con sus campos y valores.



```
> db.coleccionParticular.find()
12
> db.coleccionParticular.find()
[{"_id": ObjectId("51669ab28179e346595edeb"), "nombre": "Pepe", "edad": 20}, {"_id": ObjectId("51669ab28179e346595edec"), "nombre": "López García", "edad": 20, "dni": "12345678-0"}, {"_id": ObjectId("51669ab28179e346595edee"), "nombre": "Pepa", "apellidos": "López Madrid", "edad": 30, "dni": "87654321-P"}, {"_id": ObjectId("51669ab28179e346595edee"), "nombre": "Lolo", "apellidos": "Reyes Madrid", "edad": 30, "dni": "87654321-P"}, {"_id": ObjectId("51669ab28179e346595edef"), "nombre": "Tomás", "apellidos": "Ramiro Leal", "edad": 42, "dni": "12349876-M"}, {"_id": ObjectId("51669ab28179e346595edef"), "nombre": "Tomás", "apellidos": "Corbacho Moya", "edad": 18, "dni": "09128734-L"}, {"_id": ObjectId("51669ab28179e346595edff0"), "nombre": "Andrés", "apellidos": "González Sánchez", "edad": 43, "dni": "56761289-H"}, {"_id": ObjectId("51669ab28179e346595edff1"), "nombre": "Andrés", "apellidos": "Román Macho", "edad": 55, "dni": "12345678-J"}, {"_id": ObjectId("51669ab28179e346595edff1"), "nombre": "Rodolfo", "apellidos": "Román Macho", "edad": 55, "dni": "12345678-J"}, {"_id": 1, "item": { "name": "ab", "code": "123" }, "qty": 15, "tags": [ "A", "B", "C" ]}, {"_id": 2, "item": { "name": "cd", "code": "123" }, "qty": 20, "tags": [ "B", "C" ]}, {"_id": 3, "item": { "name": "ij", "code": "456" }, "qty": 25, "tags": [ "A", "B" ]}, {"_id": 4, "item": { "name": "xy", "code": "456" }, "qty": 30, "tags": [ "B", "A" ]}, {"_id": 5, "item": { "name": "mn", "code": "000" }, "qty": 20, "tags": [ [ "A", "B" ], "C" ]}]
```

Ahora vamos a hacer uso del comando **count()**, el cual, nos va a contabilizar los documentos que se encuentren en nuestra colección por medio del la línea de comando **db.coleccionParticular.find().count()**. Nos devuelve un valor entero, en mi caso, un 12, que son los documentos que tengo almacenados en mi colección.

```
Windows PowerShell
> db.coleccionParticular.find({tags:"B"})
{
  "_id": 1, "item": { "name": "ab", "code": "123" }, "qty": 15, "tags": [ "A", "B", "C" ] }
{
  "_id": 2, "item": { "name": "cd", "code": "123" }, "qty": 20, "tags": [ "B" ] }
{
  "_id": 3, "item": { "name": "ij", "code": "456" }, "qty": 25, "tags": [ "A", "B" ] }
{
  "_id": 4, "item": { "name": "xy", "code": "456" }, "qty": 30, "tags": [ "B", "A" ] }
>
```

```
Windows PowerShell
> db.coleccionParticular.find({qty:{$eq:20}}).pretty()
{
  "_id": 2,
  "item": {
    "name": "cd",
    "code": "123"
  },
  "qty": 20,
  "tags": [
    "B"
  ]
}

{
  "_id": 5,
  "item": {
    "name": "mn",
    "code": "000"
  },
  "qty": 20,
  "tags": [
    [
      "A",
      "B"
    ],
    "C"
  ]
}
> db.coleccionParticular.find({qty:{$eq:20}})
{
  "_id": 2, "item": { "name": "cd", "code": "123" }, "qty": 20, "tags": [ "B" ] }
{
  "_id": 5, "item": { "name": "mn", "code": "000" }, "qty": 20, "tags": [ [ "A", "B" ], "C" ] }
>
```

En éstas dos últimas capturas hago uso de los comparadores. En la primera muestro los documentos con el campo tags igual a “B” mediante **db.coleccionParticular.find({tags: “B”})**. Y en la segunda captura hago uso del operador equals que compara según el valor que le indiquemos en el filtro. En mi lugar, que tengas el campo qty igual a 20 con el comando **db.coleccionParticular.find({qty: { \$eq: 20 }}). pretty()**. El operador pretty() lo que hace es dar un formato más claro y tabulado del resultado. Si por el contrario no le ponemos el pretty() aparecerá como las dos últimas líneas.