

**Luis Villalba Palacios**

**1ºASIR A**

**Gestión de bases de datos**

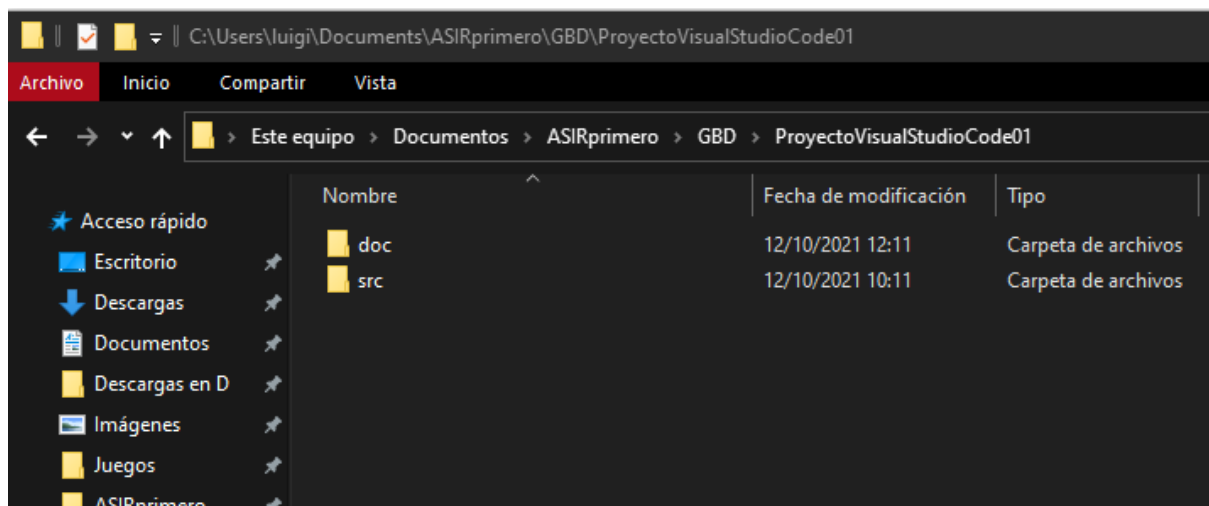
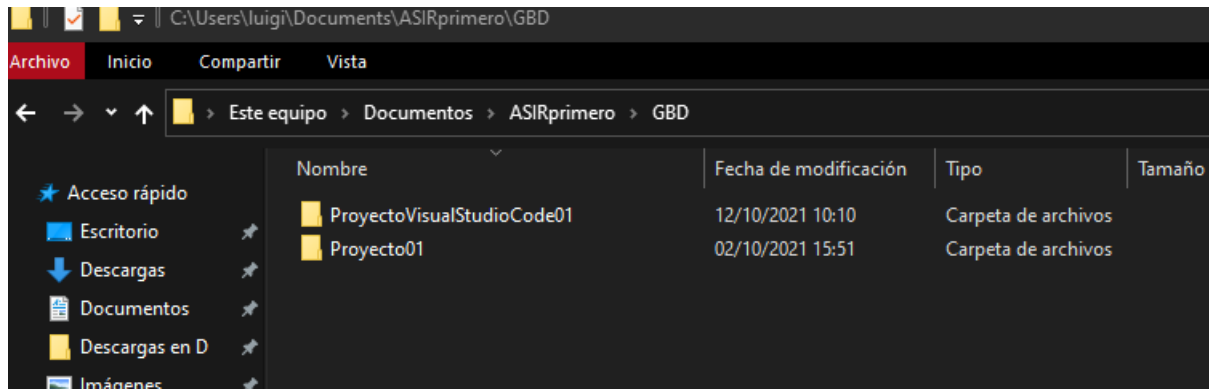
## **Práctica MongoDB**

# **Índice**

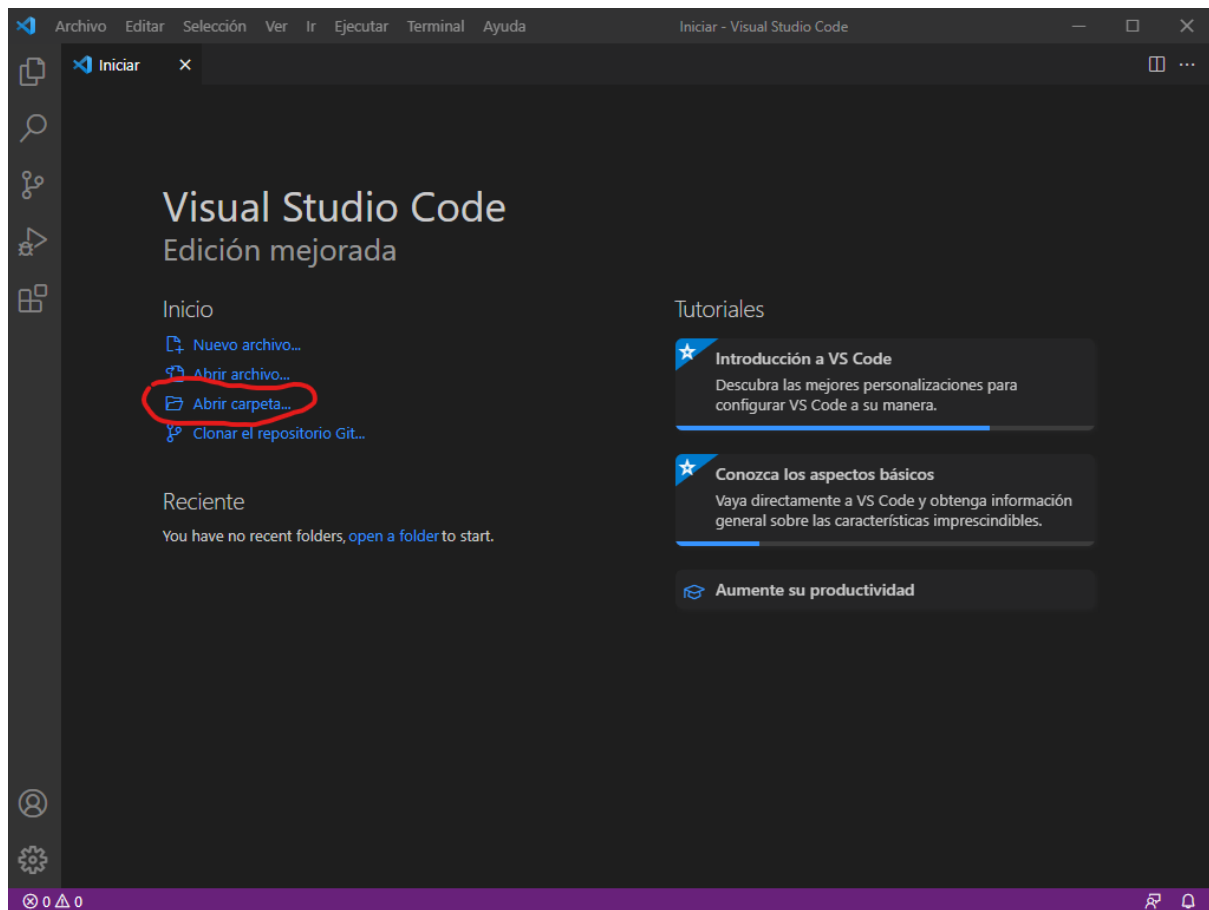
<b>Configurando Visual Studio Code</b>	<b>3</b>
<b>Creación de la primera base de datos</b>	<b>7</b>
<b>Creación de los primeros documentos con PowerShell</b>	<b>8</b>
<b>.insertOne()</b>	<b>8</b>
<b>.insertMany()</b>	<b>10</b>
<b>Creación de comentarios</b>	<b>13</b>
<b>Método find</b>	<b>14</b>

# Configurando Visual Studio Code

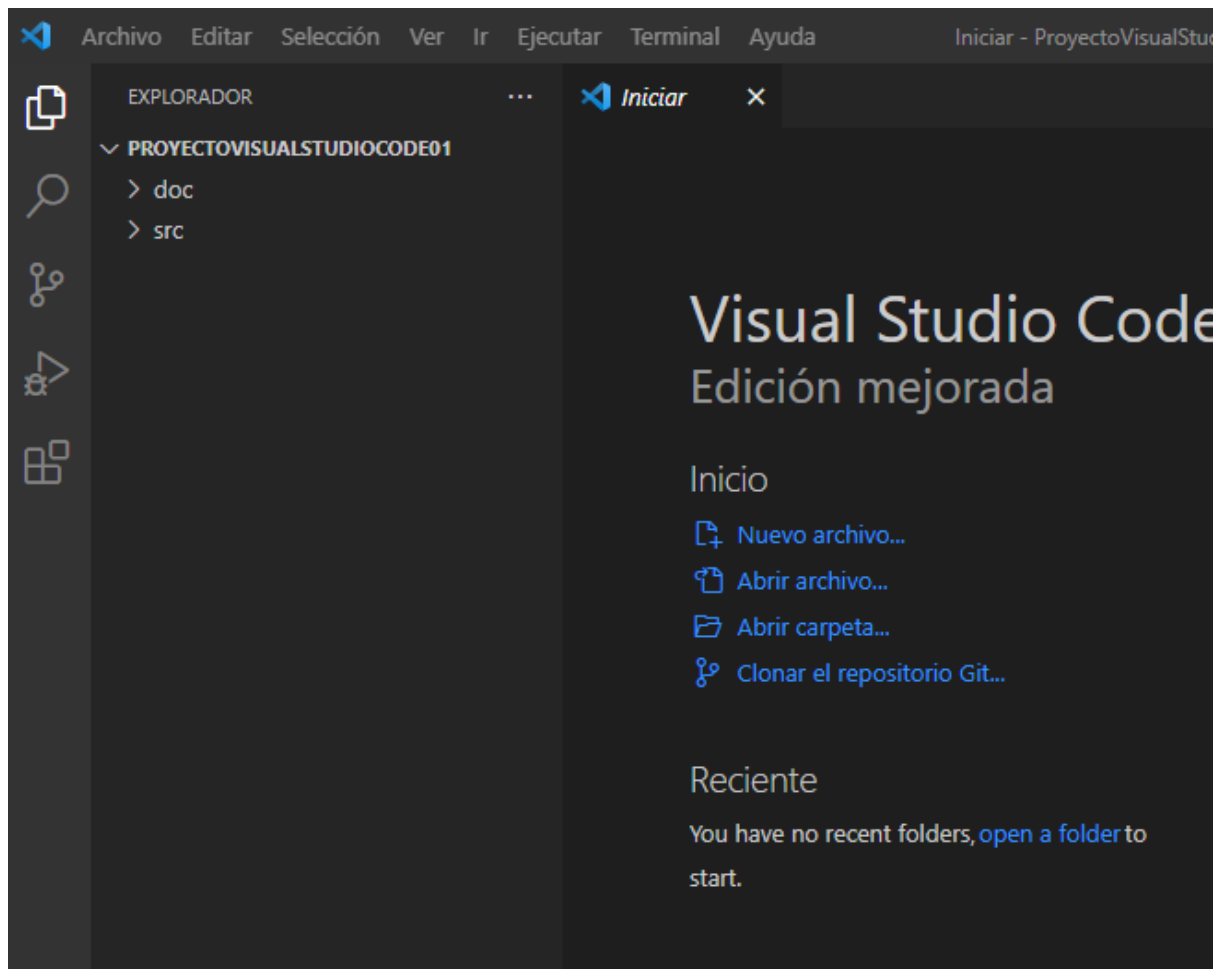
Creamos en nuestro equipo la carpeta que va a contener nuestra práctica, la he denominado “ProyectoVisualStudioCode01”. Dentro de esta carpeta crearemos dos carpetas: “src” y “doc”.



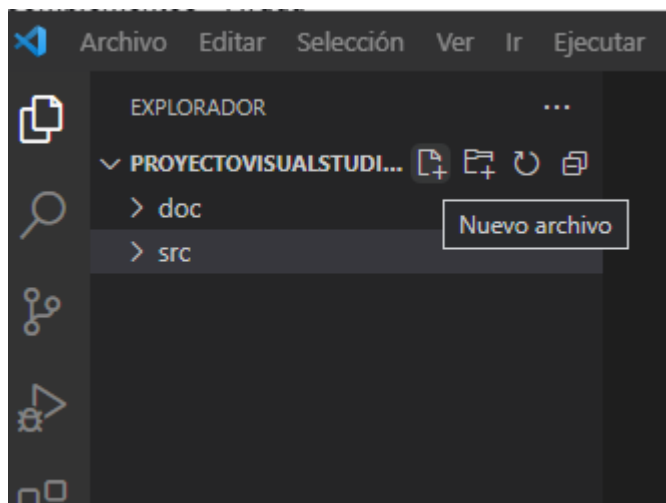
Iniciamos Visual Studio Code y pulsamos donde pone “Abrir Carpeta...”:



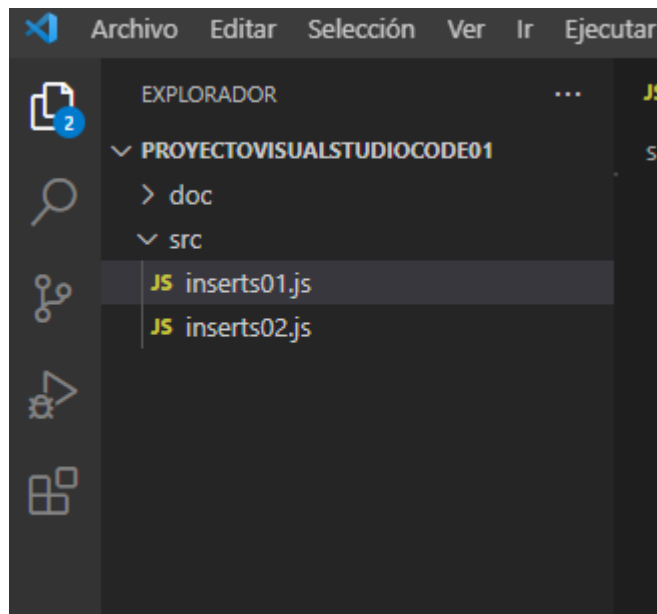
Seleccionamos la carpeta que hemos creado con el nombre "ProyectoVisualStudioCode01".



Dentro de “src” vamos a crear dos nuevos archivos:



Los vamos a llamar en este caso inserts01.js e inserts02.js respectivamente:



# Creación de la primera base de datos

Ejecutamos PowerShell y hacemos cd a nuestro directorio del proyecto:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\luigi> cd C:\Users\luigi\Documents\ASIRprimero\GBD\ProyectoVisualStudioCode01
```

A continuación ejecutamos mongo:

```
PS C:\Users\luigi\Documents\ASIRprimero\GBD\ProyectoVisualStudioCode01> mongo
MongoDB shell version v5.0.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("ce3508a7-ac03-4fbc-b028-d4fb9c7ad6aa") }
MongoDB server version: 5.0.3
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
---
The server generated these startup warnings when booting:
  2021-10-12T12:02:47.691+02:00: 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()
---
```

Ahora vamos a crear una nueva base de datos, en este caso la voy a llamar "BDPrueba01", asique para crearle hemos de escribir el comando:

>use BDPrueba01

```
Windows PowerShell

> use BDPrueba01
switched to db BDPrueba01
>
```

Nos sale un mensaje que nos indica que ahora estamos en BDPrueba01, vamos a comprobarlo escribiendo el comando:

>db

```
> db
BDPrueba01
>
```

Y nos confirma que estamos usando esa base de datos.

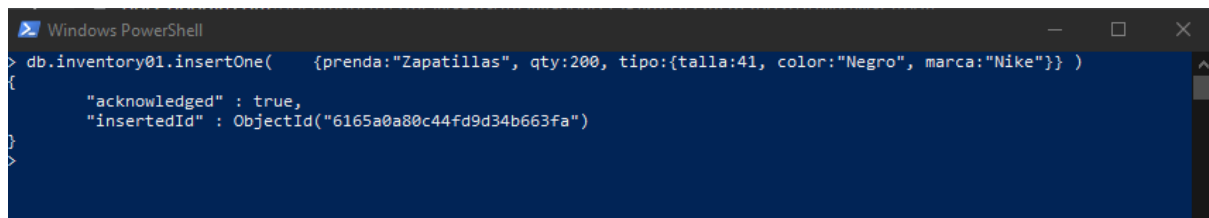
# Creación de los primeros documentos con PowerShell

## .insertOne()

Ahora vamos a insertar nuestro primer documento con PowerShell. Utilizando el comando `insertOne` lo vamos a incluir en nuestra primera colección que vamos a denominar "inventory01" en este caso:

```
db.inventory01.insertOne(      {prenda:"Zapatillas", qty:200, tipo:{talla:41, color:"Negro", marca:"Nike"},
marca:"Nike"}} )
```

Queremos registrar que tenemos 200 zapatillas de la talla 41, negras y de la marca Nike.

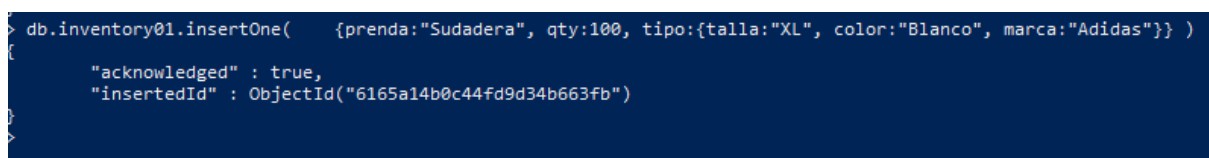


```
Windows PowerShell
> db.inventory01.insertOne(      {prenda:"Zapatillas", qty:200, tipo:{talla:41, color:"Negro", marca:"Nike"},
{
  "acknowledged" : true,
  "insertedId" : ObjectId("6165a0a80c44fd9d34b663fa")
}
>
```

Observamos que nos devuelve el valor "true", para confirmar la operación, y también le ha dotado de un identificador "6165a0a80c44fd9d34b663fa".

Vamos a añadir otro documento a la colección "inventory01".

```
db.inventory01.insertOne(  {prenda:"Sudadera", qty:100, tipo:{talla:"XL", color:"Blanco",
marca:"Adidas"}} )
```



```
> db.inventory01.insertOne(  {prenda:"Sudadera", qty:100, tipo:{talla:"XL", color:"Blanco", marca:"Adidas"}} )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("6165a14b0c44fd9d34b663fb")
}
>
```

Nos devuelve de nuevo el valor "true" y un identificador distinto a este documento.

Ahora utilizando Visual Studio Code vamos a aprender el comando `deleteMany({})` que sirve para borrar todos los documentos, y a continuación vamos a incluir los mismos documentos e incluir dos nuevos:



```
src > JS inserts01.js > ...
1 db.inventory01.deleteMany({})
2 db.inventory01.insertOne( {prenda:"Zapatillas", qty:200, tipo:{talla:41, color:"Negro", marca:"Nike"}} )
3 db.inventory01.insertOne( {prenda:"Sudadera", qty:100, tipo:{talla:"XL", color:"Blanco", marca:"Adidas"}} )
4 db.inventory01.insertOne( {prenda:"Calcetines", qty:800, tipo:{talla:41, color:"Blanco", marca:"Artengo"}} )
5 db.inventory01.insertOne( {prenda:"Pantalones", qty:50, tipo:{talla:41, color:"Azules", marca:"Bershka"}} )
6
```

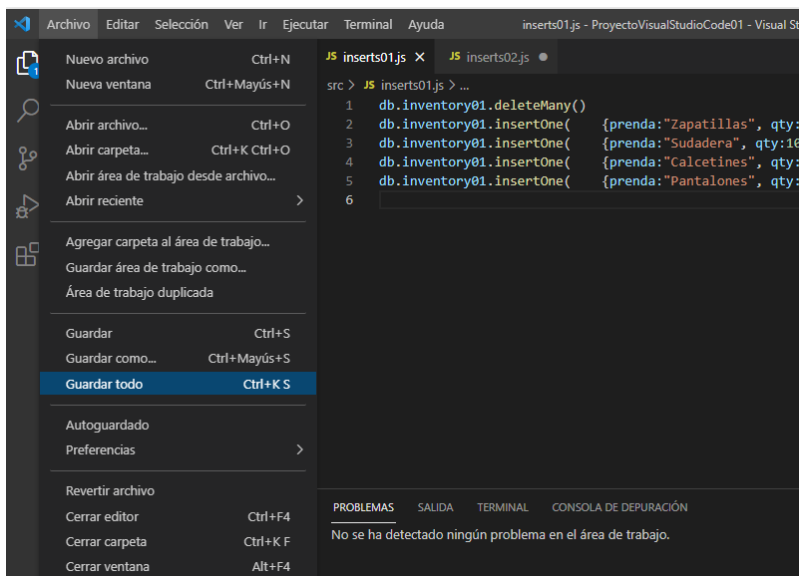
Vamos a probar diferentes métodos de añadir estos documentos a nuestra colección:

- Copiar y pegar: Copiamos desde el Visual Studio Code y lo pegamos en PowerShell:

```
insertedId : ObjectId("616a850db59c7ded723b43cd")
}
> db.inventory01.deleteMany()
uncaught exception: Error: find() requires query criteria :
Bulk/this.find@src/mongo/shell/bulk_api.js:804:23
DBCollection.prototype.deleteMany@src/mongo/shell/crud_api.js:420:20
@(<shell>):1:1
> db.inventory01.insertOne( {prenda:"Zapatillas", qty:200, tipo:{talla:41, color:"Negro", marca:"Nike"}} )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("616a850db59c7ded723b43ce")
}
> db.inventory01.insertOne( {prenda:"Sudadera", qty:100, tipo:{talla:"XL", color:"Blanco", marca:"Adidas"}} )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("616a850db59c7ded723b43cf")
}
> db.inventory01.insertOne( {prenda:"Calcetines", qty:800, tipo:{talla:41, color:"Blanco", marca:"Artengo"}} )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("616a850db59c7ded723b43d0")
}
> db.inventory01.insertOne( {prenda:"Pantalones", qty:50, tipo:{talla:41, color:"Azules", marca:"Bershka"}} )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("616a850db59c7ded723b43d1")
}
>
```

- Cargar documento .js

Nos dirigimos a Visual Studio Code y tenemos que guardar nuestro documento (Archivo>Guardar todo).



A continuación en PowerShell ejecutamos el comando load ("inserts01.js").

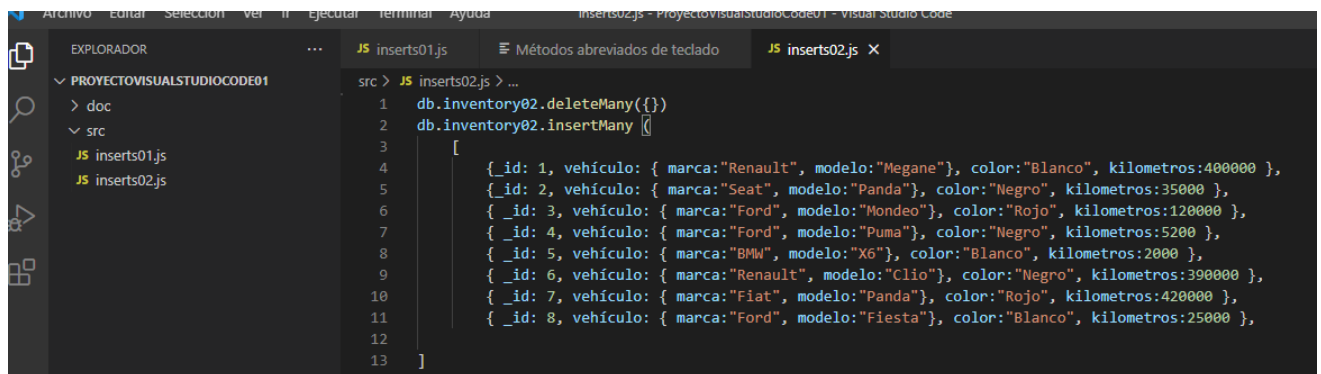
```
@(<shell>):1:1
> load("inserts01.js")
true
>
```

## .insertMany()

Con este comando podemos introducir múltiples documentos en nuestra colección. En este caso he realizado una base de datos de coches en la que he asignado a cada documento un número identificador (id), mongo siempre asigna un identificador automáticamente a cada documento, pero podemos asignarlo nosotros, dentro del campo vehículo he incluido un documento con dos operadores que son marca y modelo. Por último he añadido los campos de color y kilómetros.

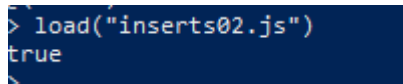
Este inventario quiero que se guarde en otra colección, por eso la vamos a llamar "inventory02" para distinguirlo del primer inventario que realizamos.

Al igual que vimos en el ejemplo anterior empiezo mi archivo con el comando deleteMany para eliminar todos los documentos anteriores de esta colección:

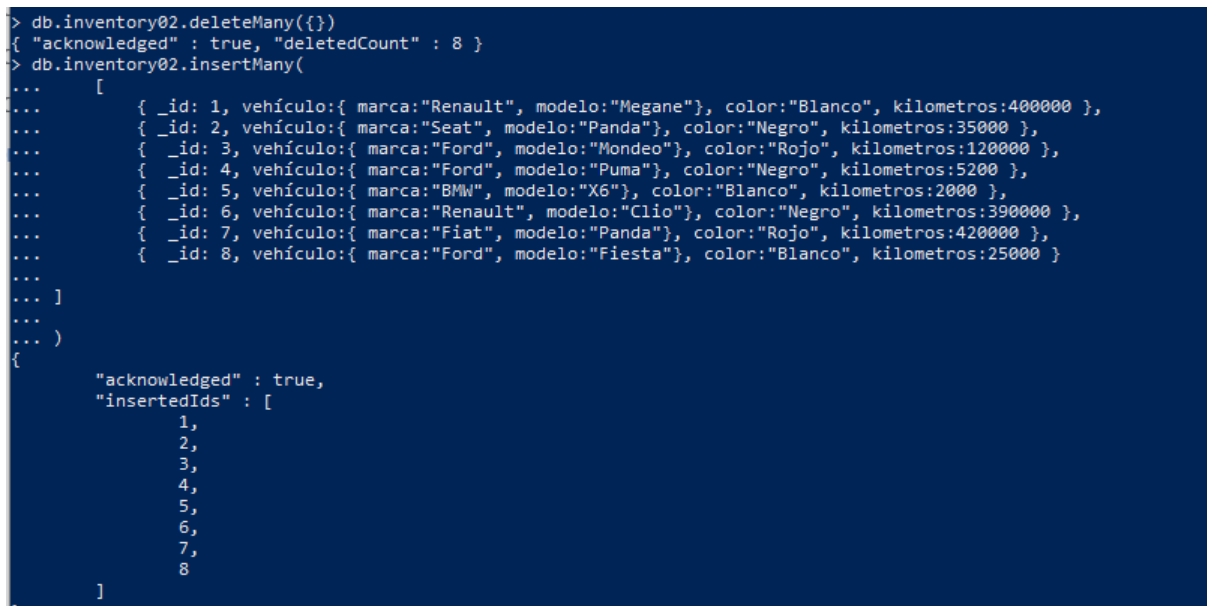


```
src > JS inserts02.js > ...
1 db.inventory02.deleteMany({})
2 db.inventory02.insertMany([
3
4     { _id: 1, vehículo: { marca: "Renault", modelo: "Megane"}, color: "Blanco", kilometros: 400000 },
5     { _id: 2, vehículo: { marca: "Seat", modelo: "Panda"}, color: "Negro", kilometros: 35000 },
6     { _id: 3, vehículo: { marca: "Ford", modelo: "Mondeo"}, color: "Rojo", kilometros: 120000 },
7     { _id: 4, vehículo: { marca: "Ford", modelo: "Puma"}, color: "Negro", kilometros: 5200 },
8     { _id: 5, vehículo: { marca: "BMW", modelo: "X6"}, color: "Blanco", kilometros: 2000 },
9     { _id: 6, vehículo: { marca: "Renault", modelo: "Clio"}, color: "Negro", kilometros: 390000 },
10    { _id: 7, vehículo: { marca: "Fiat", modelo: "Panda"}, color: "Rojo", kilometros: 420000 },
11    { _id: 8, vehículo: { marca: "Ford", modelo: "Fiesta"}, color: "Blanco", kilometros: 25000 },
12
13 ]
14
```

Al igual que hicimos en el apartado anterior, podemos copiarlo directamente a PowerShell o ejecutar el comando load("inserts02.js"):

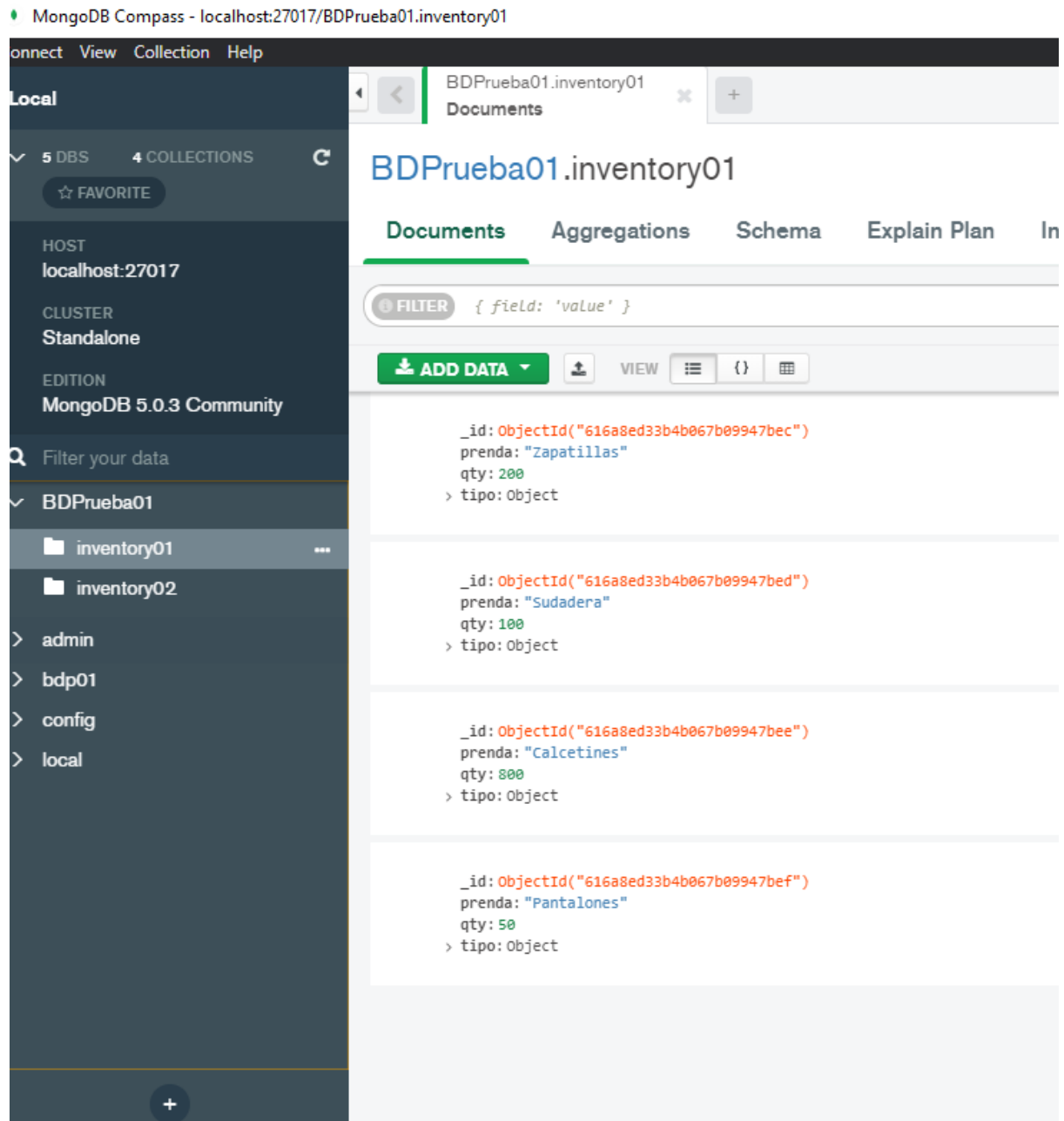


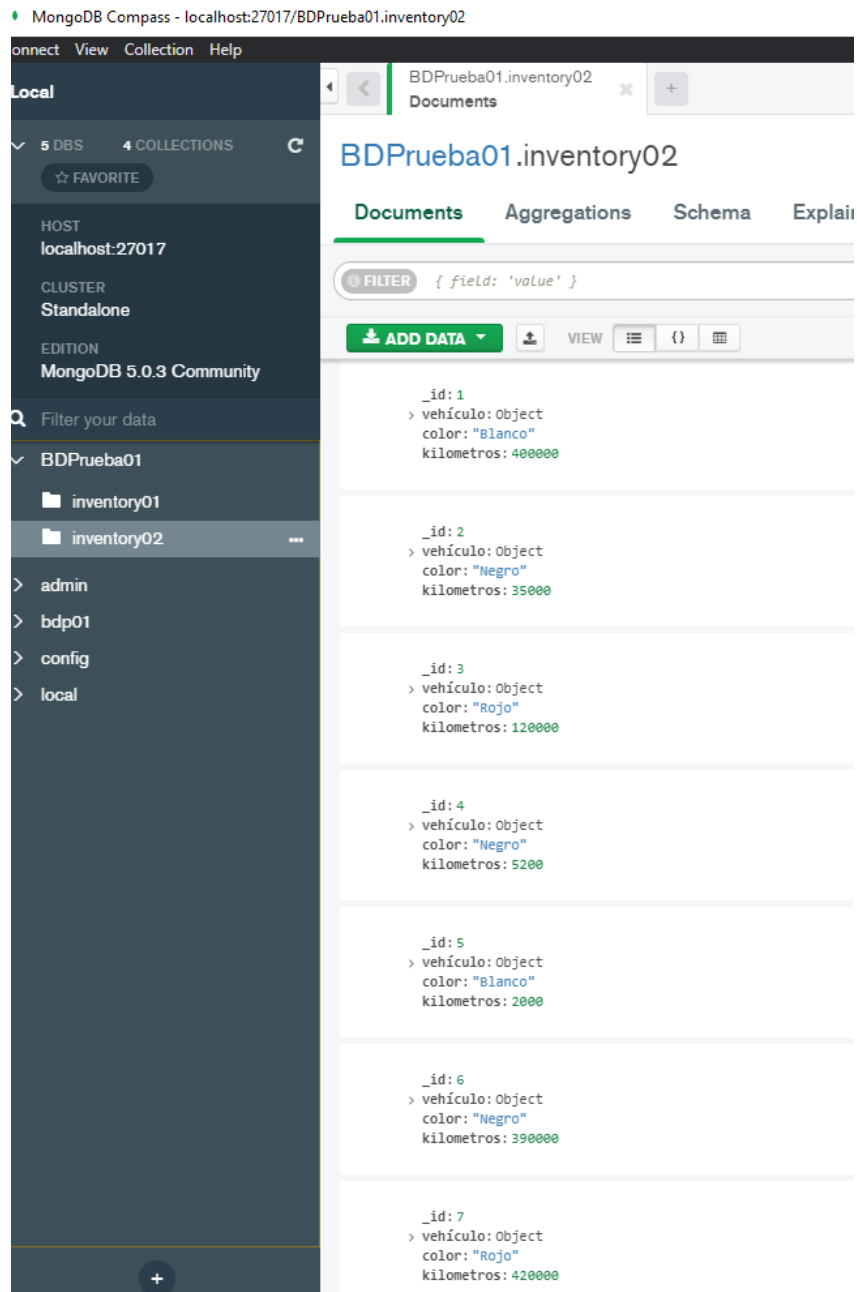
```
> load("inserts02.js")
true
>
```



```
> db.inventory02.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 8 }
> db.inventory02.insertMany(
...   [
...     { _id: 1, vehículo: { marca: "Renault", modelo: "Megane"}, color: "Blanco", kilometros: 400000 },
...     { _id: 2, vehículo: { marca: "Seat", modelo: "Panda"}, color: "Negro", kilometros: 35000 },
...     { _id: 3, vehículo: { marca: "Ford", modelo: "Mondeo"}, color: "Rojo", kilometros: 120000 },
...     { _id: 4, vehículo: { marca: "Ford", modelo: "Puma"}, color: "Negro", kilometros: 5200 },
...     { _id: 5, vehículo: { marca: "BMW", modelo: "X6"}, color: "Blanco", kilometros: 2000 },
...     { _id: 6, vehículo: { marca: "Renault", modelo: "Clio"}, color: "Negro", kilometros: 390000 },
...     { _id: 7, vehículo: { marca: "Fiat", modelo: "Panda"}, color: "Rojo", kilometros: 420000 },
...     { _id: 8, vehículo: { marca: "Ford", modelo: "Fiesta"}, color: "Blanco", kilometros: 25000 }
...   ]
... )
{
  "acknowledged" : true,
  "insertedIds" : [
    1,
    2,
    3,
    4,
    5,
    6,
    7,
    8
  ]
}
```

Si ejecutamos Mongo Compass podemos observar cómo se han creado ambas colecciones:





En mongo Compass nos aparece información de los documentos, indicando el tipo que son:

```

1  _id: 1
2  vehículo: Object
3    marca: "Renault"
4    modelo: "Megane"
5    color: "Blanco"
6    kilometros: 400000

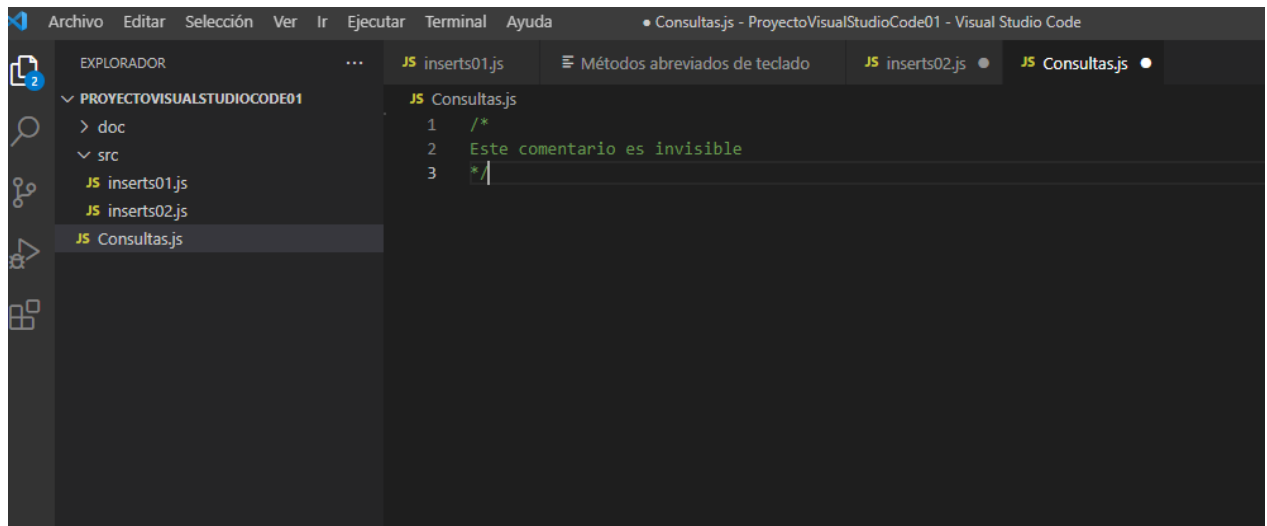
```

Double  
Object  
String  
String  
String  
Double

# Creación de comentarios

En Visual Studio Code podemos crear comentarios invisibles, para ello tenemos que escribir  
`/* <aquí nuestro comentario> */`

Vamos a crear un nuevo documento llamado Consultas.js y vamos a introducir nuestro primer comentario:



Vamos a utilizar este método para introducir los enunciados de nuestras búsquedas del próximo apartado.

## Método find

Si queremos localizar por ejemplo todos los coches que sean de color blanco tenemos que utilizar el comando `db.inventory02.find({color: "Blanco"})`

```
> db.inventory02.find({color: "Blanco"})
{ "_id" : 1, "vehículo" : { "marca" : "Renault", "modelo" : "Megane" }, "color" : "Blanco", "kilometros" : 400000 }
{ "_id" : 5, "vehículo" : { "marca" : "BMW", "modelo" : "X6" }, "color" : "Blanco", "kilometros" : 2000 }
{ "_id" : 8, "vehículo" : { "marca" : "Ford", "modelo" : "Fiesta" }, "color" : "Blanco", "kilometros" : 25000 }
```

Para que nos muestre los resultados de una manera más ordenada podemos añadir al comando `.pretty()` como vemos a continuación:

```
> db.inventory02.find({color: "Blanco"}).pretty()
{
  "_id" : 1,
  "vehículo" : {
    "marca" : "Renault",
    "modelo" : "Megane"
  },
  "color" : "Blanco",
  "kilometros" : 400000
}
{
  "_id" : 5,
  "vehículo" : {
    "marca" : "BMW",
    "modelo" : "X6"
  },
  "color" : "Blanco",
  "kilometros" : 2000
}
{
  "_id" : 8,
  "vehículo" : {
    "marca" : "Ford",
    "modelo" : "Fiesta"
  },
  "color" : "Blanco",
  "kilometros" : 25000
}
```

Si queremos que los resultados sean iguales a X tenemos que introducir el query `$eq`

Por ejemplo si queremos encontrar en `inventory01` todos los artículos de la talla 41, tenemos que tener en cuenta que talla es un campo que pertenece al objeto tipo, por eso tenemos que indicar en la búsqueda `tipo.talla`

```
> db.inventory01.find ({ "tipo.talla": {$eq: 41}})
{ "_id" : ObjectId("616a8ed33b4b067b09947bec"), "prenda" : "Zapatillas", "qty" : 200, "tipo" : { "talla" : 41, "color" : "Negro", "marca" : "Nike" } }
{ "_id" : ObjectId("616a8ed33b4b067b09947bee"), "prenda" : "Calcetines", "qty" : 800, "tipo" : { "talla" : 41, "color" : "Blanco", "marca" : "Artengo" } }
{ "_id" : ObjectId("616a8ed33b4b067b09947bef"), "prenda" : "Pantalones", "qty" : 50, "tipo" : { "talla" : 41, "color" : "Azules", "marca" : "Bershka" } }
>
```

Vamos a añadir el comando `.pretty()` para que nos lo muestre mejor:

```
> db.inventory01.find ({ "tipo.talla": {$eq: 41}}).pretty()
{
  "_id" : ObjectId("616a8ed33b4b067b09947bec"),
  "prenda" : "Zapatillas",
  "qty" : 200,
  "tipo" : {
    "talla" : 41,
    "color" : "Negro",
    "marca" : "Nike"
  }
}
{
  "_id" : ObjectId("616a8ed33b4b067b09947bee"),
  "prenda" : "Calcetines",
  "qty" : 800,
  "tipo" : {
    "talla" : 41,
    "color" : "Blanco",
    "marca" : "Artengo"
  }
}
{
  "_id" : ObjectId("616a8ed33b4b067b09947bef"),
  "prenda" : "Pantalones",
  "qty" : 50,
  "tipo" : {
    "talla" : 41,
    "color" : "Azules",
    "marca" : "Bershka"
  }
}
>
```

En el apartado de consultas he introducido una serie de enunciados que se resuelven con el método `find` y los siguientes selectores de consultas:

- **\$eq** Busca valores que sean iguales al valor que se especifique.
- **\$gt** Busca valores que sean mayores al valor que se especifique.
- **\$gte** Busca valores que sean iguales o mayores al valor que se especifique.
- **\$lt** Busca valores que sean menores al valor que se especifique.
- **\$lte** Busca valores que sean iguales o menores al valor que se especifique.
- **\$ne** Busca valores que no sean iguales al valor especificado.