

Primeros pasos en

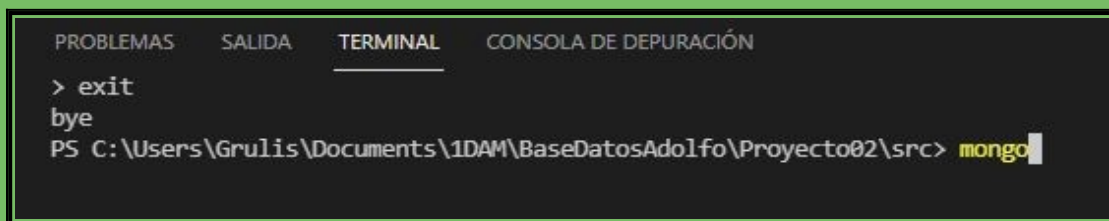
Vamos a dar nuestros primeros pasos en MongoDB y para ello nos vamos a apoyar en el uso del entorno de desarrollo Visual Studio Code que hemos instalado junto a MongoDB.

En primer lugar vamos a ejecutar el programa.

Antes de nada, después de haber instalado Visual Studio Code y MongoDB, si todo esta correctamente instalado podremos ver al abrir el Terminal que al ejecutar la instrucción

> mongo

el sistema se activa para esperar instrucciones.

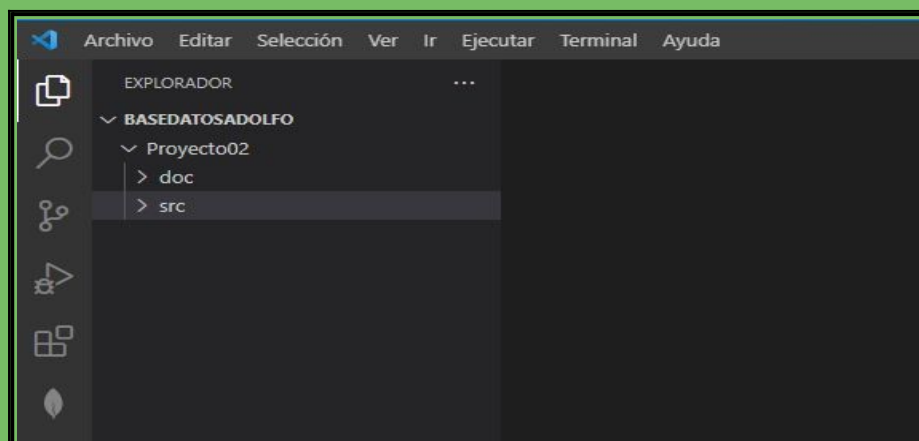


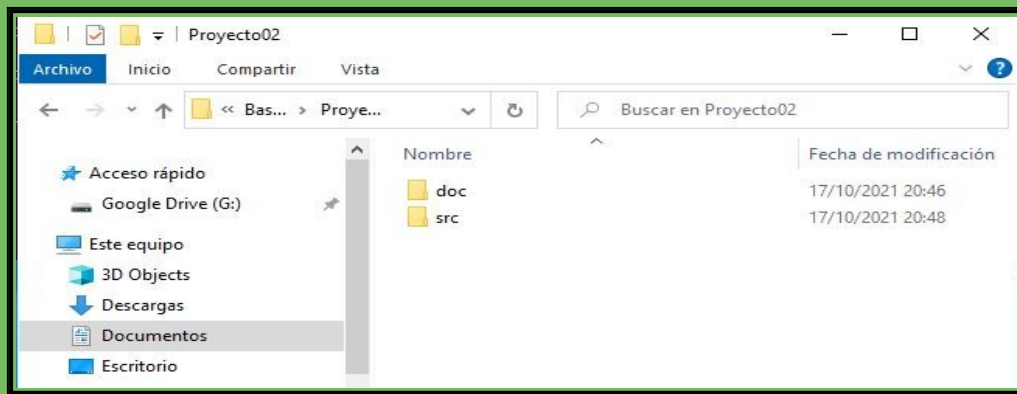
```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN
> exit
bye
PS C:\Users\Gulis\Documents\1DAM\BaseDatosAdolfo\Proyecto02\src> mongo
```

Para salir de ello y volver al PowerShell deberemos introducir la instrucción

> exit

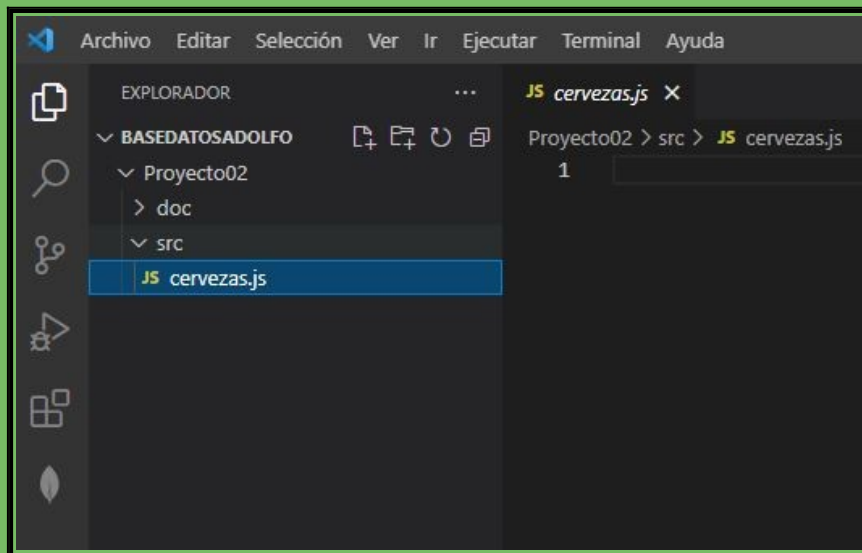
En Windows entraremos dentro de la carpeta donde guardaremos todos nuestros proyectos y crearemos una nueva carpeta llamada Proyecto02. Veremos que se ha creado automáticamente en Visual Studio también si todo esta bien enlazado. Dentro de esta carpeta Proyecto02 crearemos dos carpetas: la carpeta “Doc” que contendrá la documentación de nuestro proyecto y la carpeta “src” que contendrá los programas que vayamos creando.





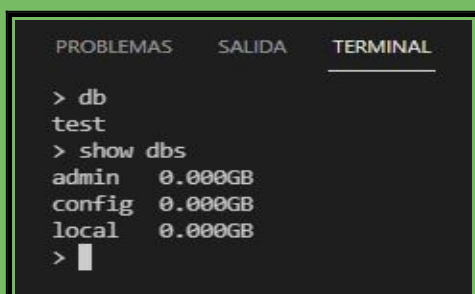
Observamos como al crear las dos carpetas en Visual Studio aparecen automáticamente en la carpeta de nuestro Proyecto en Windows.

Dentro de la carpeta “src” crearemos nuestra primera Base de Datos donde incluiremos los tipos de cerveza que nos gustan y lo que llamaremos “cervezas.js” (con la extensión de los documentos Json - JavaScript Object Notation- con los que trabajaremos en las bases de datos no relacionales).



Ok. Ya tenemos preparado nuestro sitio de trabajo y el entorno. Pues vamos a empezar:

En primer lugar desde el terminal si intentáramos insertar alguna colección o documento el sistema lo va a hacer sobre la base de datos que trae por defecto mongoDB que que es “test”. Como podemos saber esto? Pues preguntando a mongo en que base de datos esta posicionado y para ello usamos la instrucción “DB”. Al ejecutarla nos mostrara >test.



Si lo que queremos es ver todas las bases de datos que tenemos introduciremos la instrucción:

> show dbs.

También podemos usar la instrucción `show` para mostrar todas las colecciones que tenemos. Para ello ejecutamos la instrucción

> show collections

pero esta instrucción la probaremos mas adelante ya que no tenemos ninguna creada de momento.

Nosotros vamos a crear la nuestra propia para empezar a trabajar con ella. Para ello se Usa la instrucción `use` seguido del nombre de nuestra base de datos. Si no existe, la creará nueva. En nuestro caso la llamaremos “cervezas”. Seria:

> use Cervezas

Ahora, si preguntamos de nuevo por la Base de Datos nos dirá que estamos en > Cervezas.

> db

```
PROBLEMAS  SALIDA  TERMINAL
> use Cervezas
switched to db Cervezas
> db
Cervezas
> |
```

Ok. Ya tenemos nuestra base de datos creada pero esta vacía. Vamos a meter información de nuestras cervezas en ella. Para hacer esto utilizaremos la instrucción **`insertOne()`**. Por ejemplo, vamos a insertar un documento con los campos “nombre”, “grados”, “tipo”, “origen” y un array que contendrá los ingredientes principales llamado “ingredientes”. Los valores de esos campos serán “Rubita”, “5”, “pilsen”, “Checo” y [{cebada:2.5,lupulo:0.25,levadura:0.25,agua:27}]:

>db.col01.insertOne({nombre:'Rubita',grados:19,tipo:'pilsen',origen:'Checo',ingredientes:[{cebada:2.5,lupulo:0.25,levadura:0.25,agua:27}]})

```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN
> db.col01.insertOne({nombre:'Rubita',grados:19,tipo:'pilsen',origen:'Checo',ingredientes:[{cebada:2.5,lupulo:0.25,levadura:0.25,agua:27}]})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("616c8889a028145c5b648599")
}
> |
```

como vemos, ya se ha insertado una nueva colección y dentro de ella hay un documento con varios campos y sus valores. Incluso uno de los campos tiene como valor un array.

Para verlo en el terminal podemos ejecutar la instrucción:

> db.col01.find().

```
> db.col01.find()
{ "_id" : ObjectId("616c8889a028145c5b648599"), "nombre" : "Rubita", "grados" : 19, "tipo" : "pilsen", "origen" : "Checo", "ingredientes" : [ { "cebada" : 2.5, "lupulo" : 0.25, "levadura" : 0.25, "agua" : 27 } ] }
> |
```

Ahora vamos a insertar varios documentos a la vez. Para ello utilizaremos una nueva instrucción llamada **insertMany()**.

En nuestro ejemplo, vamos a introducir tres nuevas cervezas que hemos adquirido. Para ello ejecutamos la siguiente instrucción:

```
>db.col01.insertMany([{\nombre:'Negrita',grados:6.5,tipo:'stout',origen:'Ingles',ingredientes:[{\cebada:2,Tostada:0.5,lupulo:0.15,levadura:0.25,agua:25}],\nombre:'Tostadita',grados:5.5,tipo:'polter',origen:'Ingles',ingredientes:[{\cebada:2,Tostada:0.2,lupulo:0.15,levadura:0.30,agua:26}],\nombre:'Olorosa',grados:6,tipo:'lpa',origen:'Ingles',ingredientes:[{\cebada:3,lupulo:0.8,levadura:0.15,agua:23}]}])
```

y vemos que se han incorporado a nuestra Base de Datos las tres nuevas cervezas.

```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN

,levadura:0.15,agua:23}]}])gredientes:[{\cebada:2,Tostada:0.2,lupulo:0.15,levadura:0.30
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("616c92afd24575b7242a591d"),
    ObjectId("616c92afd24575b7242a591e"),
    ObjectId("616c92afd24575b7242a591f")
  ]
}
> |
```

Si ahora ejecutamos la instrucción **find()** anterior podremos ver que ahora tenemos cuatro cervezas: Rubita, Negrita, Tostadita y Olorosa.

```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN

> db.col01.find()
{ "_id" : ObjectId("616c8889a028145c5b648599"), "nombre" : "Rubita", "grados" : 19, "tipo" : "pilsen",
  "grados" : 0.25, "agua" : 27 } ] }
{ "_id" : ObjectId("616c92afd24575b7242a591d"), "nombre" : "Negrita", "grados" : 6.5, "tipo" : "stout",
  "grados" : 0.15, "levadura" : 0.25, "agua" : 25 } ] }
{ "_id" : ObjectId("616c92afd24575b7242a591e"), "nombre" : "Tostadita", "grados" : 5.5, "tipo" : "pol",
  "grados" : 0.15, "levadura" : 0.3, "agua" : 26 } ] }
{ "_id" : ObjectId("616c92afd24575b7242a591f"), "nombre" : "Olorosa", "grados" : 6, "tipo" : "Ipa", "grados",
  "grados" : 5, "agua" : 23 } ] }
> |
```

Pero que pasaría si quisiéramos buscar en concreto una de ellas? Pues usando la instrucción **find()**; podríamos buscar por ejemplo la que se llama de nombre: 'Tostadita'. Esto sería:

```
> db.col01.find({nombre:'Tostadita'})
```

```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN
> db.col01.find({nombre:'Tostadita'})
{ "_id" : ObjectId("616c92afd24575b7242a591e"), "nombre" : "Tostadita", "grados" : 5.5, "tipo" : "polter",
: 0.15, "levadura" : 0.3, "agua" : 26 } ] }
> []
```

También podemos buscar las cervezas que tengan un mismo origen por ejemplo 'Ingles' origen que comparten tres de ellas. En este caso sería:

> db.col01.find({origen:'Ingles'})

```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN
> db.col01.find({nombre:'Tostadita'})
{ "_id" : ObjectId("616c92afd24575b7242a591e"), "nombre" : "Tostadita", "grados" : 5.5, "tipo" : "polter", "origen" : "Ingles",
: 0.15, "levadura" : 0.3, "agua" : 26 } ] }
> db.col01.find({origen:'Ingles'})
{ "_id" : ObjectId("616c92afd24575b7242a591d"), "nombre" : "Negrita", "grados" : 6.5, "tipo" : "stout", "origen" : "Ingles", "i
0.15, "levadura" : 0.25, "agua" : 25 } ] }
{ "_id" : ObjectId("616c92afd24575b7242a591e"), "nombre" : "Tostadita", "grados" : 5.5, "tipo" : "polter", "origen" : "Ingles",
: 0.15, "levadura" : 0.3, "agua" : 26 } ] }
{ "_id" : ObjectId("616c92afd24575b7242a591f"), "nombre" : "Olorosa", "grados" : 6, "tipo" : "Ipa", "origen" : "Ingles", "ingre
5, "agua" : 23 } ] }
> []
```

Ahora supongamos que añadimos las tres cervezas a una col02 de tal forma que tenemos en las dos colecciones tres cervezas repetidas y decidimos eliminar una de ellas.

Para ello utilizaríamos la instrucción **deleteOne()**:

> db.col02.deleteOne({nombre:'Olorosa'})

```
PROBLEMAS  2  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN
> db.col02.deleteOne({nombre:'Olorosa'})
{ "acknowledged" : true, "deletedCount" : 1 }
> []
```

Como hemos borrado una, nos deben quedar dos en col02 para ello volvemos a usar la consulta **find()** en la col02:

> db.col02.find()

```
PROBLEMAS  2  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN
> db.col02.deleteOne({nombre:'Olorosa'})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.col02.find()
{ "_id" : ObjectId("616ca171d24575b7242a5923"), "nombre" : "Negrita", "grados" : 6.5, "tipo" : "stout", "i
0.15, "levadura" : 0.25, "agua" : 25 } ] }
{ "_id" : ObjectId("616ca171d24575b7242a5924"), "nombre" : "Tostadita", "grados" : 5.5, "tipo" : "polter"
: 0.15, "levadura" : 0.3, "agua" : 26 } ] }
> []
```

y vemos que nos quedan solo las cervezas Stout y la Polter.

No queremos mantener la col02 porque hemos decidido que no nos interesa tener información duplicada con la col01, pues eliminamos con la instrucción **deleteMany()**:

> db.col02.deleteMany({})

```
> db.col02.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 2 }
> |
```

Con lo cual, si ahora preguntamos por la información que hay en col02 nos dirá que no hay nada.

Continuará con nuevos conocimientos...