

MongoDB

MongoDB y su utilización

MongoDB es un Sistema de base de datos no relacional (noSQL) open-source y orientado a documentos. Además, MongoDB es multiplataforma, tiene un alto rendimiento y la disponibilidad de los datos es muy alta.

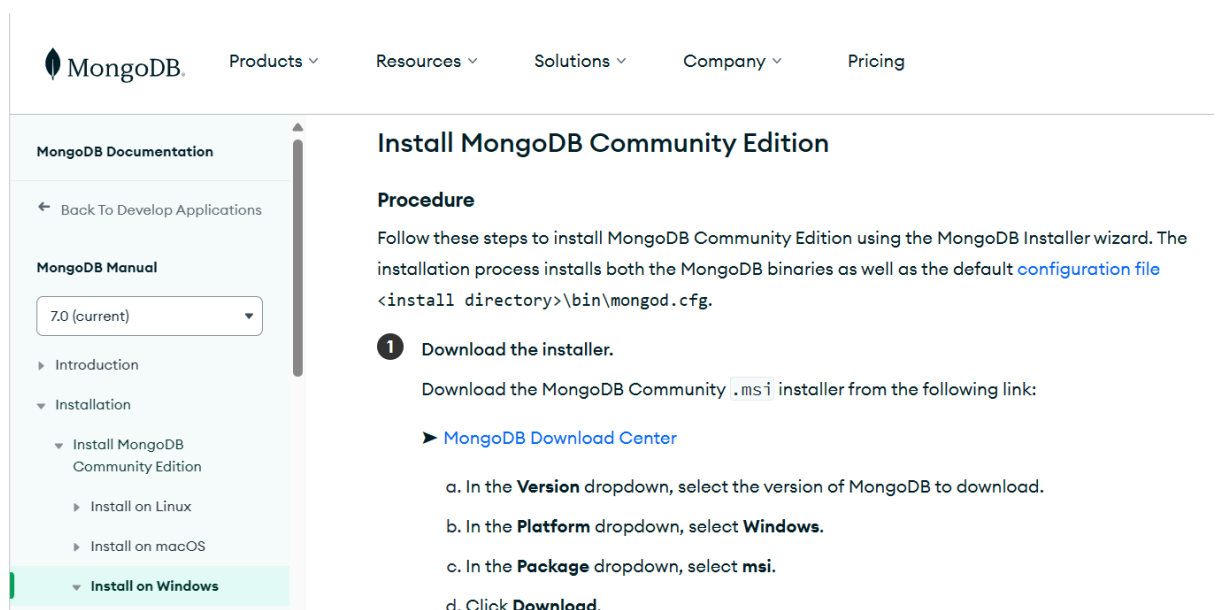
Hemos decidido utilizar una base de datos NoSQL para los datos relacionados a los Vehículos de Guiado Automático (AGV). La razón radica en que este tipo de bases de datos es la idónea para aplicaciones con un crecimiento rápido de los datos y además es una base de datos muy rápida de montar. Los AGV van a proveer grandes cantidades de información no estructurada, al ser dispositivos que se actualizan continuamente, y necesitamos una base de datos que pueda soportar el crecimiento de los datos y datos con distinta estructura.

El funcionamiento de MongoDB es bastante intuitivo. Dentro del servidor de Mongo encontramos nuestras bases de datos. En el interior de cada base de datos podemos hallar colecciones, que son el equivalente a las tablas en SQL. Dentro de cada colección encontramos documentos, análogos a los registros de las BBDD SQL. Finalmente, la representación de cada documento se lleva a cabo con pares clave-valor.

Instalación MongoDB

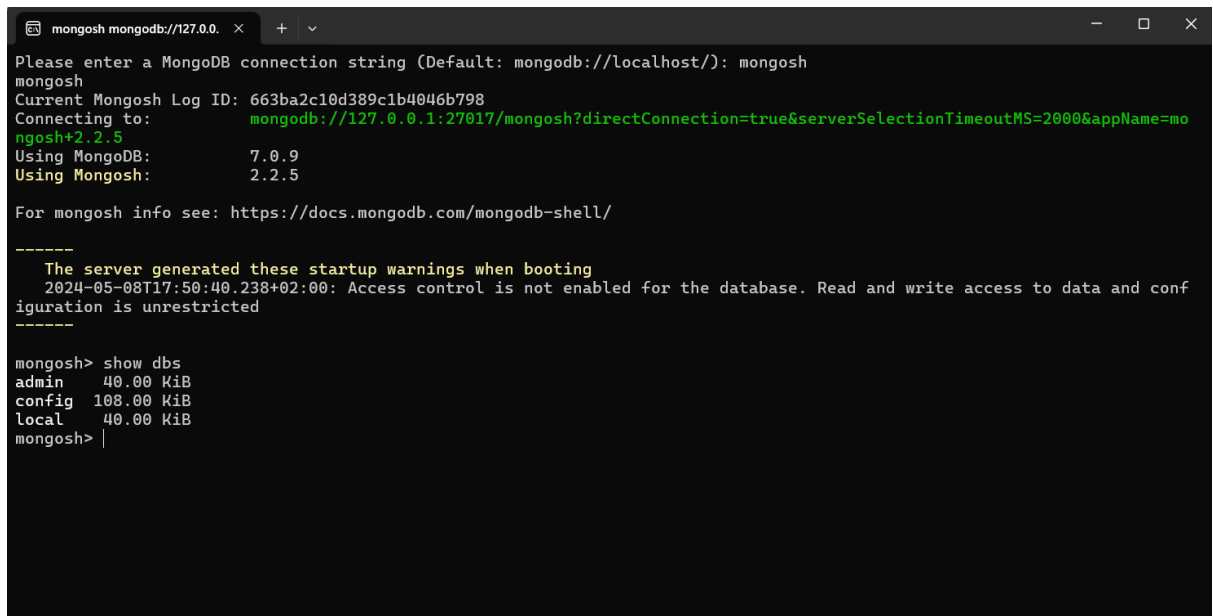
Hemos decidido instalar MongoDB en los ordenadores portátiles de dos de los miembros de nuestro equipo para conseguir una mayor velocidad de trabajo y para poder realizar la instalación en un entorno windows.

Para la instalación hemos acudido a la página oficial de MongoDB como muestra la Figura X.



Una vez instalado MongoDB, hemos instalado la shell de MongoDB. En ella, debemos escribir la instrucción “mongosh” para establecer una conexión con la base de datos.

La Figura X muestra la shell de MongoDB que vamos a utilizar como punto de comunicación con nuestra BBDD.



```
mongosh mongodb://127.0.0.1:27017/
Please enter a MongoDB connection string (Default: mongodb://localhost/): mongosh
mongosh
Current Mongosh Log ID: 663ba2c10d389c1b4046b798
Connecting to: mongodb://127.0.0.1:27017/mongosh?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.5
Using MongoDB: 7.0.9
Using Mongosh: 2.2.5

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

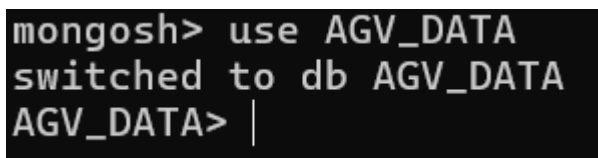
-----
The server generated these startup warnings when booting
2024-05-08T17:50:40.238+02:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

mongosh> show dbs
admin    40.00 KiB
config  108.00 KiB
local    40.00 KiB
mongosh> |
```

Creación de documentos y colecciones

Para comenzar hemos creado una base de datos llamada AGV_DATA en la que volcaremos todos los datos relacionados a los AGV.

La Figura X muestra la creación de la BBDD.



```
mongosh> use AGV_DATA
switched to db AGV_DATA
AGV_DATA> |
```

A continuación, creamos las colecciones de nuestra BBDD. Hemos decidido crear 4 colecciones para almacenar los distintos documentos de los AGV. Todos los documentos de las colecciones contienen como atributo el id del AGV para poder identificar de qué vehículo proviene la información. Las 4 colecciones creadas han sido:

- RutasTiemposAGV
- EstadoDisponibilidadAGV
- SensoresAGV
- OperacionesMantenimientoAGV

La Figura X muestra la creación de la colección “RutasTiemposAGV”.

```
AGV> db.createCollection("RutasTiemposAGV")
{ ok: 1 }
AGV> |
```

Ahora procedemos a introducir documentos en las colecciones creadas, que representen datos provenientes de los distintos AGV funcionando en la empresa. Hemos introducido un total de 21 documentos en cada colección.

La Figura X muestra un ejemplo de inserción de un documento en la colección "RutasTiemposAGV".

```
AGV> db.RutasTiemposAGV.insertOne({_id: ObjectId('00000000000000000000000000000001'), nombre: "AGV001", ruta: "Ruta 1",
punto_origen: "Almacén A", punto_destino: "Línea de producción 1", tiempo_requerido: 5})
{
  acknowledged: true,
  insertedId: ObjectId('00000000000000000000000000000001')
}
AGV> |
```

```
AGV> db.RutasTiemposAGV.find()
[
  {
    _id: ObjectId('00000000000000000000000000000001'),
    nombre: 'AGV001',
    ruta: 'Ruta 1',
    punto_origen: 'Almacén A',
    punto_destino: 'Línea de producción 1',
    tiempo_requerido: 5
  }
]
```

```

{
  _id: ObjectId('00000000000000000000000000000003'),
  nombre: 'AGV003',
  ruta: null,
  punto_origen: 'Almacén B',
  punto_destino: null,
  tiempo_requerido: null
},
{
  _id: ObjectId('00000000000000000000000000000004'),
  nombre: 'AGV004',
  ruta: 'Ruta 2',
  punto_origen: 'Almacén C',
  punto_destino: 'Línea de producción 2',
  tiempo_requerido: 8
},
{
  _id: ObjectId('00000000000000000000000000000005'),
  nombre: 'AGV005',
  ruta: 'Ruta 5',
  punto_origen: 'Línea de producción 2',
  punto_destino: 'Línea de producción 1',
  tiempo_requerido: 4
},

```

La Figura X muestra un ejemplo de inserción de un documento en la colección “EstadoDisponibilidadAGV”.

```

AGV> db.EstadoDisponibilidadAGV.insertOne({_id: ObjectId("00000000000000000000000000000001"), nombre: "AGV001", posicion: {latitud: 123.456, longitud: 789.012}, direccion: "Norte", bateria: 75, estado: "en movimiento", disponibilidad: true})
{
  acknowledged: true,
  insertedId: ObjectId('00000000000000000000000000000001')
}

```

```

AGV> db.EstadoDisponibilidadAGV.find()
[
  {
    _id: ObjectId('00000000000000000000000000000001'),
    nombre: 'AGV001',
    posicion: { latitud: 123.456, longitud: 789.012 },
    direccion: 'Norte',
    bateria: 75,
    estado: 'en movimiento',
    disponibilidad: true
  },

```

La Figura X, Figura X, Figura X y Figura X muestran un ejemplo de inserción de un documento en la colección “SensoresAGV”.

```
AGV> db.SensorAGV.insertOne({_id: ObjectId("000000000000000000000005"), nombre: "AGV005",  
  tipo_sensor: "ultrasónico", distancia_obstáculo: 0.8, alerta_seguridad: true, fecha_aler  
ta: new Date("2024-03-07T12:45:00")})  
{  
  acknowledged: true,  
  insertedId: ObjectId('000000000000000000000005')  
}
```

```
AGV> db.SensorAGV.insertOne({_id: ObjectId("000000000000000000000002"), nombre: "AGV002",  
  tipo_sensor: "velocidad-aceleracion", velocidad_actual: 0.5, aceleracion_x: 0.1, acelerac  
ion_y: 0.05, aceleracion_z: 0.02, alerta_velocidad: false})  
{  
  acknowledged: true,  
  insertedId: ObjectId('000000000000000000000002')  
}
```

```
AGV> db.SensorAGV.insertOne({_id: ObjectId("00000000000000000000000b"), nombre: "AGV011", tipo_sen  
sor: "temperatura-humedad", temperatura_ambiente: 22.8, humedad_relativa: 37, alerta_temperatura:  
false})  
{  
  acknowledged: true,  
  insertedId: ObjectId('00000000000000000000000b')  
}
```

```
AGV> db.SensorAGV.insertOne({_id: ObjectId("000000000000000000000006"), nombre: "AGV006", tipo_sensor: "vision_artificial", imagen: "data:image/jpeg;base64,  
/9j/4AAQSkZJABAAQABGSI", objetos_detectados: ["caja", "persona"], metadatos: {resolucion: "1920x1080", fecha_captura: new Date(), formato: "jpeg"}, alerta_  
deteccion: false})  
{  
  acknowledged: true,  
  insertedId: ObjectId('000000000000000000000006')  
}
```

AGV> |

La Figura X muestra un ejemplo de inserción de un documento en la colección “OperacionesMantenimientoAGV”.

```
AGV> db.OperacionesMantenimientoAGV.insertOne({_id: ObjectId("000000000000000000000015"), nombre: "AGV021",  
  evento: "Recarga de batería", fecha_hora: new Date()})  
{  
  acknowledged: true,  
  insertedId: ObjectId('000000000000000000000015')  
}
```

AGV> |

Consultas NoSQL

Para realizar consultas dentro de la BBDD que acabamos de crear, tenemos a nuestra disposición una serie de instrucciones.

-Instrucción find(). La instrucción find() nos permite visualizar todos los documentos de una colección.

La Figura X muestra el resultado de ejecutar la instrucción find() sobre la colección “RutasTiemposAGV”.

```

AGV> db.RutasTiemposAGV.find()
[
  {
    _id: ObjectId('00000000000000000000000000000001'),
    nombre: 'AGV001',
    ruta: 'Ruta 1',
    punto_origen: 'Almacén A',
    punto_destino: 'Línea de producción 1',
    tiempo_requerido: 5
  },
  {
    _id: ObjectId('00000000000000000000000000000002'),
    nombre: 'AGV002',
    ruta: 'Ruta 3',
    punto_origen: 'Almacén A',
    punto_destino: 'Línea de producción 3',
    tiempo_requerido: 10
  },
  {
    _id: ObjectId('00000000000000000000000000000003'),
    nombre: 'AGV003',
    ruta: null,
    punto_origen: 'Almacén B',
    punto_destino: null,
    tiempo_requerido: null
  },
  {
    _id: ObjectId('00000000000000000000000000000004'),
    nombre: 'AGV004',
    ruta: 'Ruta 2',
    punto_origen: 'Almacén C',
    punto_destino: 'Línea de producción 2',
    tiempo_requerido: 8
  },
  {
    _id: ObjectId('00000000000000000000000000000005'),

```

-find().sort().Esta instrucción nos permite ordenar los documentos según algún orden establecido. En el caso de la instrucción find().sort({nombre: 1}) estaríamos obteniendo los documentos de la lista en orden alfabético.

La Figura X muestra el resultado de ejecutar la instrucción `find().sort({nombre: -1})` sobre la colección “EstadoDisponibilidadAGV”. Nos devuelve los documentos ordenados en orden contrario al alfabético.

```
AGV> db.EstadoDisponibilidadAGV.find().sort({nombre: -1})
[
  {
    _id: ObjectId('000000000000000000000000000016'),
    nombre: 'AGV021',
    posicion: { latitud: 300.633, longitud: 200.001 },
    direccion: 'Este',
    bateria: 77,
    estado: 'en movimiento',
    disponibilidad: false
  },
  {
    _id: ObjectId('000000000000000000000000000015'),
    nombre: 'AGV020',
    posicion: { latitud: 450.633, longitud: 200.369 },
    direccion: 'Norte',
    bateria: 81,
    estado: 'en movimiento',
    disponibilidad: true
  },
  {
    _id: ObjectId('000000000000000000000000000014'),
    nombre: 'AGV019',
    posicion: { latitud: 269.633, longitud: 530.2 },
    direccion: 'Este',
    bateria: 60,
    estado: 'en espera',
    disponibilidad: true
  },
  {
    _id: ObjectId('000000000000000000000000000013'),
    nombre: 'AGV018',
    posicion: { latitud: 455.369, longitud: 135.745 },
```

-`find().limit()`. Esta instrucción nos permite listar el número de documentos que le indicemos a la función `limit`.

La Figura X muestra el resultado de la instrucción `find().limit(2)`. Obtenemos los dos primeros documentos de la colección “SensoresAGV”.

```
AGV> db.SensorAGV.find().limit(2)
[
  {
    _id: ObjectId('00000000000000000000000000000001'),
    nombre: 'AGV001',
    tipo_sensor: 'ultrasónico',
    'distancia_obstáculo': 1.2,
    alerta_seguridad: true,
    fecha_alerta: ISODate('2024-02-05T12:45:00.000Z')
  },
  {
    _id: ObjectId('00000000000000000000000000000005'),
    nombre: 'AGV005',
    tipo_sensor: 'ultrasónico',
    'distancia_obstáculo': 0.8,
    alerta_seguridad: true,
    fecha_alerta: ISODate('2024-03-07T11:45:00.000Z')
  }
]
AGV> |
```

-find().sort().limit(). Esta instrucción combina todas las instrucciones anteriores y nos permite mostrar el número de documentos que indiquemos en limit() y ordenados según el orden establecido en sort().

La Figura X muestra el resultado de la instrucción find().sort({nombre: 1}).limit(3). Muestra los tres primeros elementos de la colección “OperacionesMantenimientoAGV” ordenados alfabéticamente.


```

AGV> db.OperacionesMantenimientoAGV.find().sort({nombre: 1}).limit(3)
[
  {
    _id: ObjectId('00000000000000000000000000000001'),
    nombre: 'AGV001',
    evento: 'Recarga de batería',
    fecha_hora: ISODate('2024-05-09T10:21:54.486Z')
  },
  {
    _id: ObjectId('00000000000000000000000000000002'),
    nombre: 'AGV002',
    evento: 'Mantenimiento programado',
    fecha_hora: ISODate('2024-05-09T10:22:32.872Z')
  },
  {
    _id: ObjectId('00000000000000000000000000000003'),
    nombre: 'AGV003',
    evento: 'Recarga de batería',
    fecha_hora: ISODate('2024-05-09T10:22:41.849Z')
  }
]
AGV> |

```

A continuación, vamos a realizar una serie de consultas en la base de datos para mostrar nuestro control sobre esta y enseñar lo aprendido en clase de teoría.

Podemos hacer una búsqueda `find({filtro}, {proyección})` para visualizar por ejemplo la batería de todos los AGV que estén dirigiéndose hacia el norte.

```

AGV> db.EstadoDisponibilidadAGV.find({direccion: "Norte"}, {bateria:true})
[
  { _id: ObjectId('00000000000000000000000000000001'), bateria: 75 },
  { _id: ObjectId('00000000000000000000000000000005'), bateria: 67 },
  { _id: ObjectId('0000000000000000000000000000000a'), bateria: 99 },
  { _id: ObjectId('00000000000000000000000000000010'), bateria: 75 },
  { _id: ObjectId('00000000000000000000000000000015'), bateria: 81 }
]
AGV> |

```

Podemos asimismo incluir una opción de filtro en el que, por ejemplo, nos muestre el nombre de los AGV con una distancia a un obstáculo mayor de 1 m.

```

AGV> db.SensorAGV.find({'distancia_obstaculo':{'$gt':1.0}}, {nombre:true, 'distancia_obstaculo':true})
[
  {
    _id: ObjectId('00000000000000000000000000000001'),
    nombre: 'AGV001',
    'distancia_obstaculo': 1.2
  },
  {
    _id: ObjectId('0000000000000000000000000000000a'),
    nombre: 'AGV010',
    'distancia_obstaculo': 2.5
  }
]
AGV>

```

Probemos otra opción de filtro. Por ejemplo, queremos visualizar el nombre y el id de los AGV cuya `punto_origen` esté entre el conjunto [Almacén A, Almacén B y Línea de producción 2]. La Figura X muestra esta operación.

```
AGV> db.RutasTiemposAGV.find({punto_origen:{$in:['Almacén A','Almacén B','Línea de producción 2']}}, {nombre:true, _id:true, punto_origen:true})
[
  {
    _id: ObjectId('000000000000000000000001'),
    nombre: 'AGV001',
    punto_origen: 'Almacén A'
  },
  {
    _id: ObjectId('000000000000000000000002'),
    nombre: 'AGV002',
    punto_origen: 'Almacén A'
  },
  {
    _id: ObjectId('000000000000000000000003'),
    nombre: 'AGV003',
    punto_origen: 'Almacén B'
  },
  {
    _id: ObjectId('000000000000000000000005'),
    nombre: 'AGV005',
    punto_origen: 'Línea de producción 2'
  },
  {
    _id: ObjectId('000000000000000000000007'),
    nombre: 'AGV007',
    punto_origen: 'Almacén B'
  },
  {
    _id: ObjectId('00000000000000000000000c'),
    nombre: 'AGV012',
    punto_origen: 'Almacén A'
  },
  {
    _id: ObjectId('00000000000000000000000e'),
    nombre: 'AGV014',
    punto_origen: 'Almacén A'
  }
]
AGV> |
```

A continuación, vamos a aplicar una operación de filtro para comprobar si en los documentos de la colección `RutasTiempos` encontramos un campo llamado “ruta”.

```
AGV> db.RutasTiemposAGV.find({ruta:{$exists:true}})
[
  {
    _id: ObjectId('000000000000000000000001'),
    nombre: 'AGV001',
    ruta: 'Ruta 1',
    punto_origen: 'Almacén A',
    punto_destino: 'Línea de producción 1',
    tiempo_requerido: 5
  },
  {
    _id: ObjectId('000000000000000000000002'),
    nombre: 'AGV002',
    ruta: 'Ruta 2'
  }
]
```

Podemos comprobar que nos lista los documentos. No obstante, si buscamos el campo llamado “rutaa” (mal escrito), observamos que no devuelve nada.

```
AGV> db.RutasTiemposAGV.find({rutaa:{$exists:true}})
AGV>
```

Ahora vamos a proceder a hacer consultas que devuelvan el número de documentos que cumplen una determinada condición. Por ejemplo, queremos obtener el número de AGV dentro de la colección EstadoDisponibilidad cuya latitud sea mayor que 100. Como observamos en la Figura X, la consulta nos devuelve el número 18. Esta consulta utiliza la función `.countDocuments()`.

```
AGV> db.EstadoDisponibilidadAGV.countDocuments({'posicion.latitud':{'$gt:100.0'}})
18
AGV>
```

A continuación, procedemos a utilizar las operaciones OR y AND. Para la operación OR vamos a buscar todos los AGV cuya ruta sea “Ruta 1” o “Ruta 3”.

```
AGV> db.RutasTiemposAGV.find({'$or': [{'ruta': 'Ruta 1'}, {'ruta': 'Ruta 3'}]})
[
  {
    _id: ObjectId('000000000000000000000001'),
    nombre: 'AGV001',
    ruta: 'Ruta 1',
    punto_origen: 'Almacén A',
    punto_destino: 'Línea de producción 1',
    tiempo_requerido: 5
  },
  {
    _id: ObjectId('000000000000000000000002'),
    nombre: 'AGV002',
    ruta: 'Ruta 3',
    punto_origen: 'Almacén A',
    punto_destino: 'Línea de producción 3',
    tiempo_requerido: 10
  }
]
```

Para la operación AND vamos a obtener los AGV cuyo punto de origen sea “Almacén B” y cuyo punto de destino sea “Línea de producción 2”.

```
AGV> db.RutasTiemposAGV.find({'$and': [{'punto_origen': 'Almacén A'}, {'punto_destino': 'Línea de producción 2'}]})
[
  {
    _id: ObjectId('00000000000000000000000e'),
    nombre: 'AGV014',
    ruta: 'Ruta 11',
    punto_origen: 'Almacén A',
    punto_destino: 'Línea de producción 2',
    tiempo_requerido: 7
  }
]
AGV>
```

Para la siguiente consulta, nos ponemos en la situación en la que quisiéramos saber todos los puntos de origen que puede tener un AGV. No obstante, no queremos que nos muestre todos los vehículos junto con sus puntos de origen, sino que nos muestre un bloque por cada punto de origen diferente que haya. Para esto utilizaremos la función `Distinct`.

```

AGV> db.RutasTiemposAGV.distinct("punto_origen")
[
  'Almacén A',
  'Almacén B',
  'Almacén C',
  'Línea de producción 1',
  'Línea de producción 2',
  'Línea de producción 3',
  'Línea de producción A',
  'Zona de carga 1',
  'Zona de carga 2',
  'Zona de carga 3'
]
AGV>

```

A continuación, vamos a realizar la operación de modificación de un documento. Procedemos a modificar el AGV 1 para que su punto de origen sea el “Almacén B” en vez del “Almacén A”.

```

AGV> db.RutasTiemposAGV.updateOne({nombre: 'AGV001'},{$set: {punto_origen:'Almacén B'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
AGV>

```

Ahora vamos a modificar todos los AGV en los que el punto de origen es “Almacén A” para que su tiempo requerido sea 6.

```

AGV> db.RutasTiemposAGV.updateMany({punto_origen: 'Almacén A'},{$set: {tiempo_requerido: 6}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
AGV>

```

Finalmente, vamos a probar a realizar una operación de borrado de un documento de la colección RutasTiempos, en concreto el AGV número 15.

```

AGV> db.RutasTiemposAGV.deleteOne({nombre:'AGV015'})
{ acknowledged: true, deletedCount: 1 }
AGV>

```

Comprobamos que, efectivamente, se ha eliminado el documento equivalente al AGV 15.

```
{
  _id: ObjectId('0000000000000000000000000000e'),
  nombre: 'AGV014',
  ruta: 'Ruta 11',
  punto_origen: 'Almacén A',
  punto_destino: 'Línea de producción 2',
  tiempo_requerido: 6
},
{
  _id: ObjectId('000000000000000000000000000010'),
  nombre: 'AGV016',
  ruta: 'Ruta 12',
  punto_origen: 'Línea de producción 3',
  punto_destino: 'Zona de carga 1',
  tiempo_requerido: 9
},
```

Para concluir, en este apartado hemos sido capaces de crear una base de datos no relacional en la que almacenar todos los datos relacionados a los vehículos de guiado automático. Hemos podido ejecutar una amplia gama de operaciones sobre la base de datos como la creación de colecciones, inserción de documentos, modificación o eliminación de documentos, y operaciones de búsqueda dentro de la base de datos. Ahora estamos preparados para implementar bases de datos no relacionales para nuestro futuro profesional.