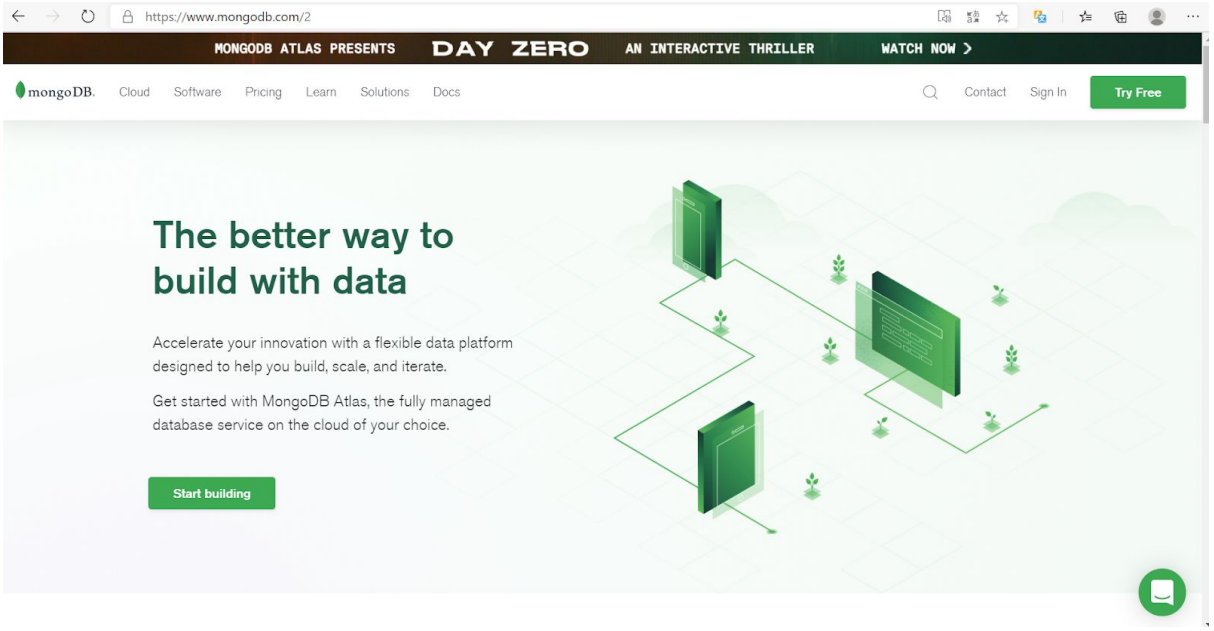
SEMINARIO 4

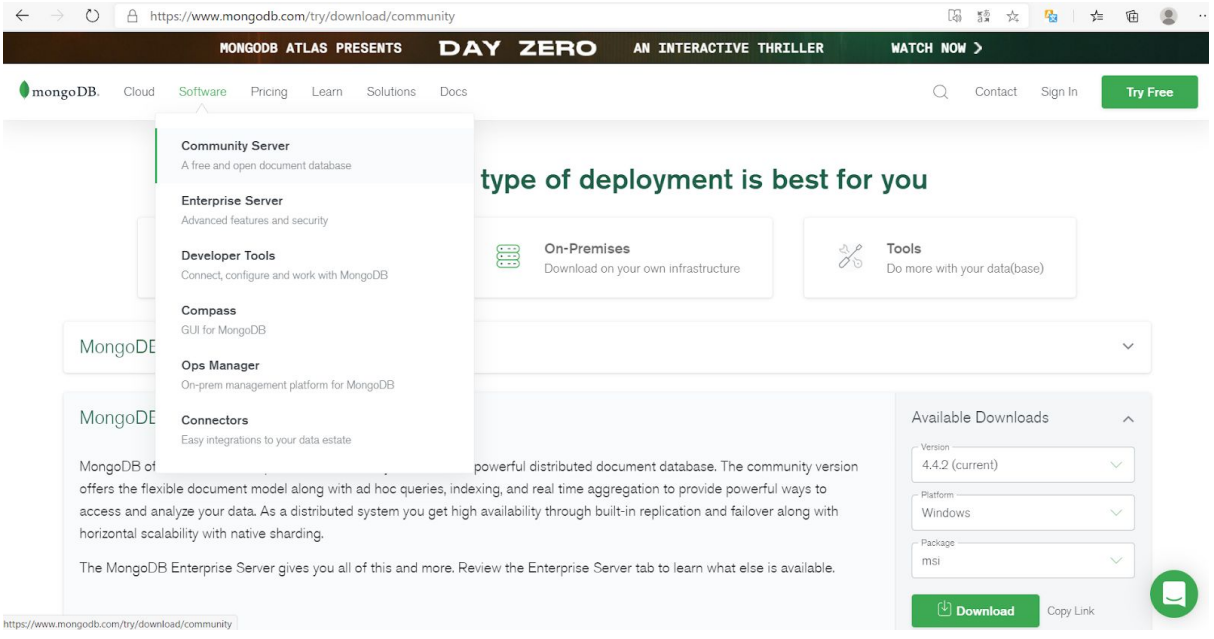
ANTONIO CARLOS MARTINEZ GARCIA
JAVIER RAMIREZ PULIDO
PABLO NUÑEZ TEJERO
PEDRO PADILLA REYES
SANTIAGO PADILLA ALVAREZ

Breve descripción de la descarga e instalación del SGBD .

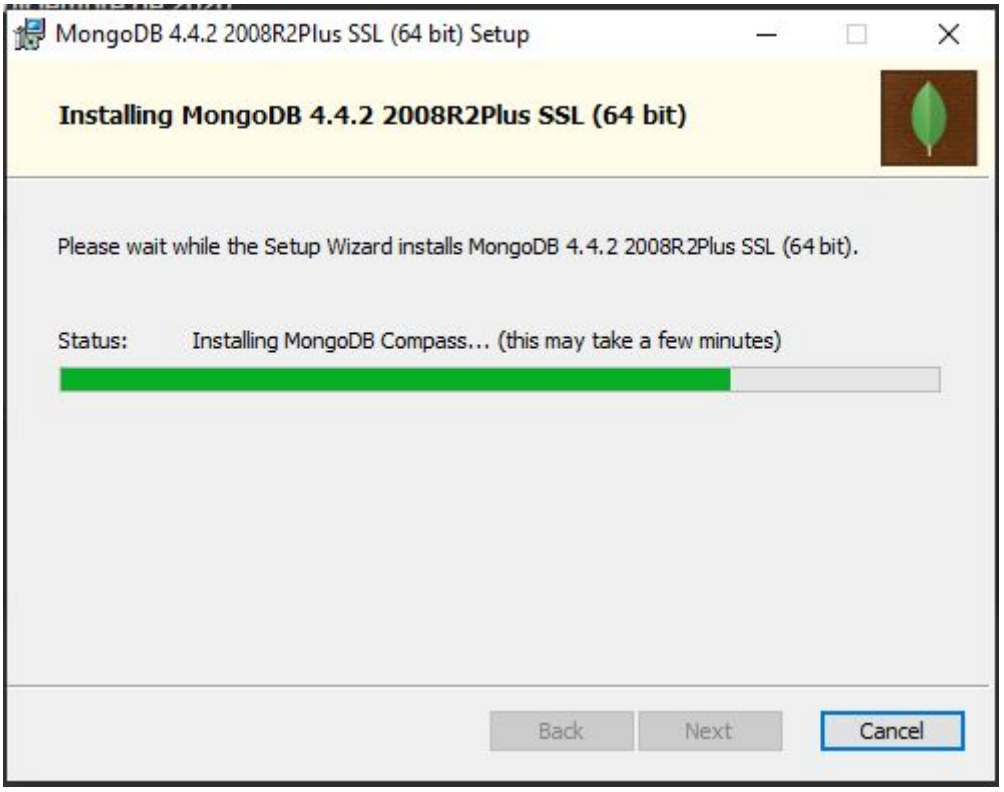
Para la realización del seminario hemos utilizado MongoDB.



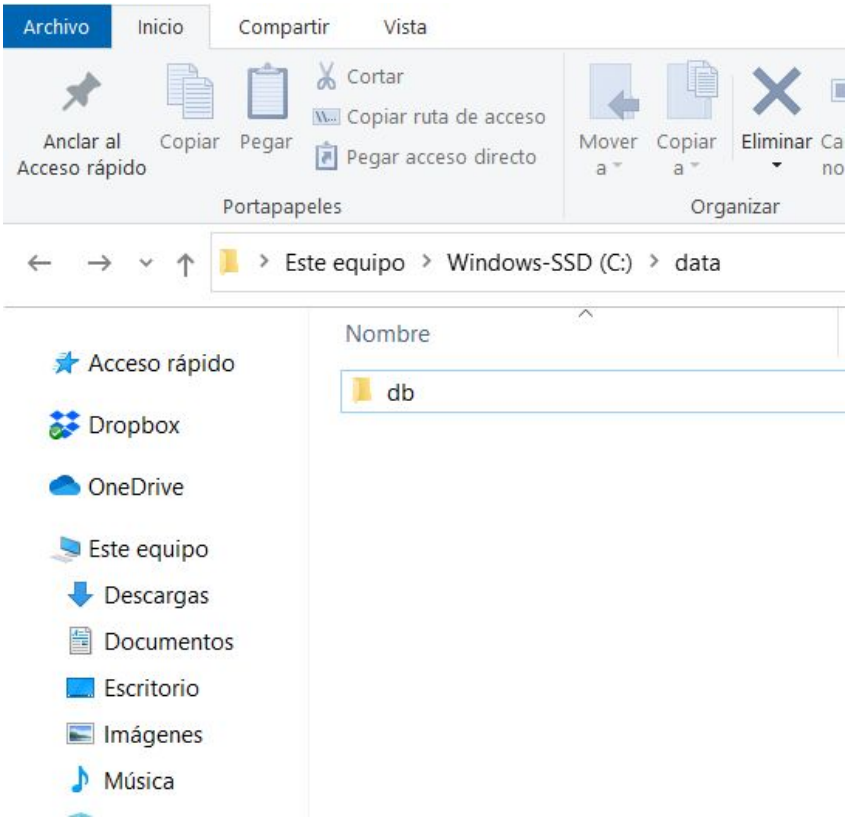
Descargamos la última versión de community server 4.4.2: Software -> Community Server



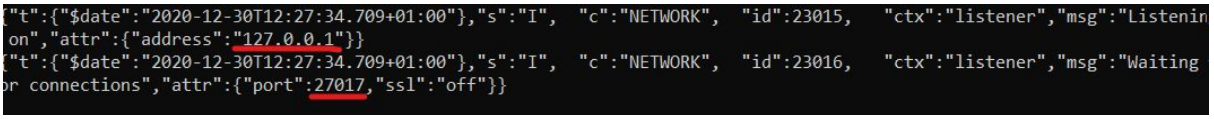
Abrimos el ejecutable y terminamos con la instalación.



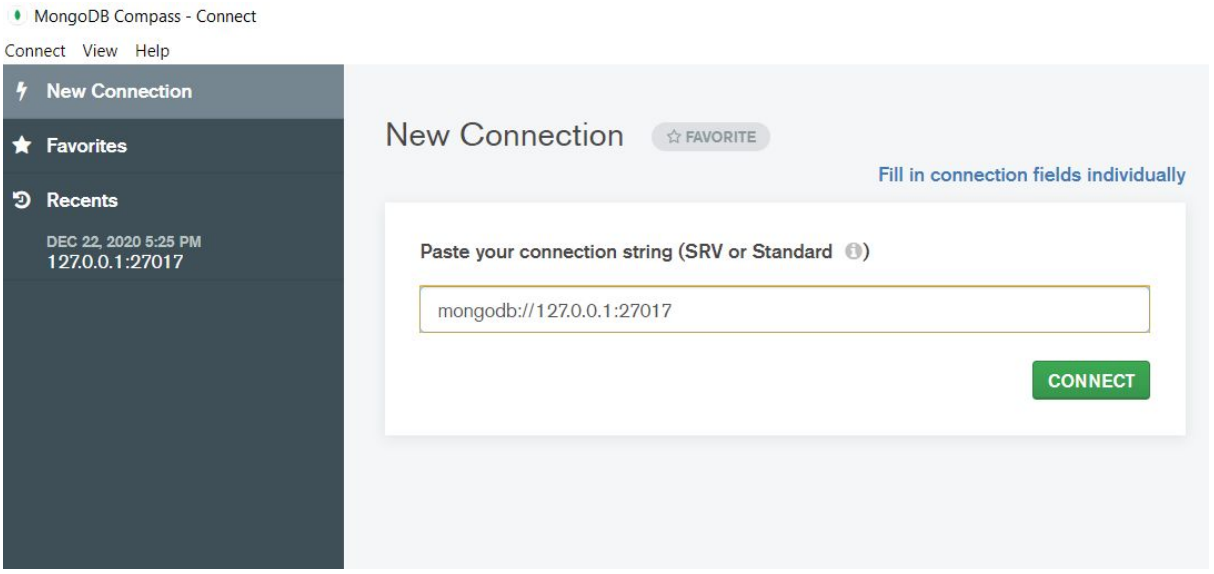
Una vez instalado creamos el directorio db dentro de otro nuevo directorio data.



Dentro del directorio bin donde se encuentran los ejecutables, ejecutamos mongod para ver la ip y el puerto.



Abrimos MongoDB y creamos la conexión.



Breve descripción del DDL y DML utilizado (no se necesita la especificación completa de la sintaxis, sólo nombrar los comandos utilizados y para qué sirven).

En MongoDB tenemos una estructura dinámica. Se pueden crear tablas sin especificar los campos, e incluso se puede crear una tabla sobre la marcha realizando una inserción.

1) Crear tablas:

MongoDB => ***db.createCollection("users")***

Equivalente en SQL:

create table users(user_id varchar2(10),name varchar2(10),age number)

2) Insertar datos en tablas (también sirve para crear la tabla sobre la marcha)**:**

MongoDB => ***db.users.insert({user_id:"ponmeun10",name:"pedro",age:20})***

Equivalente en SQL: Insert into users values("***ponmeun10***","***pedro***",***20***)

3) Update:

MongoDB => ***db.users.update({ },***

{ $set: { join_date: new Date() } },

{ multi: true })

4) Hacer drop:

A una columna:

MongoDB => ***db.users.update({ },***

{ $unset: { join_date: new Date() } },

{ multi: true })

A una tabla:

db.users.drop()

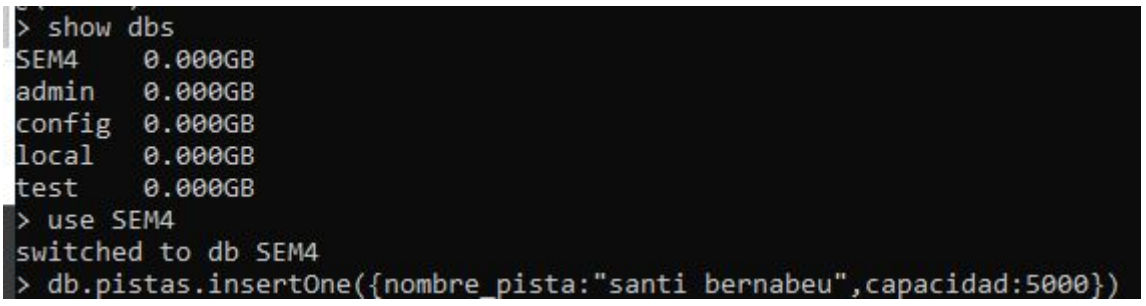
5) Seleccionar datos de una tabla (equivalente a select en sql):

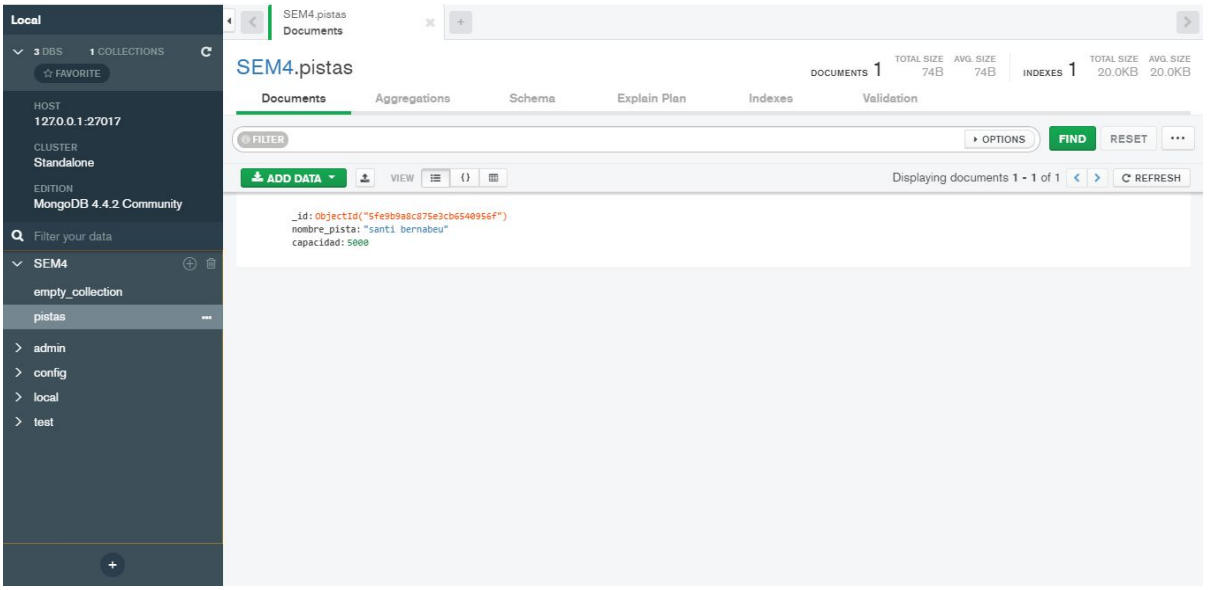
MongoDB => ***db.users.find({ age: { $ne: 20} })***

6) Borrar datos de una tabla:

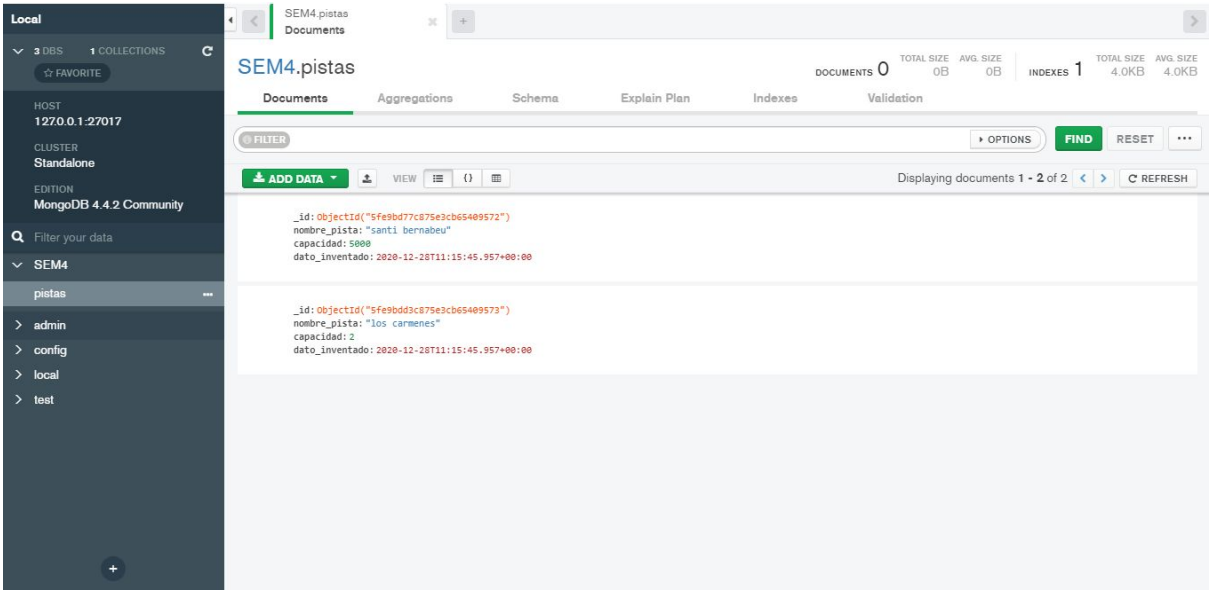
db.users.deleteMany({ age: 20})

Sentencias empleadas para la creación de estructuras, inserción/modificación/borrado de datos, y consultas.



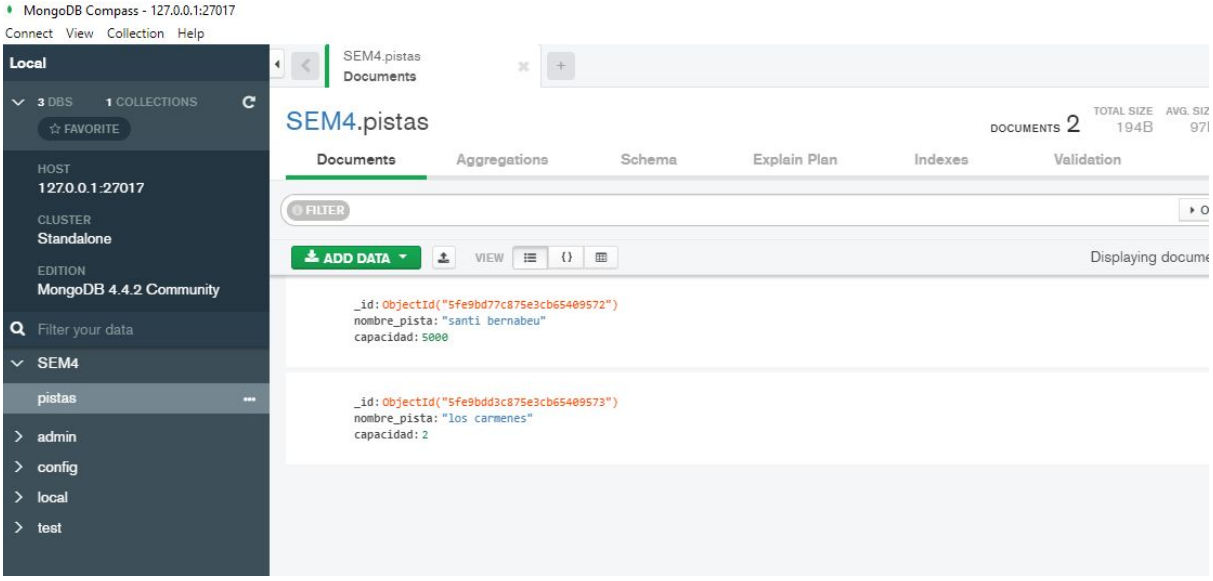



```
> db.pistas.update({},{$set:{dato_inventado: new Date()}},{multi:true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
>
```




```
> db.edicion.drop()
true
```

```
> db.pistas.update({},{$unset:{dato_inventado: new Date()}},{multi:true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
>
```

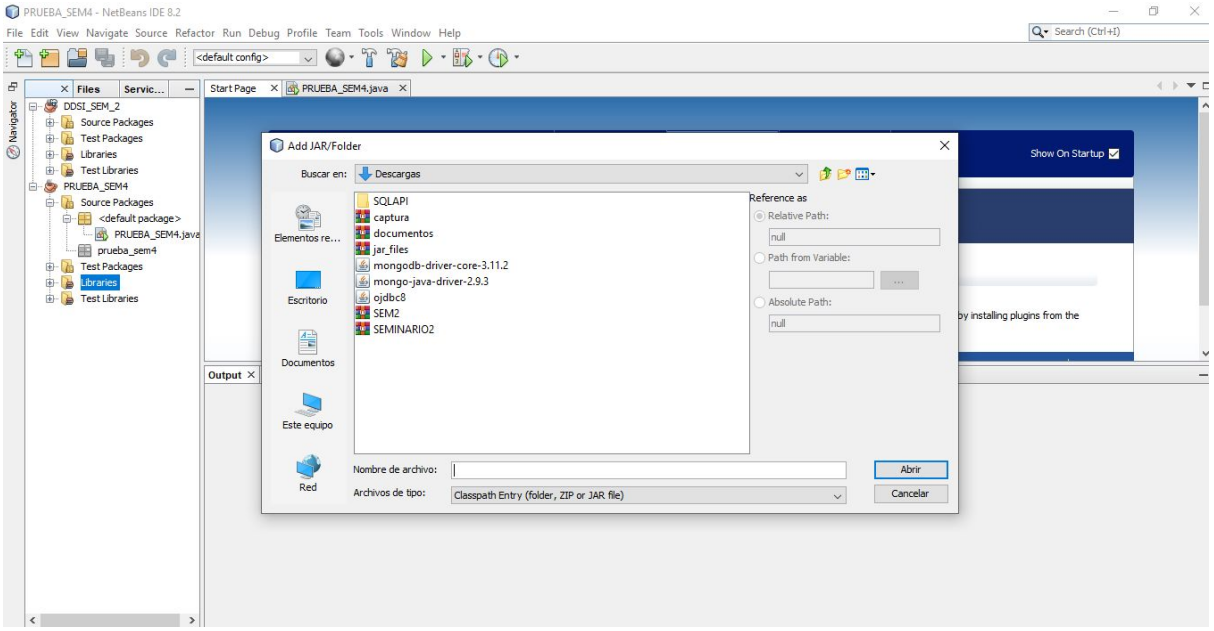



```
> db.pistas.find({nombre_pista:"los carmenes"})
{ "_id" : ObjectId("5fe9bdd3c875e3cb65409573"), "nombre_pista" : "los carmenes", "capacidad" : 2 }
>
```

```
> db.pistas.deleteMany({nombre_pista:"los carmenes"})
{ "acknowledged" : true, "deletedCount" : 1 }
>
```

Breve descripción del mecanismo de conexión al SGBD desde una aplicación.

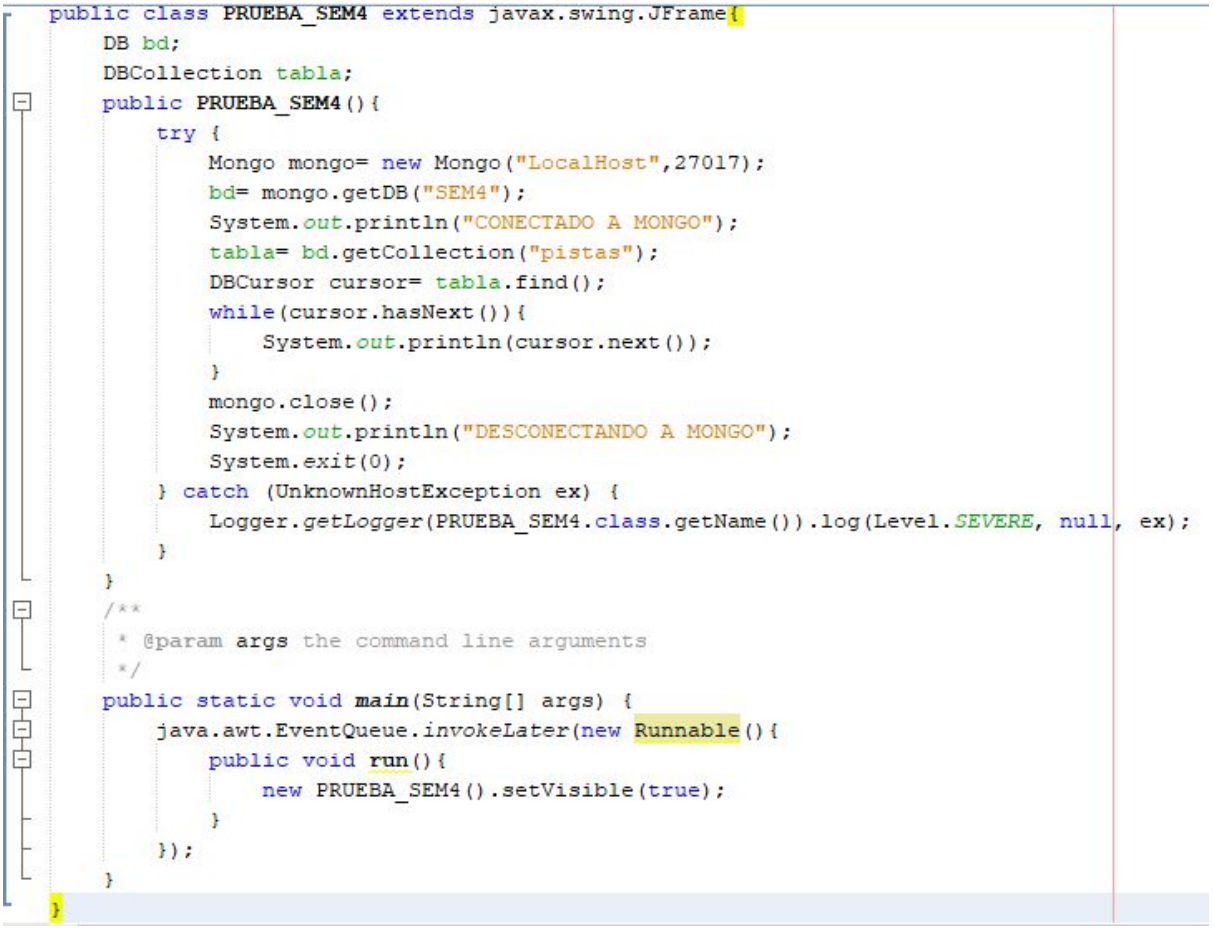
Para conectarnos a una base de datos de mongodb desde java hay que descargar el driver mongo-java-driver-2.9.3 y añadirlo a la librería.



Según vamos escribiendo tenemos que ir importando algunas librerias (NetBeans nos avisa de ello)

```
package prueba_sem4;

import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.Mongo;
import java.net.UnknownHostException;
import java.util.logging.Level;
import java.util.logging.Logger;
```



```
public class PRUEBA_SEM4 extends javax.swing.JFrame{
    DB bd;
    DBCollection tabla;
    public PRUEBA_SEM4(){
        try {
            Mongo mongo= new Mongo("LocalHost",27017);
            bd= mongo.getDB("SEM4");
            System.out.println("CONECTADO A MONGO");
            tabla= bd.getCollection("pistas");
            DBCursor cursor= tabla.find();
            while(cursor.hasNext()){
                System.out.println(cursor.next());
            }
            mongo.close();
            System.out.println("DESCONECTANDO A MONGO");
            System.exit(0);
        } catch (UnknownHostException ex) {
            Logger.getLogger(PRUEBA_SEM4.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        java.awt.EventQueue.invokeLater(new Runnable(){
            public void run(){
                new PRUEBA_SEM4().setVisible(true);
            }
        });
    }
}
```

Posteriormente creamos un objeto Mongo, e indicamos que nos conectaremos desde nuestra IP por el puerto 27017.

Después ya podemos conectarnos a la base de datos.

Breve discusión sobre si sería adecuado para implementar el SI de la práctica.

Comparado a SQL es cierto que MongoDB es más fácil y directo, pero está más orientado a sistemas más inestables y sin administración de la BD, o para la nube, fragmentación, etc.

Pero por la manera de implementarlo SQL es más fiable y más controlado, ya que, por ejemplo, al crear tablas en sql hay que especificar la estructura y en MongoDB no hace falta.

En los documentos de MongoDB no se crean relaciones con otra tabla, sino que se especifica de forma anidada. Esto puede provocar problemas de consistencia de datos. Además, MongoDB usa un lenguaje de consultas no estructurado y los usuarios de un nodo deben esperar a que los otros nodos se sincronicen para poder ser visibles y editables. Todo esto hace de MongoDB una solución poco adecuada para aplicaciones con transacciones complejas.

Por otra parte, MySQL soporta transacciones atómicas y el uso de JOIN, útiles para nuestro trabajo. Es por ello que MySQL es la mejor opción para un sistema que no crecerá mucho y mantendrá el mismo formato de los datos, como es el nuestro.

Al ser nuestro sistema uno con estructura fija y esquema definido, MongoDB no es la mejor opción.