PARA LOS ALUMNOS Y ALUMNAS CON SOLAMENTE EL SEGUNDO PARCIAL:
1. Ejercicio de SpringMVC
2. Consultas MongoDB (sin Java)
3. Ejercicios de API REST

PARA LOS ALUMNOS Y ALUMNAS CON TODO EL CURSO:
1. Ejercicio de ficheros en Java
2. Ejercicio JDBC en Java
+ los puntos anteriores (aunque con menos ejercicios).

UNIDAD 05 SPRINGMVC:

---

https://github.com/j5anchez/AC_UD05TaEvP1

https://github.com/j5anchez/AC_UD05TaEvP2

```
package eus.birt.dam;


import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication

public class Ud4TareaEvaluativaSpringApplication {

	public static void main(String[] args) {

		SpringApplication.run(Ud4TareaEvaluativaSpringApplication.class, args);

	}

}
```

INSTALAR OPENJDK EN WINDOWS

1. Descarga los binarios de la página oficial de OpenJDK:

- Pagina principal: https://openjdk.java.net/
- Binario: https://jdk.java.net/11/

2. Se descomprime el archivo zip descargado.

- Clic derecho sobre el archivo y Extraer todo.
- Seleccionamos la ubicación donde queremos que quede los binarios del JDK. En mi caso voy a utilizar: c:\\java.

- Se ha creado una carpeta llamada jdk-11.0.2. Debemos recordar esta ruta porque la configuraremos en las variables de entorno.

3. Creación de las variables de entorno.

Se debe crear la variable de entorno JAVA_HOME y se debe actualizar la variable de entorno PATH. La primera se utilizará para que java sepa dónde se encuentra la instalación del JDK y la seguna es para poder ejecutar los comandos de java (como javac, java, etc) desde cualquier lugar.

3.1. JAVA_HOME

- En la lupa de la barra de windows escribiremos: Editar las variables de entorno del sistema y damos clic en la opción que nos da.
- Se abre la ventana de Propiedades del sistema y en la pestaña de Opciones avanzadas seleccionamos Variables de entorno.
- Podemos crear las variables de entorno para el usuario actual o para el sistema. Yo sugiero que lo hagan para el sistema, con el fin de que cualquier usuario tenga acceso a la ejecución de java.
- En el grupo de Variables del sistema seleccionamos el botón de Nueva...
- En el nombre escribiremos: JAVA_HOME
- En el valor escribiremos la ruta donde se descomprimió el jdk, en mi caso será: c:\\java\\jdk-11.0.2.
- Clic en aceptar. Ya debe aparecer esta variable en el grupo.

3.2. PATH

- Si no tienes abierta la ventana de Variables de entorno abrela como se muestra en los dos primeros items del punto anterior.
- En el grupo Variables del sistema ubica la variable Path, seleccionala y da clic en Editar...
- Podrás ver todos los valores de Path. No los modifiques o elimines!.
- Agrega un nuevo valor dando clic en el botón Nuevo y digitando la ruta de la instalación de java agregando la carpeta bin. En mi caso es: c:\\java\\jdk-11.0.2\\bin.
- Das clic en aceptar, luego en aceptar y luego nuevamente en aceptar.

Ahora ya puedes abrir una consola dando clic en la lupa de la barra de windows, escribes cmd y seleccionas simbolo del sistema. Cuando te aparezca, ya puedes confirmar que la instalación ha quedado correcta digitando:

```
1java -version

2// La salida debe ser algo como esto:

3// openjdk version "11.0.2" 2018-10-16

4// OpenJDK Runtime Environment 18.9 (build 11.0.2+7)

5// OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+7, mixed mode)
```

Descargar lombok.jar y meter en la carpeta de eclipse con cmd java –jar lombok.jar y especificar carpeta eclipse

Si da error de version java:

Proyecto->build path->configure build path

Elegir java se 17 jdk 17.

Maven-update (o no, primero esperar).

Run as- spring boot app para ejecutar.

INICIALIZAR SPRINGMVC CONTROLADOR

New->-other->spring boot->spring starter project

Añadir spring boot devtools, spring web y thymeleaf

Group:eus.birt.dam

Artifact: Nombre

Package: eus.birt.dam

OK

Crear paquete eus.birt.dam.Controller y dentro la clase IndexController. Escribirmos @Controller antes de public class e importar dependencia (ctrl+mayus+o).

```
package eus.birt.dam.Controller;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;


@Controller

public class IndexController {


	@GetMapping({"/",""})

	public String holaMundo()

	{return "index";

	}

	@GetMapping("/welcome")

	public String miHelloWorld(Model model)

	{

		model.addAttribute("titulo","hello world desde modelo");


		return "index2";

	}

}
```

Guardamos y va a mirar en resources/templates a ver si tenemos index.html, lo creamos.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>insert title here</title>
</head>
<body>
<h1>holamundo</h1>
</body>
</html>
```

Cambiamos el charset a utf8 por las tildes (mirar en propiedades del html si esta cambiado).

Para verlo ejecutado boton derecho en aplication (principal)->run as-> spring boot app
Dentro de la clase:
En indexController poner @GetMapping({"/",""}) /*se puede quitar las segundas comillas si falla*/ o poner @RequestMapping (value="/", method=RequestMethod.GET) es lo mismo

Fuera de la clase:
Con @RequestMapping("/clientes") debajo de @Controller el index deberia estar en la carpeta clientes.
Si usamos model.addatribute modificar el html con <html xmlns:th=http://www.thymeleaf.org> y el texto <h1 th:text="${titulo}">bla bla bla</h1>

Añadir starter spring data jpa, h2, Lombok

Carpeta maven dependencias si no tira: c:/usuarios/xxx/.m2
Mirar lombok si falla
Mysql conector
Jdbc conector

MONTAR BD EN MEMORIA

En application.properties:

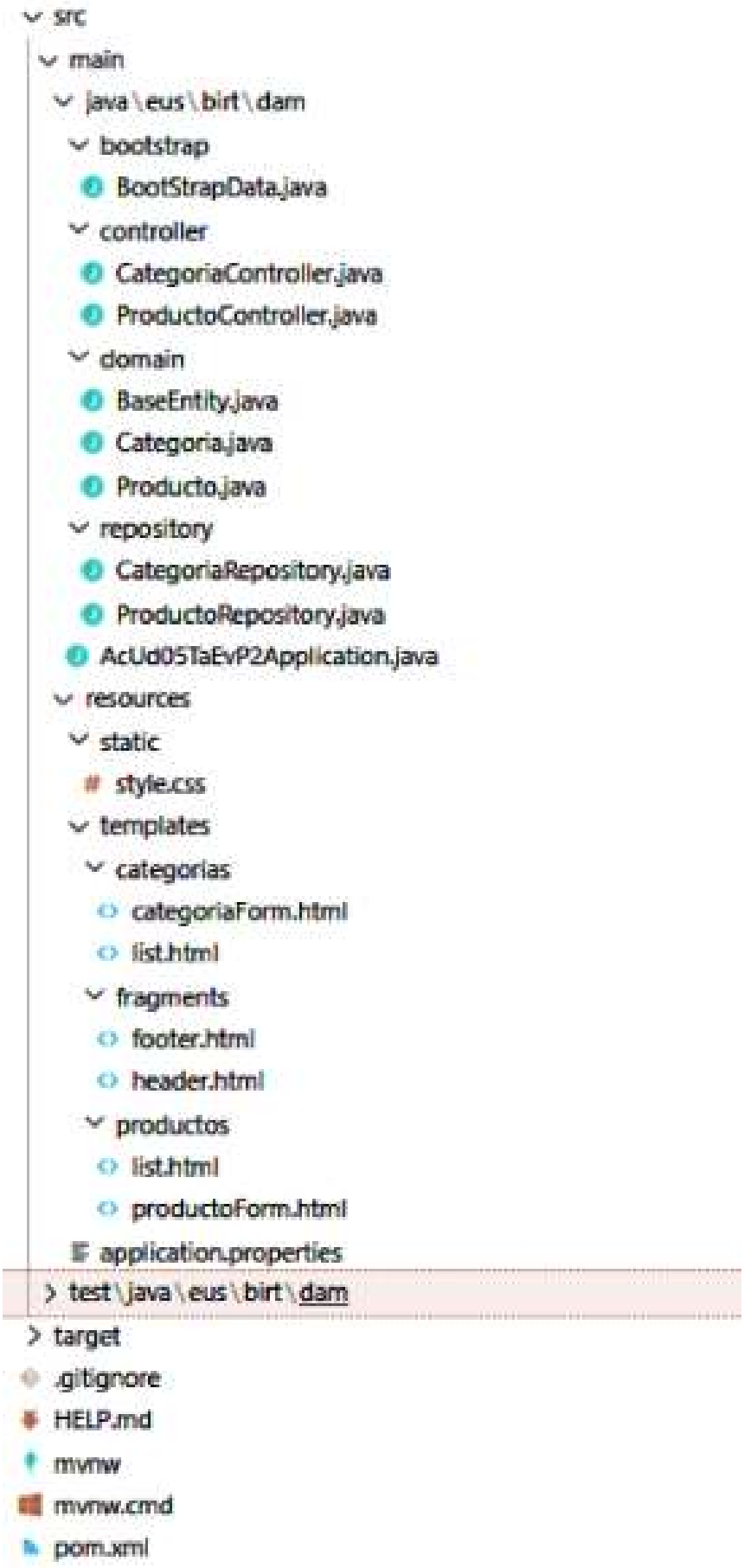
```
spring.jpa.show-sql=true
spring.h2.console.enabled=true
spring.datasource.url=jdbc:h2:mem:testdb
server.port 8085
```

En pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"

    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.6.4</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>eus.birt.dam</groupId>

    <artifactId>AC-UD05_TaEv_P2</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>AC-UD05_TaEv_P2</name>

    <description>Demo project for Spring Boot</description>

    <properties>

        <java.version>17</java.version>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-thymeleaf</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>


        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-devtools</artifactId>

            <scope>runtime</scope>

            <optional>true</optional>

        </dependency>
```

```
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Estructura:


src
main
java\eus\birt\dam
bootstrap
BootStrapData.java
controller
CategoriaController.java
ProductoController.java
domain
BaseEntity.java
Categoria.java
Producto.java
repository
CategoriaRepository.java
ProductoRepository.java
AcUd05TaEvP2Application.java
resources
static
style.css
templates
categorias
categoriaForm.html
list.html
fragments
footer.html
header.html
productos
list.html
productoForm.html
application.properties
test\java\eus\birt\dam
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml

Si falla, crear nuevo spring boot y poner en pom:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.0</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>eus.birt.app</groupId>
    <artifactId>ddd</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>app</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

STARTERS SPRINGTOOLS

SPRING BOOT

NEW->SPRING STARTER PROYECT(MAVEN PROYECT, JAR, JAVA, JDK11)

☑ SPRING WEB

SPRING WEB MVC CONTROLLERS

☑ SPRING WEB

☑ SPRING BOOT DEVTOOL

☑ THYMELEAF

SPRING WEB MVC MODEL

☑ SPRING WEB

☑ SPRING BOOT DEVTOOL

☑ THYMELEAF

☑ H2 DATABASE

☑ SPRING DATA JPA

Dentro del package Repository->Crear Interface en vez de clase

Localhost:8085/h2-console

☑ LOMBOK (Elimina getters, setters etc)

Instalar en Eclipe desde su página con instalador y añadir como starter

---

UNIDAD 06 MONGODB

Levantar servidor:

Entrar por cmd a C:\Program Files\MongoDB\Server\5.0\bin

Ejecutar mongod

MONGOSHELL crear c:/data/db si no tira

Conectar

mongodb://localhost:27017/?readPreference=primary&appname=MongoDB%20Compass&directConnection=true&ssl=false

Si no conecta localhost entrar en

mongodb+srv://joseba:pechustr0@cluster0.gygmn.mongodb.net/test?authSource=admin&replicaSet=atlas-13yuj3-shard-0&readPreference=primary&appname=MongoDB%20Compass&directConnection=true&ssl=true

Consulta desde Java (Conector-> mongo-java-driver-3.12.10.jar):

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import static com.mongodb.client.model.Projections.excludeId;
import static com.mongodb.client.model.Projections.fields;
import static com.mongodb.client.model.Projections.include;
import org.bson.Document;
import org.bson.conversions.Bson;



public class Conexion_mongo {
    public static void main(String[] args) {

MongoClient mongoClient = new MongoClient();
MongoDatabase db = mongoClient.getDatabase("moviedata");
MongoCollection<Document> movies = db.getCollection("movies");



Bson filter = Filters.and(Filters.regex("name", "the"), Filters.ne("genres", "Comedy"));
filter.toString();
for (Document query :
movies.find(filter).projection(fields(include("name","genres"),excludeId())))
{System.out.println(query);}


    }
}
```

EJEMPLO DE UN DOCUMENTO

_ID:628D09368A8715E92CB9D810

ID:1

URL:"HTTP://WWW.TVMAZE.COM/SHOWS/1/UNDER-THE-DOME"

NAME:"UNDER THE DOME"

TYPE:"SCRIPTED"

LANGUAGE:"English"

genres:Array

0:"Drama"

1:"Science-Fiction"

2:"Thriller"

status:"Ended"

runtime:60

```
premiered:"2013-06-24"
officialSite:"http://www.cbs.com/shows/under-the-dome/"
schedule:Object
        time:"22:00"
        days:Array
                0:"Thursday"
rating:Object
        average:6.5
weight:91
network:Object
        id:2
        name:"CBS"
        country:Object
                name:"United States"
                code:"US"
                timezone:"America/New_York"
webChannel:null
externals:Object
        tvrage:25988
        thetvdb:264492
        imdb:"tt1553656"
image:Object
        medium:"http://static.tvmaze.com/uploads/images/medium_portrait/0/1.jpg"
        original:"http://static.tvmaze.com/uploads/images/original_untouched/0/1.jpg"
summary:"<p><b>Under the Dome</b> is the story of a small town that is suddenly..."
updated:1529612668
_links:Object
        self:Object
                href:"http://api.tvmaze.com/shows/1"
        previousepisode:Object
                href:"http://api.tvmaze.com/episodes/185054"
```

La sentencia @Override hace uso de la sintaxis de comentarios en Java. Esta oración le permite al compilador de Java saber que deseas anular un método existente de una clase primaria (es decir, sobrescribirá cómo funciona una función en esta clase mientras exista en la clase primaria).

La manera más común de tratar las transacciones en Spring es mediante la anotación @Transactional en la cabecera del método de una clase (nunca un interfaz) gestionada por Spring: @Transactional public void hazAlgoTransaccionalmente() { // Soy transaccional! }

TAREA COMPLETA

a. Escribe una consulta que muestre el nombre, género, y la duración (runtime) de las primeras 10 series ordenados por runtime y nombre y muéstralo de forma indentada.

```
db.movies.find(
{},
{name:1,genres:1,runtime:1,_id:0})
.sort({runtime:1,name:1}).limit(10).pretty();
```

b. Escribe una consulta que muestre el nombre, el género y la puntuación de las series del género “Fantasy” (no exclusivamente) y tienen una puntuación superior a 7.

```
db.movies.find(
{genres:"Fantasy",rating:{$gt:{"average":7}}},
{name:1,genres:1,rating:1,_id:0})
```

c. Escribe una consulta que muestre el nombre, el género y la puntuación de las series que tienen una puntuación entre 9 y 9.5 (ambos incluidos).

```
db.movies.find(
{$and:[{rating:{$gte:{"average":9}}},{rating:{$lte:{"average":9.5}}}]},
{name:1,genres:1,rating:1,_id:0})
```

d. Escribe una consulta que muestre el nombre y el género de las series que tienen como género exclusivamente Drama y Comedy.

```
db.movies.find(
{genres: [ "Drama", "Comedy" ]  },
{name:1,genres:1,_id:0})
```

e. Escribe una consulta que muestre el nombre y el género de las series cuyo nombre contiene la cadena "the" y no es del género Comedy

```
db.movies.find(
{name:/the/,genres:{$nin:["Comedy"]}},
{name:1,genres:1,_id:0})
```

f. Escribe una consulta que muestre el nombre y la fecha de estreno (premiered) de las series cuya fecha de estreno es posterior a 2014-10-01.

```
db.movies.find(
{premiered:{$gt : "2014-10-01"}},
{name:1,premiered:1,_id:0})
```

g. Escribe una consulta que muestre el nombre, la fecha de estreno y el idioma (language) de las series cuya fecha de estreno se es el año 2006 y que no sean de lengua inglesa.

```
db.movies.find(
{premiered:{$gte : "2006-01-01",$lte:"2006-12-31"},language:{$not:/English/}},
{name:1,premiered:1,language:1,_id:0})
```

h. Escribe una consulta que muestre el nombre y el idioma (language) de las series cuya genero no está definido, ordenado por nombre.

```
db.movies.find(
{genres : {$exists:true, $size:0}},
{_id:0,name:1,language:1}).sort({name:1})
```

i. Escribe una consulta que muestre el nombre, y la duración (runtime) de las series que duran 30 minutos o múltiplos de 30, ordenados de menor a mayor duración y por nombre.

```
db.movies.find(
{"runtime":{$mod:[30,0]}},
{_id:0,name:1,runtime:1}).sort({runtime:1,name:1})
```

ESTRUCTURA

```
_id:628d09368a8715e92cb9d810
id:1
url:"http://www.tvmaze.com/shows/1/under-the-dome"
name:"Under the Dome"
type:"Scripted"
language:"English"
genres:Array
```

0:"Drama"
1:"Science-Fiction"
2:"Thriller"
status:"Ended"
runtime:60
premiered:"2013-06-24"
officialSite:"http://www.cbs.com/shows/under-the-dome/"
schedule:Object
time:"22:00"
days:Array
0:"Thursday"
rating:Object
average:6.5
weight:91
network:Object
id:2
name:"CBS"
country:Object
name:"United States"
code:"US"
timezone:"America/New_York"
webChannel:null
externals:Object
tvrage:25988
thetvdb:264492
imdb:"tt1553656"
image:Object
medium:"http://static.tvmaze.com/uploads/images/medium_portrait/0/1.jpg"
original:"http://static.tvmaze.com/uploads/images/original_untouched/0/1.jpg"
summary:"<p>Under the Dome is the story of a small town that is suddenly..."
updated:1529612668
_links:Object
self:Object
href:"http://api.tvmaze.com/shows/1"
previousepisode:Object
href:"http://api.tvmaze.com/episodes/185054"

BUSCAR OBJETO DENTRO DE ARRAY

```
db.restaurants.find({
                $and:[{"grades.score":46},{"grades.grade":"A"}]},
                {"restaurant_id":1,"name":1,_id:0}).sort("name")
```

POSICION ESPECIFICA DE ARRAY DENTRO DE OBJETO

```
db.restaurants.find({
              $and:[{"address.coord.1":{$gte:46}},{"address.coord.1":{$lte:52}}]},
              {"restaurant_id":1,"name":1,_id:0}).sort("name")
```

CRUD GEOLOCALIZACION

Realiza las siguientes operaciones de insertar, actualizar y borrar en una BD nueva llamada birtlhData: (2 puntos)

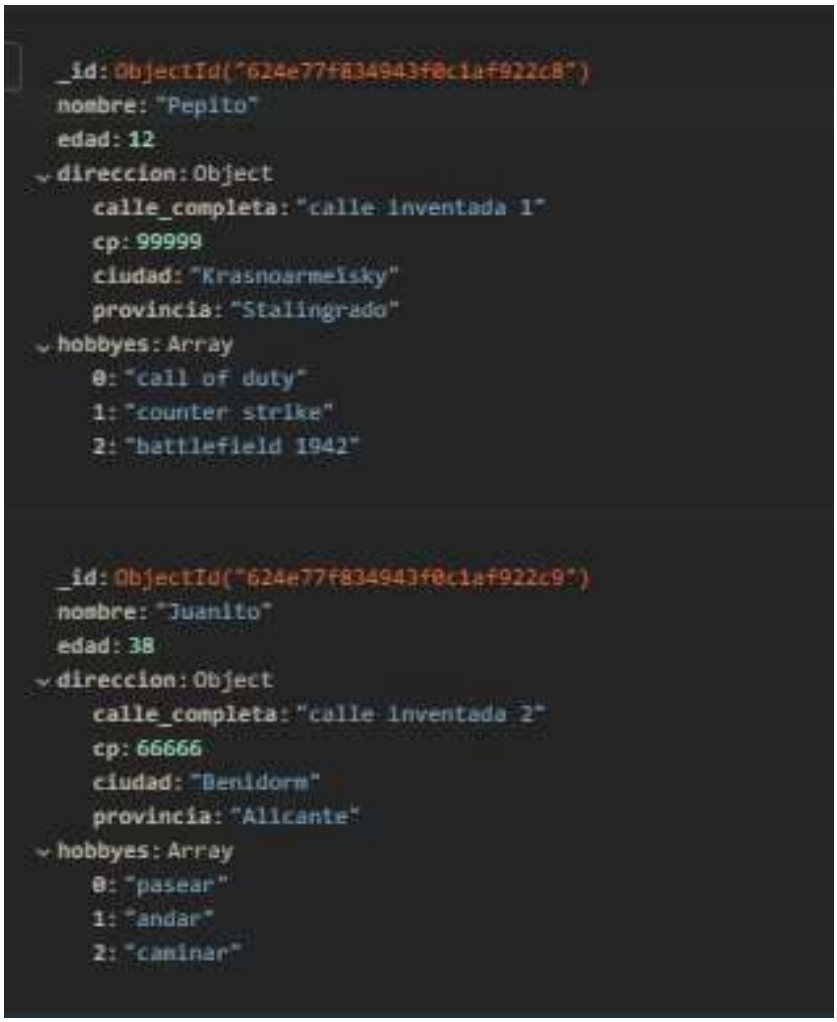
CREACIÓN → insertOne(), insertMany().

j. Introduce en la colección alumnos un nuevo alumno con los campos nombre, edad, dirección. Dirección será a su vez un nuevo documento (nested document) que contendrá los siguientes datos: calle, CP, ciudad, provincia.

```
db.alumnos.insertOne({_id:"alumno001",nombre:"Joseba",edad:34,direccion:{ calle: "maitena", CP: 20500, ciudad: "Arrasate",provincia:"Gipuzkoa" }})
```

k. Introduce en la colección alumnos dos nuevos alumnos a la vez con los campos nombre, edad, dirección. Dirección será a su vez un nuevo documento (nested document) que contendrá los siguientes datos: calle completa, CP, ciudad, provincia. De estos dos nuevos alumnos guardaremos también sus hobbies en un array con el mismo nombre.

```
db.alumnos.insertMany(
[
{
nombre:"Pepito",edad:12,
direccion:{calle_completa:"calle_inventada_1",cp:99999,ciudad:"Krasnoarmeïsky",provincia:"Stalingrado"},
hobbyes:["call of duty","counter strike", "battlefield 1942"]
},
{
nombre:"Juanito",edad:38,
direccion:{calle_completa:"calle_inventada_2",cp:66666,ciudad:"Benidorm",provincia:"Alicante"},
hobbyes:["pasear","andar", "caminar"]
}
],
{ordered:false})
```



ACTUALIZACIÓN

l. Actualiza e introduce tres hobbies también al primer alumno insertado

```
db.alumnos.updateOne({ _id : "alumno001" },{ $set: { hobbyes : ["nadar","andar","mb downhill"] } })
```

BORRADO

m. Borra los alumnos que tenga un determinado hobbie

```
db.alumnos.deleteMany( { hobbyes: "andar" } )
```

Ejercicio libre: Elige una BD geojson de la web Open Data Euskadi u otra similar y realiza 3 consultas geoespaciales diferentes utilizando al menos los operadores que hemos visto en la tarea de aprendizaje 3. ($near, $geoWithin…) (3 puntos)

Bibliotecas en un radio de 10km desde mi casa

```
db.bibliosEuskadi.find({
  geometry: {
    $geoWithin: {
      $centerSphere: [ [-2.49675,43.06623],10/6378.1 ]
      }
    }
  }
)
```

Cambiar nombre a todas las bibliotecas en un radio de 10 km

```
db.bibliosEuskadi.updateMany({geometry: {
    $geoWithin: {
      $centerSphere: [ [-2.49675,43.06623],10/6378.1 ]
      }
    }
}, {$set: {"properties.documentname":"NOMBRE INVENTADO"}});
```

Bibliotecas dentro del polígono Alto Deba

```
const p1=[-2.6031, 42.972]
const p2=[-2.4005, 43.1316]
const p3=[-2.3669, 43.0124]

db.bibliosEuskadi.find({
  geometry: {
    $geoWithin: {
      $geometry: {
        type: "Polygon" ,
        coordinates: [ [p1,p2,p3,p1] ]
      }
    }
  }
})
```

Eliminar biblioteca de Santa Marina

```
db.bibliosEuskadi.deleteOne({id: 97})
```

Bibliotecas hasta 10km de distancia

```
db.bibliosEuskadi.createIndex( { geometry : "2dsphere" } );


db.bibliosEuskadi.find(
  {
    geometry:
     { $near :
       {
         $geometry: { type: "Point",  coordinates: [ -2.49675,43.06623 ] },
                $maxDistance: 10000
       }
     }
  }
)
```

Eliminar bibliotecas hasta 10 km de distancia

```
db.bibliosEuskadi.deleteMany(   {
    geometry:
     { $near :
       {
         $geometry: { type: "Point",  coordinates: [ -2.49675,43.06623 ] },
                $maxDistance: 10000
       }
     }
  }
)
```

MÁS FUNCIONES

```
Use BASEDATOS;
db.COLECCION.find(
{"ATRIBUTO":{$PROPIEDAD}},
{INVISIBLE:0,VISIBLE:1}).sort({ORDENADOPOR:1});
```

Comandos más usados de MongoDB

Comandos básicos

`mongo`
Arranca el cliente de MongoDB
`help`
Muestra la ayuda
`db`
Muestra la base de datos que se está usando
`use base_datos`
Selecciona o crea la base de datos denominada base_datos
`show collections`
Muestra las colecciones de la base de datos que tenemos seleccionada
`exit`
Sale del cliente de MongoDB

Consultas

`db.nombre_coleccion.find()`
Lista todos los documentos que contiene la colección denominada nombre_coleccion (los primeros 10)
`it`
Paginación. Muestra los 10 siguientes resultados de la lista

Operadores para usar en los filtros - Existencia y tipo

`db.micol.find({ campo: { $exists: true } })`
Lista todos los documentos que contiene la colección denominada micol donde el campo existe
`db.micol.find( { campo: { $type: "string" } } )`
Lista todos los documentos que contiene la colección denominada micol donde el campo es de tipo string
El $type puede ser un número o un alias. Algunos tipos habituales son: string, array, bool, date, long, decimal

Operadores para usar en los filtros - Comparativos

`db.micol.find({ campo: {$gt: 90} })`
Lista todos los documentos que contiene la colección denominada micol donde el campo es mayor de 90
`db.micol.find({ campo: {$gte: 90} })`
Lista todos los documentos que contiene la colección denominada micol donde el campo es mayor o igual a 90
`db.micol.find({ campo: {$lt: 90} })`
Lista todos los documentos que contiene la colección denominada micol donde el campo es menor de 90
`db.micol.find({ campo: {$lte: 90} })`
Lista todos los documentos que contiene la colección denominada micol donde el campo es menor o igual a 90
`db.micol.find({ campo: {$in: ["string1", "string2"] } })`
Lista todos los documentos que contiene la colección denominada micol donde el campo es igual a string1 o string2
`db.micol.find({ campo: {$nin: ["string1", "string2"] } })`
Lista todos los documentos que contiene la colección denominada micol donde el campo no es igual ni a string1 ni a string2

Operadores para usar en los filtros - Lógicos

`db.micol.find({ $and: [ { campo: { $exists: true } }, { campo: { $gt:10 } } ] })`
Lista todos los documentos que contiene la colección denominada micol donde el campo existe y es mayor de 10

`db.micol.find( { $or: [ { campo1: { $lt: 20 } }, { campo2: 10 } ] } )`
Lista todos los documentos que contiene la colección denominada micol donde el campo1 es menor de 20 o el campo2 es 10

Operadores para usar en los filtros - Expresiones regulares y búsqueda de texto

`db.micol.find({ campo: /patron/ })`
Lista todos los documentos que contiene la colección denominada micol donde el campo cumple la expresión regular especificada en patron

`db.micol.find({ $text: { $search: "valor" } })`
Lista todos los documentos que contiene la colección denominada micol donde el documento contiene un campo indexado como texto que contiene el término valor

Operadores para usar en los filtros - Arrays

`db.micol.find({ campo: { $all: ["valor1","valor2"] } })`
Lista todos los documentos que contiene la colección denominada micol donde el array campo contiene el elemento valor1 y valor2

`db.micol.find({ campo: { $size: 1 } })`
Lista todos los documentos que contiene la colección denominada micol donde la longitud del array campo vale 1

`db.micol.find({ campo: { $elemMatch: { $gte: 80, $lt: 85 } } })`
Lista todos los documentos que contiene la colección denominada micol tiene algun elemento del array campo mayor o igual a 80 y menor de 85

Altas, bajas y modificaciones

`db.micol.insertOne(doc)`
Inserta el documento doc en la colección micol

`db.micol.insertMany([{...}, {...}])`
Inserta los documentos en la colección micol

`db.micol.update(filtro, campos)`
Actualiza el documento que devuelve el filtro y establece los valores definidos en campos

`db.micol.remove(filtro)`
Elimina el documento/s que devuelve el filtro

MAS INFO EN xuleta2MongoDB.docx

mongodb.db()
database.collection()
collection.find()
collection.findOne()
collection.findOneAndUpdate()
collection.findOneAndReplace()
collection.findOneAndDelete()
collection.insertOne()
collection.insertMany()
collection.updateOne()
collection.updateMany()
collection.deleteOne()
collection.deleteMany()
collection.aggregate()
collection.count()
collection.distinct()
collection.bulkWrite()

UNIDAD 07 API REST:

Crea una API REST JPA utilizando el modelo de datos que has utilizado en la Tarea evaluativa AD05. (5 puntos)

Creamos la BD en este orden para que no aparezca ningún error en la salida a consola:

Ejecutamos la query para crear el esquema desde el ide de sql:

create schema tienda_ad07;

Ejecutamos el springboot con esta configuración en *application.properties*:

```
spring.datasource.url=jdbc:mysql://localhost/Tienda_AD07
spring.datasource.username=birt
spring.datasource.password=birt
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
server.port=8090
```

Cambiamos la línea `spring.jpa.hibernate.ddl-auto` de none a update y ejecutamos para insertar las tablas.

Volvemos a ponerlo en none y añadimos la siguiente línea para insertar los datos del archivo *data.sql* reiniciando el proyecto:

`spring.sql.init.mode=`always

Entrar para verlo en

http://localhost:8090/api/producto y http://localhost:8090/api/categoria

Realiza un proyecto API REST MongoDB similar al de la tarea de aprendizaje 2, utilizando la idea del modelo de datos del ejercicio 1 de esta misma tarea evaluativa. Ten en cuenta que ya no estás trabajando en un modelo relacional y probablemente sea conveniente unir la información de dos tablas en un único documento: (5 puntos)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.6.3</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>eus.birt.dam</groupId>
	<artifactId>ud7-api-rest-mongodb</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>ud7-api-rest-mongodb</name>
	<description>api rest mongodb</description>
	<properties>
		<java.version>17</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-mongodb</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>org.projectlombok</groupId>
			<artifactId>lombok</artifactId>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
				<configuration>
					<excludes>
						<exclude>
							<groupId>org.projectlombok</groupId>
							<artifactId>lombok</artifactId>
						</exclude>
					</excludes>
				</configuration>
			</plugin>
		</plugins>
	</build>
</project>
```

Application.properties

```
spring.data.mongodb.database=tienda_ad07
spring.data.mongodb.port=27017
spring.data.mongodb.host=localhost
```