**Laboratorio 2**

****



**Integrantes:**

Tiago Mendaro, Federico González, Juan Ferreira, Agustín Bergalli.

**Docente:**

Andrés Pastorini

**Índice**

Contenido

[Marco conceptual 3](#_heading=h.gjdgxs)

[MongoDB 3](#_heading=h.30j0zll)

[Instalación y configuración 4](#_heading=h.1fob9te)

Hash- Map 6

Conclusión 7

# Marco conceptual

En este documento se realiza una pequeña guía y explicación de el procedimiento realizado con el fin de poder cumplir con lo siguiente:

- Instalación y configuración de MongoDB en localhost desde cero

- Crear un modelo Java que será una entidad MongoDB

- Crear un repositorio de interfaz Java para acceder a nuestros datos MongoDB.

- Crear un controlador para exponer los diferentes puntos finales REST

- Probar los servicios REST creados utilizando Postman.

Además, se indicarán los parámetros y características de los servicios REST realizados, y una justificación de la base de datos no relacional elegida para este trabajo.

# MongoDB

Entre las principales razones de la elección de esta base de datos NoSQL para este trabajo, se encuentran ventajas que ofrece MongoDB como su gran cantidad de documentación de muy buena calidad, amplia y detallada en comparación a otras bases de datos NoSQL. Esto además se ve aún más potenciado con el hecho de que posee una gran compatibilidad con SpringBoot que se utiliza para generar los servicios REST conectándolo con MongoDB a través de dependencias ofrecidas por SpringBoot como por ejemplo SpringData MongoDB. Como punto a favor también se tiene en cuenta que para usos no comerciales MongoDB ofrece sus herramientas completamente gratis, y además de todo esto, es ideal para entornos de pocos recursos de computación como es el caso del trabajo realizado.

En el caso hipotético de que el trabajo a realizar contará con una workload de mayor porte Mongo también ofrece soporte para la implementación de escalabilidad horizontal mediante sharding, lo cual nos permitiría dividir dicha carga de trabajo en diferentes máquinas a un nivel de colecciones, distribuyendo las mismas en diferentes shards en el cluster.

A su vez, en el caso de ser necesario, también se podrían crear un set de replicas, para mantener una redundancia de datos de forma que el sistema pueda correr de forma continua, debido a que si se llegara a tener problemas con la información en un nodo, otro podría tomar su lugar.

# Instalación y configuración

Para comenzar con la instalación y configuración es necesario tener instaladas las ultimas versiones de las siguientes herramientas:

* Robo3T
* MongoDB Community Server
* SpringToolSuite
* Postman

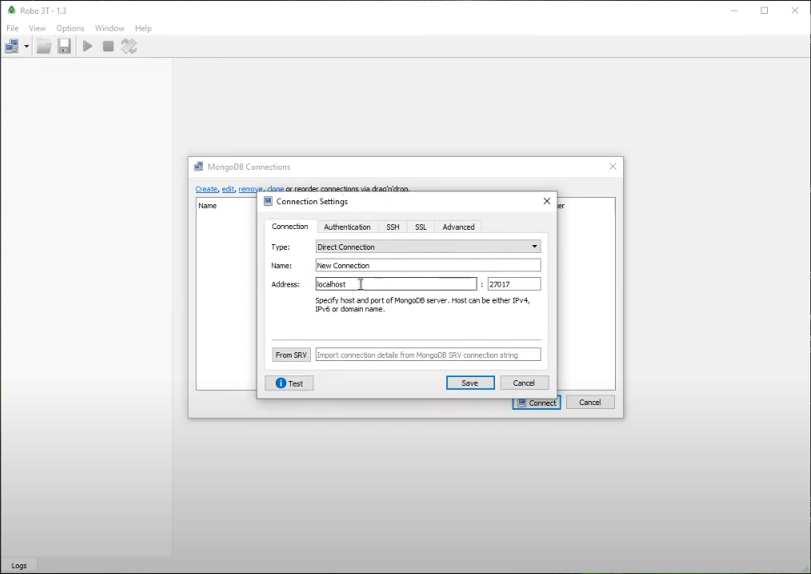
En cuanto a Robo3T, Postman y SpringToolsSuite deberían poder instalarse sin ninguna complicación, sin embargo, en cuanto a MongoDB Community, al descargarlo, por solo la primera vez se deberá realizar un procedimiento particular:

En nuestro directorio raíz, crear una carpeta llamada “data” con una carpeta vacía dentro llamada “db”.

Ir a la carpeta que se descargó de MongoDB, dentro de la carpeta “bin” y copiar la ruta de la carpeta. Dentro de la consola de comandos, dirigirse a esa dirección con el comando “cd” y ejecutar el comando “mongod”.

Si esto sale bien MongoDB quedará listo escuchado en la IP 127.0.0.1 (localhost) en el puerto 27017.

Luego, dentro de Robo3T mientras la terminal sigue abierta, se debe de crear una nueva conexión y conectarse luego de crearla, con los siguientes parámetros de creación:



Si el procedimiento se realiza correctamente, desde la terminal podemos ver cómo se conectó Robo3T con MongoDB. Ya se puede cerrar la terminal.

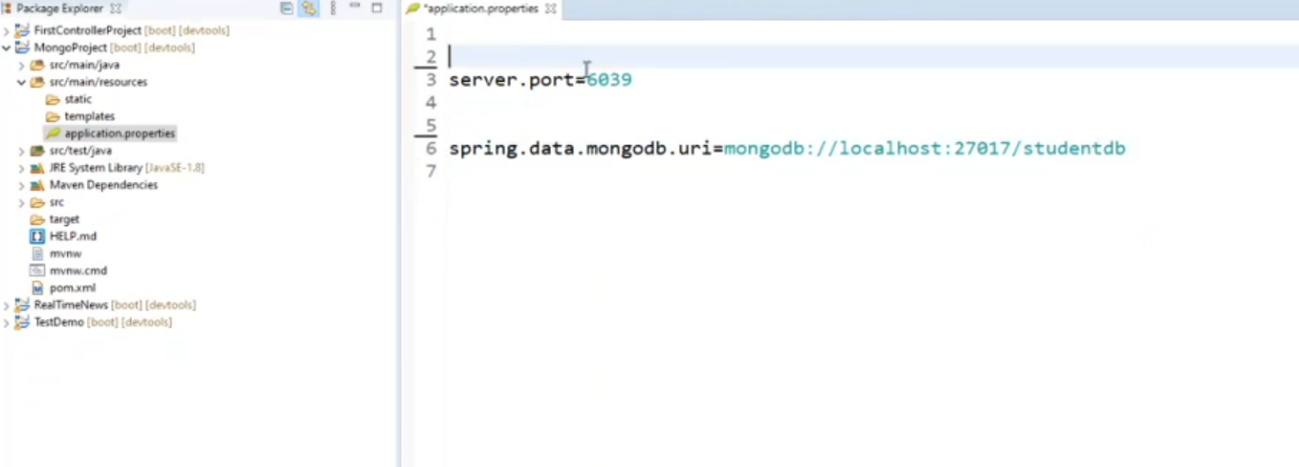
Dentro de SpringToolsSuite debemos crear el proyecto y conectarlo con la base de datos. Para esto debemos hacer lo siguiente:

Click derecho->New->Spring Starter Project

Mantener todo por default en la primera pantalla (se le puede cambiar el nombre y la versión de java). En la siguiente pantalla, buscar y elegir las dependencias Spring Boot DevTools, Spring Data MongoDB y Spring Web y darle a finalizar.

Una vez creado, dentro del proyecto vamos a src/main/resources.>application.properties y tenemos que asociar un puerto libre de la máquina e indicar la url de la base de datos tal y como la creamos en Robo3T. Para el puerto se puede utilizar cualquiera (dentro del rango de los libres), pero para el ejemplo utilizara el 6039.

En la siguiente imagen se indica cómo se debería ingresar.



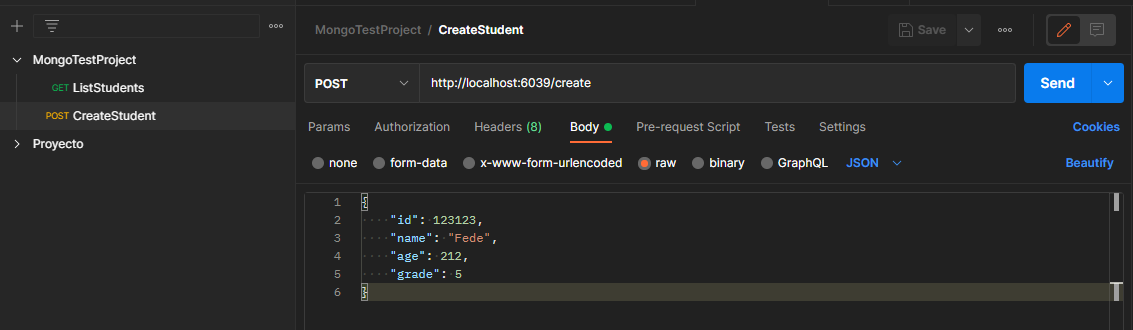
Una vez hecho esto, solamente falta crear la estructura de datos sobre la cual se van a realizar los servicios REST. Para hacer esto, utilizando las anotaciones de MongoDB que nos proporciona la dependencia añadida previamente, necesitamos crear las siguientes clases:

* Una clase para la entidad de Usuario con los sus atributos
* Una clase para el repositorio del usuario que contenga las operaciones REST y funcionen como endpoints posteriormente
* Una clase de interfaz que funcione como controlador del repositorio.

Por último, solo nos falta llamar los endpoint creados en el repositorio desde Postman.

Para esto, dentro de Postman tendremos que crear una nueva colección en nuestro workspace, y añadirle un endpoint por cada uno que hayamos creado, procurando que la url a la que se dirige el endpoint sea tal que “http://localhost:6039/create” es decir que utilice el puerto que indicamos en SpringToolsSuite y la dirección sea localhost, con el nombre que le hayamos puesto al endpoint por último.

En la siguiente imagen se ilustra cómo se debería hacer para el ejemplo dado.



En el caso de que sea un endpoint de creación, hay que indicarle a Postman el cuerpo del tipo de dato que vamos a ingresar manualmente.

Para ello, se indica en body->raw->JSON

Y en esa posición se escribe en formato de JSON el usuario que se va a ingresar.

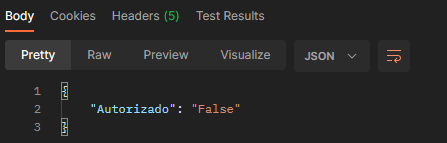
Para poder probar los endpoint se debe procurar estar conectado a la conexión realizada para Robo3T y en SpringToolsSuite este corriendo el programa, clickeando debajo a la izquierda en “local” click derecho->(Re)Start

# HashMap

Para lograr que el formato de intercambio de datos sea un JSON utilizamos un HashMap, de esta forma devolvemos el cabezal como la key del hashmap y la descripción como su valor, como podemos ver en el siguiente

**ejemplo:**



**Resultado en Postman:**

# 

# 

# 

# 

# Conclusión

En conclusión aprendimos cómo funcionan las bases NoSQL con soporte mongoDB su documentación variada y de buena calidad nos hizo este trabajo más fácil. También nos vimos beneficiados por los pocos recursos que este programa consume y por ser gratuito.

Por otro lado, trabajar con Java, que es un lenguaje con el que estuvimos familiarizados en años anteriores, nos ayudó mucho en esta tarea.

Su instalación es guiada por ende entendemos que es fácil en cambio adquiere un mayor grado de complejidad su configuración, lo cual las personas que no poseen conocimientos acerca de cómo resolverlos se les puede dificultar a la hora de emplearlo.