Integración De Aplicaciones Con Framework De Persistencia

Caso de estudio - MongoDB

Deissy Yohana Neita Nuvan

Jorge Alberto Tibaduiza Rincón

Ingeniero

Carlos Alberto Manrique Palacios



Universidad Pedagógica Y Tecnológica De Colombia

Facultad De Posgrados Ingeniería

Especialización En Bases De Datos

Tunja, Colombia

2024

Contenido

[Caso De Estudio – Vuelta a Colombia 3](#_Toc182564654)

[Modelo Entidad Relación 4](#_Toc182564655)

[Creación de esquema 5](#_Toc182564656)

[Creación de tablas 5](#_Toc182564657)

[Creación de llaves 6](#_Toc182564658)

[Creación de secuencias 6](#_Toc182564659)

[Modelo de Auditoria 6](#_Toc182564660)

[Arquitectura Auditoria 7](#_Toc182564661)

[Arquitectura del Proyecto Java 8](#_Toc182564662)

[Capa de persistencia relacional - Oracle 8](#_Toc182564663)

[Entidades – mapeo de tablas 9](#_Toc182564664)

[Servicios 9](#_Toc182564665)

[Controladores 10](#_Toc182564666)

[Capa de persistencia no relacional – Mongo DB 11](#_Toc182564667)

[Pruebas 13](#_Toc182564668)

[Anexos 18](#_Toc182564669)

[Instalación de MongoDB 18](#_Toc182564670)

[Conexión a MongoDB 20](#_Toc182564671)

[Configuración del proyecto en IntelliJ: 21](#_Toc182564672)

[Configuración del Repositorio del proyecto en GIT: 22](#_Toc182564673)

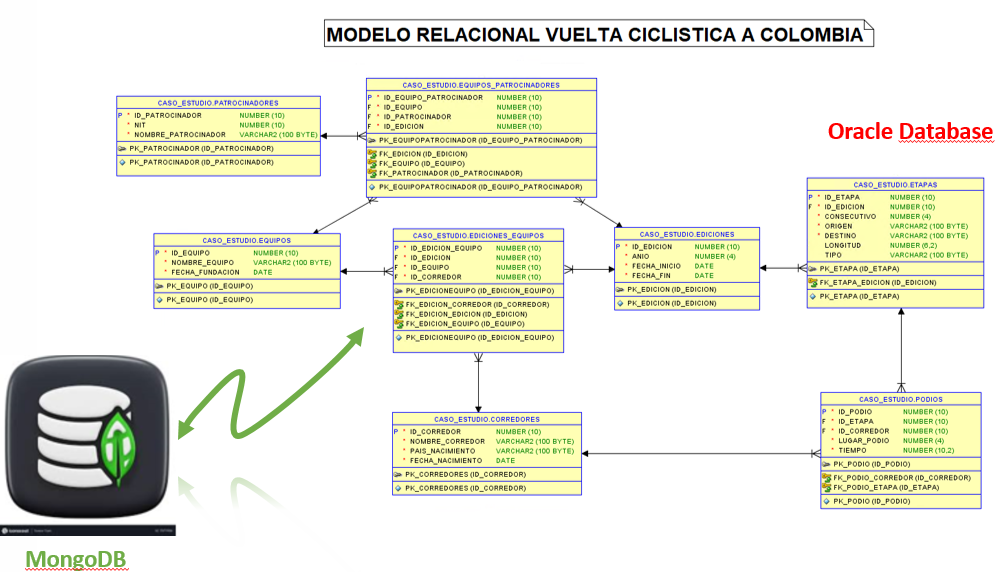
# Caso De Estudio – Vuelta a Colombia

Se desea diseñar un modelo de base de datos que permita el almacenamiento de los datos concernientes a la vuelta ciclística a Colombia, teniendo en cuenta los siguientes enunciados:

* Cada edición de la vuelta a Colombia viene identificada por el año en que tiene lugar y se desea almacenar las fecha inicial y final de la edición. Todas las ediciones tienen diferentes etapas de la cuales es importante guardar un consecutivo identificador dentro de la edición, un origen, un destino y la longitud. Las etapas también tienen un tipo (montaña, contra reloj, etc).
* En las vueltas a Colombia corren diferentes equipos. Cada equipo tiene un nombre, también queremos guardar su fecha de fundación. Los equipos pueden participar o no en una edición de la vuelta. Todos los equipos tienen un patrocinador por edición interesa guardar el NIT del patrocinador y el nombre.
* Se debe almacenar los datos de los corredores que participan, estos corredores pertenecen a un equipo e interesa guardar nombre, país de nacimiento y fecha de nacimiento, pueden cambiar de equipo de un año a otro, interesando almacenar por cuales equipos ha pasado.
* Es de intereses además almacenar el podio de cada una de las etapas, entendiéndose por el podio los ciclistas que ocuparon el primer segundo y tercer lugar en cada etapa. Además, interesa guardar los tiempos de los ocupantes del podio para cada etapa.

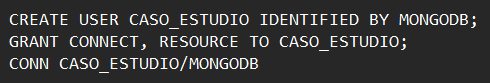
# Modelo Entidad Relación

El modelo entidad-relación se fundamenta en las entidades identificadas en el caso de estudio, definiendo y estableciendo las relaciones entre ellas. Se presenta un esquema relacional para garantizar la persistencia de los datos asociados al proceso de negocio, así como una base de datos adicional destinada al registro y seguimiento de auditorías.



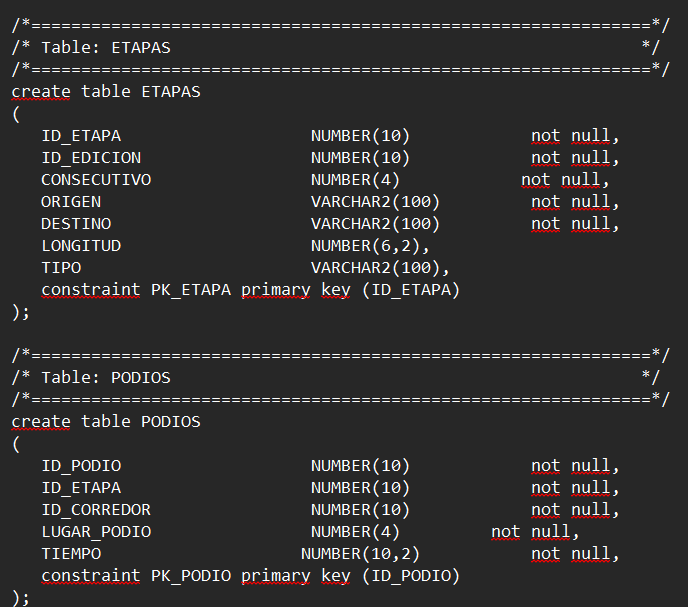
La base de datos relacional se implementa utilizando Oracle y está compuesta por ocho entidades que cubren la totalidad de la lógica de negocio. Por otro lado, la base de datos NoSQL se desarrolla con MongoDB y está vinculada a una base de datos de mongoDB denominada " auditoriaDB", la cual se encargará de almacenar los registros de auditoría correspondientes a las acciones realizadas sobre las entidades del dominio de negocio.

## Creación de esquema



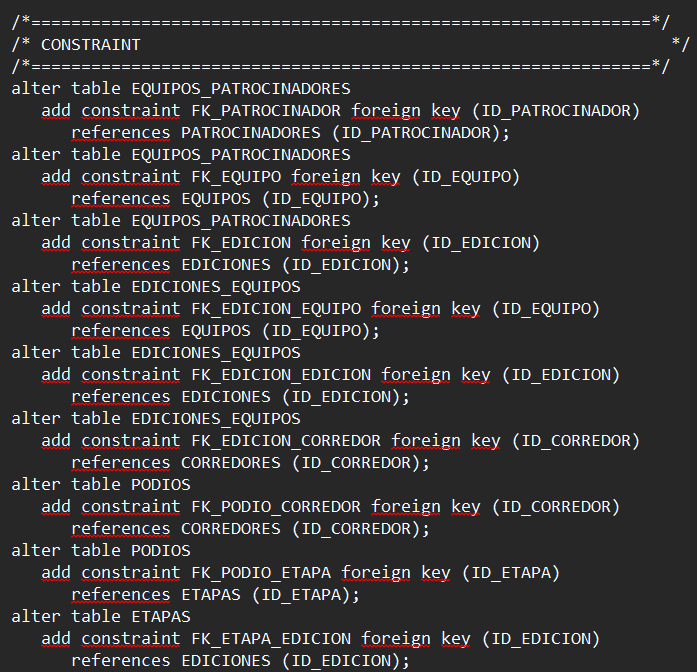
## Creación de tablas

|  |  |
| --- | --- |
|  |  |

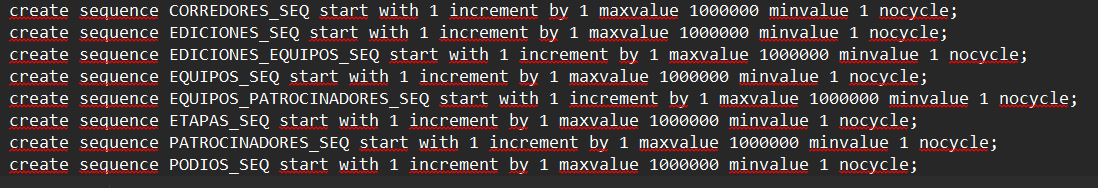


## Creación de llaves

A continuación, se presentan y detallan los scripts necesarios para la creación de las llaves foráneas en cada una de las tablas del modelo de datos, e esta manera se garantiza la integridad referencial entre las tablas, estableciendo relaciones entre las entidades y asegurando que los datos se mantengan consistentes a lo largo de las operaciones de la base de datos.

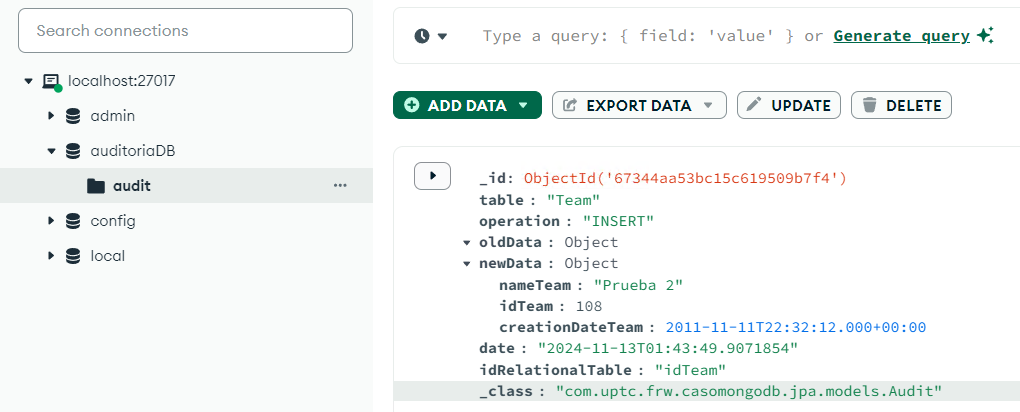


## Creación de secuencias



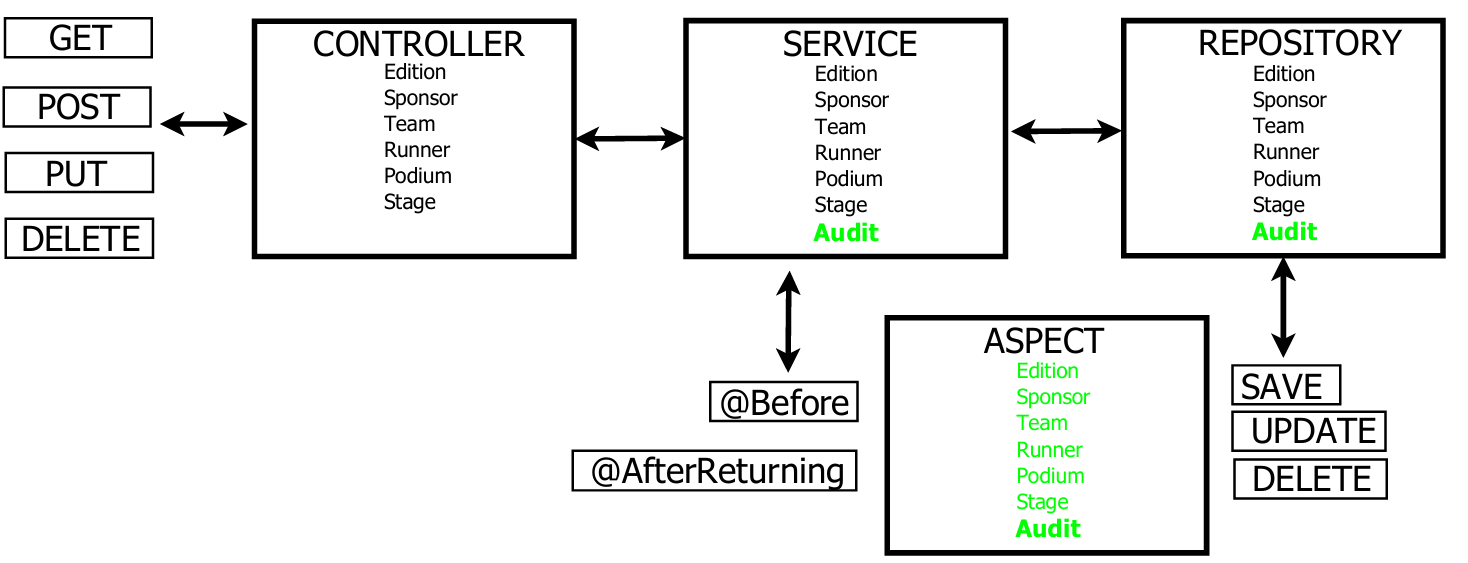
# Modelo de Auditoria

Se plantea la creación de una base de datos llamada auditoriaDB, junto con una colección llamada audit que registre todas las acciones INSERT, UPDATE y DELETE en cada una de las tablas relacionales, con la siguiente estructura:



# Arquitectura Auditoria

La estructura del proyecto está organizada de manera modular, a continuación, se describe la estructura general del proyecto, destacando los paquetes, así como las clases correspondientes a cada entidad del modelo de datos. Cada tabla tiene su respectiva implementación de clases, repositorios, servicios.



En el presente caso de estudio se analizan dos enfoques distintos para la auditoría de datos en sistemas de gestión:

* **Auditoría a través de los servicios de cada entidad**: Se centra en registrar los cambios que ocurren en los datos de las entidades, específicamente las operaciones de inserción (INSERT), actualización (UPDATE) y eliminación (DELETE). Estos cambios son capturados directamente desde los servicios responsables de la manipulación de los datos. Este método permite un control detallado de las modificaciones realizadas en el sistema, garantizando que cada acción sobre los registros sea registrada de manera precisa, lo que facilita la trazabilidad y el análisis posterior.
* **El segundo enfoque utiliza la programación orientada a aspectos (AOP)** para interceptar las operaciones sobre los datos. Mediante el uso de las anotaciones @Before y @After, se define un aspecto que se ejecuta antes y después de la ejecución de métodos específicos relacionados con las entidades. Este enfoque es similar al concepto de triggers en bases de datos, pero implementado en el código de la aplicación, lo que permite una auditoría flexible y menos invasiva. El aspecto captura los datos que se van a auditar y los registra sin necesidad de modificar directamente los servicios de las entidades. Esto ofrece una solución más modular y reutilizable para auditoria de acciones sobre las entidades.

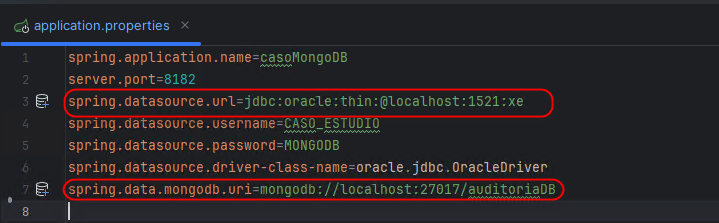
# Arquitectura del Proyecto Java

Se muestra los paquetes principales del proyecto, dentro de cada paquete, se desarrollaron clases específicas para cada tabla del sistema, incluyendo las clases de:

## Capa de persistencia relacional - Oracle

**La capa de persistencia para Oracle** se implementa utilizando JPA (Java Persistence API) junto con Hibernate como proveedor de persistencia. Esta capa gestiona las entidades relacionales, permitiendo la interacción eficiente con la base de datos Oracle. JPA facilita el mapeo objeto-relacional (ORM) para realizar operaciones CRUD sobre las tablas de la base de datos.

**La capa de persistencia para MongoDB** se implementa dentro del proyecto mediante el empleo de Spring Data MongoDB, que facilita la integración con MongoDB y permite gestionar colecciones no estructuradas de manera eficiente. Spring Data MongoDB proporciona una API simple y coherente para realizar operaciones sobre las colecciones de MongoDB, y su integración con Spring Boot asegura una configuración automática y un manejo adecuado de las conexiones.



## Entidades – mapeo de tablas

Se utiliza JPA (Java Persistence API) como proveedor de persistencia para gestionar el mapeo objeto-relacional (ORM) de las entidades y ejecutar las operaciones CRUD (crear, leer, actualizar, eliminar) sobre la base de datos Oracle.

* Models:

|  |  |
| --- | --- |
|  |  |

## Servicios

Se crearon los servicios como clases encargadas de contener la lógica de negocio de la aplicación.

* Service:

|  |  |
| --- | --- |
|  |  |

## Controladores

Como responsables de gestionar las solicitudes HTTP recibidas y devolver las respuestas correspondientes, se crearon los controladores para cada entidad a fin de que respondan a las solicitudes GET, POST, PUT y DELETE.

* Controller:

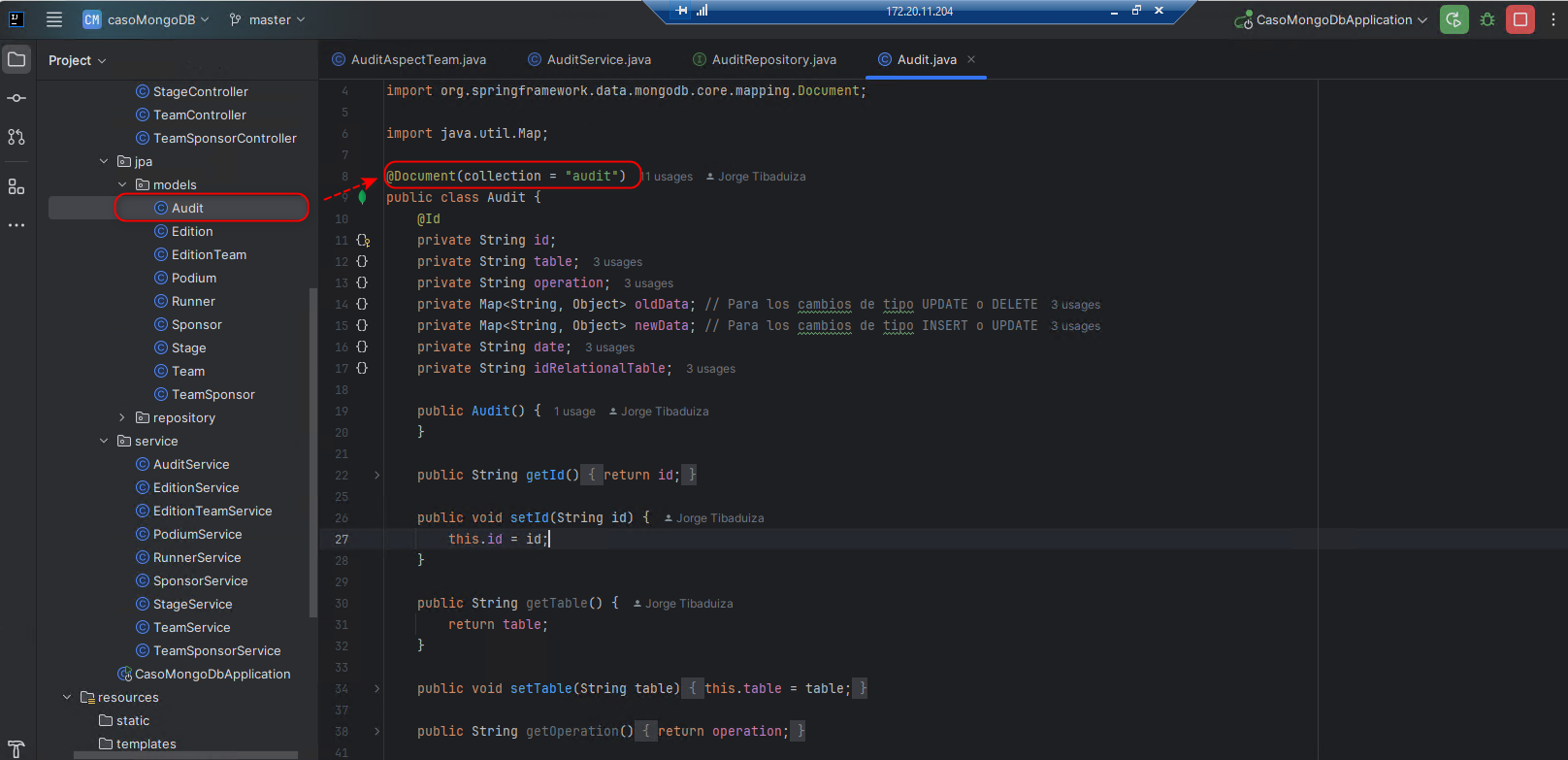
|  |  |
| --- | --- |
|  |  |

* Repository:

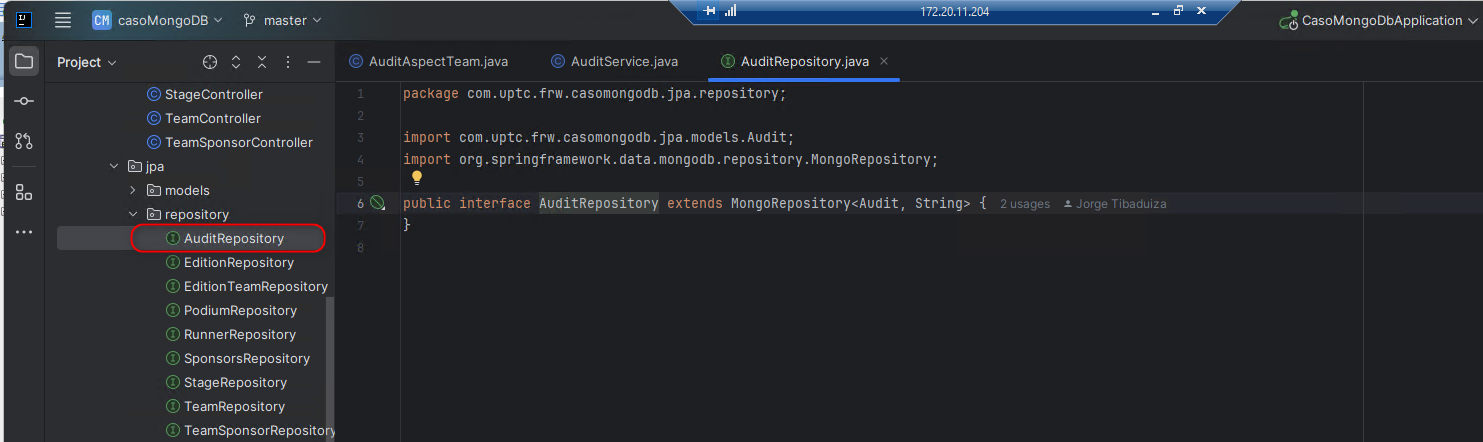
|  |  |
| --- | --- |
|  |  |

## Capa de persistencia no relacional – Mongo DB

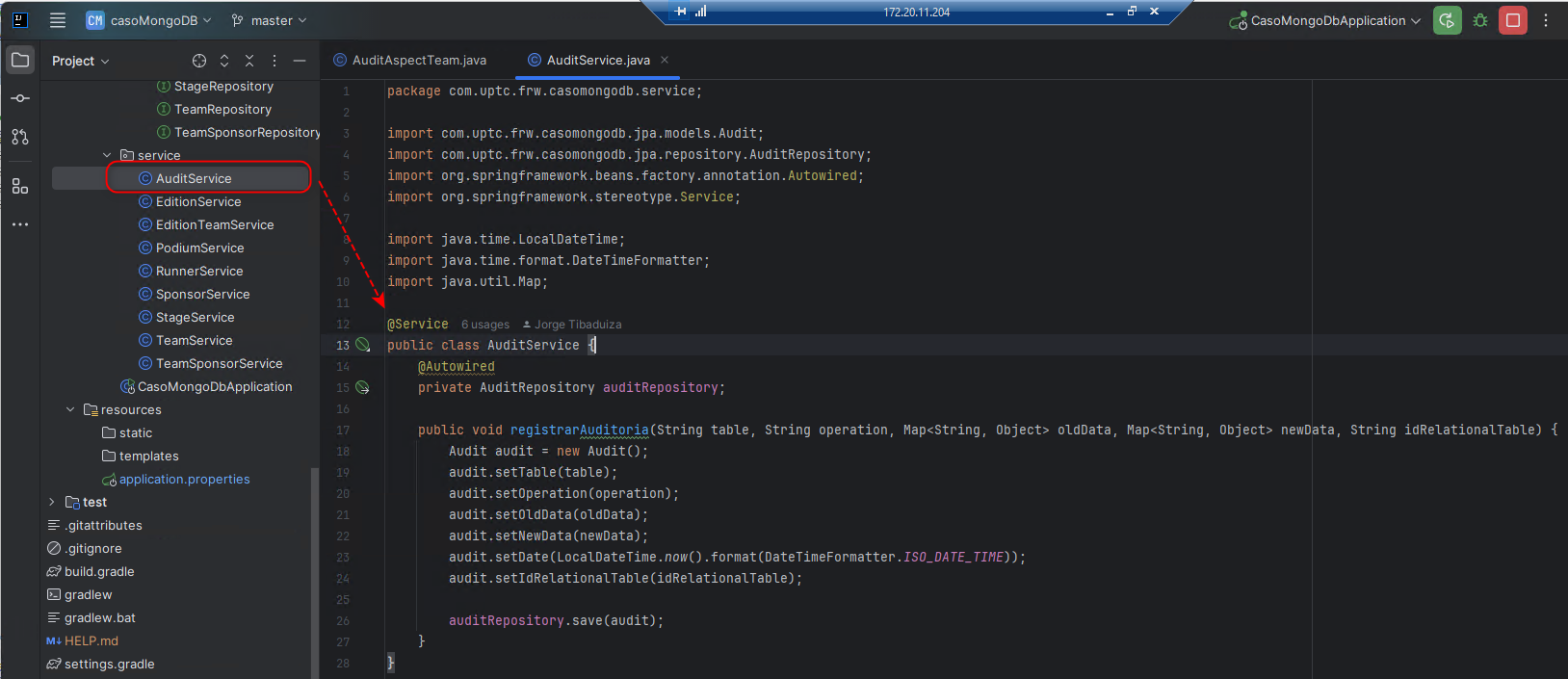
Modelo



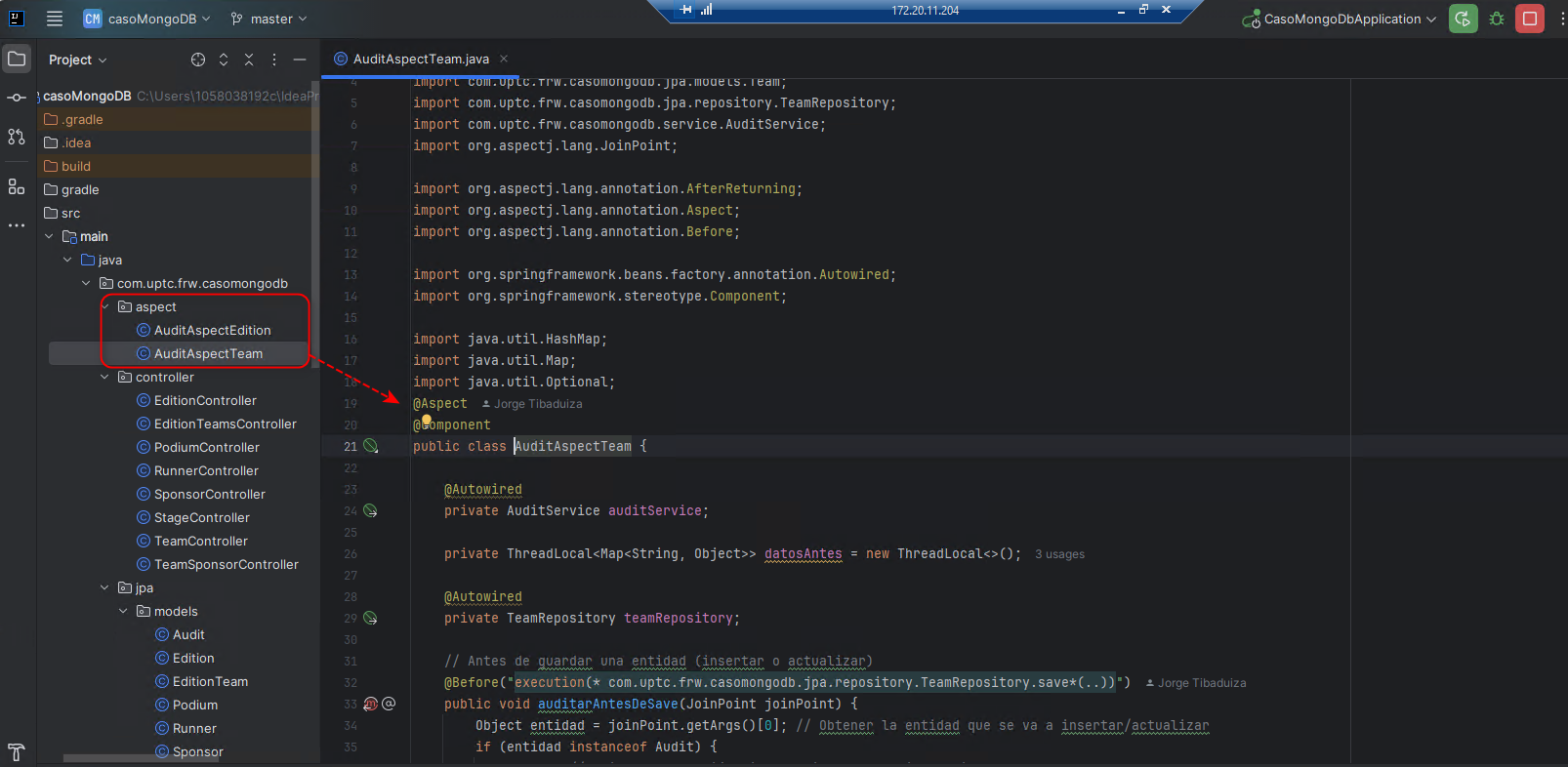
Repositorio



Servicio



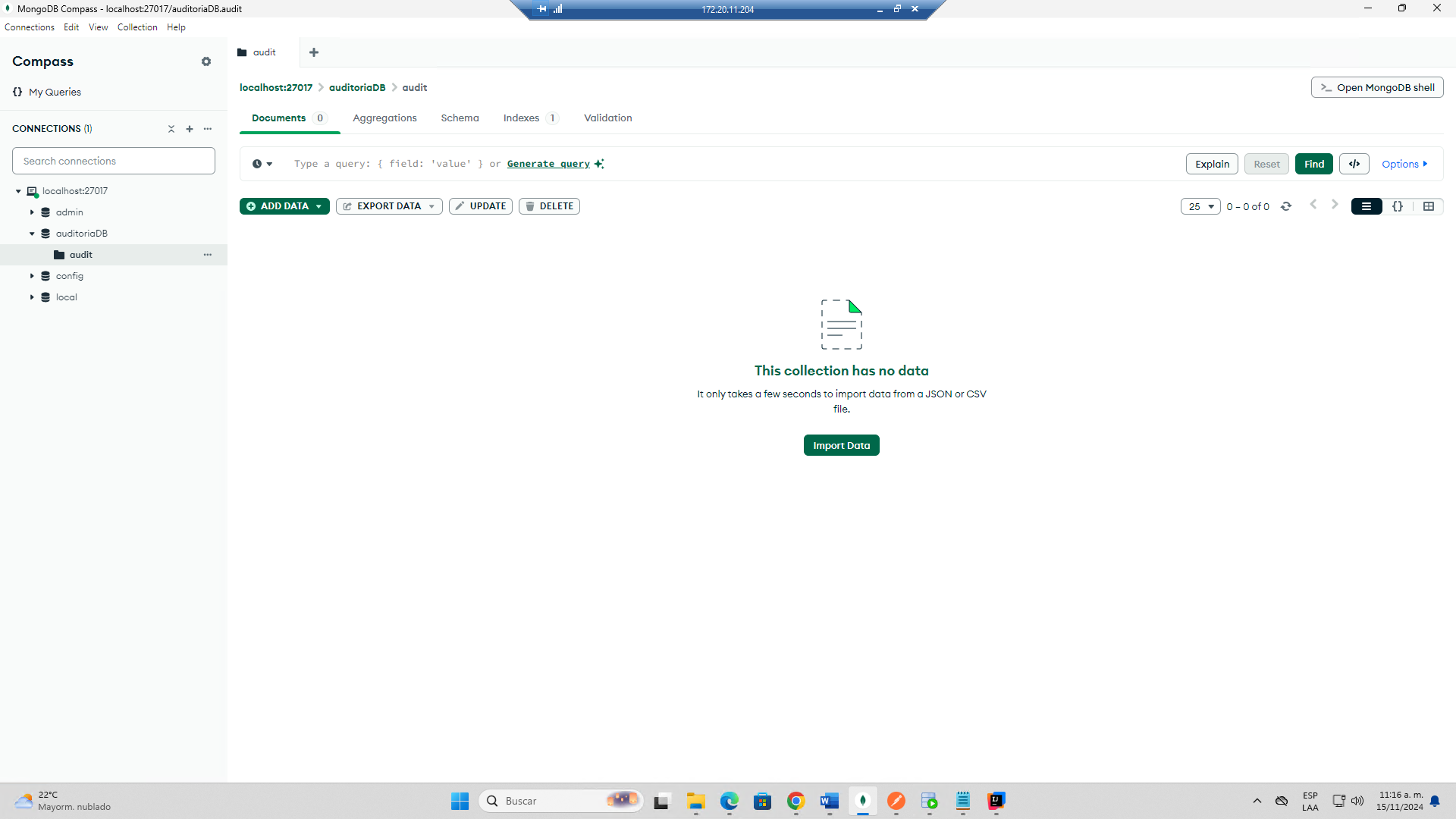
Aspecto



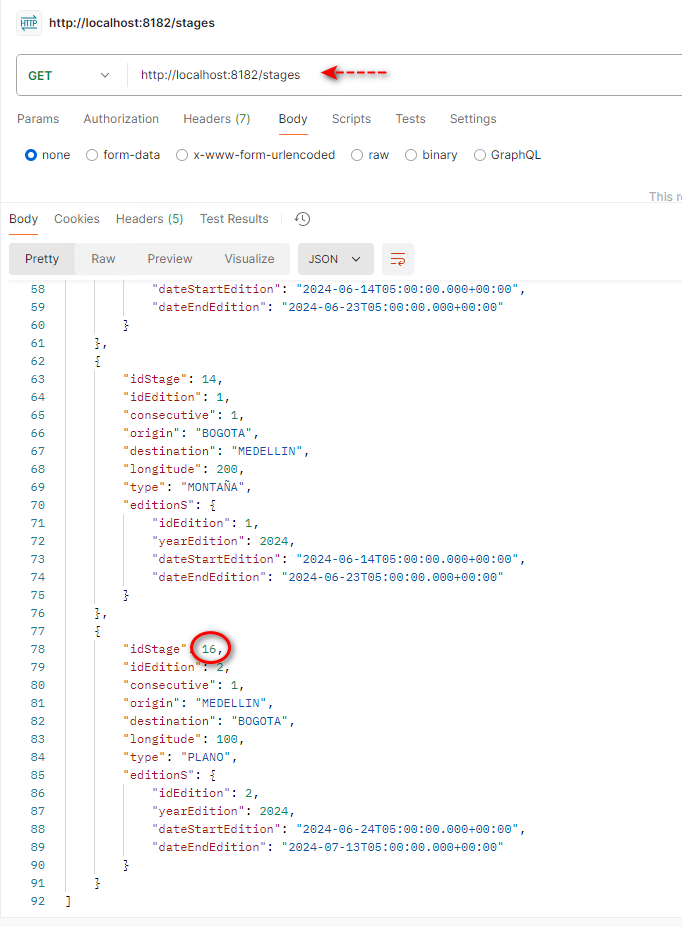
# Pruebas

El set de pruebas **Modelo 1 a continuación**, se toma como la entidad **Stage** [**Etapa**]:

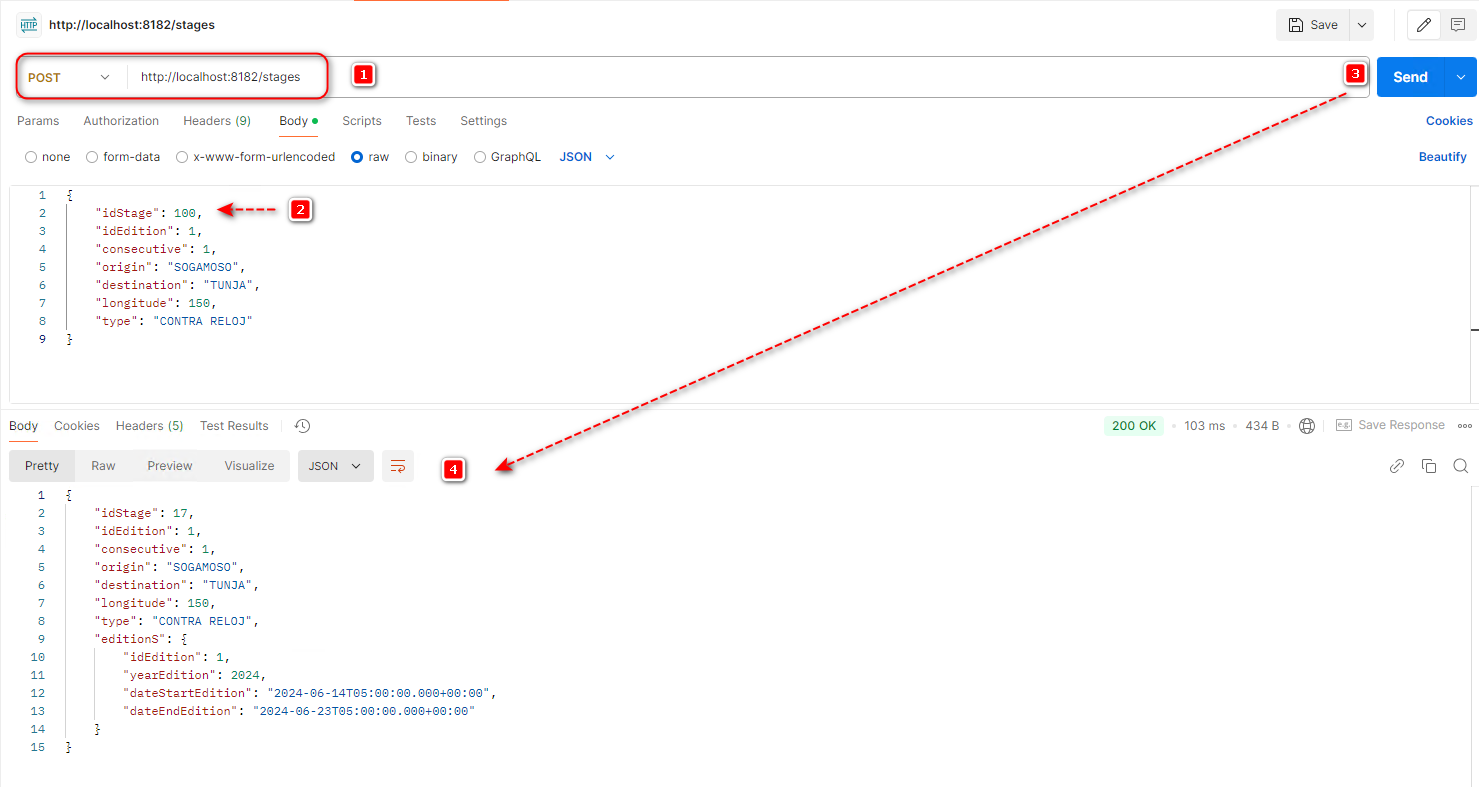
1. Validamos el BD Mongo DB de Auditoria antes de proceder con las pruebas, la cual evidenciamos se encuentra vacía.



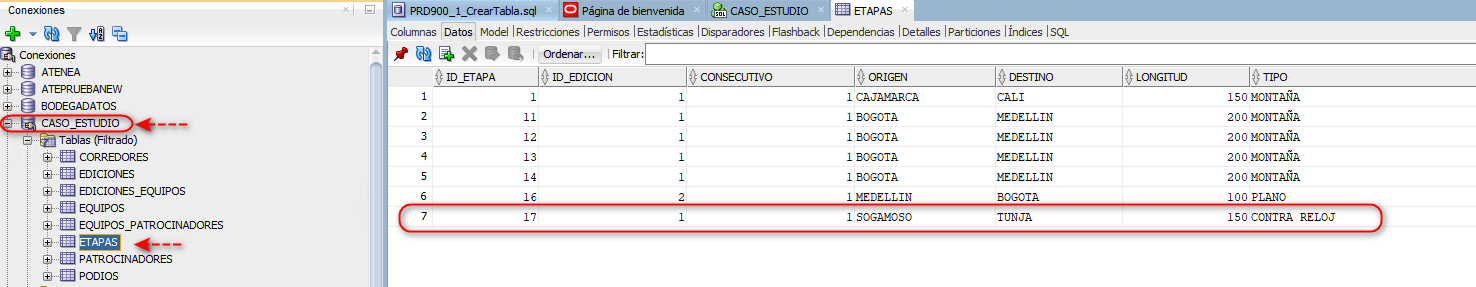
2. Consultamos los registros se **Stage** para validar el consecutivo en el que nos encontramos, el cual coincide en la BD.



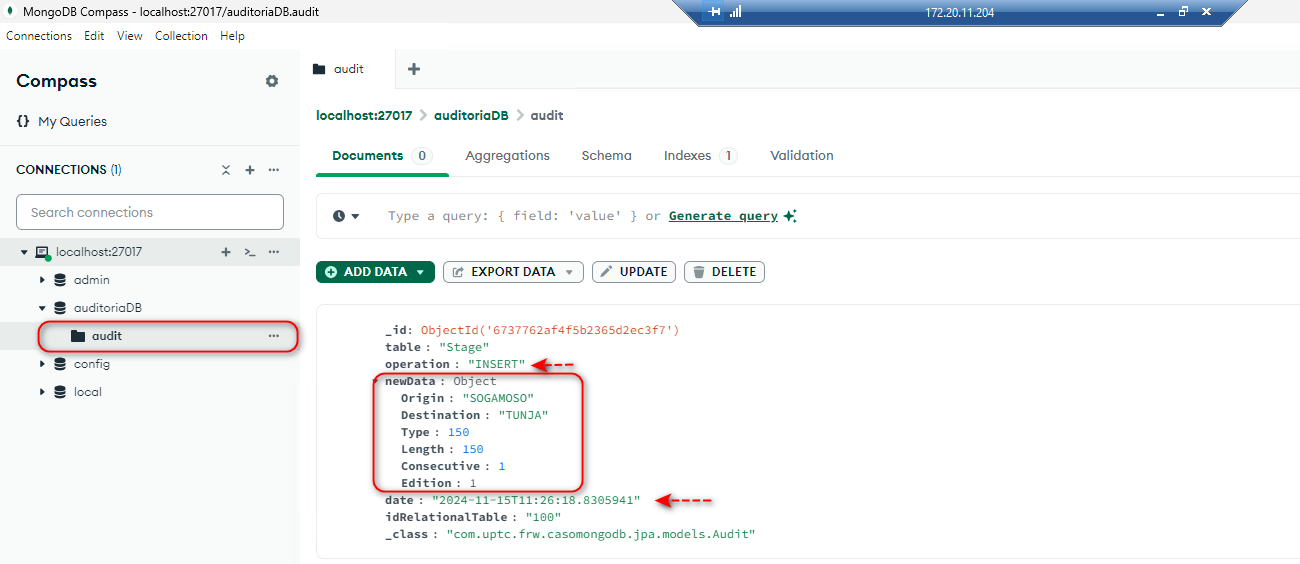
3. Se procede con la operación de **INSERT**.



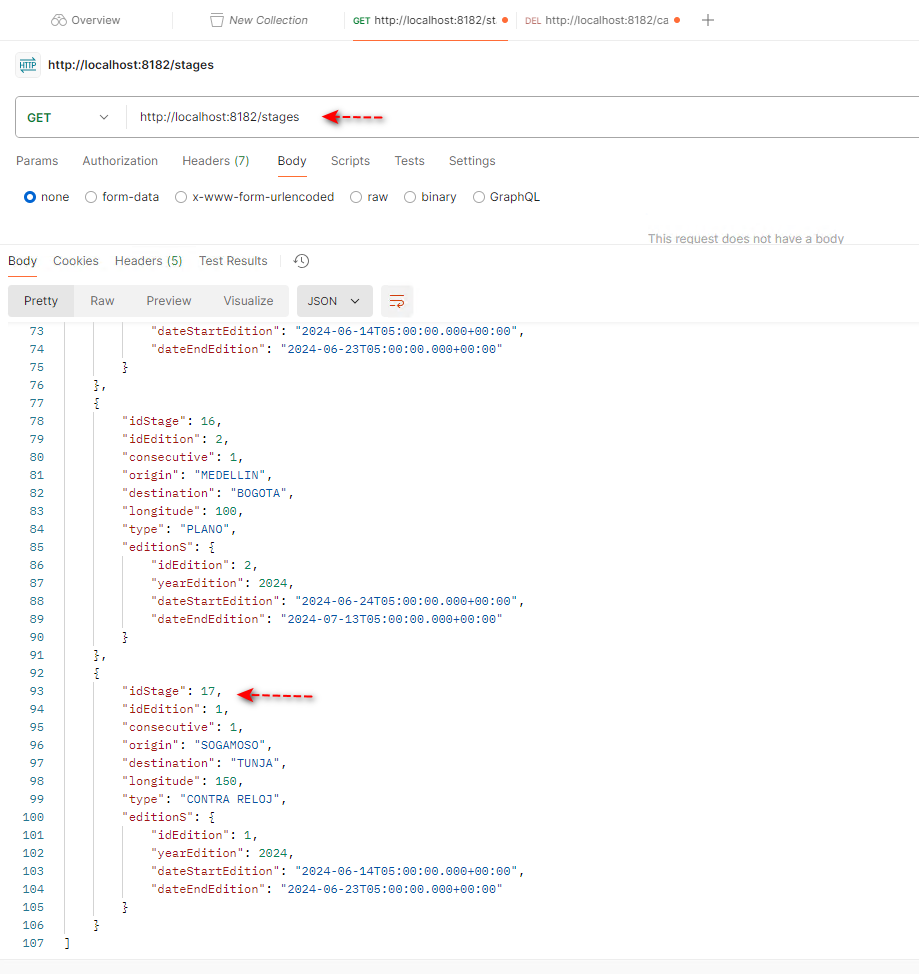
4. Se valida la Inserción en la Base de Datos de Oracle.



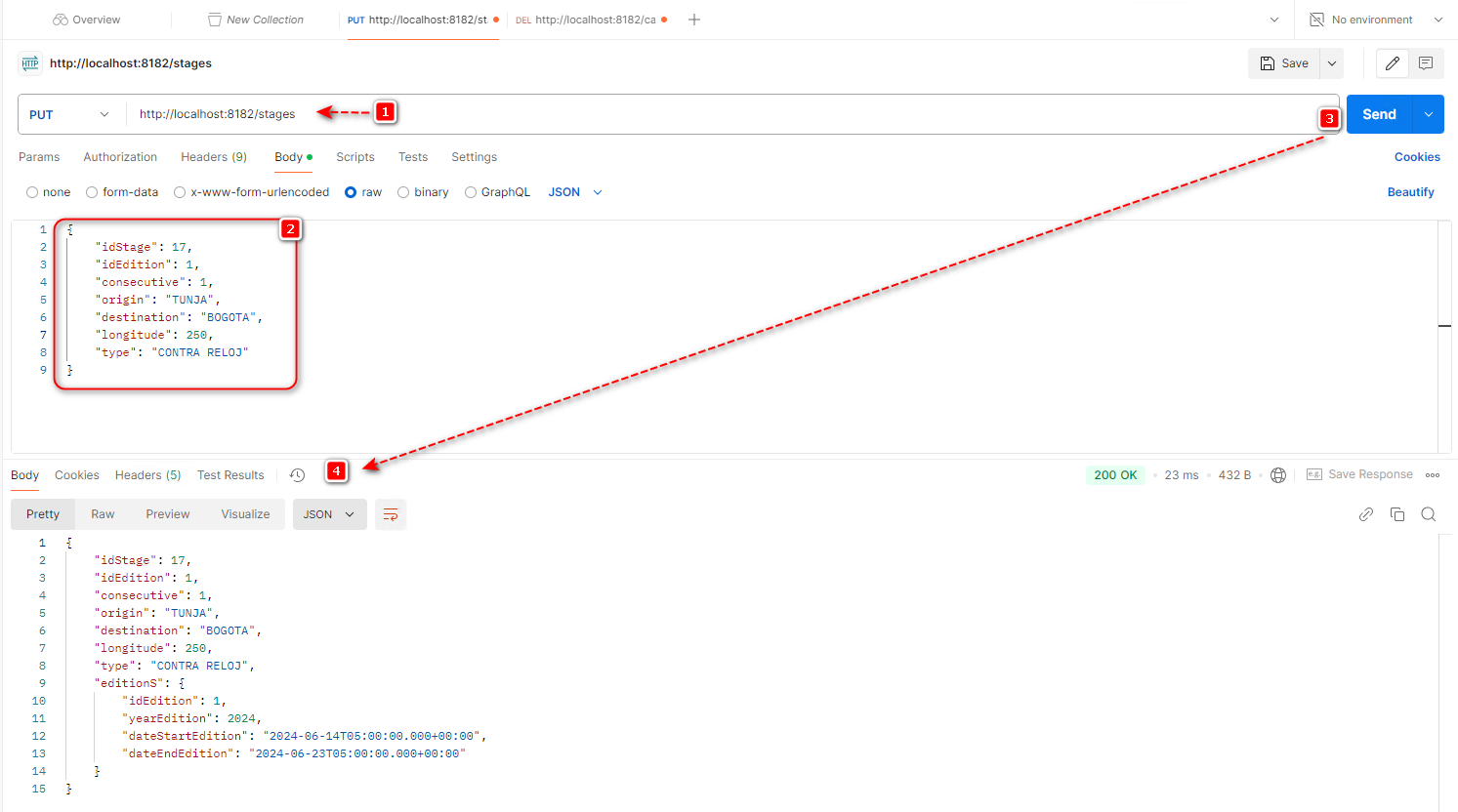
5. Validación de registro en Base de Datos de Mongo DB.



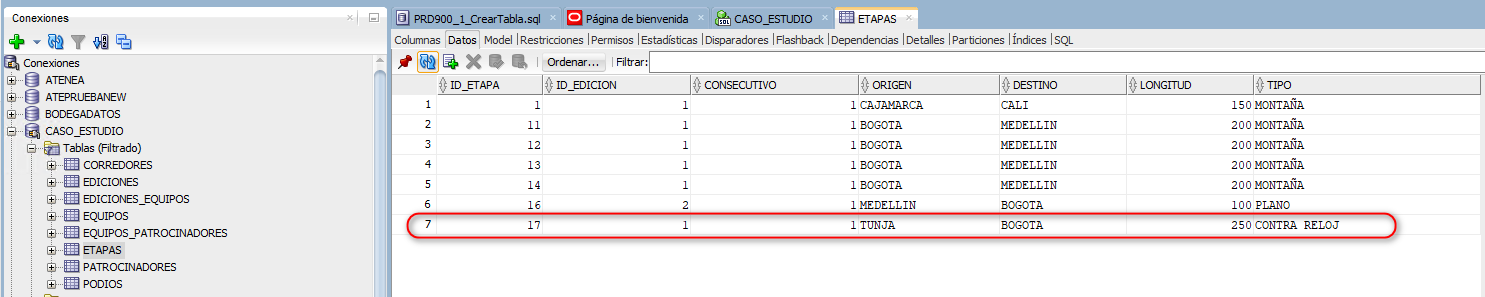
6. Se procede con la operación de **UPDATE,** afectamos el registro creado anteriormente., por lo que consultamos nuevamente los registros se Stage para validar el consecutivo en el que nos encontramos, el cual coincide en la BD.



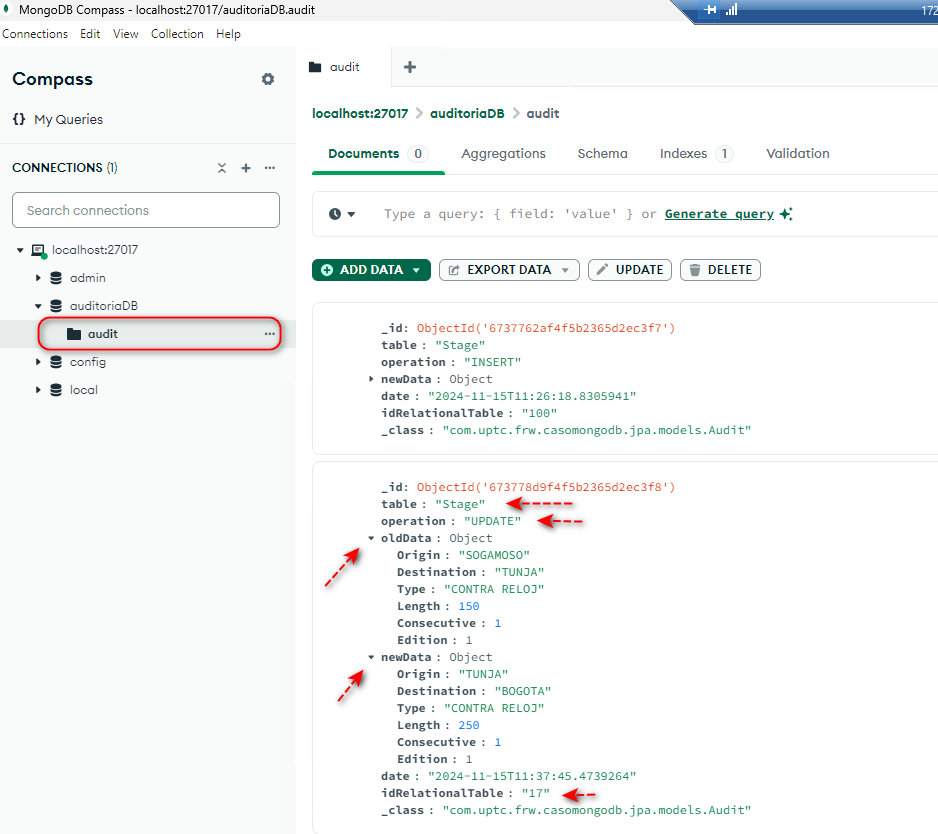
7. Ejecutamos el **UPDATE** desde el servicio.



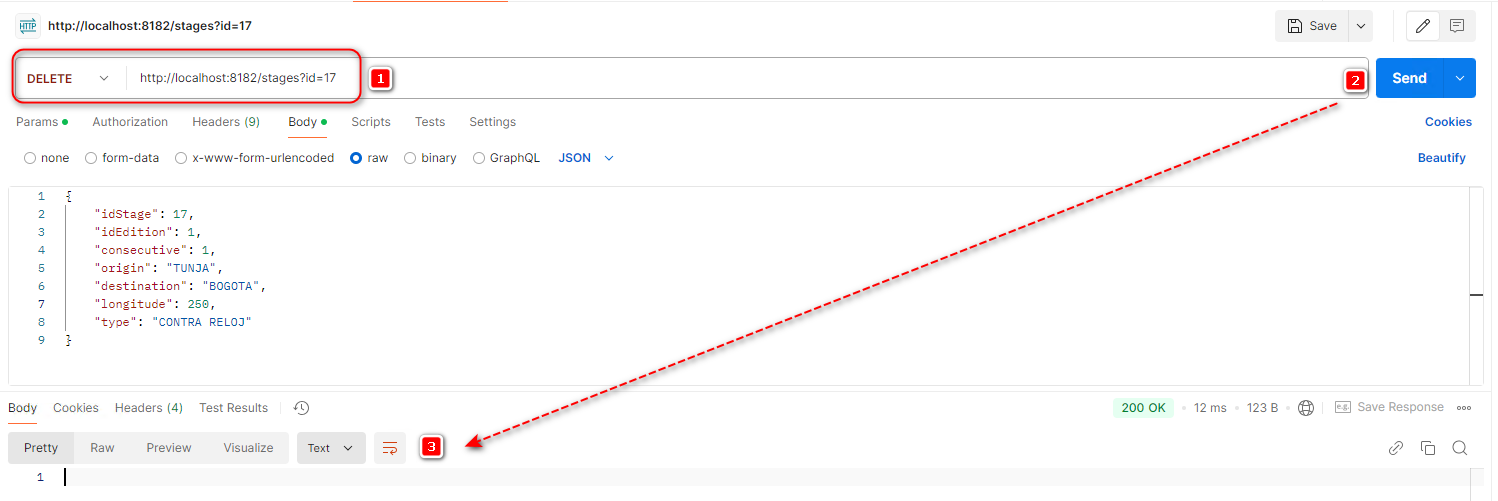
8. Se valida la Actualización en la Base de Datos de Oracle.



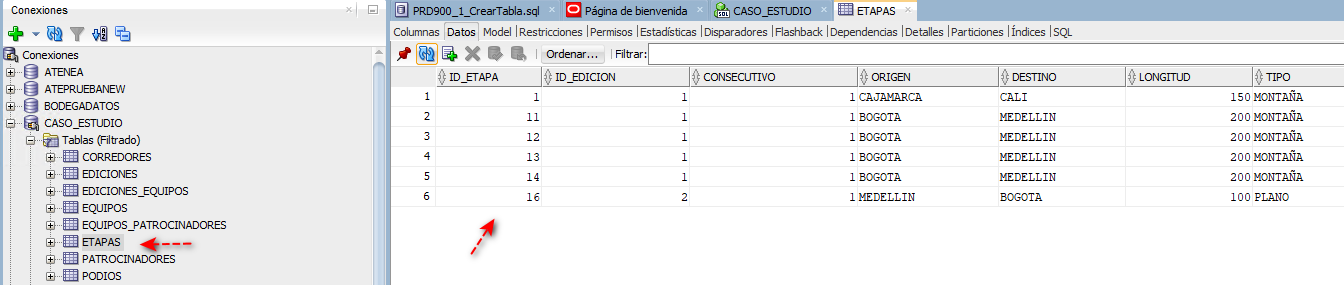
9. Validación de registro de Auditoria en Base de Datos de Mongo DB.



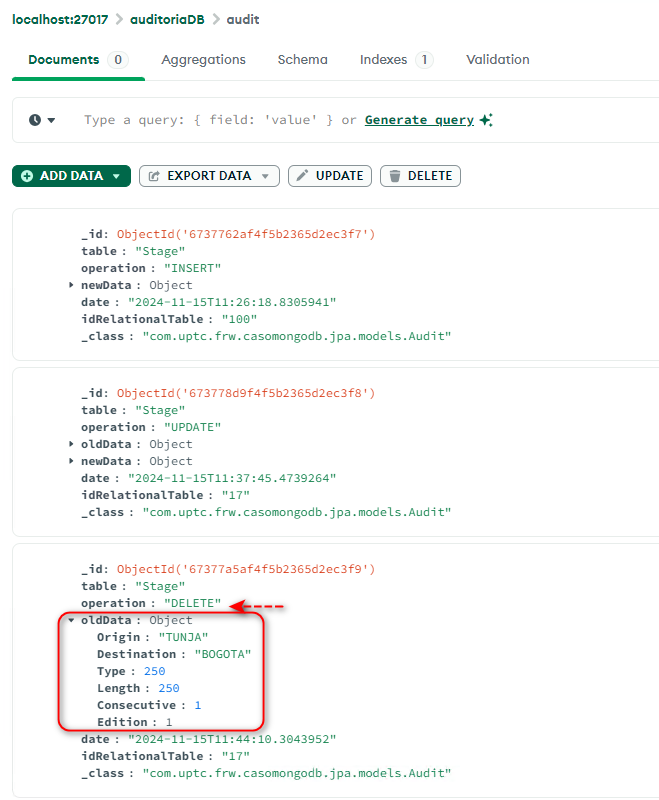
10. Finalmente se ejecuta la operación de **DELETE** desde el servicio. Se eliminará el mismo registro creado con anterioridad.



11. Se valida que en la Base de Datos de Oracle ya no existe el registro No 17.



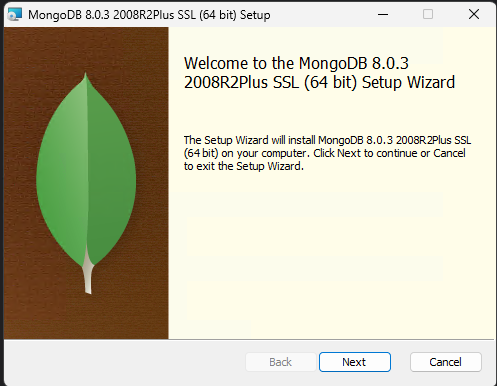
12. Validación de registro de Auditoria en Base de Datos de Mongo DB.



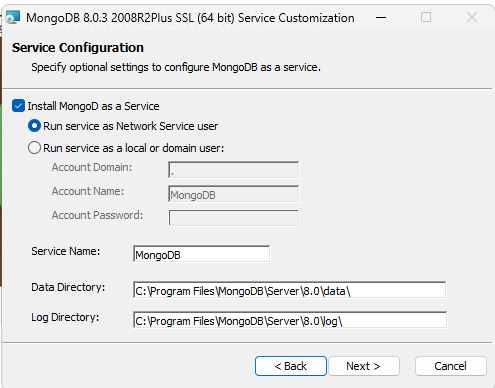
# Anexos

## Instalación de MongoDB

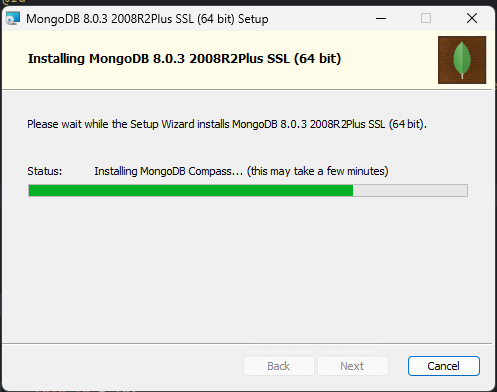
Descargamos MongoDB desde la pagina oficial: https://www.mongodb.com/try/download/community



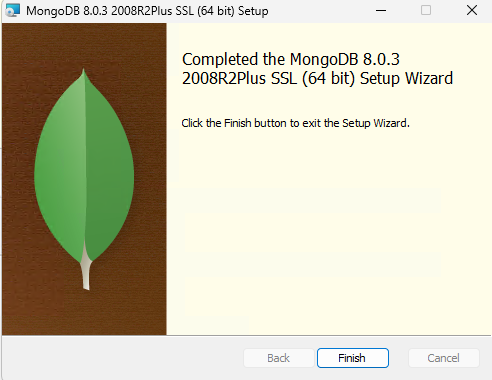
Ejecutamos como administrador y comenzamos la instalación, seleccionamos la ruta en la que deseamos quede instalado.



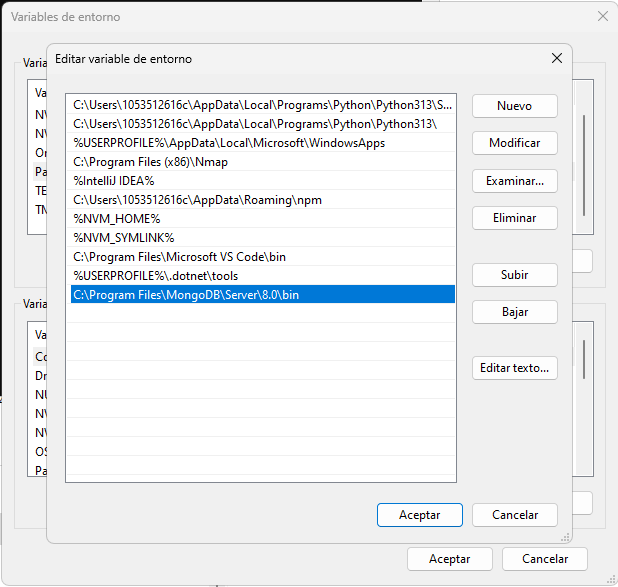
Esperamos a que finalice la copia de información y la instalación



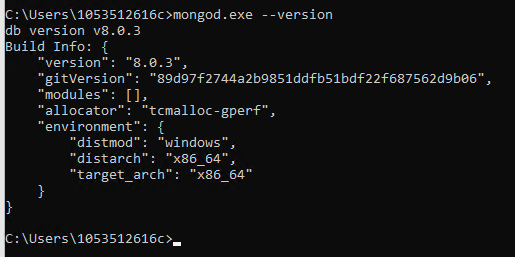
Al finalizar nos permite seleccionar la opción para instalar Mongo DB compass, a la cual le decimos que si.



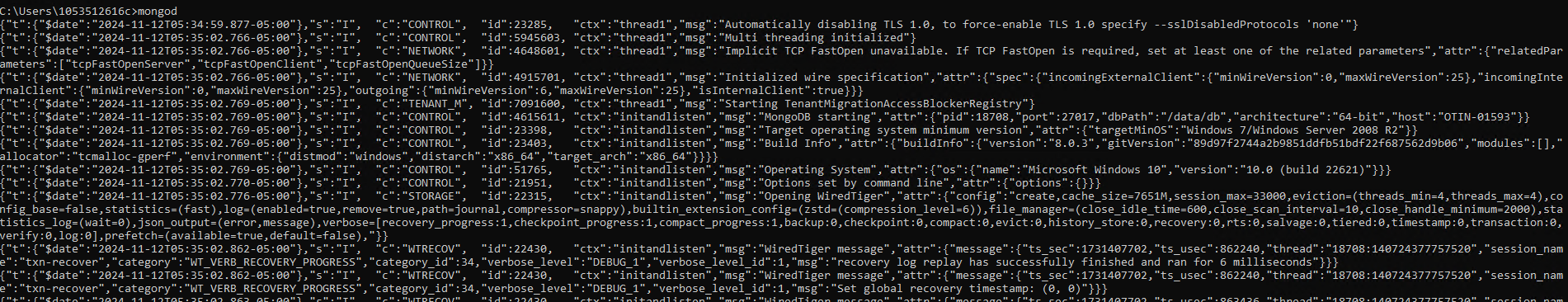
Una vez instalado se agrega la ruta del fichero BIN a las variables de entorno: C:\Program Files\MongoDB\Server\8.0\bin



Se verifica en una terminal mediante el comando **mongod.exe --version** la correcta instalación

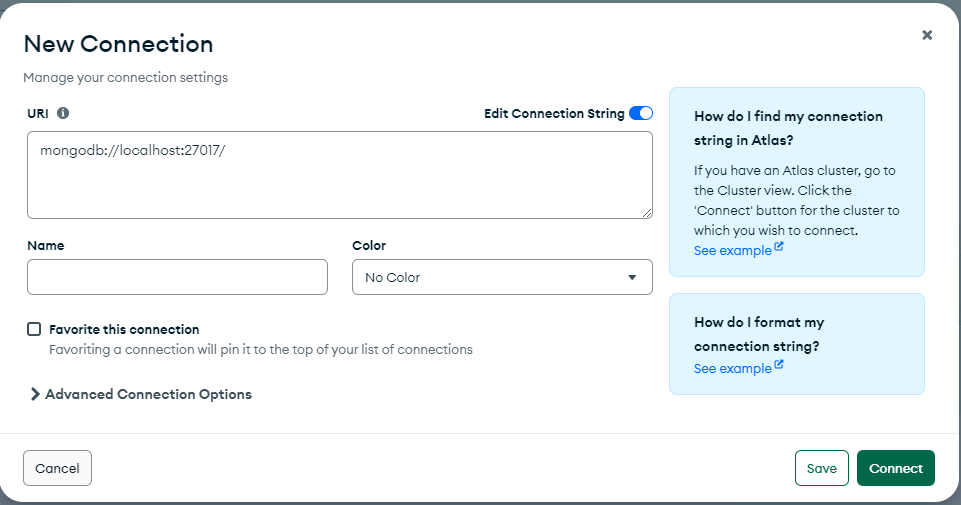


Creamos las carpetas **data** y **db** en **C:\data\db** para que MONGODB guarde allí sus archivos, posteriormente ejecutamos el comando **mongod** en la consola y subimos el servicio



## Conexión a MongoDB

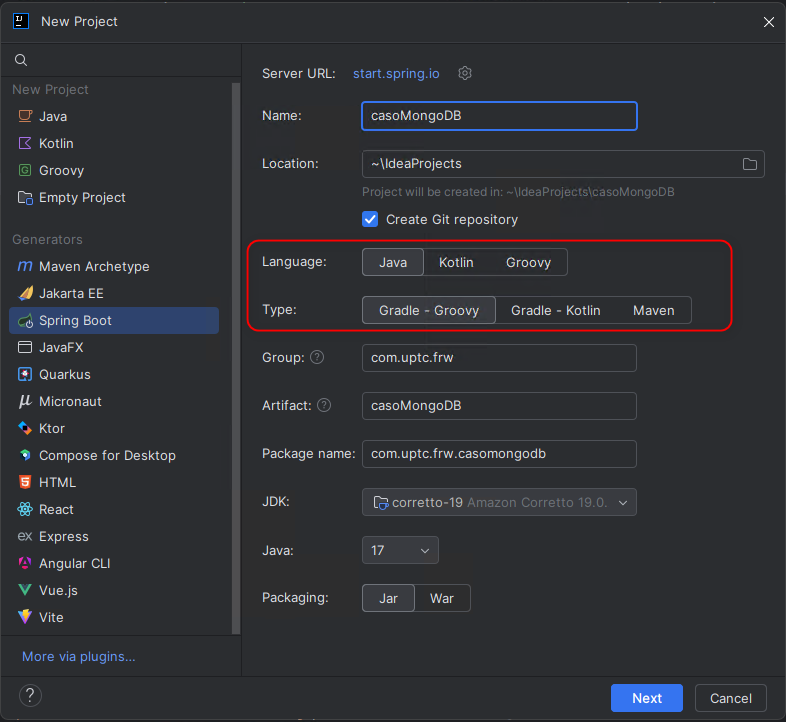
Abrimos MONGODB COMPASS y generamos una nueva conexión, la ruta de nuestro servidor local es mongodb://localhost:27017:



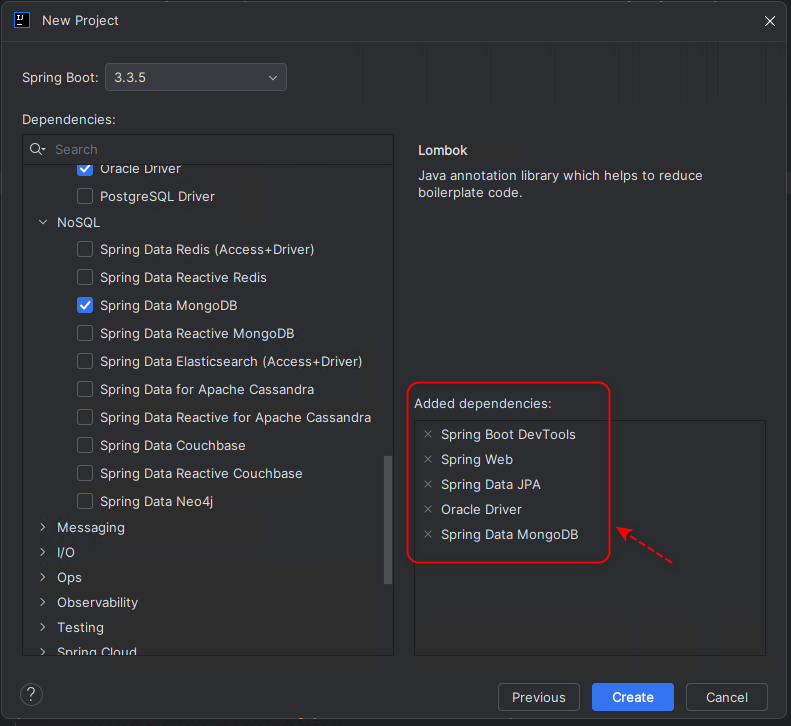
Al conectarnos nos debe mostrar estas 3 bases de datos, que son las que vienen por defecto con la instalación de mongo:



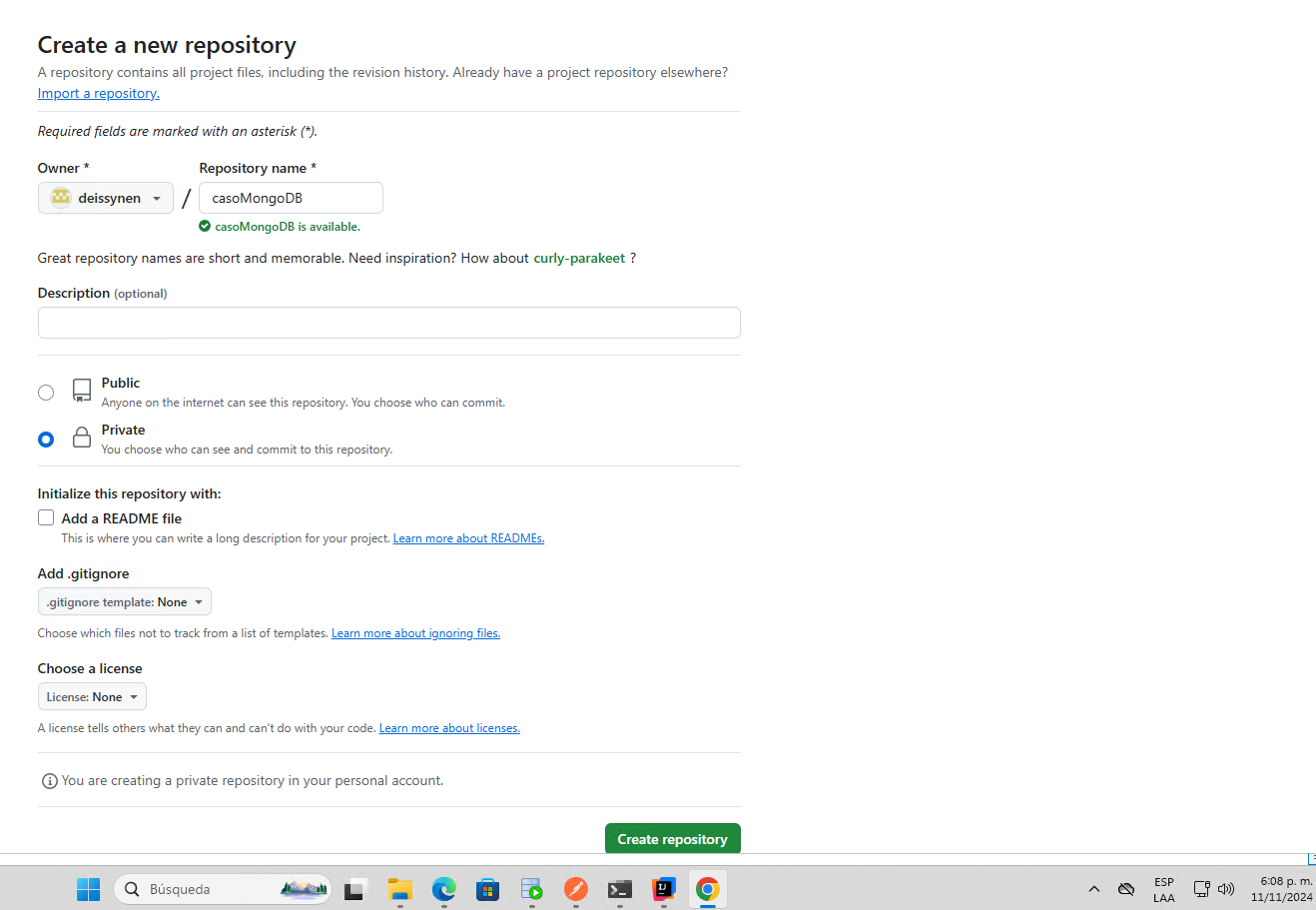
## Configuración del proyecto en IntelliJ



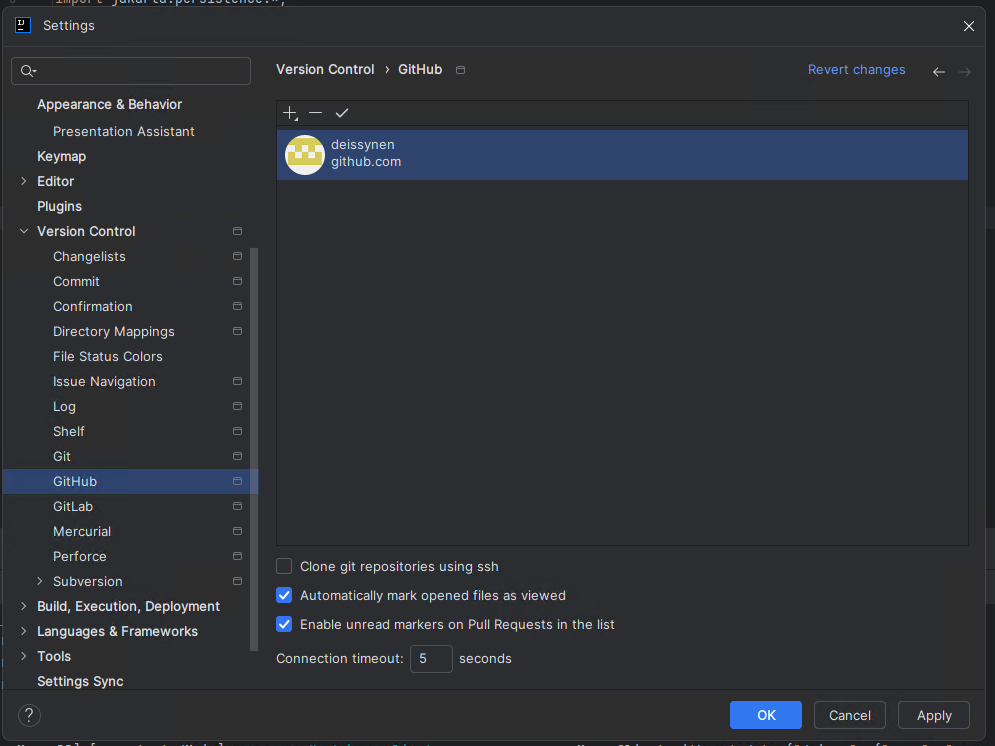
Integramos las dependencias necesarias para la base de datos no relacional **Spring Data Mongo DB**



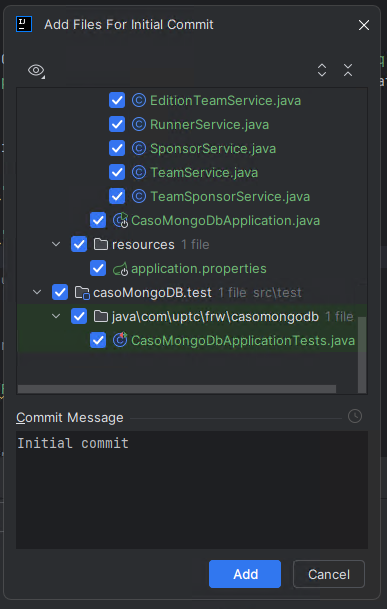
## Configuración del Repositorio del proyecto en GIT



Asociamos el repositorio Git en el proyecto IntelliJ



Una vez creada la rama Master se comparte los cambios generados desde Local a Master, proceso que se realiza para todo el proyecto.



Se comparte Link de la Rama para consulta: <https://github.com/deissynen/casoMongoDb.git>