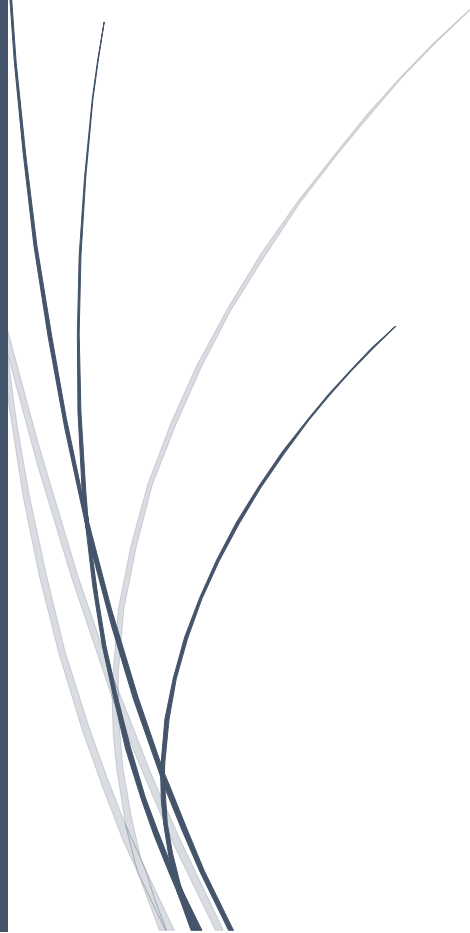


13-1-2024

RethinkDB

BDD



Jose Miguel García Navarro
2ºDAM ACCESO A DATOS

Contenido

1.	Presentación	2
2.	Características principales:	2
2.1.	Modelo de Datos Documental:	2
2.2.	Actualizaciones en Tiempo Real:	2
2.3.	Lenguaje de Consulta ReQL:	2
2.4.	Escalabilidad Horizontal:	2
2.5.	Soporte para Geodatos:	2
2.6.	Licencia Open Source:	2
3.	Casos de uso:.....	3
3.1.	Aplicaciones en Tiempo Real:	3
3.2.	Análisis Geoespacial:.....	3
3.3.	Sistemas Colaborativos:	3
3.4.	Internet de las Cosas (IoT):	3
4.	Ventajas:	3
4.1.	Actualizaciones en Tiempo Real:	3
4.2.	Modelo de Datos Flexible:	3
4.3.	Escalabilidad Horizontal Simple:	3
4.4.	ReQL:.....	3
5.	Desventajas:	4
5.1.	Madurez de la Comunidad:	4
5.2.	Complejidad Operativa:	4
5.3.	Menor Adopción en Comparación con Otros Sistemas:	4
6.	Instrucciones de uso:	4
7.	Creación de la BD y operaciones básicas CRUD con Java:	6
8.	Conclusión.....	10

1. Presentación

RethinkDB es un sistema de gestión de bases de datos NoSQL orientado a documentos y diseñado para facilitar el desarrollo de aplicaciones en tiempo real. Su arquitectura distribuida y su capacidad para gestionar flujos de datos en tiempo real lo hacen ideal para aplicaciones que requieren actualizaciones en tiempo real y escalabilidad horizontal.

2. Características principales:

2.1. Modelo de Datos Documental:

RethinkDB utiliza un modelo de datos basado en documentos JSON. Cada registro en la base de datos es un documento JSON, lo que facilita el almacenamiento de datos estructurados y no estructurados.

2.2. Actualizaciones en Tiempo Real:

Proporciona un sistema de cambios que permite a las aplicaciones suscribirse a actualizaciones en tiempo real. Esto es esencial para aplicaciones que requieren notificaciones instantáneas o información en tiempo real.

2.3. Lenguaje de Consulta ReQL:

RethinkDB utiliza ReQL (RethinkDB Query Language), un lenguaje de consulta expresivo que facilita la manipulación de datos y la realización de consultas avanzadas.

2.4. Escalabilidad Horizontal:

Puede escalarse fácilmente distribuyendo la carga en varios nodos, permitiendo el crecimiento de la capacidad de almacenamiento y la mejora del rendimiento de manera sencilla.

2.5. Soporte para Geodatos:

Ofrece funciones avanzadas para trabajar con datos geoespaciales, lo que lo hace adecuado para aplicaciones que requieren análisis y manipulación de datos de ubicación.

2.6. Licencia Open Source:

RethinkDB se distribuye bajo la licencia de código abierto Apache 2.0, permitiendo su uso, modificación y distribución de forma gratuita.

3. Casos de uso:

3.1. Aplicaciones en Tiempo Real:

Ideal para aplicaciones de chat, monitoreo en tiempo real, juegos en línea y cualquier aplicación que requiera actualizaciones instantáneas.

3.2. Análisis Geoespacial:

Útil en aplicaciones que gestionan datos de ubicación, como mapas interactivos, seguimiento de activos y análisis de patrones geográficos.

3.3. Sistemas Colaborativos:

Aplicaciones colaborativas donde múltiples usuarios acceden y editan datos simultáneamente pueden beneficiarse de la capacidad de RethinkDB para manejar concurrencia y actualizaciones en tiempo real.

3.4. Internet de las Cosas (IoT):

Excelente para almacenar y analizar datos generados por dispositivos IoT, ya que puede manejar grandes cantidades de datos y proporcionar actualizaciones en tiempo real.

4. Ventajas:

4.1. Actualizaciones en Tiempo Real:

RethinkDB es excepcionalmente fuerte en la entrega de actualizaciones en tiempo real, proporcionando una experiencia de usuario más dinámica.

4.2. Modelo de Datos Flexible:

Almacena datos en formato JSON, lo que permite una flexibilidad significativa en la representación de la información.

4.3. Escalabilidad Horizontal Simple:

La capacidad de escalar horizontalmente facilita el crecimiento de las aplicaciones sin grandes modificaciones en la infraestructura.

4.4. ReQL:

ReQL es un lenguaje de consulta potente que facilita la manipulación y extracción de datos de manera intuitiva.

5. Desventajas:

5.1. Madurez de la Comunidad:

Aunque activa, la comunidad de RethinkDB no es tan grande como algunas otras bases de datos NoSQL, lo que puede afectar la disponibilidad de recursos y plugins.

5.2. Complejidad Operativa:

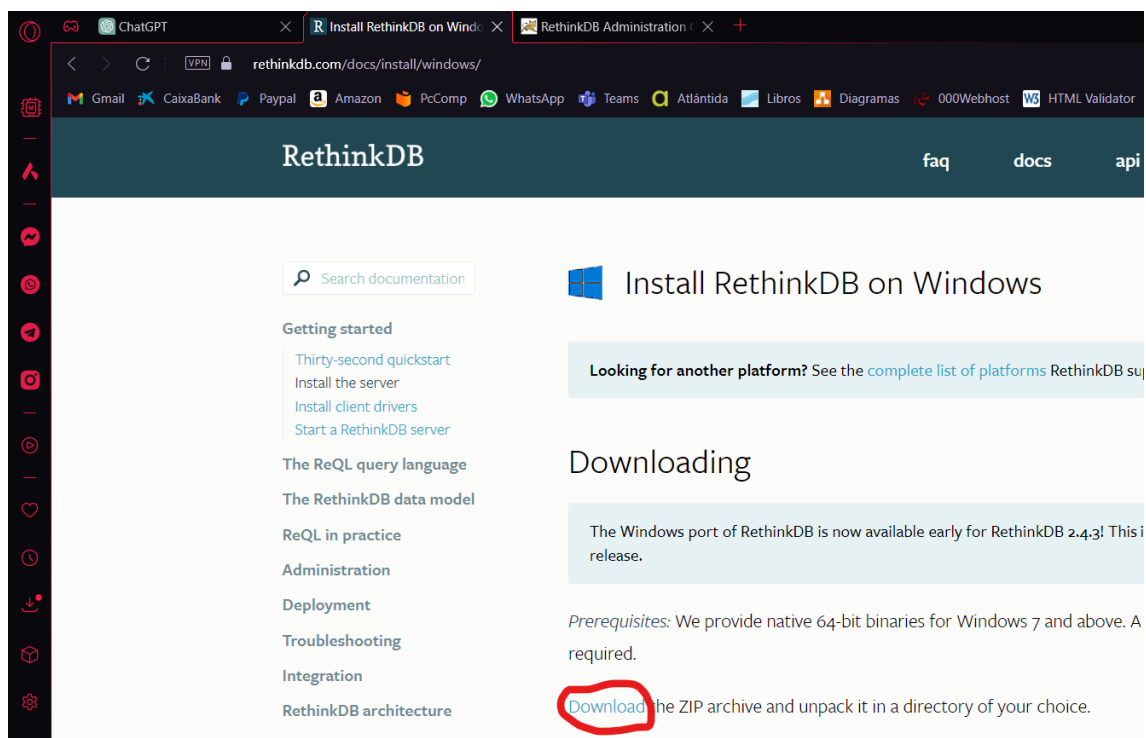
Configurar y mantener un clúster distribuido de RethinkDB puede requerir habilidades específicas, lo que puede ser una desventaja para equipos con recursos limitados.

5.3. Menor Adopción en Comparación con Otros Sistemas:

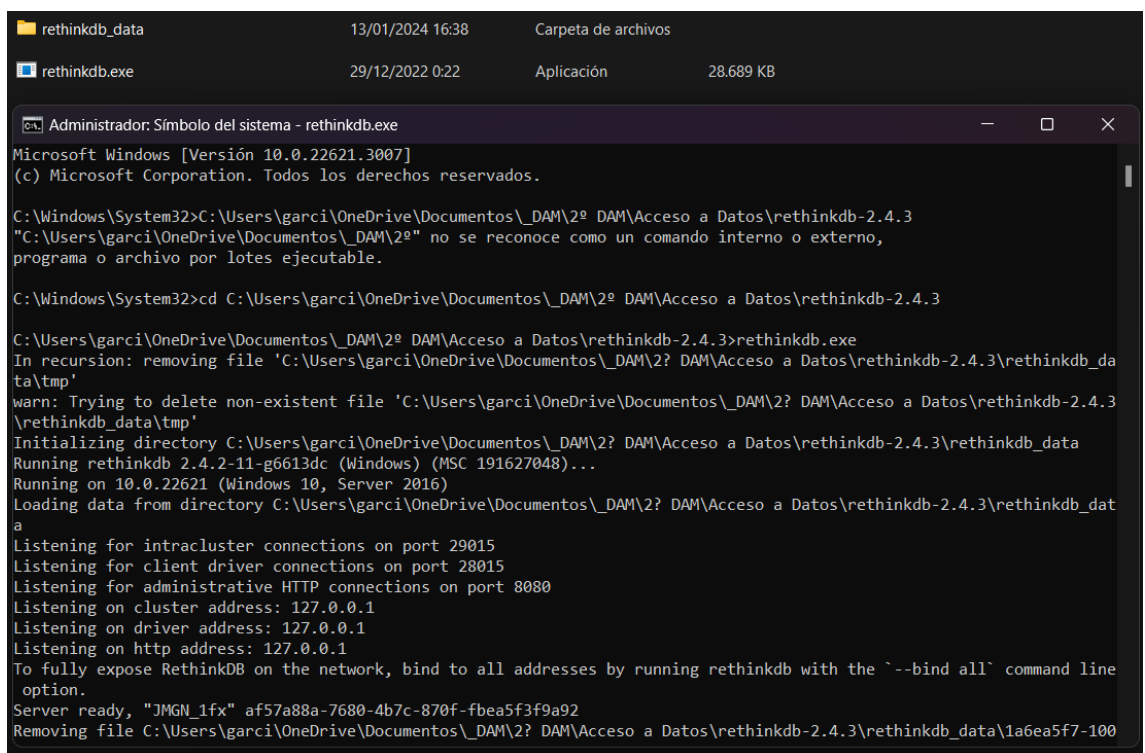
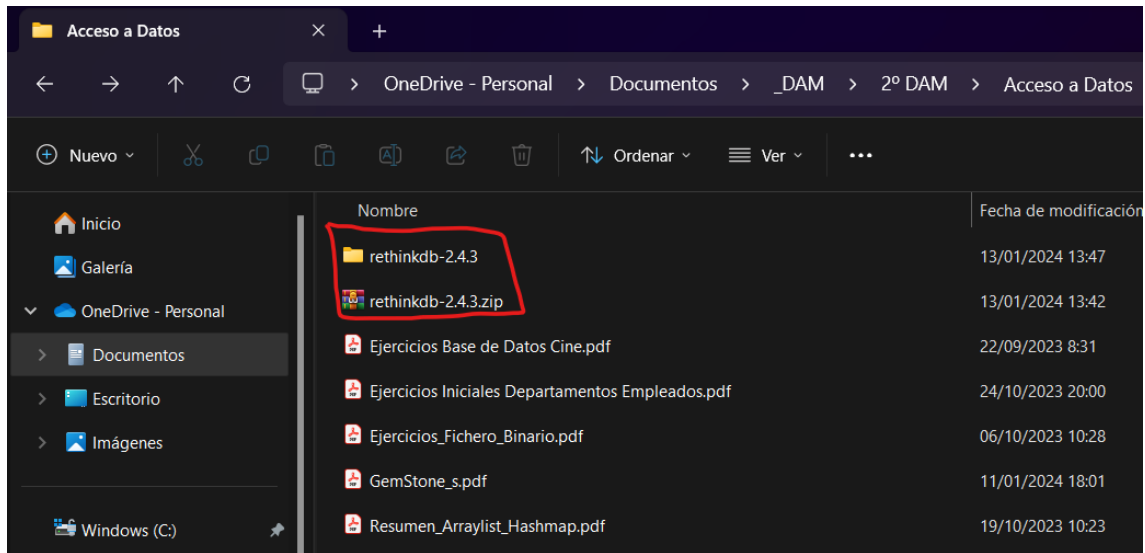
RethinkDB puede no ser la opción predilecta en entornos donde ya existen bases de datos NoSQL ampliamente adoptadas.

6. Instrucciones de uso:

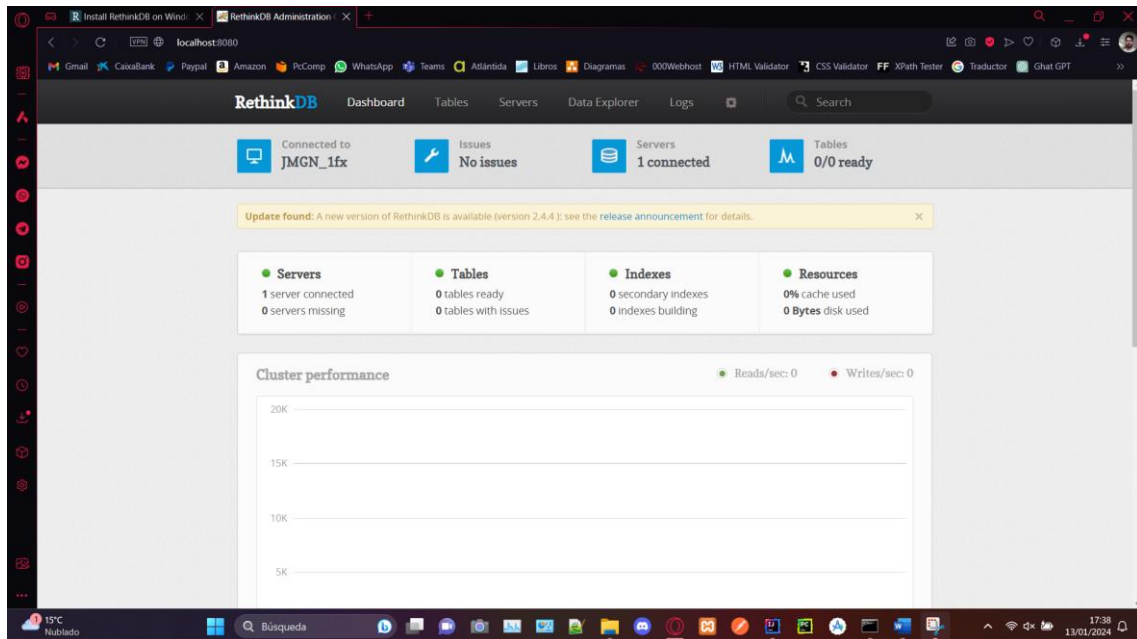
Entrar en la web oficial de RethinkDB e ir a la sección de [descargas para windows](#) y descargar el archivo que te proporciona la web.



Una vez descomprimido abrir la consola de Windows como administrador e ir hasta la ruta donde se encuentra el archivo rethinkdb.exe con cd y el path correspondiente y luego ejecutar el archivo desde la consola para inicializar el servidor.



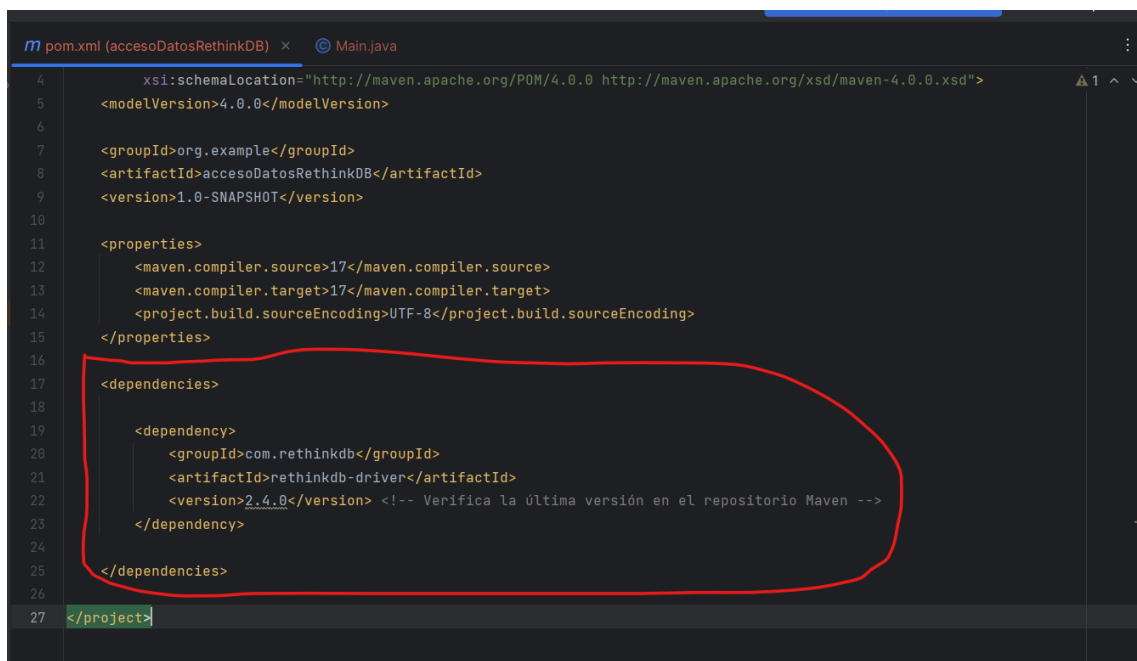
Para entrar a la base de datos tendremos que hacerlo desde el navegador web escribiendo en la barra de direcciones "localhost:8080" sin comillas y entraremos.



Arriba a la izquierda vemos que no existen tablas creadas, aquí es donde entra Java crearemos un código para crear la base de datos, las tablas (en este caso una) y sus correspondientes datos.

7. Creación de la BD y operaciones básicas CRUD con Java:

Utilizamos IntelliJ para crear el código Java. Lo primero que tenemos que hacer ya que RethinkDB no trabaja de forma nativa con java es introducir en nuestro proyecto las dependencias correspondientes para que todo funcione.



Una vez insertamos las dependencias actualizamos Maven para que las pueda usar. Y empezamos con el código.

```
m pom.xml (accesoDatosRethinkDB)  © Main.java x
1 package org.example;
2
3 import com.rethinkdb.RethinkDB;
4 import com.rethinkdb.net.Connection;
5 import com.rethinkdb.net.Cursor;
6
7 import java.util.HashMap;
8 import java.util.Map;
9
10 no usages
11 public class Main {
12
13     13 usages
14     private static final RethinkDB r = RethinkDB.r;
15     3 usages
16     private static final String dbName = "miBaseDatos";
17     7 usages
18     private static final String tableName = "2DAM";
19
20 no usages
21 public static void main(String[] args) {
22     Connection conn = r.connection().hostname("localhost").port(28015).connect();
23
24     // Crear la base de datos solo si no existe
25     if (!r.dbList().contains(dbName).run(conn).equals(true)) {
26         r.dbCreate(dbName).run(conn);
27     }
28 }
```

```
m pom.xml (accesoDatosRethinkDB)  © Main.java x
24 // Usar la base de datos recién creada
25 conn.use(dbName);
26
27 // Crear la tabla solo si no existe
28 if (!r.tableList().contains(tableName).run(conn).equals(true)) {
29     r.tableCreate(tableName).run(conn);
30 }
31
32 // Insertar datos de ejemplo
33 insertarAlumno(conn, dni: "12345678A", nombre: "Juan", apellidos: "Pérez", notaMedia: 7.5);
34 insertarAlumno(conn, dni: "98765432B", nombre: "Ana", apellidos: "Gómez", notaMedia: 4.8);
35 insertarAlumno(conn, dni: "11122334C", nombre: "Pedro", apellidos: "López", notaMedia: 6.2);
36 insertarAlumno(conn, dni: "55566666D", nombre: "Sofía", apellidos: "Martínez", notaMedia: 8.9);
37 insertarAlumno(conn, dni: "77778888E", nombre: "Luis", apellidos: "Fernández", notaMedia: 3.5);
38 insertarAlumno(conn, dni: "99990000F", nombre: "Laura", apellidos: "García", notaMedia: 9.7);
39 insertarAlumno(conn, dni: "12121212G", nombre: "Miguel", apellidos: "Rodríguez", notaMedia: 5.1);
40 insertarAlumno(conn, dni: "13131313H", nombre: "María", apellidos: "Díaz", notaMedia: 6.8);
41 insertarAlumno(conn, dni: "14141414I", nombre: "Carlos", apellidos: "Hernández", notaMedia: 4.3);
42 insertarAlumno(conn, dni: "15151515J", nombre: "Elena", apellidos: "Sánchez", notaMedia: 7.0);
43
44 // Mostrar todos los alumnos
45 System.out.println("Alumnos antes de la actualización:");
46 mostrarAlumnos(conn);
```

En estas dos capturas conectamos IntelliJ con RethinkDB, creamos la base de datos llamada “MiBaseDatos”, una tabla llamada “2DAM” e insertamos los datos con instancias al igual que después mostramos los registros insertados en la consola de IntelliJ.


```
Run
{apellidos=Díaz, aprobado=true, nota_media=6.8, id=1ff8dbac-b690-41cf-a091-d81b5f409366, nombre=Maria, dni=13131313H}
{apellidos=Rodríguez, aprobado=true, nota_media=5.1, id=a34ee384-591b-4c8a-bb8f-fbcd7d3a24e, nombre=Miguel, dni=12121212G}
{apellidos=Fernández, aprobado=false, nota_media=3.5, id=fd3c8c77-a22c-4c5d-a2bd-8639431f5d88, nombre=Luis, dni=77778888E}
{apellidos=García, aprobado=true, nota_media=9.7, id=bf7049c5-a983-4534-9151-9880d0b201d0, nombre=Laura, dni=99990000F}
{apellidos=Gómez, aprobado=false, nota_media=4.8, id=87d6c1a2-5bc1-413f-b526-01113686aff4, nombre=Ana, dni=98765432B}
{apellidos=Pérez, aprobado=true, nota_media=7.5, id=c6e8db88-4dd7-4643-b43e-8bf8164db90c, nombre=Juan, dni=12345678A}
{apellidos=López, aprobado=true, nota_media=6.2, id=98a52734-8d62-496d-b8f4-3a29605e9030, nombre=Pedro, dni=11122334C}

Process finished with exit code 0

accesoDatosRethinkDB > src > main > java > org > example > Main > main
```

Ahora en la base de datos si mostrará que existe una tabla

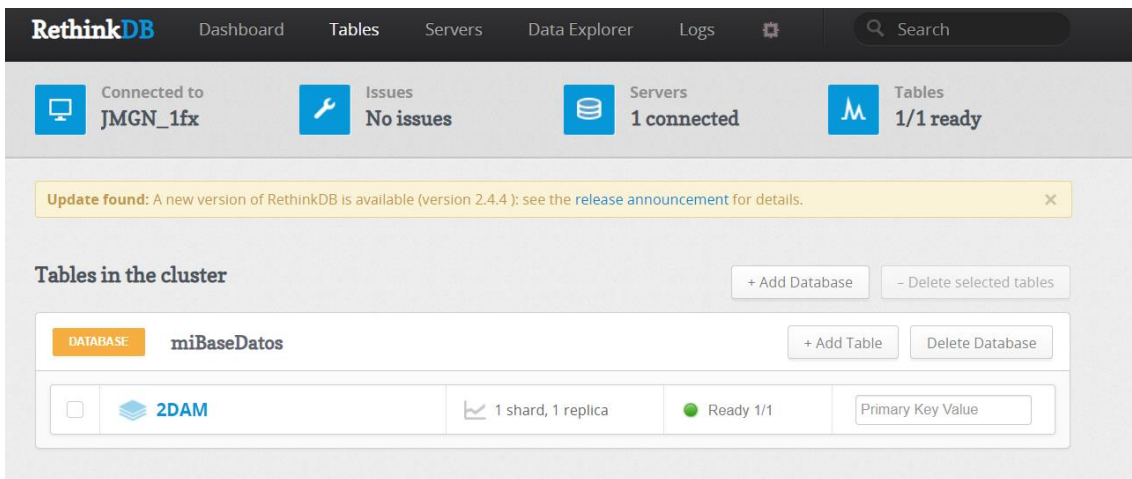
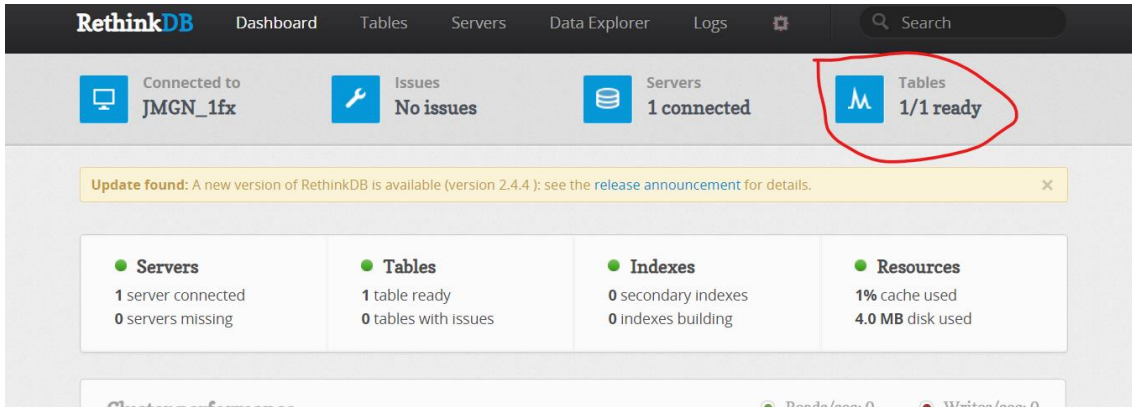


Table Viewer

Lookup: key

id	apellidos	aprobado	dni	nombre	nota_media
064afabb-6ed...	Martínez	true	55556666D	Sofía	8.9
1ff8dbac-b69...	Díaz	true	13131313H	María	6.8
332180d1-3c4...	Sánchez	true	15151515J	Elena	7
87d6c1a2-5bc...	Gómez	false	98765432B	Ana	4.8
98a52734-8d6...	López	true	11122334C	Pedro	6.2
a34ee384-591...	Rodríguez	true	12121212G	Miguel	5.1
bf7049c5-a98...	García	true	99990000F	Laura	9.7
c6e8db88-4dd...	Pérez	true	12345678A	Juan	7.5
f4903278-42f...	Hernández	false	14141414I	Carlos	4.3
fd3c8c77-a22...	Fernández	false	77778888E	Luis	3.5

Vemos que efectivamente se creo la base de datos con su tabla y datos.

Con el resto del código podemos mostrar, modificar y eliminar registros de la tabla a nuestro antojo. En este caso mostramos todos los registros de la tabla y modificamos u eliminamos por buscando por DNI. También hay un comando para eliminar todos los registros de la tabla de golpe.

```
m pom.xml (accesoDatosRethinkDB)  Main.java x
47
48 // Actualizar un alumno
49 actualizarAlumno(conn, dni: "12345678A", nuevoNombre: "Juanito", nuevosApellidos: "Pérez", nuevaNotaMedia: 8.0);
50
51 // Mostrar todos los alumnos después de la actualización
52 System.out.println("Alumnos después de la actualización:");
53 mostrarAlumnos(conn);
54
55 // Eliminar un alumno
56 eliminarAlumno(conn, dni: "12345678A");
57
58 // Mostrar todos los alumnos después de la eliminación
59 System.out.println("Alumnos después de la eliminación:");
60 mostrarAlumnos(conn);
61
62 // Borrar todos los alumnos
63 borrarTodosLosAlumnos(conn);
64
65 // Mostrar todos los alumnos después de borrarlos
66 System.out.println("Alumnos después de borrar todos:");
67 mostrarAlumnos(conn);
68
69 // Cerrar la conexión
70 conn.close();
71 }
```

En las capturas anteriores solo se mostraban las llamas a los métodos siguientes:

```
10 usages
73 private static void insertarAlumno(Connection conn, String dni, String nombre, String apellidos, double notaMedia) {
74     Map<String, Object> alumno = new HashMap<>();
75     alumno.put("dni", dni);
76     alumno.put("nombre", nombre);
77     alumno.put("apellidos", apellidos);
78     alumno.put("nota_media", notaMedia);
79     alumno.put("aprobado", notaMedia >= 5.0);
80
81     r.table(tableName).insert(alumno).run(conn);
82 }
83
84 4 usages
84 private static void mostrarAlumnos(Connection conn) {
85     Cursor<Map<String, Object>> cursor = r.table(tableName).run(conn);
86     cursor.forEachRemaining(System.out::println);
87 }
88
89 1 usage
89 private static void actualizarAlumno(Connection conn, String dni, String nuevoNombre, String nuevosApellidos, double nuevaNotaMedia) {
90     r.table(tableName).table(tableName)
91         .filter(r.hashMap( key: "dni", dni)) Filter
92         .update(
93             r.hashMap( key: "nombre", nuevoNombre)
94             .with("apellidos", nuevosApellidos)
95             .with("nota_media", nuevaNotaMedia)
96             .with("aprobado", nuevaNotaMedia >= 5.0)
97         ) Update
98     .run(conn);
99 }
```

```

100
101 1 usage
102 private static void eliminarAlumno(Connection conn, String dni) {
103     r.table(tableName).Table
104     .filter(r.hashMap( key: "dni", dni)) Filter
105     .delete() Delete
106     .run(conn);
107 }
108
109 1 usage
110 private static void borrarTodosLosAlumnos(Connection conn) {
111     r.table(tableName).delete().run(conn);
112 }

```

8. Conclusión

RethinkDB me ha parecido un gestor de bases de datos intuitivo y fácil de instalar y poner en marcha. A pesar de que no lo he investigado en profundidad a servido para el propósito para lo que lo he utilizado y no ha sido demasiado difícil encontrar como gestionarlo ya que la misma web te proporciona instrucciones básicas, eso sí, en inglés.