Universidad Nacional Autónoma de México
Facultad de Ingeniería

Proyecto Final

Bases de Datos No Estructuradas

Grupo 0600

Profesor:

Ing. Jorge Alberto Rodríguez Campos

Alumno:

Núñez Quintana Luis Axel

Ciudad de México — 07 de junio, 2024

Índice

Proyecto Final

Núñez Quintana Luis Axel

07 de junio, 2024

1. Planteamiento

El presente proyecto tiene como objetivo desarrollar una aplicación web que haga uso de una base de datos políglota, permitiendo a los usuarios registrarse, iniciar sesión, crear y compartir contenido de manera eficiente. La aplicación está diseñada para manejar grandes volúmenes de datos, ofreciendo un rendimiento y escalabilidad.

Para lograr estos objetivos, se emplea una combinación de bases de datos relacionales y no relacionales. La base de datos relacional se utilizará para manejar datos estructurados y garantizar la integridad de los datos. Se empleará Oracle al tratar con datos de usuarios y credenciales de inicio de sesión. Por otro lado, las bases de datos no relacionales se encargarán de gestionar datos no estructurados y semi-estructurados, permitiendo una mayor flexibilidad y velocidad en la recuperación y almacenamiento de información. Principalmente se encargarán de almacenar los datos de las sesiones de la página web, el contenido proveniente de la misma y también las relaciones que puedan surgir de dicho contenido.

En esta etapa inicial del proyecto, el enfoque se centrará exclusivamente en la capa de almacenamiento de datos y en la primera etapa del backend, utilizando Node.js y Express.js. Se busca implementar una API RESTful que gestione eficazmente las operaciones de registro, autenticación y gestión de contenido de los usuarios.

El desarrollo de esta aplicación no solo busca satisfacer las necesidades actuales de los usuarios, sino también prepararse para un crecimiento futuro. La elección de una base de datos políglota, junto con un backend eficaz y adaptable, asegura que la aplicación pueda escalar vertical y horizontalmente. Esto permite manejar incrementos en la cantidad de usuarios y volumen de datos sin comprometer el rendimiento o la experiencia del usuario final.

2. Modelado

La primera etapa del proyecto se enfocó en el modelado de la capa de almacenamiento, estableciendo la arquitectura sobre la cual se desarrollará la aplicación web. En esta fase inicial, se definieron las responsabilidades específicas de cada tipo de base de datos.

Se crearon esquemas detallados para cada base de datos, asegurando que tanto los datos estructurados como los no estructurados sean gestionados de manera eficiente y coherente con los objetivos del proyecto.

Esta fase no solo estableció la arquitectura sobre la cual se desarrollará la aplicación web, sino que también sentó las bases sólidas para una implementación exitosa, asegurando que cada base de datos cumpla con su papel específico.

2.1. Oracle

En primera instancia, se delegó el manejo de datos de las cuentas de los usuarios a la base de datos relacional Oracle. Esta base de datos será la encargada de almacenar las credenciales de los usuarios y la información de sus cuentas, incluyendo roles y permisos dentro de la aplicación. La elección de Oracle se fundamenta en su capacidad para gestionar de manera segura campos sensibles, como las contraseñas de los usuarios.

A continuación, se muestra el diagrama relacional de la base de datos.

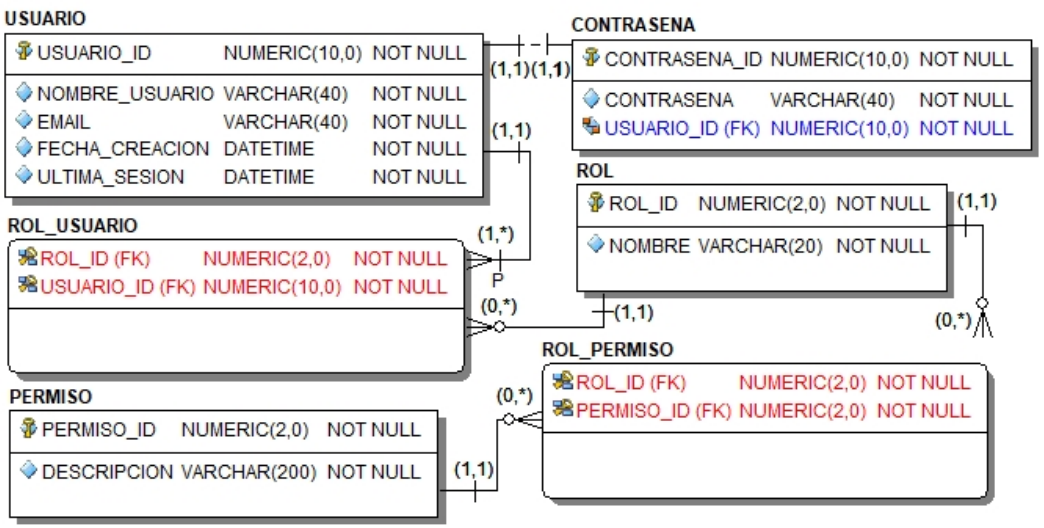

Figura 1: Modelo Relacional

2.2. Riak KV

Para gestionar los datos de la sesión de usuarios se decidió emplear Riak KV. Cada sesión de usuario se almacena como un objeto JSON asociado a una clave única, lo que permite un acceso rápido y flexible a datos como las preferencias de usuario, configuraciones temporales y registros de actividad.

A continuación, se describe cómo se puede estructurar y manejar el JSON proporcionado usando Riak KV:

```
"sesion_id" : {
    "usuario_id" : <usuario_id>,
    "inicio" : <fecha_inicio>,
    "ultimo_acceso" : <fecha_ultimo_acceso>,
    "preferencias" : {
        "tema" : <oscuro || claro>,
        "idioma" : <idioma>,
        "vista" : <expandida || compacta>
    },
    "configuraciones_temporales" : {
        "diagrama_abierto" : {
            "diagrama_id" : <diagrama_id>,
            "estado" : <estado>,
            "ultima_actualización" : <fecha>
      },
      "busqueda_reciente" : {
        "query" : <query>,
        "filtros" : {
            "tipo" : <'ER' || 'R' || 'F'>,
            "fecha" : <fecha>
        }
      }
    }
}
```

2.3. MongoDB

Continuando, se eligió utilizar MongoDB para gestionar los documentos resultantes de la aplicación de manera eficiente. Cada diagrama se almacena como un documento BSON en una colección, permitiendo una estructura flexible y anidada que incluye entidades, relaciones, y sus atributos correspondientes.

A continuación, se muestra un ejemplo del documento de un diagrama:

```
{
  "_id": <diagrama_id>,
  "usuario_id": <usuario_id>,
  "nombre_diagrama": <Titulo diagrama>,
  "tipo_diagrama": <ER || R || F>,
  "contenido": {
    "entidades": [
      {
        "nombre": <nombre_entidad>,
        "atributos": [<atributo_1>, <atributo_2>, ... , <atributo_n>]
      },
    ],
    "relaciones": [
      {
        "nombre": <nombre_relacion>,
        "entidades": [<entidad_1>, <entidad_2>],
        "cardinalidad": <N:M>
      }
    ]
  },
  "fecha_creacion": <fecha_creacion>,
  "fecha_modificacion": <fecha_modificacion>
}
```

2.4. Neo4J

Finalmente, se emplea Neo4j para la gestión de relaciones entre usuarios y sus diagramas. Las entidades como usuario, diagrama y etiqueta se representan como nodos, mientras que las relaciones entre ellos se modelan como aristas. Esto incluye relaciones como sigue, crea, comenta, comparte, favorito y tiene. Este modelado permite una representación natural y eficiente de la interconexión entre usuarios y sus actividades en la aplicación.

A continuación, se ilustra su uso mediante un diagrama.

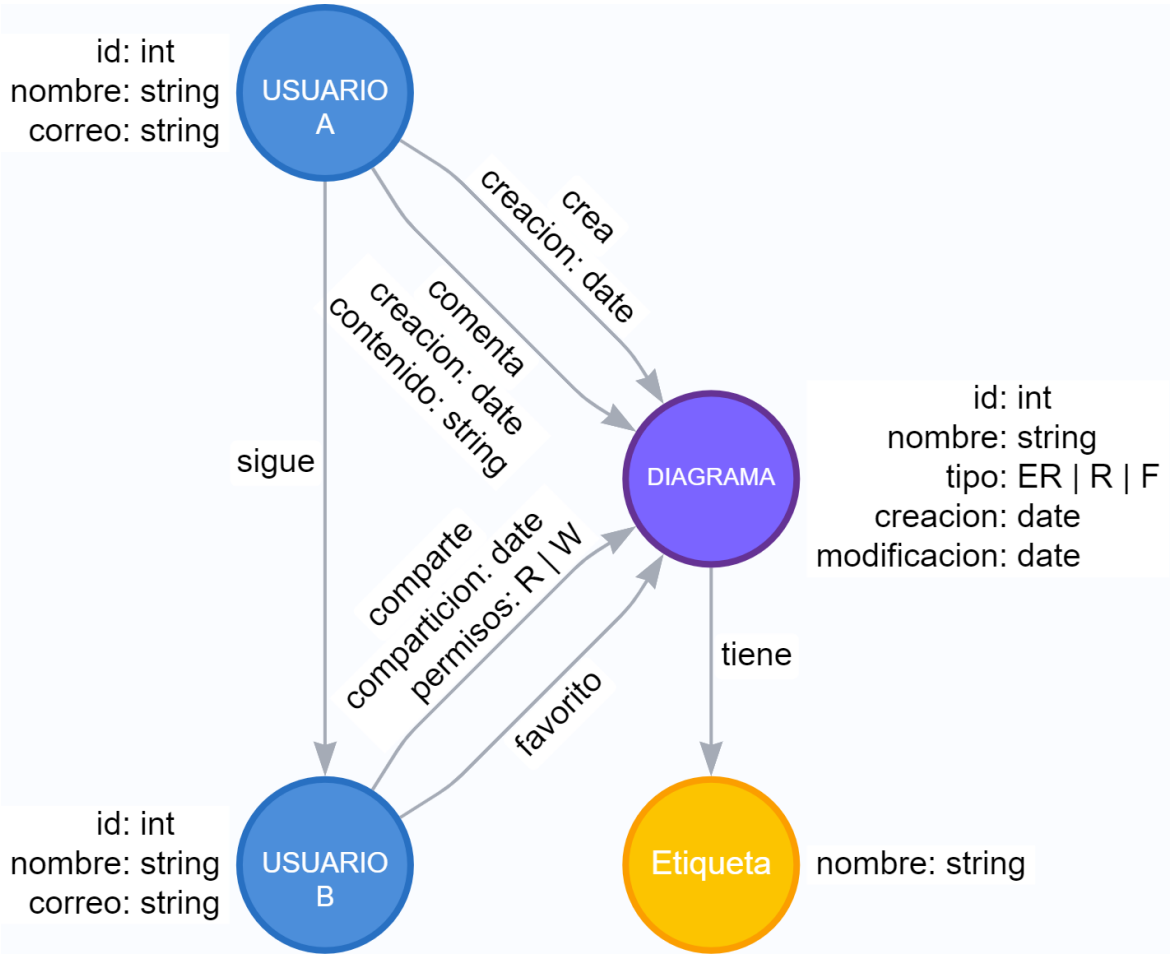

Figura 2: Diagrama de Neo4J

3. Capa de almacenamiento

3.1. Oracle

3.1.1. Entorno

3.1.2. Creación de entidades

3.1.3. Carga inicial

3.2. Riak KV

3.2.1. Entorno

3.2.2. Carga inicial

3.3. MongoDB

3.3.1. Entorno

3.3.2. Creación de esquema

3.3.3. Carga inicial

3.4. Neo4J

3.4.1. Entorno

3.4.2. Creación de entidades inicial

3.4.3. Creación de relaciones inicial

4. Backend

4.1. Comunicación

4.1.1. Oracle

4.1.2. Riak KV

4.1.3. MongoDB

4.1.4. Neo4J

4.2. Métodos CRUD

4.2.1. Oracle

4.2.2. Riak KV

Referencias

[1] Basho Technologies. *Riak KV Documentation*. Recuperado: 2024-05-29. 2024. URL: https://docs.riak.com/riak/kv/latest/.

[2] MongoDB, Inc. *MongoDB Documentation*. Recuperado: 2024-05-29. 2024. URL: https://docs.mongodb.com/.

[3] Neo4j, Inc. *Neo4j Documentation*. Recuperado: 2024-05-29. 2024. URL: https://neo4j.com/docs/.

[4] Oracle Corporation. *Oracle Database Documentation*. Recuperado: 2024-05-29. 2024. URL: https://docs.oracle.com/en/database/.