Technologic Of Costa Rica

Computer Engineer

IC4302 - Bases de Datos 2

Project 1

Professor

Erick Fabian Hernández Bonilla

Student

Walter A. Segura Barboza/2023072838

Hidalgo Paz Carmen/2020030538

Ilama Gamboa Naomi Joseph/ 2021114064

Morales Vargas David Enrique/2021052762

October, 2025

Table of Contents

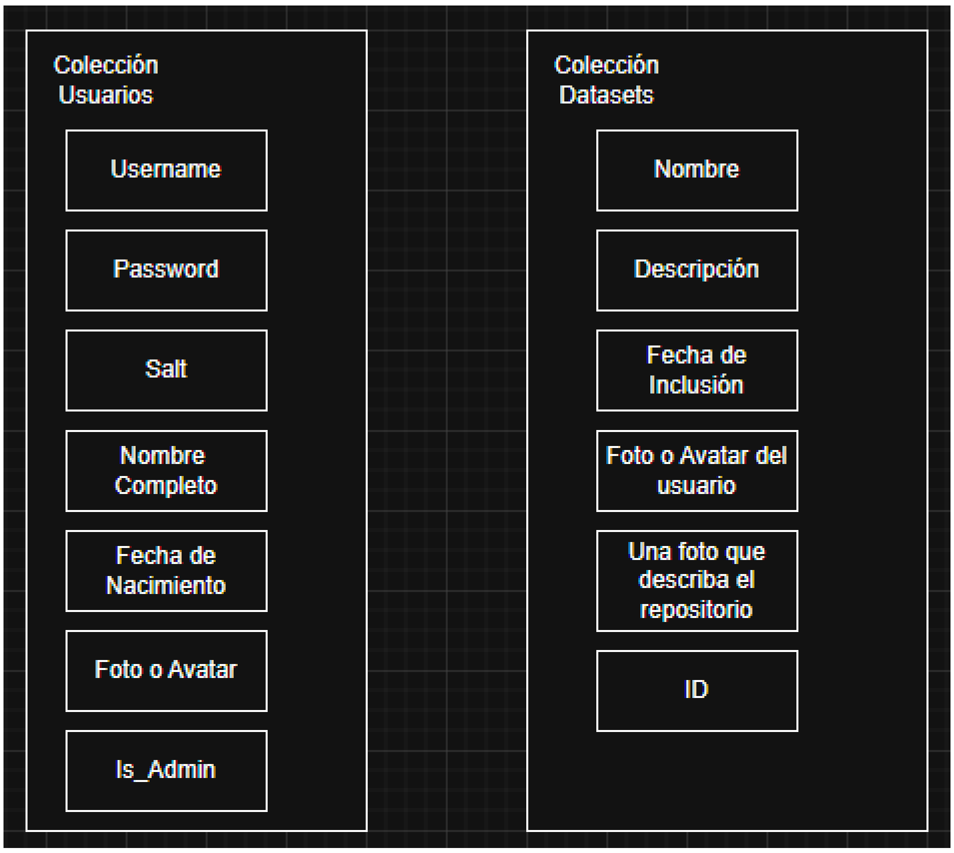
Figure Index

The databases selected for this project are: MongoDB, Neo4j, Redis and Cassandra

MongoDB

This NoSQL database saves information via documents, for this reason it was selected to manage all the information of the users and the datasets. Mongo has a system where you can pass a lot of information with a simple JSON file from the interface to the database. Since Users and Datasets are pretty close to the interface and have a lot of data that can be simply put in a JSON file, saving them in this database facilitates the work of saving all this information as well as reading it to show it in the interface.

Figure 1: MongoDB diagram



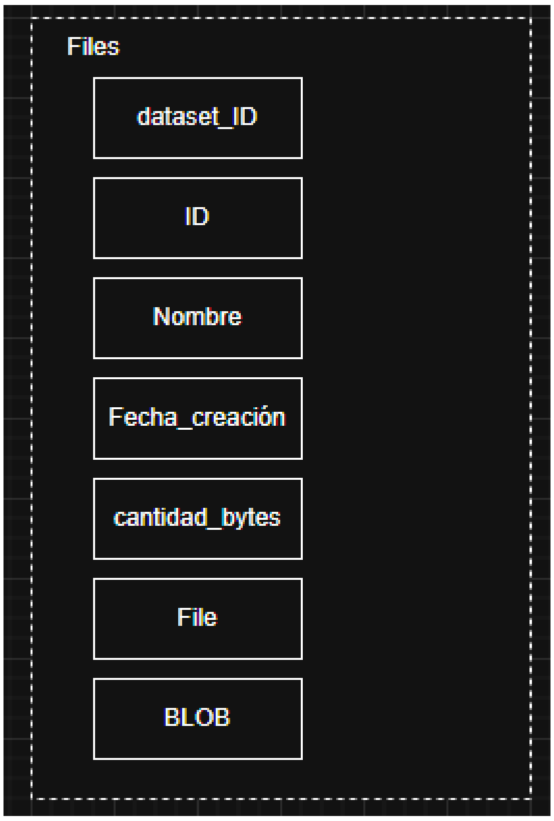
Source: own draw.io

As shown in Figure 1, Mongo saves all the basic information for the users, along with an identifier to know if they are admins or not, as well as all of a dataset's basic information.

Cassandra

Cassandra is a peculiar database, since it works similar to SQL. It uses the same system of tables, rows and columns, but instead of working with tables and rows, it uses columns as their primary storage to save and search for information. Being able to read, update or insert information just by accessing the necessary information stored in columns is very useful to handle large volumes of information, in this case BLOBs. The easy mechanics to access just the necessary columns and the BLOBs in Cassandra were the main reason to use Cassandra as the one in charge of saving the internal data of the datasets, like all the files and large volumes of text, videos and all objects you can upload to it.

Figure 2: Cassandra diagram



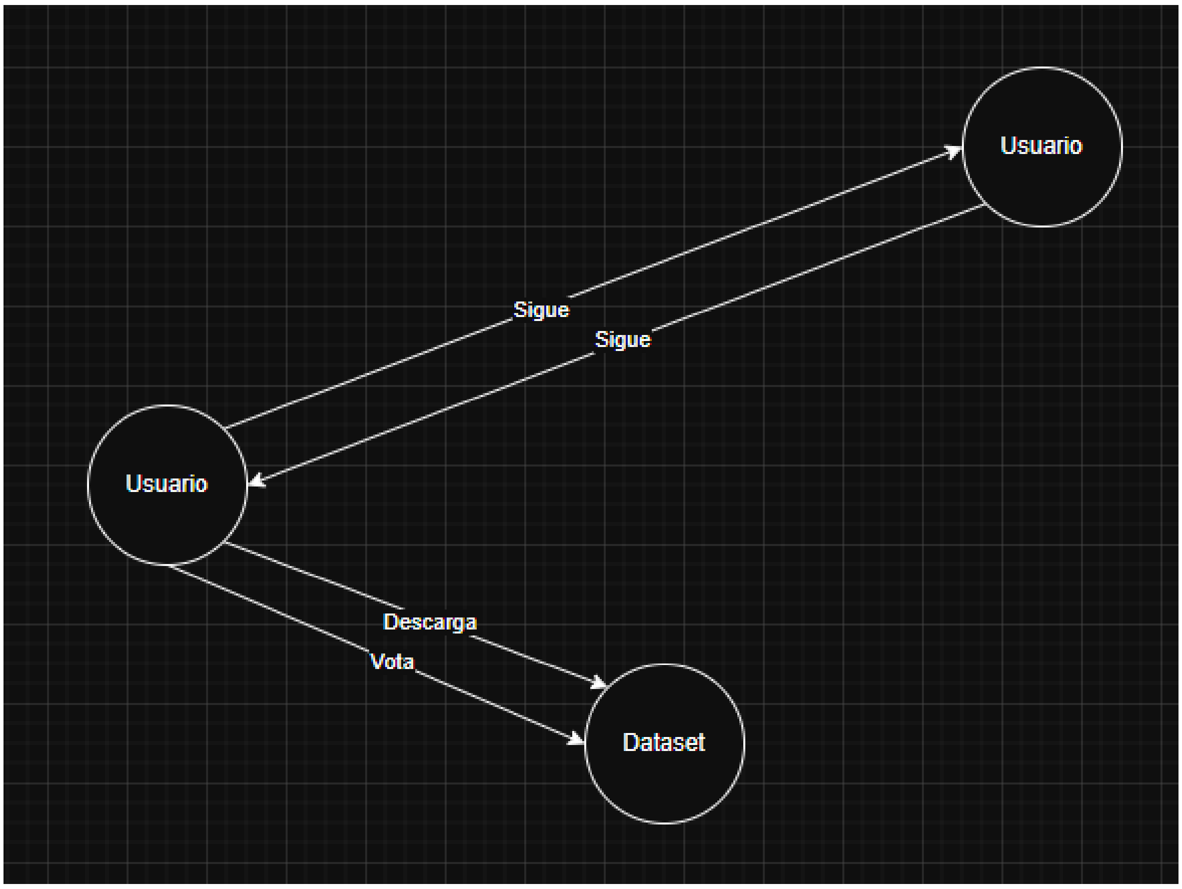
Source: own draw.io

Here Cassandra will save the volumes of information in a BLOB space, with all the information to recognize and read the dataset's file information.

Neo4j

This is a database that works with graphs. Each node in the graph is an object where you can save information and the connection between the nodes are relationships between them. These relations can also have their own properties, like a name or even some important data. All reads, insertion and upload operations are made with graph algorithms, making this type of DB good for nodes with a lot of relationships, for this reason it was the best option to save the follower system, downloads and votes system. The project requires a follower system for the users. A user can follow another one and should obtain certain information if they are following someone. This can be easily made by making all users be a node, saving only the important information like the ID, and make relationships between all the nodes to know all the users someone is following. Another point is the download and vote system. Datasets can be voted and downloaded, this information has to be saved in a way we can know how many votes a specific user gave to the dataset, and who downloaded it, making the datasets also nodes with only the necessary information like the ID and some counters, and make the relationships for downloads and votes. Votes in this case could also save the amount of votes that the user gave to the dataset.

Figure 3: Neo4j diagram

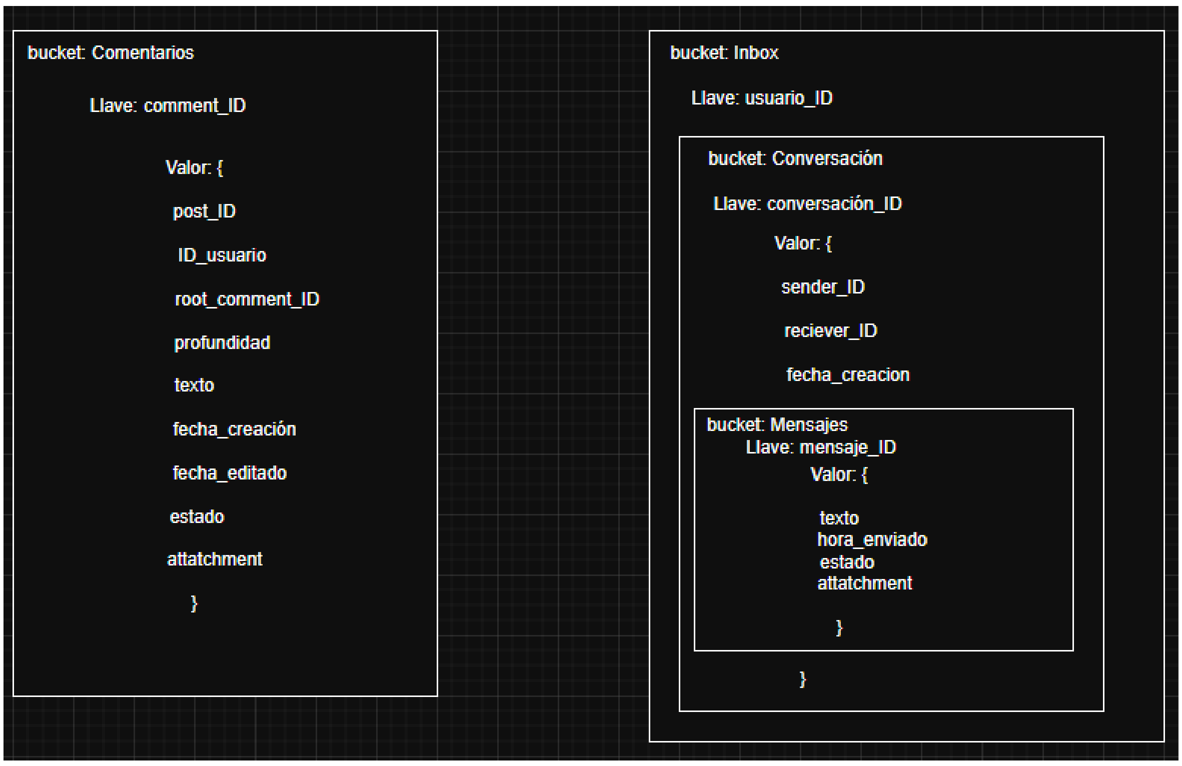


Source: own draw.io

Redis

The last thing we need are the comments system, for this we will use Redis which is an already used database for these jobs. It uses a hashtable to save data with keys, which will be very useful to obtain all the comments saved of a certain dataset, but more importantly it has very useful data structures to handle and save the comments, so we can save all the hierarchy of comments in one object identified by the dataset's ID.

Figure 4: Redis diagram



Source: own draw.io

Bibliography

Neo4j. (n.d.). Clustering with Docker. Neo4j Graph Data Platform.https://neo4j.com/docs/operations-manual/5/tutorial/tutorial-clustering-docker/

Neo4j. (n.d). Connect. Neo4j Graph Data Platform: JavaScript Driver Manual. https://neo4j.com/docs/javascript-manual/current/connect/

Redis (n.d). Node.js. Redis Documentation. https://redis.io/docs/latest/develop/clients/nodejs/

Docker. (n.d.). Install Docker Engine on Ubuntu. Docker Docs. https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository

Docker. (n.d.). Install Docker Compose on Linux. Docker Docs. https://docs.docker.com/compose/install/linux/#install-using-the-repository