

NoSQL Document Databases Assessment: Couchbase, CouchDB, and MongoDB

Inês Carvalho¹^a, Filipe Sá^{1,3}^b and Jorge Bernardino^{1,2}^c

¹*Polytechnic of Coimbra – ISEC, Rua Pedro Nunes, Quinta da Nora, 3030-199 Coimbra, Portugal*

²*CISUC – Centre of Informatics and Systems of University of Coimbra, Pinhal de Marrocos, 3030-290 Coimbra, Portugal*

³*CISeD – Research Centre in Digital Services of Polytechnic of Viseu, Campus Politécnico, 3504-510 Viseu, Portugal*

Keywords: NoSQL Document Databases, OSSpal Methodology, Couchbase, CouchDB, MongoDB.

Abstract: NoSQL document databases emerged as an alternative to relational databases to deal with large volumes of data. In this paper, we assess the top three free and open-source NoSQL document databases: Couchbase, CouchDB, and MongoDB. Through this analysis, we identify the main characteristics of each database. The OSSpal methodology, which combines quantitative and qualitative measures to assess open-source software was used. This methodology defines seven categories: functionality, operational software characteristics, software technology attributes, documentation, support and service, community and adoption, and development process. At the end, it is obtained a score that identify which is the best NoSQL document database.

1 INTRODUCTION

The NoSQL Databases, also known as non-relational or not only SQL, emerged in the late 2000. NoSQL databases started as an alternative to relational databases and began to gain more popularity when the leading Web 2.0 companies, such as Amazon and Google, needed to deal with the data volume problem. These companies created their own databases, Dynamo and Big Table, which inspire many current databases (Leavitt, 2010; Elmasri and Navathe, 2016).

NoSQL databases are characterized into four types (Elmasri and Navathe, 2016):

- Key-Value Store databases;
- Column-Oriented databases;
- Document Store databases;
- Graph databases.


NoSQL databases are based on the BASE properties (Basically Available, Soft State, Eventually Consistent), which are characterized by focusing on the availability of data even in the presence of multiple failures and little consistency.


Relational databases are known for being based on ACID properties (Atomic, Consistent, Isolated, Durable), which are characterized by being consistent in their transactions and their data (Abramova, Bernardino, and Furtado, 2014).


Document databases store and organize their data in the form of document collections as JSON (JavaScript Object Notation) format rather than basic columns/rows. Documents from the same collection must be similar and may have different fields unlike tables in relational databases. Document databases do not have a pre-defined structure and do not depend on each other (Tannir, 2013; Elmasri and Navathe, 2016).

The main objective of this work is to evaluate the three most popular free and open source NoSQL document databases: Couchbase, CouchDB, and MongoDB. This choice is according to the DB-Engines Ranking (2022). For this, the OSSpal methodology was used, which allows evaluating open-source software using qualitative and quantitative measures.

The OSSpal methodology emerged from the evolution of the OpenBRR methodology, which is

^a <https://orcid.org/0000-0002-7981-4263>

^b <https://orcid.org/0000-0002-7846-8397>

^c <https://orcid.org/0000-0001-9660-2011>

considered the best methodology to apply in the evaluation of open-source platforms (Ferreira, 2018). This methodology makes it possible to obtain a classification for each database. Based on this classification, we intended to know, which is the best of the three databases.

The rest of this paper is organized as follows. Section 2 explains the OSSpal methodology. Section 3 describes the three NoSQL document databases. Section 4 presents the evaluation of the databases with the application of the OSSpal methodology. Finally, section 5 presents the conclusions and future work.

2 OSSpal METHODOLOGY

The OSSpal methodology aims to help organizations choose free and open-source software. It combines qualitative and quantitative measures for software evaluation in several categories, which results in an accepted value that allows comparing the software platforms (Ferreira, 2018).

The original version of OSSpal methodology consists of seven categories (Wasserman *et al.*, 2017; Oliveira and Bernardino, 2019):

- **Functionality:** Does the software meet the average user's requirements?
- **Operational Software Characteristics:** Is the software secure and scalable? Is it easy to install, configure, deploy, and maintain? How well does the software perform? Does it have a pleasant interface, and is it easy for the end-user to use?
- **Support and Services:** Does it have commercial support and, or community support? Are there people or organizations that can provide training or consulting services?
- **Documentation:** Are there tutorials and documentation for the software?
- **Software Technology Attributes:** How is the software architecture? Is the software modular, portable, flexible, extensible, open, complete, error-free, and easy to integrate? What is the quality of software design, code, and testing?
- **Community and Adoption:** Does the market, industry, and community adoption of the software? Is the software community active?
- **Development Process:** How professional are the development process and the project organization?

The assessment process for all categories, except for the functionality category, consists on four steps:

1. Identify and build a list of software components to be analyzed;
2. Weights must be assigned to the categories and measures:
 - 2.1. Assign a percentage of importance to each category, making a total of 100%;
 - 2.2. For each characteristic within a category, it is necessary to order and classify the characteristic according to its importance;
 - 2.3. For each characteristic within a category, assign a percentage according to its importance, making a total of 100% of all the characteristics of a category;
3. Collect data for each characteristic used in each category and calculate their weight in a range of 1 to 5 (1 – Unacceptable, 2 – Poor, 3 – Acceptable, 4 – Very Good, 5 – Excellent);
4. Calculate the result based on the qualification of each category with its weighting factor.

The category Functionality serves to analyze and evaluate the characteristics that the platforms have or should have. To evaluate this category, we must follow these steps:

1. Define the characteristics to be analyzed, evaluating them from 1 to 3 (from least important to most important);
2. Rank the characteristics in a cumulative sum from 1 to 3;
3. Standardize the previous result on a scale of 1 to 5, according to the follow:
 - Under 65%, Score = 1 (Unacceptable);
 - 65% - 80%, Score = 2 (Poor);
 - 80% - 90%, Score = 3 (Acceptable);
 - 90% - 96%, Score = 4 (Good);
 - Over 96%, Score = 5 (Excellent).

3 NoSQL DOCUMENT DATABASES

There is currently a wide variety and growing number of NoSQL Document databases. Document databases, store and organize their data in the form of document collections. They were created to be easily scaled as they grow, and their main advantage is dealing with unstructured data such as text files, email, and multimedia files (Elmasri and Navathe, 2016).

We selected the top three free and open source Document databases according to the DB-Engines Ranking: MongoDB, Couchbase, and CouchDB (DB-Engines Ranking, 2022).

In the following subsections, we describe the main characteristics, advantages, and weaknesses of each database.

3.1 Couchbase

Couchbase is a document database developed in C++ by Couchbase Inc in 2011. It is designed for web and mobile applications. Couchbase is a schema-free database, where the documents are in JSON format and stored in buckets. A bucket is a collection of documents. Couchbase has their own query language, which is N1QL. Couchbase has three versions: Couchbase Server, Couchbase Mobile, and Developer SDK. Couchbase can be downloaded at <https://www.couchbase.com>

In Figure 1, we can see the interface of Couchbase. It offers access to the features of the database, such as queries and indexes, and it shows the performance of the database.

Couchbase Server is organized into a set of services managed by the Couchbase Server cluster. The services are the Eventing, Indexing, Full-Text Search, Analytics, Mobile, Data, and Query.

Couchbase has a clustered architecture. The cluster consists in a group of nodes, using a peer-to-peer topology, where services are run and managed on each node. Components of a Couchbase node include the cluster manager and, optionally, the data, query, index, analytics, search, and other services. There are also the underlying managed cache and storage components (Couchbase, 2020).

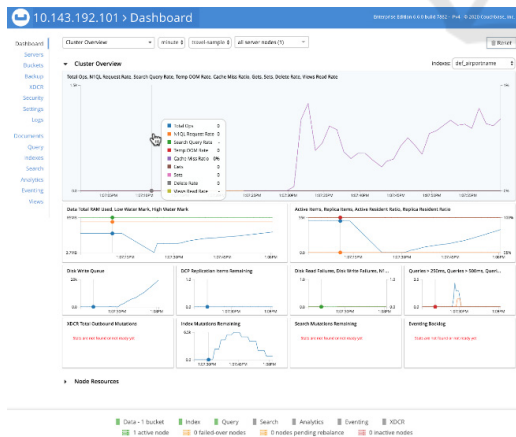


Figure 1: Couchbase Interface. Source: Couchbase Inc (2022).

Nodes can be added or removed through a rebalance process that redistributes the data evenly across all nodes. This process is done online and requires no system downtime. Couchbase supports

live-cluster topology. This ability to map services to sets of nodes is called Multi-Dimensional Scaling (MDS) (Hubail *et al.*, 2019; Couchbase Inc, 2022).

In Figure 2, we can see that, the Couchbase cluster consists of one or more instances of Couchbase Server, each running on an independent node. Data and services are shared across the cluster. The application servers communicate with the cluster overall, but because they are also aware of the individual node topology, they can adapt as needed (Couchbase, 2020).

The cluster manager supervises server configuration and interaction between servers. It manages replication and rebalancing operations in Couchbase. The cluster manager executes locally on each cluster node, but it elects a cluster-wide orchestrator node to oversee cluster conditions and execute cluster management functions. If the orchestrator node crashes, existing nodes will detect that it is no longer available and will elect a new orchestrator immediately (Couchbase, 2020).

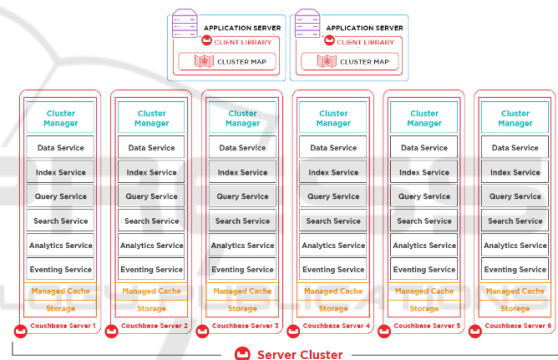


Figure 2: Couchbase Server Cluster. Source: Couchbase (2020).

The main advantages and characteristics of Couchbase are (Couchbase Inc, 2022; DB-Engines Ranking, 2022):

- Supports multiple programming languages, such as .Net, C, Go, Java, JavaScript, PHP, Python, Ruby, and Scala;
- Provides triggers, secondary indexes, server-side scripts, CRUD operations, and MapReduce;
- It supports horizontal and vertical scaling;
- Uses MVCC (MultiVersion Concurrency Control);
- It supports master-master replication and master-slave replication;
- Does have a SQL-like query language (N1QL).

The main weaknesses of Couchbase are (Couchbase Inc, 2022; DB-Engines Ranking, 2022):

- It is not being ACID compliant;
- Indexing takes up a lot of RAM.

3.2 CouchDB

CouchDB is a document database developed in Erlang by Apache Software Foundation in 2005. CouchDB is ideal for web applications that can handle a large amount of data. It is suitable for CRM (Customer Relationship Management) and CMS (Content Management System) (Nayak, Poriya and Poojary, 2013; CouchDB, 2022).

CouchDB is a schema-free database that provides a built-in web application called FULTON, which can be used for administration. CouchDB can be downloaded at <https://couchdb.apache.org/>. In Figure 3, we can see the CouchDB interface. It gives access to the database features and shows the existed databases and a few details about them.

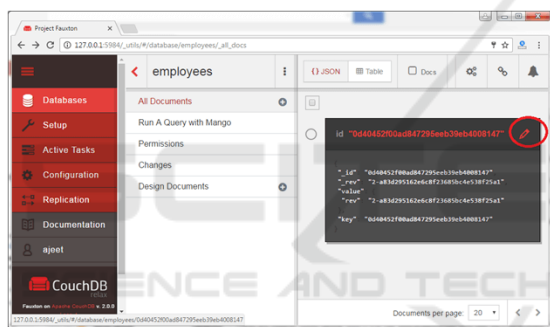


Figure 3: CouchDB Interface. Source: JavaPoint (2021).

In CouchDB, documents are stored in JSON format and JavaScript is used as a query language. It uses multi-version concurrency control (MVCC), which allows simultaneous access to multiple users. When the network connection is not possible to establish, CouchDB keeps working (Anderson, Lehanardt, and Slater, 2010; CouchDB, 2022).

CouchDB design is based on web architecture and the concepts of resources, methods, and representations. In Figure 4, we can see that all clients communicate with the server by making an HTTP request and receive either JSON or HTTP responses. The CouchDB is based on a B-tree and the data is accessed by keys or key ranges which map directly to the underlying B-tree operations. B-trees are a generalization of binary search trees. It is a tree data structure, which stores data and allows searches, sequential access, insertions, and deletions (Manyam *et al.*, 2012; CouchDB, 2022).

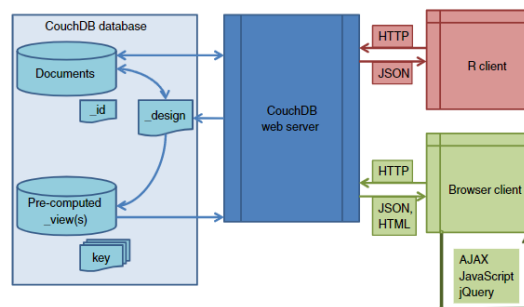


Figure 4: The architecture of CouchDB. Source: Manyam et al (2012).

The main advantages and characteristics of CouchDB are (CouchDB, 2022; DB-Engines Ranking, 2022):

- It supports horizontal scaling;
- Provides triggers, secondary indexes, server-side scripts, CRUD operations, and MapReduce;
- Supports multiple programming languages, such as C, C#, Erlang, Java, JavaScript, PHP, PL/SQL, Python, Ruby, Haskell, and more;
- Uses master-master replication;
- Ideal for web applications;
- It can handle a large amount of data.

The main weaknesses of CouchDB are (CouchDB, 2022; DB-Engines Ranking, 2022):

- Does not have an SQL-like query language;
- Views are temporary;
- Does not have support for ad-hoc queries.

3.3 MongoDB

MongoDB is a document database developed in C++ by 10gen in 2009. It is used for content management systems, archiving, real-time analytics, and mobile applications. MongoDB has four versions: Atlas, a multi-cloud database platform; Enterprise Advanced; Community Edition, and Realm, which provides data services for mobile and web. MongoDB offers the GUI (Graphical User Interface), the MongoDB Compass, and a built-in web application called Mongo Express. MongoDB can be downloaded at <https://www.mongodb.com>.

In Figure 5, we can see the interface of MongoDB Compass, where it is possible to create, change and delete the databases and collections. Also, is possible to explore and manipulate data, create queries, aggregations pipelines, views, indexes, build schema validation rules, and more (MongoDB, 2021).

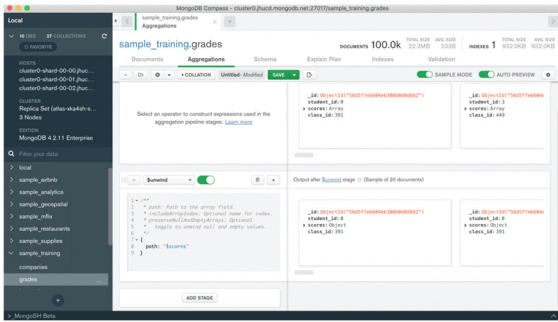


Figure 5: MongoDB Compass Interface. Source: Fiori (2021).

MongoDB is a schema-free database that provides the MongoDB Query Language. It stores data in JSON-like files (BSON), which is a binary format at JSON, and it is considered more efficient in storage space.

MongoDB uses sharding that is a method for distributing or partitioning data across multiple machines and enables horizontal scaling. MongoDB native sharding gives the following options (MongoDB, 2021a):

- Ranged sharding: involves dividing data into ranges based on the shard key values. Each chunk is then assigned a range based on the shard key values;
- Hashed sharding: this option guarantees a uniform distribution of writes across shards. The hashed sharding involves computing a hash of the shard key fields value. Then, each chunk is then assigned a range based on the hashed shard key values;
- Zoned sharding: on sharded clusters, it is possible to create zones of sharded data based on the shard key, and associate each zone with one or more shards in the cluster. A shard can associate with any number of zones. MongoDB migrates chunks covered by a zone only to those shards associated with the zone. Each zone covers one or more ranges of shard key values.

In Figure 6, we can see the MongoDB cluster architecture. The main core components in MongoDB architecture are (Edward and Sabharwal, 2015; MongoDB, 2021):

- Mongod: it handles with all the data requests, it manages the data format, and performs operations for background management;
- Mongos: it is used in sharding. It acts as a routing service that processes queries from the application layer and determines where in the sharded cluster the requested data is located;
- Mongo: is the interactive MongoDB Shell.

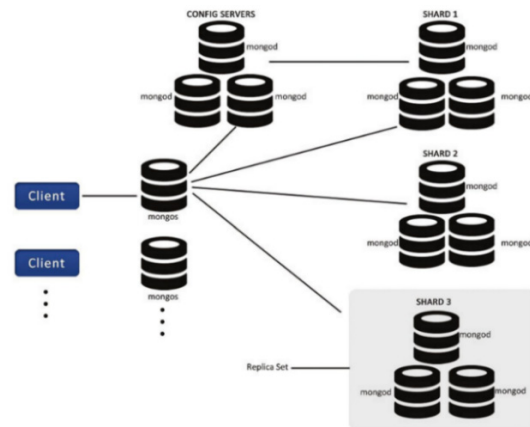


Figure 6: MongoDB cluster architecture. Source: Edward and Sabharwal (2015).

The main advantages and characteristics of MongoDB are (MongoDB, 2021; DB-Engines Ranking, 2022):

- It has a flexible structure;
- Very easily scalable, it offers vertical, horizontal, and tiered scaling;
- It has an amicable interface and provides a GUI;
- It is ACID compliant;
- Provides indexing, ad-hoc queries, CRUD operations, and MapReduce;
- Aggregation pipeline that allows for transformation and analysis of data;
- Provides native drivers for programming languages and frameworks;
- MongoDB replica sets enable the creation of up to fifty copies of data, which can be stored across multiple separate nodes, data centers, and geographic regions;
- Through native sharding, MongoDB allows the scale of the database as the applications grow across multiple nodes to handle write-intensive workloads and growing data sizes;
- MongoDB sharding is automatic and built into the database;
- MongoDB architecture is based on four essentials: availability, workload isolation, scalability, and data locality;
- It provides a replica set and a failover mechanism.

The main weaknesses of MongoDB are the following (MongoDB, 2021; DB-Engines Ranking, 2022):

- Does not use triggers, only in the Atlas version;
- Does not have an SQL-like query language;
- Data can easily be eliminated by mistake due to the lack of relations;

- Uses master-slave replication which is easier to lose data;
- Indexing takes up a lot of RAM.

4 DATABASE EVALUATION USING OSSpal

In this section, we assess the three databases using the OSSpal methodology. The information for the metrics was collected on the websites of the respective tools, books, papers, and some platforms such as GitHub, Stackshare, Google Scholar, and Stack Overflow. The first step is to assign weights to the categories in order of importance. Table 1 shows the weights assigned to each category.

Table 1: The weight assigned to each category.

Category	Weight
Functionality	30%
Operational Software Characteristics	15%
Software Technology Attributes	15%
Documentation	15%
Support and Service	10%
Community and Adoption	10%
Development Process	5%

The Functionality category is considered the most important because it gathers the most relevant characteristics that a NoSQL Document database should have. Therefore, it received the highest weight (30%).

The next step is defining and evaluating important characteristics for NoSQL Document Databases tools to analyze the functionality category. As shown in Table 2 a weight is assigned to each feature in the functionality category according to its relevance (1 – slightly important, 2 – important, and 3 – very important).

Table 2: Weight for the characteristics of the functionality category.

Characteristics	Weight
CRUD operations	3
Aggregations	3
Triggers	2
MapReduce	2
Indexes	2
Functions and procedures	2
Ad hoc queries	1

With most relevance we choose the CRUD operations, such as create, read, update, and delete. We also choose as most relevant the aggregations,

i.e., if the databases have aggregation functions, such as, sum, count, avg, etc. For the next most relevant features we choose the existence of triggers, the support of MapReduce functions, indexes, and functions and procedures. The feature with the least relevance is the support for *ad-hoc* queries.

In the second position, we have the Operational Software Characteristics that bring together features such as scalability, security, user interface, replication, sharding, installation, and configuration of databases. This category received a value of 15%, as did the categories of Software Technology Attributes, and Documentation. The category of Software Technology Attributes includes features such as the number of bugs, supported APIs, supported operating system servers, and supported programming languages. In the Documentation category, the existence of various types of documentation is considered, as a well-documented platform helps with installation, configuration, maintenance, and usability.

The category Support and Service obtained a value of 10%. The characteristics of this category are the existence of commercial and community support. The Community and Adoption category also received a value of 10%. It brings characteristics such as interaction in forums, and adoption by companies.

Finally, the category with the least weight (5%) was the Development Process. To assess this category, we considered the number of people who contributed to the development of the databases on the GitHub platform.

After assigning the weights to all categories, we obtain the results for each category (evaluation from 1 to 5), as shown in Table 3.

Table 3: Score obtained by the databases for each category.

Category	Couchbase	CouchDB	MongoDB
Functionality	5.0	3.0	4.0
Operational Software Characteristics	4.5	3.6	4.7
Software Technology Attributes	4.0	4.6	3.7
Documentation	2.0	3.0	5.0
Support and Service	4.3	4.3	5.0
Community and Adoption	3.0	3.0	5.0
Development Process	2.0	4.0	5.0

To analyze the features of each category, we used the following metrics, such as:

- In the Functionality category, we find out if the databases supported the features described in the Table 3, for all the versions that they offer;
- For the Operational Software Characteristics category, the metrics used were, the number of security certificates and safety mechanisms that are provided by each database for the feature security. The number of documents, offer by each database website, that provide information for the installation and configuration of the database. The number of sharding techniques supported, the types of scalability supported, and the types of replication topologies;
- The metrics used for the Software Technology Attributes category were the number of programming languages supported, the number of operating systems supported, the number of APIs supported, and the number of bugs that have been reported for each database on the GitHub platform;
- To evaluate the Documentation category, we count the number of papers released from January 1st, 2022 to April 15th, 2022 on the Google Scholar;
- For the Support and Service category, we evaluate the community support using two metrics. The number of followers at each database on the Stackshare platform, and the number of questions asked about each database on the Stack Overflow platform. And we find out if the databases offer commercial support;
- For the Community and Adoption category, we evaluate the number of followers in blogs related to each database on the Stackshare platform. And, we count the number of companies that adopt each database;
- For the Development Process category, we count the number of persons that have been contributed to the development of the database through the GitHub platform.

After analyzing Table 3, it is possible to conclude that:

- In the Functionality category, the database that stood out for having the all features, that we consider most important, is the Couchbase. Following the MongoDB, that have every features like Couchbase, but it just supports triggers on the Atlas version. In last place, we got CouchDB because it does not support ad-hoc queries and it just uses indexes on views;
- In the category of Operational Software Characteristics, MongoDB obtained the

highest score because it is the database with more certificates and security mechanisms, supports more types of sharding, and more types of scalability compared to the others databases;

- In the category of Software Technology Attributes, CouchDB obtained the highest score following Couchbase. And MongoDB obtained the lowest score, because was the database with more bugs report on GitHub platform;
- For the rest of the categories, we concluded that MongoDB always obtained the highest score than the other databases, probably because it has been on the market for the longest time and has the biggest development community. On the other hand, Couchbase and CouchDB obtained, almost, the same scores in the rest of the categories. Except for the Development Process category, CouchDB got ahead of Couchbase.

After evaluating each category, the final score is calculated for each database. Each category must be multiplied by its score and its respective weight assigned in Table 1. The results are shown in Table 4.

Table 4: OSSpal final score.

Category	Couchbase	CouchDB	MongoDB
Functionality	1.50	0.90	1.20
Operational Software Characteristics	0.68	0.54	0.71
Software Technology Attributes	0.60	0.69	0.56
Documentation	0.30	0.45	0.75
Support and Service	0.43	0.43	0.50
Community and Adoption	0.30	0.30	0.50
Development Process	0.10	0.20	0.25
TOTAL	3.91	3.51	4.47

With a score of 4.47, MongoDB was the database with the highest score with the application of the OSSpal methodology. Following Couchbase with a score of 3.91 and CouchDB with a score of 3.51.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we selected the top three document databases according to the DB-Engines ranking. We created a summary table that allowed us to put together the main characteristics of the databases.

To analyze and evaluate the NoSQL document databases we use the OSSpal methodology that allows the evaluation of open-source software.

The application of the OSSpal methodology was found to be very useful as it made it possible to compare the various databases in distinct categories. It also allows us to conclude that the best database with the highest score was MongoDB.

As future work, we intend to evaluate these Document databases, MongoDB, Couchbase, and CouchDB through their performance using the YCSB benchmark. We also intend to evaluate more Document NoSQL databases and compare them with relational databases.

REFERENCES

- Abramova, V., Bernardino, J. and Furtado, P. (2014) "Experimental Evaluation of NoSQL Databases," *International Journal of Database Management Systems*, 6(3), pp. 01–16.
- Anderson, J.C., Lehanardt, J. and Slater, N. (2010) *CouchDB: The Definitive Guide: Time to Relax*. O'Reilly Media, Inc.
- Calçada, A. and Bernardino, J. (2019) "Evaluation of Couchbase, CouchDB and MongoDB using OSSpal," in *IC3K 2019 - Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. SciTePress, pp. 427–433.
- Couchbase (2020) "Couchbase Under the Hood: An Architectural Overview." Santa Clara, California.
- Couchbase Inc (2022) *Couchbase Documentation*. Available at: <https://docs.couchbase.com/home/index.html> (Accessed: April 24, 2022).
- CouchDB (2022) *Apache CouchDB® 3.2.0 Documentation*. Available at: <https://docs.couchdb.org/en/stable/> (Accessed: April 23, 2022).
- DB-Engines Ranking (2022) *DB-Engines*. Available at: <https://db-engines.com/en/ranking> (Accessed: April 23, 2022).
- Edward, S.G. and Sabharwal, N. (2015) *Practical MongoDB, Practical MongoDB*.
- Elmasri, R. and Navathe, S.B. (2016) *Fundamentals of Database Systems*. 7th edn.
- Ferreira, T. (2018) *Integração de business intelligence no e-Commerce para PME*. Instituto Politécnico de Coimbra.
- Fiori, A. (2021) *MongoDB Compass – Extract Statistics Using Aggregation Pipeline*. Available at: <https://flowygo.com/en/blog/mongodb-compass-extract-statistics-using-aggregation-pipeline/> (Accessed: May 3, 2022).
- Hubail, M. al *et al.* (2019) "Couchbase Analytics: NoETL for Scalable NoSQL Data Analysis," *Proceedings of the VLDB Endowment*, 12(12), pp. 2275–2286.
- JavaPoint (2021) *CouchDB Create Document*. Available at: <https://www.javatpoint.com/couchdb-create-document> (Accessed: April 28, 2022).
- Leavitt, N. (2010) *Will NoSQL Databases Live Up to Their Promise?* Available at: www.leavcom.com.
- Manyam, G. *et al.* (2012) "Relax with CouchDB - Into the non-relational DBMS era of bioinformatics," *Genomics*, 100(1), pp. 1–7.
- Martins, P., Abbasi, M. and Sá, F. (2019) "A Study over NoSQL Performance," in *Advances in Intelligent Systems and Computing*. Springer Verlag, pp. 603–611.
- MongoDB (2021a) *MongoDB Architecture Guide*. Available at: <https://www.mongodb.com/collateral/mongodb-architecture-guide> (Accessed: March 29, 2022).
- MongoDB (2021b) *Welcome to the MongoDB Documentation*. Available at: <https://www.mongodb.com/docs/> (Accessed: March 25, 2022).
- Nayak, A., Poriya, A. and Poojary, D. (2013) "Type of NOSQL Databases and its Comparison with Relational Databases," *International Journal of Applied Information Systems (IJ AIS)*, 5(4), pp. 16–19.
- Oliveira, A. and Bernardino, J. (2019) "Evaluating Open Source Project Management Tools using OSSpal Methodology," in *WEBIST 2019 - Proceedings of the 15th International Conference on Web Information Systems and Technologies*. SciTePress, pp. 343–350.
- Tannir, K. (2013) *RavenDB 2.x*. Edited by A. Albuquerque et al. PACKT Publishing.
- Wasserman, A. *et al.* (2017) "OSSpal: Finding and Evaluating Open Source Software," in *Open Source Systems: Towards Robust Practices*, pp. 193–203.
- Wu, Y. *et al.* (2017) "An Empirical Evaluation of In-Memory Multi-Version Concurrency Control," *Proceedings of the VLDB Endowment*, 10(7), pp. 781–792.