Healthcare Information System using ArangoDB, a Graph-based Database

EDGAR TORRE, ANDRÉ SANTOS, and JOÃO AFONSO ANDRADE, Faculty of Engineering of the University of Porto, Portugal

This report presents the development and implementation of an application built using ArangoDB, a multi-model database system, to store and manage information related to diseases, symptoms, patients, hospitals, and treatments, as well as the relationships between them. The application aims to provide a comprehensive and efficient solution for organizing and retrieving healthcare-related data.

CCS Concepts: • **Information systems** → *Data management systems*; **Graph-based database models**; **Network data models**; • **Applied computing** → Health informatics; Health care information systems; • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: Graph databases, Healthcare information system, Data modeling

ACM Reference Format:
Edgar Torre, André Santos, and João Afonso Andrade. 2024. Healthcare Information System using ArangoDB, a Graph-based Database. 1, 1 (May 2024), 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' address: Edgar Torre, up201906573@fe.up.pt; André Santos, up201907879@fe.up.pt; João Afonso Andrade, up201905589@fe.up.pt, Faculty of Engineering of the University of Porto, Rua Dr. Roberto Frias, Porto, Portugal, 4200-465.

XXXX-XXXX/2024/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

ArangoDB is a robust multi-model database that provides a versatile and effective way to manage, store, and query data in a variety of use scenarios. ArangoDB appears to be a superb option in the context of a medical platform that links hospitals, patients, illnesses, therapies, and symptoms.

The medical platform built intends to offer a complete system that makes it easier for hospitals, patients, and various medical entities to communicate information in an efficient manner. ArangoDB can manage the intricate interactions and varied data structures present in the medical domain since it supports numerous data models, including key-value, document, and graph.

The graph model in ArangoDB is particularly beneficial for representing the intricate relationships between hospitals, patients, diseases, treatments, and symptoms. Graph databases excel in modeling and querying interconnected data, enabling the platform to establish connections between patients and hospitals, track disease occurrences, and analyze treatment effectiveness based on symptom data.

2 TECHNOLOGY OVERVIEW - ARANGODB

2.1 History

ArangoDB was first developed by ArangoDB GmbH in 2012 as an open-source project. It was written in C++ for high performance and built to work at scale, in the cloud or on-premises. The company also aimed to create a modern database system that could handle the increasing complexity and diversity of data types prevalent in contemporary applications. Over the years, ArangoDB has evolved with regular updates and enhancements and has established itself as a robust and reliable database solution. At the time of this report, it sits on version 3.11.

2.2 Licensing Model

ArangoDB adopted the Apache 2.0 open-source license as its licensing model for the Community Edition. The Apache 2.0 license is a permissive open-source license that allows users to freely use, modify, and distribute the software. It is recognized by the Open Source Initiative (OSI) as one of the widely used and respected open-source licenses.

Under the Apache 2.0 license, users have the freedom to use ArangoDB for any purpose, modify the source code to suit their needs, and distribute the modified or original version of the software. This license also ensures that users have the necessary permissions and legal protection to use ArangoDB in their projects, whether they are personal, academic, or commercial in nature.

This leads to a vivid and cohesive community who is encouraged to contribute and collaborate further. Users actively participate in the development and improvement of the software, sharing their enhancements and bug fixes with the community.

Conversely, its Enterprise and Professional edition are under the commercial license, as they are paid software.

2.3 Community Support and Resources

ArangoDB has a vibrant and active community of developers, database administrators, and enthusiasts. The community offers extensive support through various channels, including forums, mailing lists, and chat platforms. Users can seek assistance, share knowledge, and collaborate on projects within this community-driven ecosystem. Built into the official website (here) is the "Stay informed" category, where one can find the Blogs, Resources, Events and Community subcategories.

2.4 Documentation

A lot of updated and easy-to-read materials and tutorials for ArangoDB are available and easily accessible. All documentation can be found on ArangoDB's official website: https://www.arangodb.com/docs/stable/.[1]

3 MAIN FEATURES

ArangoDB is a versatile and flexible open-source multi-model database system. It is designed to handle a wide range of data models and offers support for key-value, document, and graph data models in a single database engine. ArangoDB combines the advantages of different database paradigms, providing developers with a unified and versatile platform for storing, querying, and manipulating their data. Key features and characteristics of ArangoDB include:

Multi-Model Support: Arguably, the most important characteristic about ArangoDB and what distinguishes it from its alternatives. This tools allows developers to work with multiple data models within a single database and back-end. It supports document-based data storage, key-value pairs, and graph data structures, enabling developers to choose the most suitable data model for their specific use cases and even combine them for extended customization.

ArangoDB Query Languag: AQL is a powerful and expressive query language which is exclusive to ArangoDB. This language allows developers to perform complex queries, joins, aggregations, and graph traversals across multiple data models, providing a unified and comprehensive way to retrieve and manipulate data.

Built-in Graph Database Features: As a graph-based ArangoDB has native support for graph data structures and provides efficient graph traversal and querying capabilities. It allows developers to store and process interconnected data, enabling the modeling and analysis of complex relationships and networks.

Flexible Data Modeling: Developers can define their data schemas using ArangoDB using key-value pairs or JSON-like documents. Data modeling can be fluid and dynamic, allowing schemas to change as application requirements do due to this flexibility.

Distributed Design: ArangoDB is designed for high availability and scalability. It supports distributed data storage, allowing data to be partitioned across multiple nodes in a cluster. This distributed architecture ensures fault tolerance, horizontal scalability, and efficient data processing in large-scale deployments. It also provides replication and sharding capabilities, allowing data to be replicated and distributed across multiple servers. Replication ensures data redundancy and fault tolerance, while sharding enables horizontal scalability by partitioning data across multiple shards.

Multiple Language Drivers: ArangoDB provides client libraries and language-specific drivers for most of the well-known programming languages including JavaScript, Python, and more. These drivers provide easy application integration and offer practical APIs for database interaction.

ArangoDB is appropriate for a variety of use cases, including content management systems, social networks, recommendation engines, and data analytics applications due to its combination of numerous data models, flexible data modeling, robust querying capabilities, and distributed architecture. Its popularity and adoption among developers and businesses is further aided by its open-source nature, thriving community, and thorough documentation.

4 DATA MODEL

Unlike most database engines that focus only on one data model, ArangoDB supports multiple data models, which is one of its biggest strengths. It is able to use paradigms such as the Geospatial model, but the main supported ones are:

Key-Value Pairs: This model store and retrieve data using a simple key-value pair. The key-value data model is useful for scenarios where quick and direct access to data based on a unique identifier (key) is required.

Document Model: In this model, data is organized into flexible, semi-structured documents (usually JSON). Documents can have nested key-value pairs, arrays, and other complex structures. This model is suitable for handling diverse and evolving data structures.

Graph Model: ArangoDB also supports graph models, which are actually their main focus and are the most used and debated in this report. This model allows you to store and manage highly interconnected data, such as relationships between entities, as a graph structure. ArangoDB's graph capabilities include storing vertices (nodes) and edges, along with powerful graph traversal and querying functionality, which will be mentioned on the next section.

5 SUPPORTED DATA OPERATIONS

Along with its multiple data models, ArangoDB also supports a wide range of data operations for them. Here are some of the most notable data operations supported by ArangoDB:

CRUD: Like most database engines, ArangoDB supports the standard Create, Read, Update, and Delete operations for managing data. You can create new documents, vertices or key-value pairs, read them by their keys or using queries, update existing data, and delete documents, vertices and key-value pairs from the database.

Queries: By using the powerful ArangoDB Query Language that is native to ArangoDB, ArangoDB allows you to perform complex queries on your data. AQL supports various query constructs, including filtering, sorting, aggregation, joins, and graph traversals, enabling you to retrieve specific subsets of data based on your requirements.

Indexes: To improve query performance, ArangoDB allows for indexing. Indexing allows for extremely efficient data lookup and retrieval, especially when querying large datasets. Therefore, in ArangoDB the developer can create various types of indexes, such as hash indexes, skiplist indexes, and geo indexes, based on the data and its access patterns.

Transactions: Since ArangoDB ensures data integrity in multi-operation scenarios, it also provides transactions. Multiple database operations can be grouped into a transaction, and ArangoDB guarantees that all the operations within the transaction either succeed or fail as a unit. This helps maintain data consistency and integrity, because otherwise some operations that were only correct in the context of the others may go through without the others, and cause unexpected behavior.

6 APIS AND CLIENT LIBRARIES

Developers may interface with the database using a number of programming languages and frameworks thanks to ArangoDB's APIs, libraries and drivers. These are the official ones:

REST HTTP API ArangoDB provides a RESTful HTTP API that allows communication with the database using HTTP requests. This API provides endpoints for various operations (mainly GET, POST, PATCH, PUT, and DELETE) on multiple resources, identified by their URI. This communication is stateless. Official documentation can be found here. [2]

JavaScript Driver The official JavaScript driver for ArangoDB is ArangoJS. Developers may communicate with ArangoDB from JavaScript applications using this client library, which is a API made available by ArangoJS to connect to ArangoDB. It provides a high-level promise-based API for executing queries, managing collections, and working with documents. Official documentation can be found here. [3]

Java Driver ArangoDB has built-in support for Java developers to interact with the database. This is one of the most used client library, and this API offers a comprehensive set of classes and methods to perform CRUD (Create, Read, Update, Delete) operations, query executing and much more. Official documentation can be found here. [**?**]

Go Driver The "arangodb-go-driver" is the official driver provided by ArangoDB for the Go programming language. With the help of this tool, programmers may communicate with ArangoDB from Go programs in a quick and effective manner. Official documentation can be found here. [4]

.NET Driver ArangoDB offers an official .NET driver called "arangodb-net-standard" that allows developers to integrate ArangoDB into C# /.NET applications. Like the previous ones, this .NET driver provides a set of classes and methods for performing database operations, executing AQL queries, and working with collections and documents. Official documentation can be found here. [5]

Python Drivers ArangoDB offers two official drivers for python. The first and most popular one is called "python-arango" and the second one is called "pyarango". Both are supported and maintained, and the main difference between both is the built-in validation present in pyArango that python-arango does not provide. Official documentation for python-Arango can be found here [6], and official documentation for pyArango can be found here.[7]

There are more drivers that weren't mentioned because they are not directly and officially endorsed by ArangoDB, such as the PHP driver and the low-level C++ driver. However, many of these are up-to-date, well-maintained and usable by developers who wish to do so.

7 CONSISTENCY FEATURES

Consistency is a fundamental property in database systems that ensures the correctness and reliability of data. It refers to maintaining the integrity and validity of data across different parts of the database, even in the presence of concurrent transactions or system failures. ArangoDB provides several mechanisms to ensure this in distributed environments. Here are some of the key aspects of how consistency works in ArangoDB:

7.1 ACID Transactions

ArangoDB supports ACID (Atomicity, Consistency, Isolation, Durability) transactions, which ensure that database operations are executed in an all-or-nothing manner. Through the preservation of data integrity throughout concurrent processes, ACID transactions provide the consistency. With the help of ArangoDB, developers may impose data consistency on connected entities by allowing transactions to span multiple documents and collections.

7.2 Data Validation

ArangoDB provides data validation mechanisms through its document schema definition. Developers can define validation rules, constraints, and data types for documents, ensuring consistent data structure and integrity across collections. Data validation helps enforce consistency during write operations and prevents invalid or inconsistent data from being stored.

7.3 Conflict Resolution

In distributed environments, conflicts may occur when concurrent transactions modify the same data. ArangoDB employs conflict resolution mechanisms to handle such scenarios. It provides conflict detection and resolution strategies, such as last-write-wins or user-defined conflict resolution functions, allowing developers to define how conflicts should be resolved based on their specific application requirements.

7.4 Adjustable Consistency Level

Developers may select the ideal trade-off between consistency and performance thanks to ArangoDB's several consistency levels. It provides many degrees of consistency and isolation for read and write operations by offering serializable, snapshot isolation, and read-committed isolation levels.

7.5 Universal Query Language

AQL allows for complex queries and joins across multiple collections. Since this can retrieve and combine related data from different collections in a single query, atomicity is better achieved this way, which also helps to ensure consistency.

7.6 Replication Model

Replication models also help to assure consistency by maintaining several copies of the data in order to restore potential lost data after a failure by reading the last state of the database. More on this on the next segment.

8 REPLICATION FEATURES

Replication involves creating and maintaining multiple copies of data across different nodes or servers. It is used to enhance data availability, fault tolerance, and performance in distributed systems. ArangoDB natively supports replication. This support is offered both as synchronous and asynchronous replication.

8.1 Synchronous Replication

Synchronous replication is used between the DB-Servers of an ArangoDB Cluster. Usually it is only used for critical data which must be accessible at all times, because the extra availability comes with the cost of more latency. On a general note, it stores a copy of a shard's data on another DB-Server and keeps it in sync. The way ArangoDB does this is by assuring that, when storing data, every replica within the cluster must write that data before the leader replica validates the write request to the client that requested it. This method means that singular failures due to a replica's unavailability are not signaled to the client. In that case, instead of canceling the write, the leader replica re-tries a few times to override a possible one-time failure of a replica.

8.2 Asynchronous Replication

Asynchronous replication is generally the most used and is based a different approach. Here the followers connect to a leader and apply all the events from the Leader log in the same order locally. This way, the followers end up with the same state of data as the leader, however this does not happen when data is written to the leader, but when the follower pulls the leader's log. This also means that followers are read-only instances, otherwise, data conflicts may occur that cannot be solved automatically, and this makes the replication stop. However, this restriction comes with a positive point, which is that the leader does not need to be aware of the Followers that replicate from it. This is because the only communication present in this process is from the follower to the leader. A consequence of this characteristic is, when a follower fails, the replication continues normally, and once the failed follower comes back, it simply pulls the log from the leader and applies all changes done while it was failed, and is once again up-to-date.

9 DATA PROCESSING FEATURES

Data processing features refer to higher-level capabilities and functionalities that enable advanced data manipulation, analysis, and computation. These features are built upon the underlying data operations previously mentioned in section "Supported Data Operations" and provide more specialized operations for specific use cases. In ArangoDB, the available features are:

9.1 Map-Reduce

This paradigm allows for distributed processing of large datasets and enables efficient data transformation, aggregation, and analysis by mapping data elements to intermediate key-value pairs and reducing them to

produce the desired output. This feature facilitates many complex data processing tasks, such as data mining, analytics and batch processing.

9.2 Joins and Graph Traversals

ArangoDB supports joins and graph traversals for querying and navigating relationships in the data. For the document data model, the developer can perform joins using AQL to combine data from multiple collections based on common attributes, while in the graph data model, it can also traverse relationships between vertices using various traversal algorithms and patterns.

9.3 Aggregation

ArangoDB supports the typical aggregation operations, such as grouping, counting, summing, averaging, and other statistical calculations. Developers can use the AQL aggregation functions to perform computations and generate aggregated results from your data, which was also used in this report.

9.4 Geospatial Data and Operations

By using the graph-based model, ArangoDB also includes geospatial query capabilities, enabling spatial queries, indexes and other operations on geospatial data. Moreover, documents with geographic coordinates, perform distance-based searches, and execute spatial operations like intersection, containment, and buffering can all be used in this tool.

9.5 Full-Text Search

Finally, ArangoDB also provides full-text search capabilities, allowing developers to perform text-based searches on document fields. Full-text indexes on specific fields can be created and executed search queries to find documents that match specific keyword or text patterns.

10 DESCRIPTION OF ADEQUATE USE CASES, AND PROBLEMATIC SCENARIOS

Starting with the use cases where ArangoDB is a good choice, a few examples are:

10.1 Knowledge Graphs

Multi-data model engines like ArangoDB excel in knowledge graph applications. Knowledge graphs represent varied and complex relationships between entities and allow for semantic querying and analysis. ArangoDB's graph capabilities enable the modeling and querying of intricate relationships, making it an ideal choice for knowledge graph applications in domains like e-commerce, recommendation systems, and information retrieval.

10.2 Multi-Domain Complex Projects

ArangoDB is well-suited for applications that involve multiple data domains or data models. Its ability to support key-value, document, and graph data models in a unified manner simplifies the development and management of such complex applications. For example, the healthcare application built in this report greatly used this characteristic to its advantage, since ArangoDB efficiently handled storing and querying data related to patients, medical records, hospitals, and the relationships between them.

10.3 Content Management Systems

ArangoDB's support for document storage and querying makes it suitable for CMSs. Managing and searching for various material kinds, such as articles, photographs, videos, and user-generated content, are typical tasks

for CMS solutions. Fast content retrieval and flexible content arrangement are made possible by ArangoDB's document model's flexible schema and effective indexing.

However, not all applications use ArangoDB to its full potential, and some need more specific functionalities, so it shouldn't be used universally. Some not ideal uses of ArangoDB are:

10.4 Small, Single-Model Systems

If an application has a straightforward data structure and only requires a single data model, opting for a multi-data model engine like ArangoDB might introduce unnecessary complexity. In such cases, a specialized database that aligns with the specific data model may be more appropriate and simpler to work with.

10.5 High-Concurrency, Write-Intensive Workloads

Multi-data model engines like ArangoDB may not perform as well as specialized databases in high-concurrency, write-intensive scenarios. If your application heavily focuses on write operations with frequent updates, inserts, or transactions, a database designed specifically for high-throughput writes may offer better performance and scalability, because that is not the focus of ArangoDB.

10.6 Data Warehouse or Analytics Applications

In contrast to analytical or data warehousing jobs, multi-data model engines like ArangoDB are often tuned for operational or transactional workloads. Due to their particular optimizations for such use cases, specialist analytical databases like Apache Hadoop, Apache Spark, or columnar databases may be better appropriate for applications that require complicated data analysis, large-scale batch processing, or advanced analytical queries.

11 COMPARISON TO OTHER SOLUTIONS

ArangoDB provides a lot of advantages when compared with other graph-based tools, The most prevalent being:

11.1 Unified Query Language

AQL (ArangoDB Query Language), which supports queries across many data models, is the unified query language used by ArangoDB. Complex joins, traversals, and aggregations between documents and graphs are supported by this unified query language, while most of the similar tools' query languages are specific to graph data. For example, in Neo4J searching is carried out using the Cypher query language, which is specific to graph data.

11.2 Horizontal Scalability

Users can distribute data across a number of servers or clusters thanks to ArangoDB's native support for horizontal scaling. This makes it possible to manage massive datasets and workloads with high flow rates very smoothly. Neo4j, on the other hand, typically takes a more vertically scalable strategy which isn't as distributed as ArangoDB's approach.

11.3 Cost

Being an open-source solution, ArangoDB can be used freely, which is a big advantage when used in personal or small scale projects. It's important to consider the cost implications, especially for those with budget constraints. Some similar tools, such as Amazon Neptune, comes with pricing based on usage, which is not always feasible in self-hosted or open-source solutions.

11.4 Certification

ArangoDB also offers an online exam that verifies a user's skills and knowledge about this tool. This can be a great way of quality assurance for developers who may look for a job on this field. Other graph-based solutions such as Neo4J for example, don't have this concept, which may make ArangoDB more appealing to new developers.

However, it also comes with some drawbacks which should be accounted for before using this technology. The most impactful ones are:

11.5 Overall Integration

Despite growing in popularity, some established databases may be better integrated in the market and are more explored than ArangoDB due the latter to not being used as much. This could mean fewer third-party integrations, community plugins, or extensive documentation for very specific use cases.

11.6 Performance Trade-offs

While ArangoDB offers multi-model capabilities, it may not excel in certain specialized areas. For instance, in use cases that predominantly require complex graph traversals, dedicated graph databases might offer better performance and optimization.

11.7 Learning Curve

ArangoDB introduces a different data model and query language, which may require a learning curve for developers who are more accustomed to SQL-based databases. This could impact initial development efforts and require additional training.

12 PROTOTYPE

12.1 Dataset

When considering what would be the best topic to showcase the capabilities of ArangoDB, many options were brought to the table. The key point involved in the decision was the need for the existence of various types of relations, that could take advantage of the graph database paradigm. After going through different suggestions, the choice ended up being a healthcare information system.

This theme is very versatile and scalable, given the nature of the topic. Many approaches and class could be generated from the problem and the focus ended up to fall around a disease search system.

Even though the idea was pertinent, the lack of data that could be aggregated and processed into a dataset was a concern. Despite that, 'Wikidata' knowledge base ended up being sufficient, since it contained some possible relations, such as the diseases symptoms or possible treatments were available. The query language used to get data from the web application was SPARQL. Some of the queries were not trivial. Besides that, in order to solidify the entities, and since this information was not available, some data was artificially generated, such as hospitals, patients or simply the attributes that describe a disease.

12.2 Data models

12.2.1 Conceptual Data Model. As it has been previously mentioned, the healthcare theme could include a great variety of entities and relations. For example, since hospitals and patients were included, other types of nodes, such as doctors, appointments, material...could have been also used, but the chosen ones were sufficient to showcase the different ArangoDB capabilities, and it was more important to focus on quality rather than quantity.

The different nodes were centered around diseases and patients. From there, symptoms, treatments, hospitals and areas were instantiated. Their relations are described and may be seen in the image below.

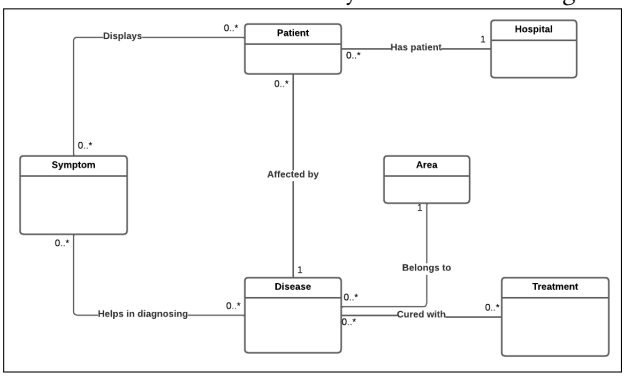

Fig. 1. Conceptual Data Model

12.2.2 Physical Data Model. As much as having real data would enrich the dataset, the reality is that none of these entities had linkable information that could be used. Despite that, many attributes were generated to exemplify what could be some possible use cases in real life. Between them, there are some indices such as a disease transmission and mortality or simply the hospital infrastructure data, such as their workforce and number of beds. Again, evidently more attributes could be used considering a real-life scenario, More information on it is specified below in the data model, that already contains the attributes and respective types. In ArangoDB the variables adjust their size automatically.

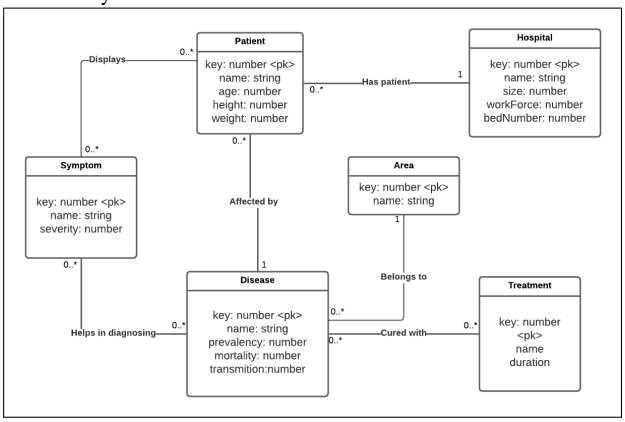

Fig. 2. Physical Data Model

12.3 Data Processing

All numbers in the attributes were generated randomly, which means that a high mortality rate disease in real life may not have a high value and vice-versa. In some criteria, the the low range let the values maintain a certain approximation to reality, but for example in the treatment duration attribute the high variety of possibilities makes the value approximation significantly worse. The symptoms were attributed to the patients considering the number of possible symptoms for a said disease.

12.4 Data Structures

As its been previously explained, one of ArangoDB's main advantages is its versatility and the possibility to use various database paradigms. As such, our data was stored in the collections as documents, and used as nodes.. Afterwards, these vertices are put together by creating relations between them as edges in an edge collection.

12.5 Architecture

12.5.1 Introduction. The prototype in itself was created with the purpose of facilitating the life of the user when exploring and dealing with our dataset and database engine. It allows the user to perform almost all of hospital related tasks such as adding patients to a hospital, deleting them, editing the hospital's information, etc...

It also allows for the user to easily see and compare information regarding each of the classes available in our dataset.

12.5.2 WebServer and Application Layer. In order to develop our web server, handle client requests and correctly establish connection between the server, the application layer and ArangoDB we used Node.js with Express as the web server technology and framework. We also used Arango.js and Docker.

Node.js is a powerful JavaScript runtime that enables server-side JavaScript code execution. It provides non-blocking and event-driven I/O operations, which efficiently handle concurrent requests. Express, a popular web application framework for Node.js, provides a powerful set of features and utilities for building web applications and APIs.

When it comes to the communication with the ArangoDB database engine, this is done using the Arango.js library. Arango.js is the official JavaScript driver for ArangoDB, allowing transparent communication between the web server and the database and also allowing the web server to execute various database operations, such as inserting, querying, updating, and deleting data stored in ArangoDB.

12.5.3 Client-Side Components. When it comes to the client-side technologies that were used in the prototype, we utilized HTML, CSS and REACT which is used as a JavaScript library.

By using HTML, CSS, JavaScript, and React together, we manage to create an engaging and interactive user interface. HTML structures the content, CSS styles the components, JavaScript adds interactivity, and React makes it easy to efficiently render and manage UI elements.

12.5.4 Communication Protocols. Even though Arango.js in itself is not a communication protocol, it provides a convenient API and methods for connecting to ArangoDB, executing database queries, and performing various data operations. In order to do this it supports multiple communication protocols, from which we used HTTP.

12.5.5 Security Features. Even though we had no need to implement them, ArangoDB combines various security features, including SSL/TLS encryption and authentication, to protect data. Authentication ensures that only authorized users or applications can access the database.

SSL/TLS encryption encrypts data in transit, preventing unauthorized access.

Access control allows administrators to manage user privileges and restrict access to sensitive data. Auditing and logging features help monitor and track database activities for security and compliance purposes. By leveraging these features, ArangoDB ensures the protection and integrity of the data stored in the database.

12.5.6 Scalability and Performance. ArangoDB performs the best regarding the scalability to handle large volumes of data and works efficiently under demanding workloads. It achieves this through flexible data models, distributed architecture for horizontal scalability, optimized graph traversal, indexing, caching, optimization. integrated query and cluster management. It also ensures smooth scalability and fast performance for various data operations.

12.5.7 Features. The application has various features. Summarizing the main pages, three great units come out:

Hospital Map / Home Page In the main page there is a map with the hospitals location. Each marker leads to the hospital page. There's also a button that leads to the disease list page.

Hospital Page Inside the hospital page there is information and diverse statistics about its patients and their symptoms and diseases, showcasing multiple aggregates and database queries. There is also the list of patients and the ability to add or delete them and to edit the hospital information. With this feature all CRUD operations are applied. Important to note that the delete operation does not automatically delete related edges in ArangoDB, so it has to be done manually.

Disease List In this page all the diseases are showcased in a list. There, the user can use filters to narrow the search(sliders on the node attributes and related edges and also key-value text search). Inside the disease, treatment and symptom pages, one can also find the list to the other connecting edges.

12.5.8 How to Run. In order to run the application. Two commands inside the src folder. The first one to build the container, and the second to start it every time.

```
docker-compose build
docker-compose up
```

If by any chance that does not work there is the possibility to do it manually by running these sequentially. In the beginning, the official Arango image should be pulled from docker. Then, one command is used to start the Arango Database and other to populate it afterwards. Then, to start the backend and frontend, respectively, the last 2 commands will do. The prototype will be available for use at localhost:3000. Commands are based on the main directory.

```
Install Arango:
docker pull arangodb

Inicialize database:
docker run -e ARANGO_NO_AUTH=1 -p 8529:8529 arangodb

Populate database:
cd src/backend
npm install (if necessary)
node populate.js

========================================

Initialize backend:
cd src/backend
node api.js

Initialize frontend:
cd src/frontend
npm install (if necessary)
npm start

========================================
```

13 REFERENCES

Various references were used so some will be specially highlihted. One of those, is the ArangoDB official website, since it provided plenty of information about the technology clearly and efficiently. We also made use of ChatGPT capabilities when developing the prototype since it proved to be a pretty efficient tool to assert and build AQL queries. It also proved to be very useful regarding succinct explanations of the concepts we did not understand. [8] Complementing that, Wikidata was also really important in order to build the dataset. [9]

REFERENCES

[1] [n. d.]. ([n. d.]). Arango Documentation, Last Access Date: 16/05/2023.
[2] [n. d.]. ([n. d.]). Rest Api, Last Access Date: 16/05/2023.
[3] [n. d.]. ([n. d.]). Arango JavaScript Driver, Last Access Date: 16/05/2023.
[4] [n. d.]. ([n. d.]). Arango Go Driver, Last Access Date: 16/05/2023.
[5] [n. d.]. ([n. d.]). Arango .NET Driver, Last Access Date: 16/05/2023.
[6] [n. d.]. ([n. d.]). Python-Arango, Last Access Date: 16/05/2023.
[7] [n. d.]. ([n. d.]). PyArango, Last Access Date: 16/05/2023.
[8] [n. d.]. ([n. d.]). ChatGPT, Last Access Date: 16/05/2023.
[9] [n. d.]. ([n. d.]). Wikidata, Last Access Date: 16/05/2023.