

Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies

XXXXXXXX

Peter Plevko

**Development of web application
components for semantic annotation of
datasets**

Master's thesis

Supervisor: Ing. Miroslav Rác

January 2024

Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies

XXXXXXXX

Peter Plevko

**Development of web application
components for semantic annotation of
datasets**

Master's thesis

Study program: Intelligent Software Systems

Field of study: Computer Science

Place: Institute of Computer Engineering and Applied Informatics, FIIT STU,
Bratislava

Supervisor: Ing. Miroslav Rác

January 2024

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce¹

Študent:

Meno, priezvisko, tituly: Peter Plevko, Bc.
Študijný program: Inteligentné softvérové systémy
Kontakt: <https://www.facebook.com/peterplevkoo/>

Výskumník:

Meno, priezvisko, tituly: Miroslav Rác, Ing.

Projekt:

Názov: Vývoj súčastí webovej aplikácie na sémantickú anotáciu datasetov
Názov v angličtine: Development of web application components for semantic annotation of datasets
Miesto vypracovania: Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU
Oblasť problematiky: Sémantická anotácia datasetov

Text návrhu zadania²

V oblasti dátovej vedy sa využíva štatistika, procesy a algoritmy na extrakciu poznatkov a znalostí zo štruktúrovaných aj neštruktúrovaných dát. Dátoví vedci potrebujú infraštruktúru na zjednodušenie práce s datasetmi a vyhodnocovania modelov umelej inteligencie. Ich potreby zahŕňajú uchovávanie, organizovanie a anotáciu datasetov, a tiež ich distribúciu cez sieť. Cieľom dátovej vedy je zjednotiť štatistiku, informatiku a súvisiace metódy s cieľom pochopiť a analyzovať skutočné javy pomocou dát.

Cieľom práce je vytvoriť webové prostredie, ktoré bude podporovať pridávanie, anotáciu a odosielanie súboru dát na požiadanie, pričom výsledný webový nástroj umožní efektívne pracovať s dátami a bude podporovať vyhodnocovanie modelov umelej inteligencie. Analýzou možností vkladania, odosielania a anotáciou dát, navrhnete vhodnú metódu a formát pre anotáciu súboru dát a ich následné odosielanie. Vytvorte databázový model pre ukladanie týchto dát a umožnite používateľovi pozrieť sa na vytvorenú reprezentáciu dát a na vzťahy medzi nimi. Implementované webové prostredie by malo umožňovať pracovať s dátami vo forme zrozumiteľnej pre človeka (user interface) aj pre stroj (API).

Navrhnuté riešenie, ktoré bude schopné na základe analytických vstupov ukladať a odosielať anotovaný súbor dát, implementujte ako online verziu. Samotnú funkčnosť a korektnosť webového nástroja overte a vyhodnoťte prostredníctvom testov.

¹ Vytlačiť obojstranne na jeden list papiera

² 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra³

- Kellou-Menouer, K., Kardoulakis, N., Troullinou, G. et al. A survey on semantic schema discovery. The VLDB Journal 31, 675–710 (2022). <https://link.springer.com/article/10.1007/s00778-021-00717-x>
- Reeve, L., & Han, H. (2005, March). Survey of semantic annotation platforms. In Proceedings of the 2005 ACM symposium on Applied computing (pp. 1634-1638). <https://dl.acm.org/doi/pdf/10.1145/1066677.1067049>

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Peter Plevko, konzultoval(a) a osvojil(a) si ho Ing. Miroslav Rác a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 3.1.2024

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / nie⁴

Dňa:

Podpis garanta predmetov

³ 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uvedte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

⁴ Nehodiace sa prečiarknite

Čestne vyhlasujem, že som túto prácu vypracoval samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, Január 2024

.....

Peter Plevko

Acknowledgement

I would like to express my deepest gratitude to Ing. Miroslav Rác, my thesis supervisor, for his invaluable guidance and insightful counsel throughout the process of writing my master's thesis. His exceptional support and willingness to address even the most trivial questions have been instrumental in shaping the quality of my work. I am truly grateful for his expertise and dedication to helping me succeed.

Furthermore, I extend my heartfelt appreciation to my colleagues from the faculty. Their unwavering assistance and mutual encouragement have been invaluable. Together, we have consistently fostered an environment of collaboration and support, enabling each other's growth and academic success. I am thankful for their friendship and shared commitment to excellence.

In addition, I would like to express my sincere thanks to my family for their unwavering support throughout this journey. Their encouragement, understanding, and belief in my abilities have been a constant source of motivation. I am truly fortunate to have such a loving and supportive family by my side.

Lastly, I want to acknowledge my friend and roommate, Matej Delicak, for constantly improving my mood and providing a sense of companionship during this challenging time. His presence and positivity have made a significant difference in my overall well-being, and I am grateful for his friendship.

To all those mentioned above, I extend my heartfelt gratitude. Without your support, encouragement, and assistance, this accomplishment would not have been possible. Thank you for being a part of my academic journey and for making it a truly rewarding experience.

Annotation

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Study program: Informatics

Author: Bc. Peter Plevko

Master's thesis: Development of web application components for semantic
annotation of datasets

Supervisor: Ing. Miroslav Rác

January 2024

The aim of the thesis is to analyze the architecture and data model requirements for the purpose of annotating datasets with metadata in the field of data science. The web environment is to be used to add, annotate and send datasets on demand, while allowing efficient handling of datasets and supporting the evaluation of AI models. At the same time, it analyzes the possibilities of inserting, sending and annotating data in order to propose a suitable method and format for annotating datasets and then sending them. The thesis also includes a database model and an overview of the created data representation and its relationships for the user. The implemented web environment allows working with the data through both the user interface and the application user interface. Theoretical part - All programming languages and technologies that would be appropriate to use are briefly described. Their advantages and disadvantages are described and then the best ones for our use-case are selected. Practical part - The developed solution, which can store and send annotated datasets based on analytical inputs, is implemented in the form of an online version. The functionality of the web-based tool is evaluated using tests. An efficient data model for the annotated data is created.

Anotácia

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLOGIÍ

Študijný program: Inteligentné softvérové systémy

Autor: Peter Plevko

Diplomová práca: Vývoj súčastí webovej aplikácie na sémantickú
anotáciu datasetov

Vedúci práce: Ing. Miroslav Rác

January 2024

Cieľom diplomovej práce je analyzovať požiadavky na architektúru a dátový model pre účely anotácie datasetov metadátami v oblasti dátovej vedy. Webové prostredie má slúžiť na pridávanie, anotáciu a odosielanie súborov dát na požiadanie, pričom umožní efektívnu prácu s datasetmi a podporí vyhodnocovanie modelov umelej inteligencie. Súčasne sa venuje analýze možností vkladania, odosielania a anotácie dát s cieľom navrhnúť vhodnú metódu a formát pre anotáciu súborov dát a ich následné odosielanie. Diplomová práca zahŕňa aj databázový model a prehľad vytvorenej reprezentácie dát a ich vzťahov pre používateľa. Implementované webové prostredie umožňuje prácu s dátami prostredníctvom používateľského rozhrania aj užívateľského rozhrania pre aplikácie. Teoretická časť - Sú v nej stručne popísane všetky programovacie jazyky a technológie ktoré by bolo vhodné použiť. Sú popísane ich výhody a nevýhody a následne vybraté tie najlepšie pre náš use-case. Praktická časť - Vytvorené riešenie, ktoré dokáže ukladať a odosielať anotované súbory dát na základe analytických vstupov, je implementované vo forme online verzie. Funkčnosť webového nástroja je vyhodnotená pomocou testov. Je vytvorený efektívny dátový model pre anotované dáta.

Contents

1	Introduction	1
2	Analysis	4
2.1	Knowledge base	5
2.2	Knowledge graph	5
2.3	Web page	6
2.4	Open source	6
2.5	Framework	7
2.6	Docker	8
2.7	Security	10
2.7.1	HTTP	10
2.7.2	Login	11
2.7.3	User types	12
2.7.4	JSON web token	13
2.8	Frontend	14
2.8.1	HTML	15
2.8.2	CSS	16
2.8.3	JavaScript	17
2.9	Backend	19
2.9.1	Database	21
2.9.2	Adapter pattern	23
2.10	Data model	25
2.10.1	Artificial intelligence	27

2.10.2	Machine learning	29
2.10.3	Artificial neural networks	30
2.10.4	Natural language processing	33
2.10.5	Large language model	34
2.10.6	Deep learning	35
2.11	Data format	38
3	Related Works	41
4	Design	44
4.1	Technologies	45
4.2	Web application specification and requirements	47
4.2.1	Usability	47
4.2.2	Awareness	48
4.2.3	Accessibility	48
4.2.4	Ease of use	48
4.3	Target group	49
4.3.1	Target audience characteristics	49
4.3.2	Device usage statistics	50
4.4	Database model	50
4.5	Diagrams	53
4.5.1	Login diagram flowchart	53
4.5.2	Add dataset diagram flowchart	55
4.5.3	Annotate dataset diagram flowchart	57
4.5.4	Google cloud infrastructure diagram	59
4.5.5	Google cloud CI/CD diagram	61
4.6	User interface	62
4.6.1	Find a dataset wireframe	63

4.6.2	Selected dataset wireframe	64
4.6.3	Find annotation wireframe	65
4.6.4	Selected annotation wireframe	66
4.6.5	Add annotation wireframe	67
5	Implementation	70
5.1	App implementation steps	70
5.2	Api endpoints	81
5.2.1	NestJS API endpoints	82
5.2.2	Firestore cloud functions endpoints	83
5.3	Screens	83
5.3.1	Find a dataset page	84
5.3.2	Selected dataset page	85
5.3.3	List annotations page	88
5.3.4	Add annotation page	91
5.3.5	Admin page	93
6	Future work	97
6.1	Deployment	97
6.2	Microservices	98
6.3	Annotation suggestions	99
6.4	User testing	100
6.5	Using a different database with the repository pattern	100
6.6	Removing restriction for dataset size	101
7	Evaluation	104
	References	107

A	Diploma thesis evaluation	A-1
A.1	Work plan for phase DP1	A-1
A.2	Evaluation of the work plan for the DP1 phase	A-1
A.3	Work plan for phase DP2	A-3
A.4	Evaluation of the work plan for the DP2 phase	A-4
A.5	Work plan for phase DP3	A-6
A.6	Evaluation of the work plan for the DP3 phase	A-6
B	User Guide	A-7
B.1	Program function	A-7
B.2	Installing the program	A-8
C	Description of the digital part	D-1

List of Figures

2.1	Software Framework [12]	8
2.2	Adapter design pattern [39]	23
2.3	Adapter design pattern in project	25
2.4	Artificial intelligence development and expansion. [40]	26
2.5	Single-layer feedforward networks [49]	31
2.6	Multilayer feedforward network with a single hidden layer [49]	32
2.7	Components of Natural Language Processing [52]	34
2.8	Biological neuron [56]	37
2.9	Artificial neuron - perceptron [57]	37
4.1	Neo4j database model	51
4.2	Login diagram flowchart	53
4.3	Add dataset diagram flowchart	55
4.4	Annotate dataset diagram flowchart	57
4.5	Google cloud infrastructure diagram	59
4.6	Google cloud CI/CD diagram	61
4.7	Find a dataset figma wireframe	63
4.8	Selected dataset figma wireframe	64
4.9	Find annotation figma wireframe	65
4.10	Selected annotation figma wireframe	66
4.11	Add annotation figma wireframe	67
5.1	Find your dataset page	84
5.2	Selected dataset page 1	85
5.3	Selected dataset page 2	86
5.4	List annotations page	88
5.5	Add annotation page	91

5.6	Admin page	93
-----	----------------------	----

Abbreviations list

SQL	Structured Query Language
Back-end	Data Access Layer
Front-end	Presentation Layer
API	Application Programming Interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
DNS	Domain Name system
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JS	Javascript
JSON	JavaScript Object Notation
URL	Uniform Resource Locator
OOP	Object-Oriented Programming
IP	Internet Protocol
SSL	Secure Sockets Layer
AJAX	Asynchronous JavaScript and XML

1 Introduction

Data science has emerged as a crucial field, encompassing the utilization of statistics, processes, and algorithms to extract valuable insights and knowledge from both structured and unstructured data. With the ever-increasing availability of data, data scientists require robust infrastructures that facilitate efficient handling of datasets and the evaluation of artificial intelligence (AI) models. The ability to store, organize, annotate, and distribute datasets becomes imperative for their work. Furthermore, data science seeks to integrate statistics, computer science, and related methodologies to comprehend and analyze real-world phenomena through data.

This research aims to develop a web-based environment specifically tailored to support the needs of data scientists. The environment will enable seamless addition, annotation, and transmission of datasets, thereby empowering efficient data manipulation and facilitating the evaluation of AI models. Through a comprehensive analysis of various methods and formats for data insertion, transmission, and annotation, this study will determine the most suitable approach. Additionally, a database model will be devised to store the data, allowing users to explore the created data representations and the relationships between them. Ultimately, the web environment will provide data scientists with a user-friendly interface as well as a machine-readable API, ensuring data accessibility and understanding.

To achieve these objectives, it is crucial to gain a deep understanding of the concepts and terminologies that data scientists employ in their daily work. By familiarizing ourselves with these concepts, we can design a tool that aligns with

their requirements. In the subsequent sections, we will analyze and explain these concepts, progressing from fundamental to advanced topics. We will explore artificial intelligence, machine learning, and artificial neural networks, delving into their applications in various domains. Additionally, we will discuss deep learning and its significance in the field of data science.

By thoroughly examining these concepts and their interconnectedness, we aim to gain valuable insights that will inform the design and development of an effective tool for data scientists. This thesis endeavors to contribute to the field of data science by addressing the pressing needs of data scientists and advancing the capabilities of their infrastructure. Through this research, we aspire to facilitate enhanced data manipulation, support AI model evaluation, and ultimately empower data scientists in their pursuit of knowledge extraction and analysis from diverse datasets.

2 Analysis

Our website serves as a comprehensive knowledge base tailored specifically for data scientists, encompassing a multitude of data structures that collectively represent statements about the world [1]. In this context, the system diligently stores data following each user-annotated dataset. For instance, if a user annotates a column named “area“ the system not only recognizes it as a physical quantity denoting area but also retains information about the associated unit of measurement, such as meters. These annotations are securely saved within our web page, enabling subsequent users to conveniently utilize these pre-existing annotations when annotating tables with similar names. However, if a required value has not been previously added, users possess the autonomy to contribute and introduce new annotations. This approach showcases a semi-automatic nature, as it necessitates user input while effectively managing the storage aspect. It is important to acknowledge that the complete automation of annotation creation remains an ongoing challenge in the field [2].

Knowledge engineering, an integral field of artificial intelligence (AI), endeavors to emulate the judgment and reasoning capabilities of human experts within specific domains [3]. Our platform offers a range of APIs that empower data researchers to access pertinent data for training AI models or assessing their performance following training. Acting in a passive manner, our website requires users to initiate API requests to provide the desired information. Consequently, our API seamlessly delivers the requested dataset accompanied by the associated annotated metadata.

Developing an efficient and robust web page prompts critical considerations surrounding the selection of programming languages for both the client and server components, the identification of an optimal framework, architectural decisions between server-based or serverless models, the choice between relational, non-relational, or graph databases, and determining the ideal format for transmitting annotated dataset metadata. In the subsequent sections, we will delve into the possibilities surrounding these questions, while offering academic perspectives and insights.

2.1 Knowledge base

A knowledge base is a centralized repository of information that can be accessed and used by individuals or systems. It can contain a wide range of information, including facts, rules, procedures, and best practices. Knowledge bases can be used in various fields, such as computer science, medicine, and economics, to support decision-making, problem-solving, and learning. Examples of knowledge bases include expert systems, databases, and wikis [4].

2.2 Knowledge graph

A knowledge graph is a graph-structured data model used to store descriptions of entities such as people, places, and events. It can be defined as a collection of ordered triples (h, r, t) , where h and t are the head and tail entities, and r is the relation between them. Knowledge graphs play a significant and growing role in semantics-based support of a wide variety of applications, and they are useful for tasks such as question answering and reasoning [5]. They are also being used in the Solid ecosystem to improve pod access through different Web APIs that act

as views into the knowledge graph, providing improved opportunities for storage, publication, and querying of decentralized data in more flexible and sustainable ways [6]. The management of knowledge graphs has evolved, and there are now KG management platforms supporting the lifecycle of KGs from their creation to their maintenance and use [7].

2.3 Web page

According to the information gathered from the sources cited, a website represents a collection of interconnected web pages that are accessible through the internet [8, 9]. Two primary classifications of websites exist: static and dynamic. Static websites are constructed using “fixed code” and employ languages such as HTML, JavaScript, and CSS [8]. The content of static web pages remains unaltered until manual modifications are made. In contrast, dynamic websites offer interactivity and are implemented using languages such as CGI, AJAX, ASP, and ASP.NET [8]. Dynamic web pages exhibit variability in content depending on visitor interactions, though at the expense of increased loading times compared to static web pages. This dynamism proves particularly useful in scenarios where frequent information updates are required, such as displaying real-time stock prices or weather information.

2.4 Open source

As per the information provided by sources such as [10, 11], the concept of open source pertains to computer software that is released under a licensing framework, granting users specific rights to utilize, study, modify, and distribute the software and its corresponding source code. This licensing approach enables individuals

to access and leverage the software freely, without restrictions, while fostering an environment of collaboration and community-driven development. Open source software often undergoes public and collaborative development processes, involving multiple contributors working together to enhance its functionality and address issues.

In addition to the technical aspects, the term “open source” encompasses a broader set of values that promote principles like open exchange, participatory collaboration, transparent development, rapid prototyping, meritocracy, and community-oriented engagement. These values support an inclusive and collaborative environment where individuals can openly contribute to the improvement and evolution of the software.

2.5 Framework

A programming framework can be described as a valuable tool that offers developers a collection of pre-built components, libraries, support programs, and APIs. Its primary objective is to accelerate software development by providing standard low-level functionalities, thereby enabling developers to concentrate on the distinctive aspects of their projects. The underlying principle governing frameworks is the inversion of control, where the framework takes charge of the flow of control in the application.

Frameworks play a crucial role in software development by enhancing reliability, expediting programming time, and simplifying the testing process. By providing a foundation of standardized functionality, frameworks alleviate the need for developers to reinvent the wheel for common tasks. This not only saves time and effort but also promotes consistency and reduces the likelihood of errors.

Furthermore, frameworks offer a support system, benefiting developers with access to a community of fellow practitioners who can provide guidance and assistance. This collective knowledge pool can prove invaluable when encountering challenges during the development process. Additionally, frameworks often incorporate security measures and best practices, ensuring that developers can build applications with robust security foundations.

Overall, the adoption of frameworks contributes to improved efficiency, reduced development time, enhanced software reliability, and cost savings [12]. Acknowledging the significance of frameworks in programming can aid in making informed decisions when selecting the appropriate framework for developing a knowledge base website targeting data scientists.

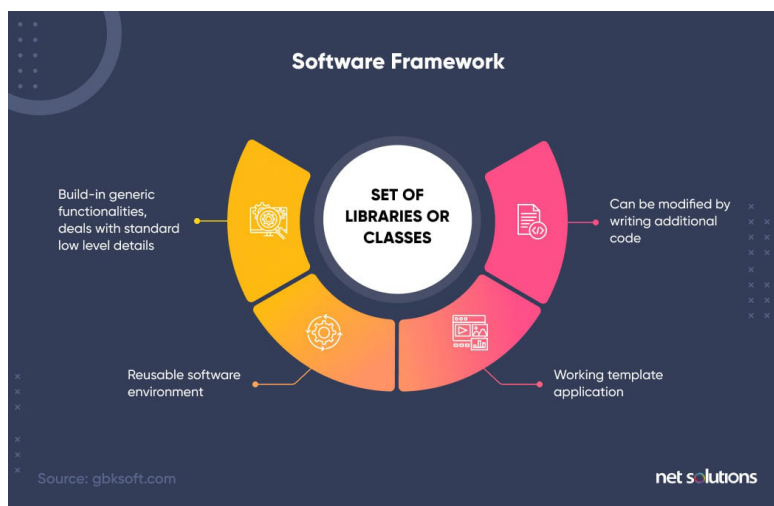


Figure 2.1: Software Framework [12]

2.6 Docker

Docker, an open source containerization platform, has gained widespread recognition as a powerful tool for packaging applications into portable and self-contained

units known as containers. These containers encapsulate the application's source code, along with the necessary operating system libraries and dependencies, enabling consistent and reliable execution across diverse computing environments. The rise in popularity of cloud systems has further propelled Docker's adoption, as it facilitates the seamless deployment and execution of applications on various host machines [13].

One of Docker's key advantages lies in its ability to ensure application consistency and portability. By encapsulating the application and its dependencies within a container, Docker enables developers to build and test applications in isolation, independent of the underlying host system. This containerized approach provides a standardized environment, eliminating compatibility issues and enabling the application to run consistently across different computing environments.

Furthermore, Docker simplifies the process of sharing and deploying applications. The containerized nature of Docker allows developers to package an application along with its dependencies into a single artifact, ensuring that the application runs reliably and predictably on any compatible Docker host. This portability significantly reduces the complexities associated with configuring and setting up an application on different computers, enhancing development efficiency and facilitating seamless collaboration.

Moreover, Docker promotes scalability and efficiency through its lightweight and resource-efficient containerization model. Containers provide a lightweight alternative to traditional virtualization, enabling faster startup times, efficient resource utilization, and the ability to run multiple isolated containers on a single host machine. These characteristics make Docker an ideal choice for orchestrating microservices and deploying distributed applications, where scalability and efficient resource allocation are paramount.

2.7 Security

2.7.1 HTTP

As described in the book “HTTP: The Definitive Guide“ [14], HTTP serves as the universal language for communication between web browsers, servers, and related web applications on the internet. It facilitates the reliable transmission of web content from servers to clients, ensuring the integrity of the data exchanged. Web content is stored on HTTP servers, which respond to requests from HTTP clients by providing the requested data along with relevant information such as object type and length. This interaction between HTTP clients and servers forms the fundamental basis of the World Wide Web.

Building upon the foundation of HTTP, HTTPS (HyperText Transfer Protocol Secure) serves as an enhanced and more secure version, as indicated in the referenced articles [15, 16]. It operates as an application-specific implementation that encapsulates HTTP within SSL (Secure Sockets Layer) or TLS (Transport Layer Security) protocols. HTTPS plays a vital role in ensuring secure data transmission over the internet. By encrypting the data exchanged between servers and clients, HTTPS mitigates the risk of interception and unauthorized access by malicious entities. It serves as a crucial method for securing web servers and ensuring network security.

A comprehensive analysis of HTTPS deployments and associated challenges was conducted, as mentioned in one of the cited articles [15]. This analysis highlights the effectiveness of HTTPS in establishing end-to-end secure connections, even when modifications are introduced during the communication between clients and web servers. The utilization of HTTPS in the design of the webpage in question will significantly enhance its security measures.

2.7.2 Login

Authentication is a fundamental aspect of user login in web applications, serving the purpose of verifying the claimed identity of a user. There are three primary types of authentication: single-factor, two-factor, and three-factor authentication [17].

Single-factor authentication, which relies on knowledge-based factors, is commonly used in web applications. This method typically involves the use of a username and password combination for user verification.

The second type, two-factor authentication, requires the user to provide two independent pieces of information for authentication. For instance, in addition to a username and password, a user may be required to enter a code received through SMS.

Three-factor authentication, as the most stringent form, necessitates the provision of three independent factors for authentication. This may include a combination of a username, password, SMS code, and even a retina scan.

Authorization, on the other hand, is the process of granting access rights to users for accessing specific information or functionalities within an application. It ensures that each user type is allocated appropriate access privileges based on their role or level of authorization [17].

In our proposed implementation, we will incorporate a model that encompasses authentication through single-factor authentication using a username and password. Additionally, we will address authorization by assigning different access levels and permissions to distinct user types within the application.

2.7.3 User types

The application will cater to three distinct user types, each possessing varying levels of authority and privileges. The most common user category is the “normal user“ while the least common is the “admin“. Furthermore, users will be categorized as either “registered“ or “unregistered.“ Registration will grant access to additional features, but it is essential to acknowledge that the requirement to register may deter some users from contributing content to the platform.

The first user type is the “visitor“ who possesses the lowest level of authority. Visitors can browse the web page, view its contents, and download annotated datasets. However, to contribute to the platform, a visitor must undergo the registration process. This measure aims to identify and prohibit users from adding inappropriate or nonsensical datasets.

The “normal user“ category enjoys the same privileges as a visitor but gains the additional capability of adding datasets to the platform. This empowers them to actively contribute to the content pool.

The “admin“ user type possesses the highest level of authority within the system, granting them comprehensive privileges. Administrators have unrestricted access and control over all aspects of the application, enabling them to perform various administrative tasks and maintain the platform’s integrity and functionality.

It is crucial to establish these user categories and corresponding privileges to ensure a structured and controlled environment within the application, promoting user engagement and content quality.

2.7.4 JSON web token

All the information provided in this section was taken from jwt.io website [18]. JSON Web Token (JWT) is a widely adopted open standard (RFC 7519) that establishes a way to securely transfer data between parties using a JSON object. This data can be trusted because it's digitally signed, either by a secret key using the HMAC algorithm or a public/private key using RSA or ECDSA.

Mainly, JWT is employed for two purposes: Authorization and secure information exchange.

Authorization: JWT is very helpful in authorization scenarios. When a user logs in, they receive a JWT which needs to be included in their subsequent requests to get access to resources. This token confirms that the user has permission to access these resources. Single Sign On services widely use JWT due to its small overhead and its ability to be used across various domains.

Information Exchange: JWT is an effective means for securely transferring data between parties. As JWTs are signed, we can verify the authenticity of the sender. Moreover, as the signature is based on the header and the payload, we can also confirm that the content wasn't altered.

How does it work? Upon successful login, the user is returned a JWT. Thereafter, every time the user tries to access a protected resource, they must send the JWT, usually in the Authorization header. The server checks for a valid JWT in this header and if it's valid, access is granted. It's important to take precautions while handling tokens due to their sensitive nature. Storing tokens longer than necessary or keeping sensitive data in browser storage should be avoided.

2.8 Frontend

According to the Stack Overflow survey conducted in 2022, the prevailing client-side languages utilized in web development include JavaScript, HTML/CSS, Python, Java, C#, and PHP [19]. These languages are widely employed by developers to create robust and interactive user interfaces for web applications. The incorporation of frameworks further enhances development efficiency by providing libraries and pre-built components that reduce the amount of code required.

Frameworks offer numerous advantages, particularly in simplifying the implementation of common tasks encountered during web application development. Tasks such as form validation, data sanitization, and CRUD operations often involve repetitive processes. Instead of reinventing solutions for these tasks, frameworks provide developers with pre-existing functions and modules specifically designed to handle them, streamlining the development process. Prominent frameworks widely used in the industry include PHP Laravel, Symfony, JavaScript React, Vue.js, Python Django, and others [20].

The front-end of a web application typically comprises three primary languages: HTML, CSS, and JavaScript. Each language serves a distinct purpose in shaping the visual and interactive aspects of the user interface. HTML (HyperText Markup Language) defines the structure and organization of web content, specifying the elements and layout of the page. CSS (Cascading Style Sheets) governs the presentation and styling of HTML elements, controlling visual aspects such as color, typography, and layout. JavaScript, on the other hand, adds interactivity and dynamic behavior to the web page, facilitating client-side scripting and enhancing user engagement.

Considering the widespread popularity of JavaScript, employing it as the primary

client-side language is a good choice. JavaScript boasts a substantial user base, ample resources available on platforms like Stack Overflow, and a plethora of JavaScript implementations and libraries. These factors contribute to a vibrant development ecosystem that supports the language's continuous growth and adaptation to evolving industry needs.

2.8.1 HTML

HTML, an acronym for Hypertext Markup Language, serves as a fundamental markup language extensively employed in crafting web pages and various forms of online content for rendering in web browsers. It operates by utilizing a collection of tags designed to annotate and structure elements within a web page, encompassing elements such as headings, paragraphs, images, and hyperlinks. Originally derived from the Standard Generalized Markup Language (SGML), HTML has progressively developed to establish its distinctive standard. Presently, the World Wide Web Consortium (W3C) undertakes the role of its custodian, assuming responsibility for the ongoing development and maintenance of HTML and associated web standards.

Functioning as a client-side language, HTML is subject to interpretation by the web browser deployed on the user's device. Upon user request, an HTML page is transmitted from the web server to the client's browser, which subsequently interprets and renders the HTML code, displaying the resulting web page on the user's screen [21]. The accessibility and ease of use associated with HTML contribute to its popularity and widespread adoption, enabling a diverse range of individuals to engage with the language. Moreover, HTML offers significant customizability, empowering developers to fashion web pages exhibiting a broad spectrum of visual styles and functional attributes.

In recent times, HTML has undergone notable advancements, most prominently with the introduction of HTML5. This iteration introduced a host of new elements and attributes tailored to accommodate multimedia content and interactive experiences. Furthermore, HTML5 embraces cutting-edge technologies like WebGL and WebSockets, facilitating the development of immersive web applications and captivating games [22]. The continuous evolution of HTML reinforces its pivotal role in shaping the modern web and underpins its enduring significance in facilitating captivating and interactive online experiences.

2.8.2 CSS

During the early stages of the web, HTML faced limitations in terms of structural markup, leading to the introduction of presentational elements like `` and `<BIG>` to cater to the growing demand for new elements. However, this approach resulted in a conflation of structural and presentational concerns, impeding effective content indexing, accessibility, and maintainability. Consequently, the need arose for a solution that would disentangle presentation from structure, enabling greater flexibility and sophistication in styling. As a response to these challenges, Cascading Style Sheets (CSS) was developed.

CSS was designed to separate the presentation layer from the structural markup, affording enhanced control over styling. Unlike HTML, CSS offers a comprehensive range of styling options, encompassing the ability to define colors, create borders, regulate spacing, and incorporate various other features. By employing CSS for presentation control, developers are empowered to structure content in a manner that optimizes accessibility and indexing while simultaneously achieving the desired visual effects [23].

The introduction of CSS brought about a paradigm shift in web design, fostering

improved modularity, maintainability, and extensibility. By decoupling presentation concerns from the underlying structure, CSS facilitates a more structured and sustainable approach to web development, ensuring the longevity and adaptability of web content.

2.8.3 JavaScript

JavaScript is a versatile programming language utilized to provide instructions for diverse computational tasks, which are sequentially processed by computers. As an interpreted language, JavaScript necessitates a program to convert its code into machine-readable instructions each time it is executed. It is important to note that JavaScript is distinct from Java and is renowned for its user-friendly nature and robust capabilities. Initially known as LiveScript in Netscape Navigator 2, it was subsequently renamed JavaScript, while Microsoft introduced its own variant called JScript. Due to its widespread availability and integration with popular web browsers, JavaScript has emerged as a favored choice among scripting languages. In the web context, web servers store web pages identified by IP addresses, while domain name servers perform the crucial task of converting these addresses into more human-readable domain names [24].

TypeScript, on the other hand, is a statically typed programming language that incorporates static type checking during compilation to mitigate potential runtime issues. Serving as a superset of JavaScript, TypeScript offers advanced structuring mechanisms for managing complex codebases, including class-based object orientation, interfaces, and modules. The type checking capabilities of TypeScript operate exclusively during design time, ensuring that no additional runtime overhead is incurred. The resulting code is highly compatible with web browsers and can target ECMAScript 3, 5, and 6. TypeScript is designed to align with existing

and forthcoming ECMAScript proposals and is freely available as a cross-platform development tool under the open-source Apache license [25].

Node.js, in turn, empowers developers to execute JavaScript outside the confines of web browsers. Within the Node.js environment, all JavaScript code is processed by the V8 engine, originally developed for the Google Chrome browser. The widespread adoption of Google Chrome reflects the significance of V8 in web development, owing to its exceptional speed and seamless integration across platforms [26].

The adoption of frameworks has become indispensable in modern JavaScript development. Notably, the primary frameworks in the JavaScript ecosystem encompass Angular, React, and Vue. Comparing these frameworks, a study conducted by Elar Saks [27] provides valuable insights. Analyzing data from GitHub, NPM, and Stack Overflow, it is evident that Vue exhibits the highest popularity on GitHub, whereas React dominates the NPM ecosystem. In terms of Stack Overflow, React and Angular share the lead. Vue stands out for its intuitive syntax and ease of learning, while Angular proves to be the most challenging framework. Regarding performance, Vue demonstrates superior speed, whereas Angular lags behind, exhibiting the largest production build. React generally outperforms Angular in various benchmark tests. In summary, React emerges as the most popular choice, making it advantageous for job seekers. Vue shines in terms of simplicity and performance, and Angular excels in building large-scale applications. It is noteworthy that Angular and React default to TypeScript, whereas Vue grants users the flexibility to choose their preferred language. Angular, React, and Vue are all open-source frameworks, ensuring active community maintenance and offering unrestricted download, usage, and modification rights.

Considering the aforementioned comparison, Angular emerges as the most suitable

framework for constructing large-scale applications. React, being highly sought after by employers, holds significance for job seekers. Meanwhile, Vue stands out for its user-friendly learning curve and exceptional performance. However, the selection of an appropriate front-end framework should be guided by various factors, such as project scale, complexity, scalability, and team expertise. In my case, as a non-job seeker and considering the size and requirements of my webpage, Vue.js presents itself as the optimal choice.

Vue.js will be used together with vitebuild tool. As said in [28]. Vite is a frontend development tool created by Evan You, the founder of Vue.js. It offers a faster and more efficient development experience for modern web projects. Key features include native ES module support, faster dev server startup, quick updates with Hot Module Replacement (HMR), and an optimized production build. Vite leverages Rollup's flexible plugin API for better performance and flexibility. It's compatible with Vue.js, TezJS, and React, and ongoing development efforts aim to enhance its performance further. Overall, Vite stands out for its speed, streamlined workflow, and commitment to modern web development practices.

2.9 Backend

When deliberating on the architectural approach for developing a web application, two primary options come to the forefront: server-based architecture and serverless architecture. Serverless architecture can be classified into two domains: Backend as a Service (BaaS) and Functions as a Service (FaaS). BaaS involves transferring the backend functionality to external servers offered by third-party companies, while FaaS provides a platform for users to write application-specific functions that respond to events without the burden of managing underlying infrastructure. It is noteworthy that BaaS encompasses the entirety of server functionality, while FaaS

focuses solely on event-driven functions. An inherent advantage of serverless architecture lies in its pay-per-use model, facilitating cost optimization. However, in the traditional serverless approach, charges may still apply for idle time [29].

Conversely, server architecture refers to the conventional approach of constructing and deploying applications, where dedicated servers handle incoming requests and perform the necessary processing. Opting to use JavaScript/TypeScript on the backend when following server architecture proves judicious, given its compatibility with the frontend. Such a decision offers the advantages of streamlined development processes, reduced compatibility issues, and seamless data exchange between the frontend and backend using JavaScript-based JSON. Furthermore, by employing NestJS, a framework constructed on top of Node.js and TypeScript, developers can build modular and scalable architectures. NestJS is tailored for efficient and scalable server-side applications, supporting TypeScript and incorporating elements of Object-Oriented Programming (OOP), Functional Programming (FP), and Functional Reactive Programming (FRP). It utilizes Express as the default HTTP server framework and can optionally be configured to use Fastify. Nest abstracts underlying frameworks while exposing their APIs, allowing developers the flexibility to utilize third-party modules for enhanced adaptability.[30].

Ultimately, the choice between server and serverless architecture hinges upon the specific requirements of the application at hand. Nevertheless, serverless architecture presents several compelling advantages. Firstly, it enables developers to primarily concentrate on application logic, obviating the need for extensive server-side management and configurations. Secondly, serverless computing fosters improved scalability by automatically allocating resources based on demand, culminating in enhanced performance. Moreover, serverless architecture has the potential to yield cost savings and reduced latency compared to traditional server-based computing

[31].

Multiple serverless platforms exist, and the optimal selection is contingent on individual needs. Among these platforms, Google Cloud Functions stands out by offering numerous advantages over competitors like AWS Lambda and Azure Functions. Google Cloud Functions provides an attractive free offering, encompassing \$300 in free credits during the initial year and 5GB of perpetual free storage. It seamlessly integrates with other Google Cloud Services, such as Kubernetes Engine, App Engine, and Compute Engine. The platform features comprehensive documentation, ensuring ease of use and manageability. Developers can build and test functions using a standard Node.js runtime alongside their preferred development tools. Notably, Google Cloud Functions does impose a limit of 1,000 functions per project, which surpasses that of Azure Functions but falls short of AWS Lambda. Considering the benefits and features offered by Google Cloud Functions, particularly its enticing free tier, it emerges as a fitting choice for the present project [32, 33].

2.9.1 Database

When considering the selection of a database for the web application, it is crucial to store data and their relationships efficiently. Various types of databases are available, including relational databases like PostgreSQL, graph databases such as Neo4j, non-relational databases like MongoDB, and NewSQL databases like SurrealDB. NewSQL databases belong to the relational database class and aim to provide scalability comparable to non-relational databases while preserving ACID (Atomicity, Consistency, Isolation, Durability) properties [34]. ACID represents a set of principles that ensure reliable and robust data operations. Careful consideration of data structure, required database operations, and the best fit for the

project is necessary when making a decision.

For knowledge graph databases, several options are available, and the choice depends on the specific use case. Popular choices include Neo4j, RDF triple stores, and graph databases like Amazon Neptune and Microsoft Azure Cosmos DB [35].

Neo4j and Amazon Neptune are both graph databases suitable for knowledge graphs. Neo4j is a scalable, ACID-compliant graph database that has been extensively tested for performance and scalability. It enables users to establish connections between text and data in large knowledge graphs. On the other hand, Amazon Neptune is a cloud-based graph database that offers speed and reliability. It supports both RDF and Gremlin graph models simultaneously. In terms of performance, Amazon Neptune has room for improvement, but with the resources available from Amazon, enhancements are expected. In contrast, Neo4j has a proven track record of delivering high performance and scalability [35, 36].

Ultimately, the choice between Neo4j and Amazon Neptune depends on the specific needs and use cases of the project. Neo4j can be deployed locally on a personal computer or on AWS using a Partner Solution developed by AWS and Neo4j. On the other hand, Amazon Neptune is a cloud service and does not offer local deployment. Considering these arguments, it is evident that Neo4j is the preferred choice, given its battle-tested performance and scalability and the flexibility of deployment options [35, 36].

In the deployment phase, Neo4j serves as the chosen graph database solution. The deployment options include setting up a virtual machine and installing Neo4j locally or opting for Neo4j Aura. As detailed in the source [37], Neo4j Aura stands out as Neo4j’s fully managed cloud service. It provides a highly efficient, dependable, and scalable graph database solution delivered entirely as a cloud service.

Neo4j Aura is characterized by its exceptional speed, reliability, and scalability, offering a seamless and automated experience for managing graph databases in the cloud environment.

2.9.2 Adapter pattern

The adapter pattern, as described in Harmes et al. (2008), is a design pattern that facilitates the adaptation of incompatible interfaces between classes. This pattern involves the use of objects, often referred to as wrappers, that encapsulate another object within a new interface. Adapters play a crucial role for programmers and interface designers in situations where existing APIs cannot be directly used with certain interfaces. By employing adapters, it becomes possible to incorporate these classes into the system without the need for direct modifications. The image below illustrates the various components that need to be implemented to effectively utilize the adapter design pattern [38].



Figure 2.2: Adapter design pattern [39]

In the context of software engineering, design patterns like the adapter pattern serve as valuable tools for addressing interface compatibility challenges. They provide a structured and reusable approach to enable communication and interaction between classes that would otherwise be incompatible. By employing the adapter pattern, developers can achieve flexibility, maintainability, and extensibility in their software systems. The adapter pattern is particularly beneficial when integrating legacy code, utilizing third-party libraries or APIs, or when designing flexible

systems that can accommodate future changes in interface requirements [38].

In our project, currently utilizing Neo4j as the database system, the significance of the adapter design pattern becomes apparent. Anticipating potential shifts in the choice of databases or even a transition from the NestJS framework to another, the adapter design pattern emerges as a fitting solution. This pattern allows us to seamlessly switch between different frameworks while minimizing the need for extensive modifications to the existing codebase.

Consider the scenario where a change in the database system, perhaps a transition to PostgreSQL or another database, is contemplated. In such cases, the repository design pattern proves to be a better match for handling database changes. However, in the broader context of adapting to various backend modifications, including a change in the choice of databases or the framework, the adapter design pattern remains a robust solution.

The implementation of adapters acts as intermediaries between the application frontend and the backend, shielding the frontend from the intricacies of changes at the backend, such as alterations in the choice of databases.

The adapter design pattern introduces a layer of abstraction that enables the frontend to remain agnostic to the technologies employed in the backend. This means that even if there are changes in the backend, be it the choice of database or the framework the frontend can remain untouched, provided the API stays the same.

In the provided example (see 2.3), the frontend serves as the client, the backend functions as the adapter, and the database acts as the adaptee. The frontend communicates with the backend through a well-defined API, maintaining loose coupling with the backend and insulating itself from the underlying changes. The

backend, acting as an adapter, translates and adapts requests between the frontend and the database, ensuring a seamless interaction.

This modular and adaptable architecture aligns with the principles of loose coupling and abstraction. The separation between the frontend, backend, and database creates a system where changes to the backend, such as a switch in databases or modifications in the internal implementation, do not necessitate alterations in the frontend as long as the API remains consistent.



Figure 2.3: Adapter design pattern in project

2.10 Data model

To create an effective data model, it is crucial to acquire a comprehensive understanding of the terminologies used in the field of AI and Data Science. This knowledge will serve as a foundation for designing a data model that aligns with the principles and concepts prevalent in this domain. Additionally, exploring the technologies employed in AI and Data Science and engaging in direct discussions with experts from this field can offer valuable insights.

Artificial intelligence (AI) has been a prominent field for several decades, encompassing various subfields such as machine learning (ML) and deep learning (DL). DL, specifically, refers to a type of AI that involves neural networks with intricate layers. The key distinction between DL and other neural networks lies in its capacity to accommodate a higher number of layers, neurons, and computational

power, facilitating the representation of complex models and automatic feature extraction. DL is, therefore, considered a subset of ML, which, in turn, falls under the broader umbrella of AI.

Over time, the field of AI has witnessed significant technological advancements, as demonstrated in (see 2.4). These advancements have contributed to the development and adoption of sophisticated algorithms, increased computational capabilities, and the availability of vast amounts of data. Such progress has propelled AI and ML to the forefront of various industries, enabling organizations to leverage data-driven insights for decision-making, automation, and problem-solving. [40]

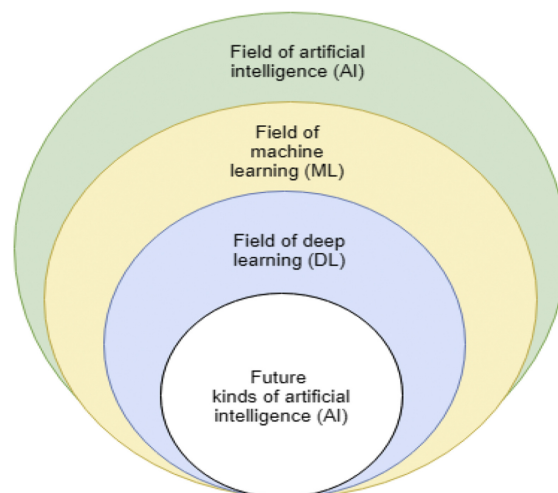


Figure 2.4: Artificial intelligence development and expansion. [40]

To ensure the adequacy and effectiveness of our data model, it is advisable to actively engage with professionals and experts in the field of AI and Data Science. Collaborating with individuals specializing in these domains will provide invaluable guidance and expertise regarding the specific AI concepts, techniques, and technologies that are relevant to our project. By seeking direct interaction and knowledge exchange, we can refine our understanding, gain firsthand insights into best

practices, and make informed decisions when constructing our data model.

By incorporating the fundamental AI concepts, exploring cutting-edge technologies, and tapping into the expertise of domain specialists, we can create a robust data model that not only adheres to the principles of AI and Data Science but also enables accurate analysis, efficient processing, and the extraction of meaningful insights.

In the forthcoming sections, an exhaustive exploration of pivotal facets within the realm of artificial intelligence (AI) shall be undertaken. The primary objective is to explicate the aforementioned concepts in a lucid and accessible manner, affording readers a comprehensive comprehension of AI and its associated lexicon within a broader context.

2.10.1 Artificial intelligence

Artificial Intelligence (AI) represents a rapidly evolving technology that empowers machines to engage in cognitive functions encompassing perception, reasoning, learning, and interaction. This transformative paradigm is anticipated to revolutionize industries across the globe, with projected revenue increments exceeding ten trillion dollars. The realization of AI hinges upon the orchestration of algorithms, big datasets, and growing computational capabilities. Central to the AI endeavor is the pursuit of pattern identification, necessitating the acquisition of relevant datasets and the instruction of algorithms to discern discernible patterns. The efficacy of algorithms assumes paramount importance, as they form the bedrock upon which AI models are constructed. Mathematical constructs underpin the decision-making process in AI, involving the multiplication of incoming data with weighted vectors, thereby enabling informed judgments and responses. Over the years, the field of AI has witnessed big breakthroughs, fostering advancements in algorithms

and nurturing the growth of this transformative technology [41].

The origins of AI can be traced back to the imaginative realms of philosophers and science fiction writers. The emergence of early AI research was fueled by remarkable developments, such as the introduction of “The Turk“, a chess-playing automaton, during the 18th and 19th centuries. This invention served as a catalyst, stimulating further exploration into the realms of AI. Significant milestones further propelled the field, including Isaac Asimov’s captivating short story “Runaround“ in 1942 and Alan Turing’s creation of “The Bombe“, the first operational electro-mechanical computer. Turing’s pioneering work culminated in the formulation of the Turing test in 1950, a seminal contribution to evaluating machine intelligence. The watershed moment arrived in 1956 with the Dartmouth Summer Research Project on Artificial Intelligence, which marked the formal inception of AI as an independent field of study. Notwithstanding subsequent setbacks attributed to waning government support, a pivotal resurgence occurred with the advent of Google’s AlphaGo in 2015. Powered by Deep Learning techniques and artificial neural networks, AlphaGo astounded the world by defeating the reigning Go champion, underscoring the remarkable progress achieved in the realm of AI [42].

It is imperative to recognize that AI’s transformative potential is contingent upon a multidisciplinary approach, engaging scholars and experts across diverse domains. A scholarly exploration of the underlying principles, advancements, and historical context of AI, coupled with a steadfast commitment to ongoing research and collaboration, shall foster a comprehensive understanding of this big field. By embracing the academic discourse and continuously monitoring the latest developments, stakeholders can position themselves at the vanguard of AI, capitalizing on its immense potential to reshape industries and drive innovation.

2.10.2 Machine learning

Throughout history, humans have harnessed a diverse array of tools and machines to streamline tasks and fulfill various needs, spanning domains such as transportation, industrial processes, and computing. Within the realm of computing, machine learning has emerged as a pivotal field, enabling machines to acquire knowledge and enhance their performance without explicit programming. Coined by Arthur Samuel, machine learning involves equipping computers with the ability to learn from data and improve their efficiency in handling information. Given the abundance of datasets available today, the demand for machine learning methodologies has surged, with industries leveraging its power to extract meaningful insights from vast volumes of data. At the core of machine learning lies the objective of learning from data, necessitating the utilization of diverse algorithms tailored to specific problem domains, the number of variables involved, and the optimal model for the given context [43].

Supervised learning serves as a prominent paradigm within machine learning, finding applications across various domains, including text mining and web applications. It revolves around leveraging past data to enhance performance in real-world tasks, with a primary focus on learning a function that predicts discrete class attributes. Supervised learning has witnessed extensive research and practical adoption, particularly in the realm of web mining applications [44].

In contrast, unsupervised learning represents a methodology wherein a system learns to represent input patterns based on statistical structures without explicit target outputs or environmental evaluations. This form of learning holds considerable significance, as it aligns closely with the natural learning processes observed in the human brain. Unsupervised learning methods operate on observed input patterns, extracting underlying statistical regularities and relevant information.

Within the domain of unsupervised learning, two distinct classes can be identified: density estimation, wherein statistical models are constructed, and feature extraction, which directly extracts statistical regularities from input data [45].

Furthermore, semisupervised learning presents a viable approach that leverages both labeled and unlabeled data to train models. Particularly in scenarios involving large datasets, the process of labeling data, i.e., assigning outcomes to instances, often proves time-consuming and resource-intensive. In an effort to enhance model performance, semisupervised learning supplements sparsely labeled data with a wealth of unlabeled data, with research demonstrating the potential of unlabeled data to improve classifier performance. However, careful model selection remains paramount in achieving optimal outcomes within the realm of semisupervised learning [46].

2.10.3 Artificial neural networks

Artificial Neural Networks (ANNs) represent computational systems inspired by the intricate structure and functionality of biological neural networks. Comprising numerous interconnected processors, ANNs serve as potent tools in the realm of information processing, catering to tasks such as pattern recognition, forecasting, and data compression. These networks are characterized by inputs that undergo transformation via multiplication with assigned weights and subsequent activation through mathematical functions within individual neurons. Activation occurs through the summation of weighted inputs, with the weights themselves determined through a process of learning or training aimed at minimizing error. ANNs prove particularly effective in tackling complex problems like pattern recognition, clustering, categorization, and prediction. Within the domain of ANNs, two primary categories exist: feed-forward networks, which produce a singular set of output

values, and recurrent (or feedback) networks, which generate a sequence of values based on given inputs. The multilayer perceptron architecture represents a notable example of a fully connected, three-layer feed-forward network, featuring an input layer, hidden layers, and an output layer. Additionally, the radial basis function network utilizes radial basis functions as activation functions, further expanding the repertoire of ANN applications encompassing function approximation, time series prediction, and system control [47].

Feedforward neural networks (FNNs) are renowned for their exceptional performance in supervised learning scenarios and find application across diverse fields. Among the simplest variants of FNNs is the single-layer feedforward network (SLFN) encompassing a solitary hidden layer. SLFNs gain popularity due to their approximation capabilities and inherent simplicity, making them viable for deployment in domains such as time-series prediction, control systems, signal processing, and more. The learning process associated with SLFNs involves two primary objectives: determining the optimal network size and seeking the optimal configuration of parameters. Tunable parameters within SLFNs encompass nonlinear parameters from the hidden layer and linear parameters from the output layer, collectively shaping the network's behavior and performance [48].

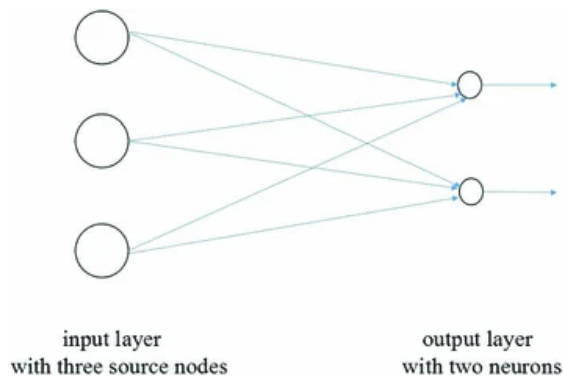


Figure 2.5: Single-layer feedforward networks [49]

A multilayer feedforward network, a prominent variant of artificial neural networks (ANNs), constitutes a sophisticated architecture comprising an input layer, an output layer, and one or more hidden layers interleaved between them. This network structure is employed to establish a mapping between input patterns and corresponding output representations. The underlying behavior and performance of the network are influenced by various factors, including the number of processing units within the hidden layers, the weight and threshold values assigned to connections, and the choice of activation function employed. The primary objective in utilizing a multilayer feedforward network lies in determining the set of functions that can be effectively approximated by the network. This entails characterizing the closure, which pertains to the collection of limit points, encompassing the functions that can be computed by the network. By unraveling the boundaries and capabilities of such networks, researchers and practitioners can gain insights into the expressive power and limitations of multilayer feedforward architectures, further advancing their understanding and utilization in diverse applications [50].

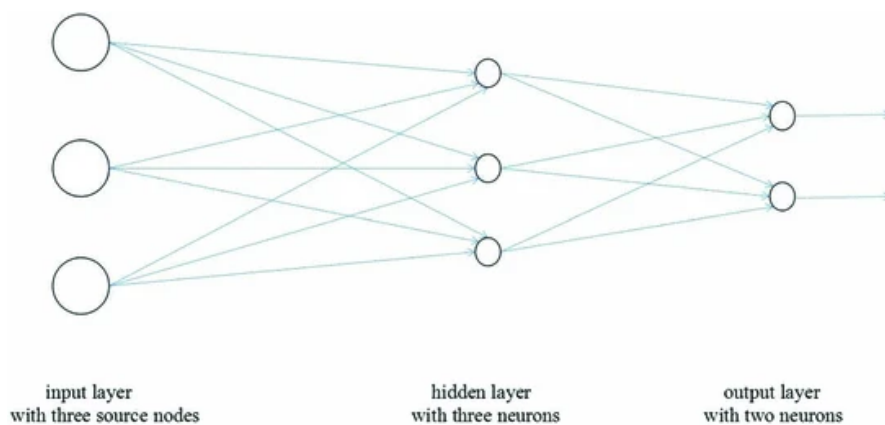


Figure 2.6: Multilayer feedforward network with a single hidden layer [49]

2.10.4 Natural language processing

Natural Language Processing (NLP) is an interdisciplinary field that combines the realms of linguistics, computer science, and artificial intelligence to investigate, comprehend, and generate human language. It encompasses a broad spectrum of computational techniques and theories aimed at representing and processing natural language across various levels of linguistic analysis. The fundamental objective of NLP is to attain a level of language processing that emulates human-like capabilities, enabling its application in diverse tasks and domains. NLP systems operate on authentic, naturally occurring texts, encompassing a multitude of languages and genres, including both oral and written forms. It is essential that the texts under analysis are derived from genuine usage rather than being specifically crafted for analytical purposes. NLP systems employ different levels of linguistic analysis, either individually or in combination, leading to some confusion among non-specialists regarding the precise nature of NLP. Distinctions arise based on whether a system utilizes a subset of these analytical levels, categorizing it as either “weak” or “strong” NLP. The ultimate objective of NLP lies in achieving language processing capabilities that mirror human capabilities across a wide array of tasks and applications. NLP is predominantly regarded as a means to accomplish specific objectives rather than an end in itself. Consequently, NLP finds application in numerous domains, such as information retrieval, machine translation, question-answering systems, and many others [51].

The field of computational linguistics encompasses two primary branches: computational linguistics and theoretical linguistics. Computational linguistics focuses on the development of algorithms and methodologies for analyzing and generating natural language, while theoretical linguistics delves into the study of grammatical competence and language universals. NLP entails intricate processes such as sen-

tence analysis, discourse analysis, and dialogue structure analysis, with sentence analysis further partitioned into syntax and semantic analysis. Sentence analysis, as a pivotal component, endeavors to ascertain the intended meaning of a sentence, often involving the translation of input into a language with simpler semantics, such as formal logic or a database command language. Syntax analysis typically serves as the initial stage in this multifaceted process, aiding in the determination of structural relationships and dependencies within a sentence [52].

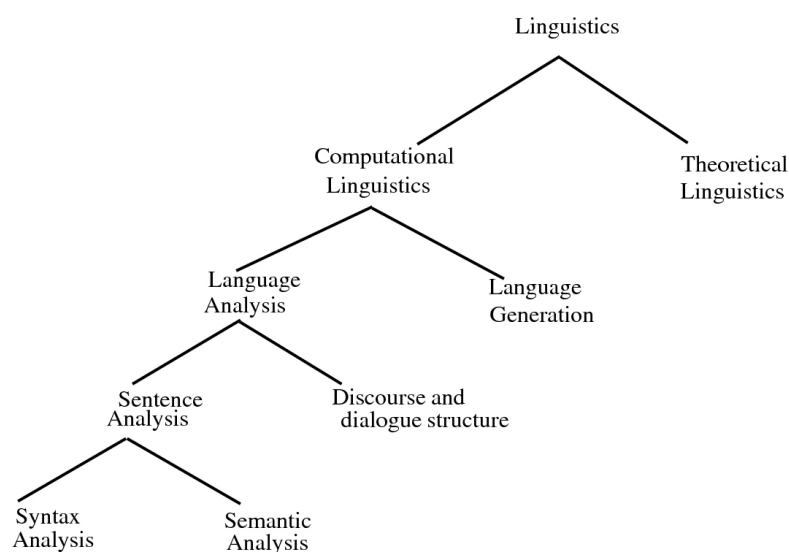


Figure 2.7: Components of Natural Language Processing [52]

2.10.5 Large language model

Large language models (LLMs) such as Bert and GPT-2 have ushered in a paradigm shift in the realms of natural language processing (NLP) and machine learning (ML). These sophisticated models have been specifically devised to generate coherent and contextually appropriate responses to given prompts, harnessing the power of statistical distributions derived from human-generated text. However, it is of utmost importance to discern the inherent nature of LLMs as purely mat-

hematical constructs, devoid of consciousness or a comprehensive understanding of the world akin to that of humans.

While LLMs exhibit remarkable capabilities across an array of tasks, they lack the shared “form of life“ that underpins the foundation of mutual understanding and trust among human beings. Consequently, LLMs are susceptible to generating language that may be deemed inappropriate or biased, particularly when confronted with unfamiliar or sensitive subject matter. Instances have arisen wherein LLMs have produced sexist or racist language due to the presence of biases within the training data.

It is imperative to refrain from anthropomorphizing LLMs and ascribing to them language that insinuates human-like capacities or beliefs. LLMs lack the inherent ability to discern veracity from falsity autonomously, with their responses being solely based on statistical patterns embedded within the training data. Hence, an comprehension of the limitations of LLMs is essential, deploying them as tools rather than surrogates for human intelligence [53].

2.10.6 Deep learning

Deep neural networks represent a prominent class of machine learning algorithms that encompass multiple hidden layers, enabling the automated discovery of intricate representations from raw input data. These networks incorporate advanced neurons equipped with convolutional capabilities and multiple activations, offering enhanced functionality compared to simple artificial neural networks or shallow machine learning approaches. While certain shallow machine learning algorithms are deemed “white boxes“, revealing their decision-making process, the majority of advanced machine learning algorithms, including deep neural networks, are characterized as “black boxes“ with untraceable internal mechanisms lacking interpre-

tability. Deep learning techniques excel particularly in processing high-dimensional data such as text, images, videos, speech, and audio. However, for low-dimensional data and scenarios with limited training data availability, shallow machine learning algorithms often yield superior and more interpretable outcomes. It is crucial to note that deep neural networks do not possess the capacity to address challenges requiring strong artificial intelligence capabilities, such as literal understanding and intentionality. In essence, deep learning represents a potent machine learning approach that provides advanced functionality for processing vast and high-dimensional datasets, while acknowledging that shallow machine learning algorithms can outperform and offer greater interpretability for low-dimensional data and limited training data scenarios [54].

The roots of this field trace back to 1957 when Frank Rosenblatt formulated the perceptron algorithm, a foundational component of deep learning. The perceptron, akin to a neuron, mirrors the fundamental functional unit of the brain and can be expressed through an equation, as depicted in (see 2.9). This equation involves the input vector, denoted as x , a corresponding set of weights, indicated as w , a bias term, represented by b , and an activation function, typically denoted as f . The perceptron encompasses $D+1$ adjustable parameters, encompassing D weights and a bias term, and can be viewed as a form of multiple linear regression augmented by a nonlinear output function f . While the initial activation function employed a step function, contemporary perceptrons utilize diverse monotonic functions such as sigmoidal functions. The output, denoted as y , is determined by the summation of the weighted input vector, together with the bias term, passed through the activation function f [55].

$$y = f\left(\sum_{i=1}^D w_i x_i + b\right) \quad (1)$$

Where:

y is output value,

f is activation function

D is the dimension of input space

w_i is a set of weights corresponding to the input vector

x_i x is the input vector

b is bias,

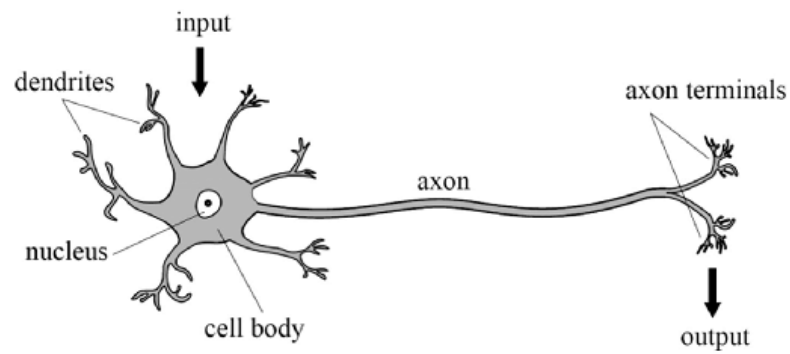


Figure 2.8: Biological neuron [56]

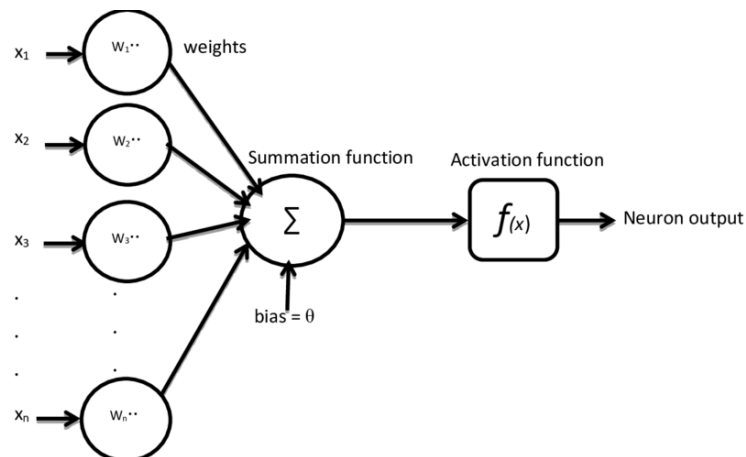


Figure 2.9: Artificial neuron - perceptron [57]

2.11 Data format

The question of data format arises when considering the appropriate representation for data transmission and storage. Different formats, such as JSON and CSV, have their own advantages and considerations. For instance, JSON (JavaScript Object Notation) is a widely used format for structuring data that is human-readable and easily processed by various programming languages. On the other hand, CSV (Comma-Separated Values) is a tabular format often used for representing data in a simple and concise manner. The choice of format depends on factors such as the intended usage, compatibility with existing systems, and the need for structured or unstructured data representation.

When it comes to storing data in databases, the choice of database technology can influence the storage format. For example, a JSON document database like MongoDB offers native support for storing and querying JSON objects, allowing for seamless integration and retrieval of structured data. In contrast, if a PostgreSQL database is used, which primarily deals with relational data, storing JSON objects would require transforming the metadata into binary data. This can be achieved by serializing the dataset into a pickle object, commonly used in Python programming language, and then storing it as a binary array in PostgreSQL. Similarly, for CSV data, it would need to be transformed into a binary format before storage in PostgreSQL.

An inherent challenge arises when dealing with datasets in different formats, as they may have varying column structures and sizes. This necessitates addressing the problem of data format consistency and compatibility. Data preprocessing techniques can be employed to handle these variations, such as mapping or reshaping the data to a standardized format before further analysis or storage. Ensuring

data quality and integrity through data validation and normalization processes is crucial in mitigating such challenges.

In the context of data annotation or labeling, the issue of potential ambiguity arises when two different entities share identical names or labels. Resolving such cases requires disambiguation techniques to distinguish between entities that may appear similar but hold distinct meanings. One approach is to utilize contextual information, such as hover functionality or tooltips, to provide additional specifications or details that disambiguate the identical names. This can aid in conveying the intended meaning and ensuring accurate understanding and interpretation of the annotated data.

Overall, addressing data format considerations, ensuring compatibility between storage systems, handling variations in dataset formats, and resolving ambiguity in annotations are essential aspects of data management and analysis in the realm of information technology and data science.

3 Related Works

The schema.org webpage [58] serves as a scholarly compendium, jointly established by renowned entities such as Google, Microsoft, and Yahoo. It provides an authoritative guide for the systematic development of structured data schemas. This platform offers valuable insights into leveraging existing types of annotation, enabling a comprehensive understanding of vanished columns and their interrelations. Moreover, schema.org acts as a wellspring of inspiration, facilitating the adoption of established structural frameworks and pre-existing types, thus expediting the initial stages of schema development.

In the realm of triage machine learning models, the Kaggle platform [59] assumes a pivotal role in accessing relevant datasets. It offers a excess of resources, inspiring researchers and developers alike. Users can help themselves of downloadable datasets accompanied by comprehensive descriptions, shedding light on the data contained therein. Furthermore, Kaggle provides specific statistics, unveiling the cardinality and distribution of unique values. This information enables researchers to comprehend the nature and scope of each dataset column, including its associated data type. Additionally, metadata pertaining to dataset origins and contributors can be found, along with usage statistics such as views and downloads, which offer valuable insights into the dataset’s popularity and engagement.

In scholarly discourse, the creation of comprehensive schemas assumes significance, particularly in cases where a system is being defined for the first time or remains partially articulated [60]. With an increasing influx of weakly structured and irregular data sources, the extraction of schema information is vital for diverse tasks

like query answering, exploration, and summarization. While semantic web data may contain schema information, it often lacks completeness or is entirely absent. In this context, the academic community has endeavored to survey and categorize schema information extraction approaches. These approaches can be classified into three distinct families: (1) those that exploit the implicit structure of data, irrespective of explicit schema statements; (2) those that utilize explicit schema statements within the dataset to enhance the overall schema; and (3) those that discover structural patterns within datasets. A comparative analysis of these approaches reveals their distinct methodologies, advantages, and limitations. Furthermore, the identification of open challenges underscores the need for continued research in this domain.

4 Design

Designing a project involves careful consideration of the technologies to be employed, the database model, and the overall structure of the web page. This section provides an overview of the technologies that will be utilized, followed by a comprehensive database model showing firebase and Neo4j. Furthermore, the sections showcase diagrams that depict the functionality of the web page, and conclude with the design of high-fidelity wireframes, illustrating the envisioned appearance of the page.

The selection of appropriate technologies is crucial for the success of any project, as it ensures robustness and efficiency. Various modern technologies will be employed, leveraging their unique features and capabilities.

A well-designed database model serves as the foundation for organizing and managing data efficiently. In this project, the Neo4j graph database model will be utilized due to its advantages in handling complex relationships and interconnected data. Neo4j represents entities as nodes and relationships as edges, enabling seamless data traversal and facilitating data-driven decision making. Firebase will primarily be utilized for its authentication functionalities, with additional utilization of its Firestore database.

To provide a comprehensive understanding of the functionality of the web page, a set of diagrams will be presented. These diagrams will illustrate the various components and interactions within the system, including user interfaces, data flows, and system architecture. The use of visual representations aids in conveying the conceptual design and identifying potential bottlenecks and areas for improve-

ment.

4.1 Technologies

In the culmination of the comprehensive analysis conducted, we have carefully examined various technologies and evaluated their respective merits and drawbacks. Based on this assessment, we have made informed decisions regarding the selection of technologies that will be most suitable for the successful execution of our project. This section serves as a recapitulation and final enumeration of the chosen technologies, which will be employed in the development of a globally accessible web page.

In the development of this project, a range of technologies will be employed, resulting in a web page accessible worldwide. The implementation will primarily utilize open source technologies, ensuring transparency and community support. To ensure secure communication, the web page will be deployed with the HTTPS protocol, guaranteeing data confidentiality and integrity. Additionally, a DNS record will be configured to enhance the ease of use and accessibility of the web page.

The front-end of the application will be built using the Vue.js JavaScript framework, specifically version 3. To enhance the maintainability and scalability of the codebase, TypeScript will be incorporated, providing static type-checking and improved code documentation. The Vue Router will enable seamless navigation and routing within the application. For efficient development and building processes, the Vite build tool will be utilized, offering fast and optimized builds. Furthermore, the Composition API will be employed to leverage its reactivity and composition features. To manage the application's state, the Pinia state management library

will be utilized, ensuring effective data management and synchronization across components.

In terms of styling, the Tailwind CSS framework will be employed, offering a utility-first approach that facilitates rapid development and consistent styling throughout the application. By leveraging Tailwind CSS, the design process becomes more efficient, enabling customization and responsiveness. PrimeVue is the project's key component library, providing a versatile toolkit for seamless UI development. Its reliability and community support ensure a strong foundation for long-term adaptability to evolving design trends and technology.

For streamlined deployment and scalability, the application's front-end will be hosted on Firebase Hosting.

On the back-end, the application will utilize Google Cloud Functions as serverless compute solutions. These functions, written in JavaScript, enable the execution of discrete and scalable application logic without the need for managing server infrastructure. To streamline development and enhance code organization, the Nest.js framework will be employed, providing a structured and modular architecture for building scalable and maintainable server-side applications.

The database for this project will be implemented using the Neo4j graph database. Neo4j's graph-based model excels in handling complex relationships and interconnected data, allowing for efficient data traversal and querying. This choice aligns with the project's requirements, enabling seamless integration with the application's data model and facilitating data-driven decision making. This project will also utilize a firebase authentication for the login/registration process and firestore for saving the roles of a user and the information about the dataset.

To ensure code maintainability and minimize redundancy, various design patterns

will be employed throughout the development process. Notably, the Adapter design pattern will be utilized, enabling the flexibility to change database and back-end without the need of changing the front-end if we don't change the endpoints. This design pattern promotes code reusability, scalability, and adaptability, enhancing the overall functionality of the project.

The entire application will be hosted on the Google Cloud Platform (GCP), a cloud computing service that provides a robust and scalable infrastructure. Leveraging GCP ensures reliable performance, scalability, and availability of the web page to users worldwide.

4.2 Web application specification and requirements

This section presents the specifications and requirements for the development of a web application targeting data scientists seeking annotated datasets for training their artificial intelligence models. The application will also provide a platform for users to upload and annotate their own datasets. Users will have the flexibility to access the web application from any internet-connected device. The application will be structured into multiple web pages, categorized based on user preferences. Furthermore, the system will incorporate multiple user types, each with distinct rights and privileges.

4.2.1 Usability

The primary objective of the web application is to ensure usability, enabling data scientists to easily browse and download annotated datasets or annotate their own datasets. Emphasis will be placed on designing a user-friendly interface that facilitates efficient navigation through the available datasets. Intuitive search and

filtering mechanisms will be implemented to streamline dataset discovery, enhancing the overall user experience.

4.2.2 Awareness

The web application will serve as a comprehensive repository of information, encompassing all relevant aspects pertaining to the topic. Detailed descriptions and metadata for each dataset will be provided, equipping users with the necessary knowledge to make informed decisions regarding dataset selection. Additionally, the application will offer insights into the dataset annotation process, guidelines, and best practices, fostering user awareness and understanding of the annotation procedures.

4.2.3 Accessibility

The web application will be publicly accessible on the internet as a free web page, ensuring ease of access for data scientists worldwide. Visitors will have the privilege to view and download datasets without mandatory registration. However, to contribute or annotate datasets, users will be required to register an account. It is important to acknowledge that registration may pose a potential barrier to user participation, as it might discourage individuals from contributing information to the platform.

4.2.4 Ease of use

The web application will prioritize ease of use, aiming to deliver a seamless and intuitive user experience. The navigation system will be thoughtfully designed, allowing users to swiftly locate and access their desired datasets. Consistent and

intelligible user interface elements, such as navigation menus and buttons, will guide users throughout the application, enabling them to effortlessly perform tasks and navigate between pages. Moreover, responsive design principles will be implemented to ensure optimal user experience across diverse devices and screen sizes.

4.3 Target group

This chapter focuses on the target group analysis for the web application, considering the individuals who are interested in data science and data annotation. The aim is to create a clear and intuitive website that is accessible and easy to navigate for users of all ages. It is crucial to provide a seamless user experience to encourage repeated visits and minimize the need for users to seek alternative platforms. Furthermore, considering the diverse range of devices used to access the website, including mobile phones, readers, tablets, and computers, it is essential to develop a responsive site that adapts to various screen resolutions.

4.3.1 Target audience characteristics

The target audience comprises individuals of varying ages who possess an interest in data science and data annotation. The website's design and functionality should cater to this broad demographic, ensuring usability and accessibility for users of any age. The navigation system should be intuitive, employing clear menus that encompass all main categories. By creating a user-friendly interface, the website can retain users, providing a comprehensive and engaging experience that discourages them from seeking alternative platforms.

4.3.2 Device usage statistics

An analysis of device usage statistics is crucial for understanding the preferred platforms through which users access the web application. Research conducted in [61] reveals that mobile devices, including mobile phones, accounted for approximately 51.3% of internet usage in 2016, and this figure rose to 53% in 2019. Although the desktop web traffic is relatively higher at 56.7% in the same year. It is evident that mobile internet use is steadily increasing and has the potential to surpass desktop usage in the near future. Despite the higher proportion of page views from desktop computers, it is important to note that mobile users tend to spend more time on the site. Therefore, it is imperative to develop a responsive website that can adapt to different screen resolutions, providing an optimal user experience across various devices.

4.4 Database model

In (see 4.1), an illustrative depiction of a database representation utilizing the diagram.io platform is presented. The diagram portrays a visual representation of the underlying data structure and relationships within the database system.

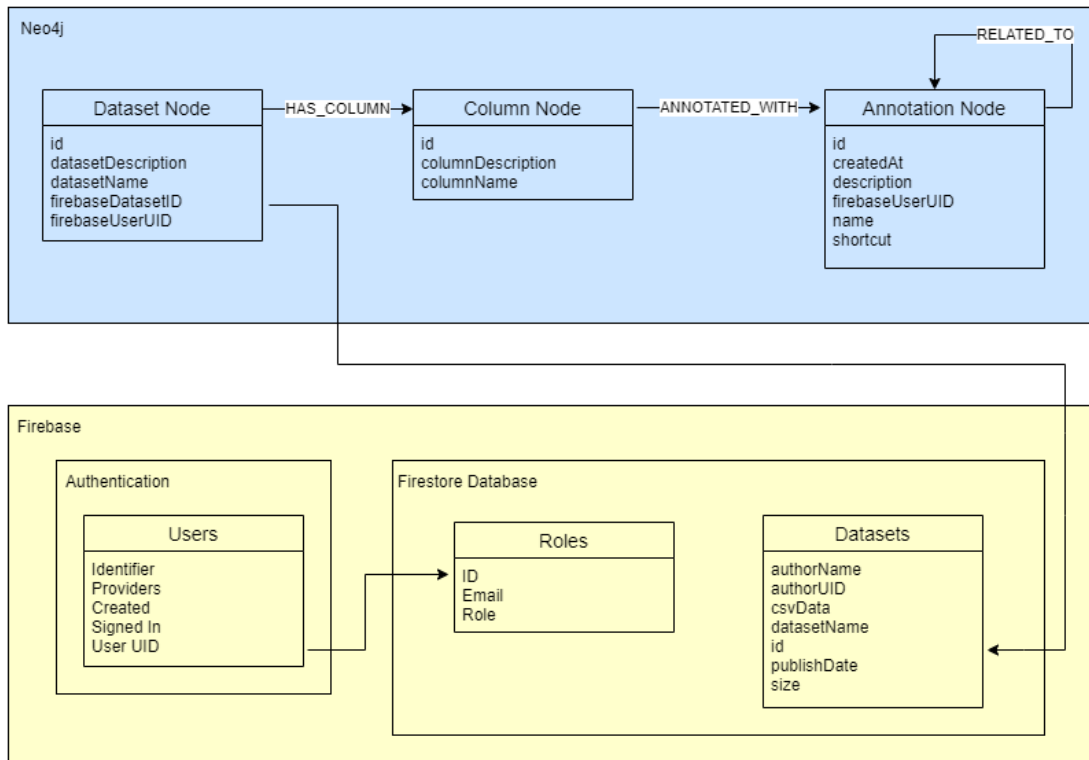


Figure 4.1: Neo4j database model

The following text describes a graph data model in Neo4j and firebase, with a structure suitable for a knowledge base that stores data and user annotations about datasets. Here's a breakdown of what each part does:

Firestore:

- **Firestore Authentication Users:** Represents a user of the system.
- **Firestore Database Roles:** Represents the role of a user the role is mapped to the user by UserUID.
- **Firestore Database Datasets:** Represents a dataset that the user has uploaded.

Neo4j:

- **Node Dataset:** Represents a dataset that the user has uploaded. Inside of neo4j database. This dataset is mapped to the one on firestore by firebase-DatasetID.
- **Node Column:** Represents a column in the dataset.
- **Relationship HAS_COLUMN:** Indicates that the column belongs to the dataset.
- **Node Annotation:** Represents an annotation that user has added into our knowledge base.
- **Relationship ANNOTATED_WITH:** Indicates that the column has been annotated by this annotation.
- **Relationship RELATED_TO:** Indicates that this annotation is related to another annotation.

4.5 Diagrams

4.5.1 Login diagram flowchart

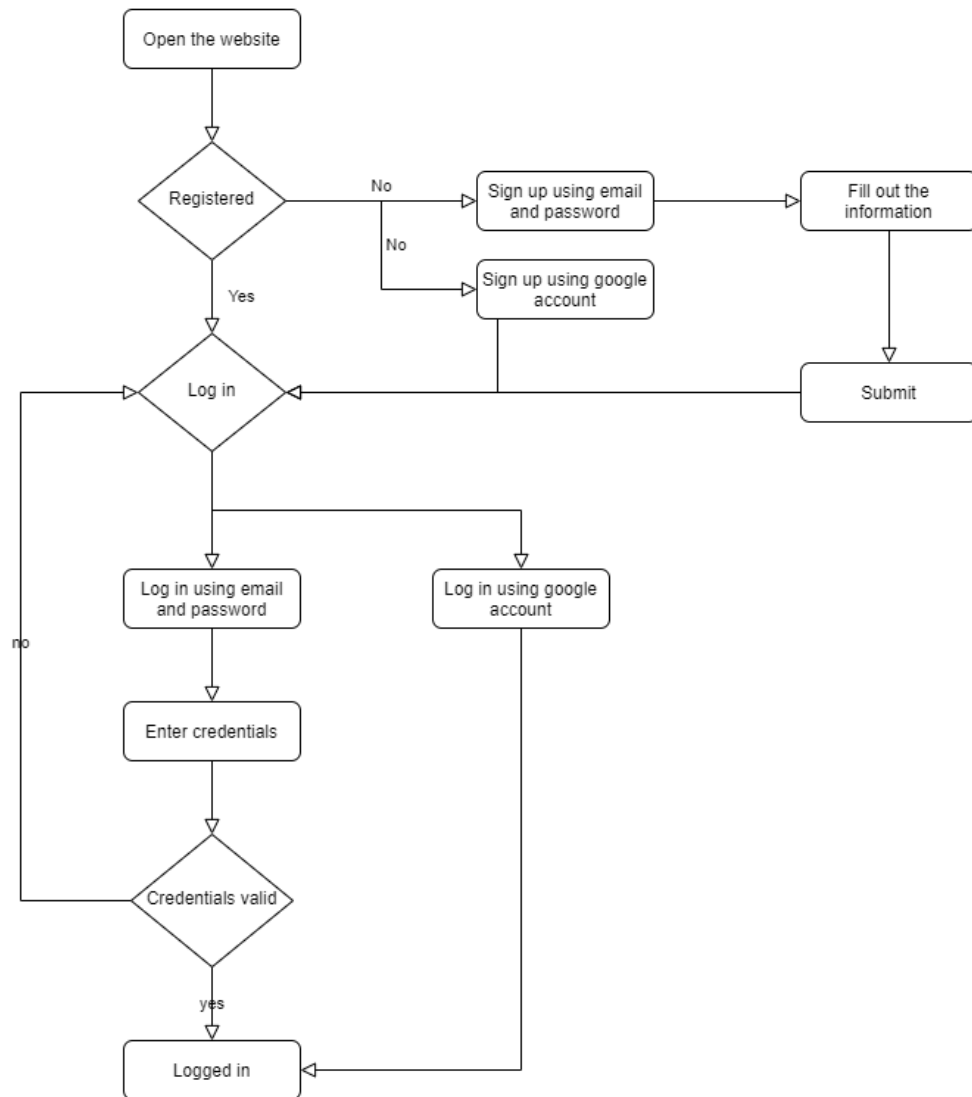


Figure 4.2: Login diagram flowchart

The flowchart outlines the user journey upon accessing the website. It begins by checking the user's registration status. If the user is not registered, they are directed

to the sign-up process, where they provide necessary information and submit the form. Alternatively, users can opt for a streamlined sign-up process using their Google account, facilitated by Firebase Authentication.

After successful registration, or if the user is already registered, the flowchart proceeds to the login process. Users enter their credentials, and the flowchart validates the information. If the credentials are valid, the user is successfully authenticated and logged into the system. In case of invalid credentials, users are redirected to the login step for correction. Additionally, users have the option to log in using their Google account as an alternative method.

4.5.2 Add dataset diagram flowchart

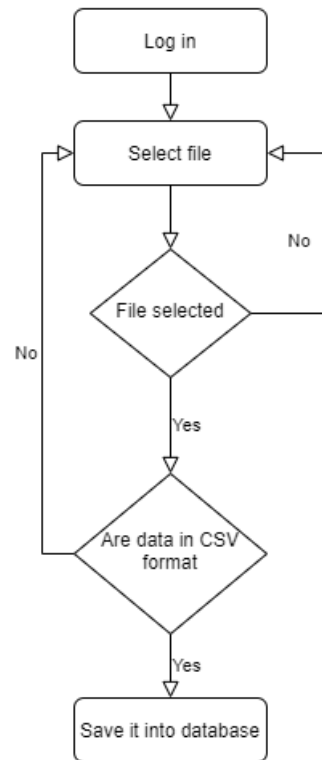


Figure 4.3: Add dataset diagram flowchart

The flowchart begins with login then the flowchart proceeds to the next step, which involves selecting a file. If the user fails to select a file, the flowchart loops back to the “Select File“ stage, prompting the user to choose a file.

If the file is not in the CSV format, the flowchart loops back to the “Select File“ stage, allowing the user to select a different file. On the other hand, if the data is indeed in CSV format, the flowchart proceeds to the next step, which involves saving the csv data inside firestore and also saving other information as who added this dataset, when and how big it is.

4.5.3 Annotate dataset diagram flowchart

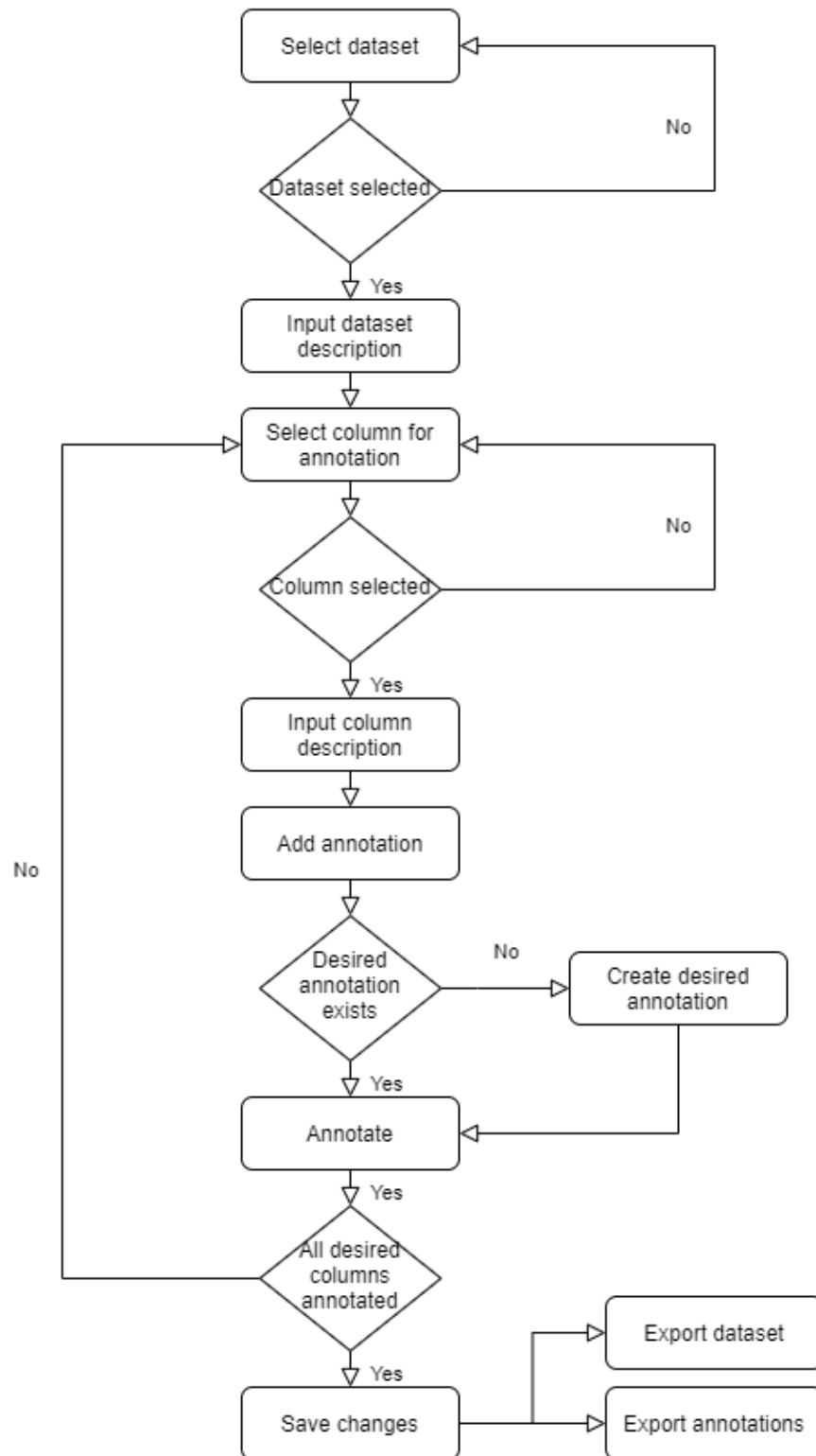


Figure 4.4: Annotate dataset diagram flowchart

The flowchart commences with the “Select Dataset“ stage, where the user is prompted to choose a dataset for annotation. If the user fails to select a dataset, the flowchart directs them back to the “Select Dataset“ stage, allowing them to make a dataset selection.

Once a dataset is chosen, the flowchart progresses to the “Input dataset description“ step. This step is optional and involves adding a description for the dataset.

Once a dataset is chosen, the flowchart progresses to the “Select Column for annotation“ step. In the event that the user does not select a column within the dataset, the flowchart redirects them to the “Select a Column“ stage, enabling them to choose a column.

Upon selecting a column, the flowchart proceeds to the “Input column description“ stage. Here, the user can add description for the column to better describe it.

Upon selecting a column, the flowchart proceeds to the “Add annotation“ stage. Here, the user starts the annotation process. If the desired annotation already exists for the selected column, the flowchart transitions into the “Annotate“ step, allowing the user to proceed with annotation.

If the desired annotation doesn’t exist, the flowchart moves to the “Create desired Annotation“ stage, where the user defines the annotation. Then, it transitions to the “Annotate“ step for annotation.

If user did not annotate all desired column he is moved back into “Select column for annotation“. If all the descriptions and annotation are added, user then presses “Save changes“ which saves all the data into the database. Here user can export the csv dataset or export annotations which he has annotated the dataset with.

4.5.4 Google cloud infrastructure diagram

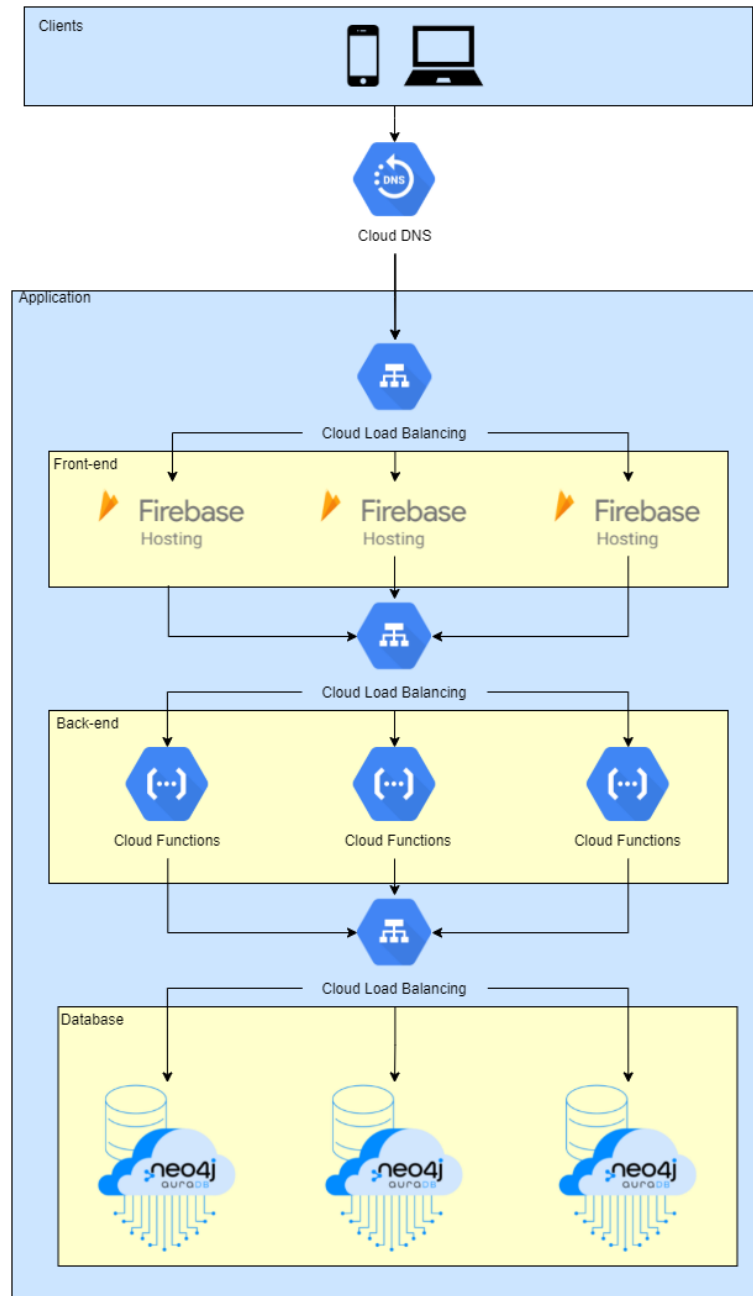


Figure 4.5: Google cloud infrastructure diagram

The diagram in Figure 4.5 can be effectively conceptualized as a hierarchical depiction of the Google Cloud Platform (GCP) architecture, illustrating the pathway from a client endpoint to a Neo4j Aura database using multiple distributed services.

The top of the hierarchy begins with the client-side, from which a connection is made to the GCP infrastructure via the Cloud DNS service. This represents the domain name system, a fundamental internet service that transforms human-readable hostnames into IP addresses, ensuring the client can communicate effectively with the GCP resources.

Once the DNS resolution is performed, the client is routed through the Cloud Load Balancer. This load balancer is designed to ensure the equitable distribution of network traffic across multiple compute instances, preventing any single instance from becoming a bottleneck and consequently improving overall system performance.

Following the first load balancing layer, the system architecture branches into three identical paths, each leading to an instance of the Firebase hosting. This service acts as the host for the front-end Vue.js applications, providing automatic scaling, built-in security, and a developer-friendly environment for managing and deploying web applications.

Subsequent to this, network traffic from each front-end instance is routed to another layer of Cloud Load Balancing. Similar to the first, this load balancer ensures that traffic is distributed evenly among the subsequent tier of cloud functions.

The subsequent tier is a trio of Cloud Functions, Google's serverless execution environment. These functions are an event-driven computing solution that allows developers to execute their code without the need to manage or provision servers,

making it an optimal solution for microservices architecture.

The Cloud Functions connect to a Cloud Load Balancer, distributing traffic to the terminal tier, which includes three instances of Neo4j Aura. Neo4j Aura is a fully-managed, always-on graph database with horizontal scalability and high availability.

4.5.5 Google cloud CI/CD diagram

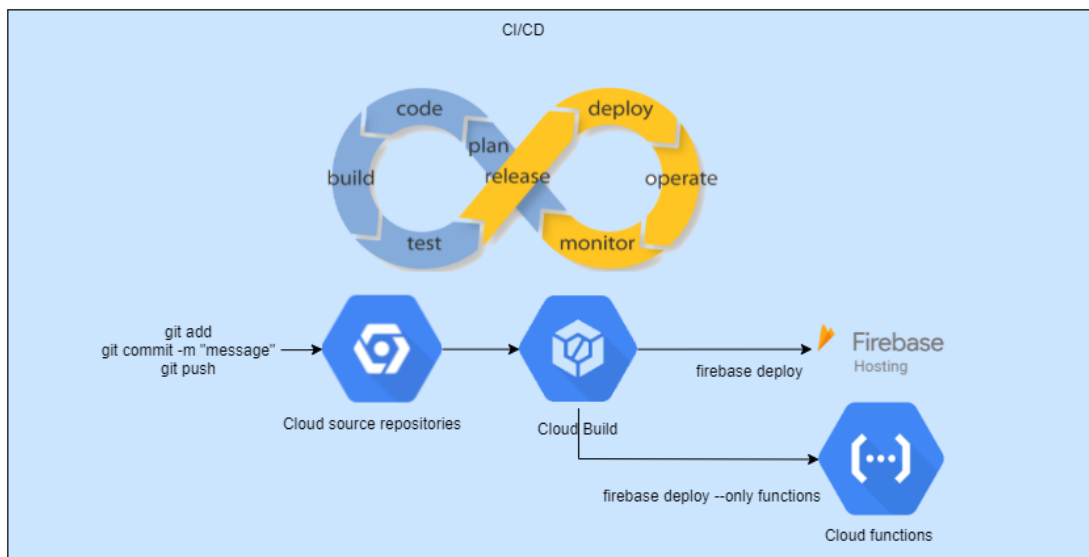


Figure 4.6: Google cloud CI/CD diagram

The Continuous Integration and Continuous Deployment (CI/CD) workflow for the application unfolds as follows:

Initially, the developer executes modifications to the application's source code on their local development environment. Upon completion of the modifications, the developer stages the changes using the Git command `git add`, followed by encapsulating the modifications into a commit via `git commit`. Subsequently, these

commits are pushed to a remote repository hosted on Google Cloud Source Repositories using the `git push` command.

Google Cloud Source Repositories serves as the version-controlled storage mechanism for the application's source code, tracking all changes. It is intricately linked to Google Cloud Build such that any changes pushed to the repository initiate an automatic trigger in Google Cloud Build.

In the CI/CD pipeline, Google Cloud Build manages the end-to-end deployment of both the Vue.js frontend application and Cloud Functions. It orchestrates the build, testing, and deployment processes, ensuring a streamlined workflow for the entire application stack.


4.6 User interface

In this segment of my master's thesis, I will use the design tool Figma to create detailed, high-fidelity wireframes for my proposed application. The aim is to craft a visual guide that represents the layout and features of the application, with a focus on usability and aesthetics.

It's important to note that not all wireframes developed during this process will be included in this section. Instead, I'll highlight the most critical ones that offer significant insights into the design and functionality of the application.

However, the rest of the screens, though not directly discussed here, remain essential for a comprehensive understanding of the overall design. Therefore, I've included these additional wireframes in the project folder of my thesis, within the Figma project file.

4.6.1 Find a dataset wireframe

AnnotateDataset.info 

[Find a dataset](#) [List annotations](#) [Log in / my account](#)

Find your dataset

Dataset name	Author name	Publish date	Size
Global Temperature Trends	Dr. John Smith	2023-05-31	Medium
COVID-19 Worldwide Cases	Prof. Emma Johnson	2023-03-20	Big
European Bird Migration Patterns	Dr. Sofia Martinez	2023-04-10	Small

[Previous](#) [1](#) [2](#) [3](#) [4](#) [5](#) [Next](#)

Figure 4.7: Find a dataset figma wireframe

In (see 4.7), we have a wireframe of a webpage called “Find Your Dataset“. This page has search boxes where we can type in a dataset name or author name to look for specific data. We can also search using the date the dataset was published or how big it is.

When we hit the search button, a table appears that lists all the datasets that match what we searched for. The table is split into multiple pages to make it easier to read. If we want to see more about a dataset, we just click on its row in the table, and that dataset will open up.

Also on this page, there’s a button labeled “Add New Dataset“. If we press this button, we’re asked to choose a dataset to add to the system.

4.6.2 Selected dataset wireframe

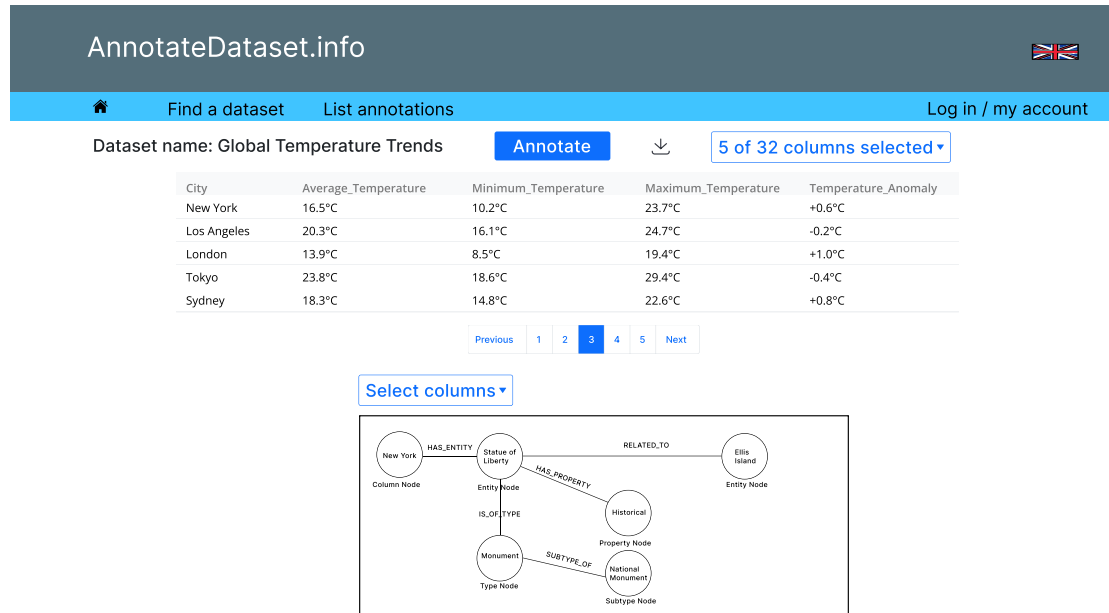


Figure 4.8: Selected dataset figma wireframe

In (see 4.8), we have a sketch or wireframe of a webpage called “Selected Dataset”. This page shows the name of the dataset we’ve chosen. We can also download this dataset in a format called CSV by clicking on a download icon.

An interesting feature of this page is that we can choose what information or columns we want to see. We can select all of them, a few, or even none. The dataset is shown in a table that’s split into multiple pages to make it easier to look through.

The page also has a tool that lets us pick a column and see a graph representation of how this column is annotated

Another cool thing about this page is the “Annotate” button. When we click it, a popup appears asking us to pick a column to mark up. We can then use existing

marks or create new ones to note down important details.

4.6.3 Find annotation wireframe

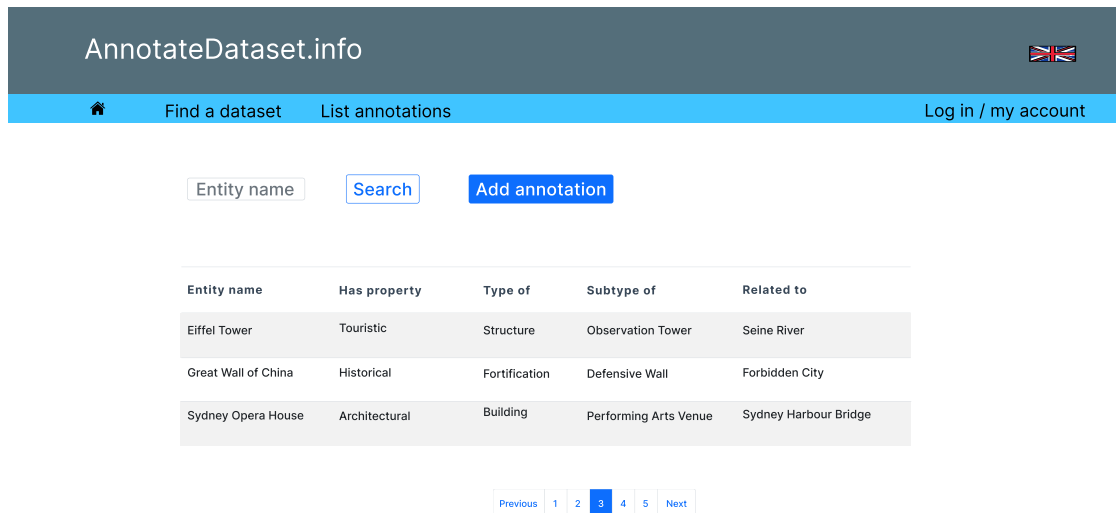


Figure 4.9: Find annotation figma wireframe

In (see 4.9), an illustrative wireframe representation of the page, “Find Annotation” is displayed. This page allows for searching a specific annotation, and as a result, provides a comprehensive list of annotations accompanied by pertinent information.

Each column in the displayed table can be sorted, thus enabling an enhanced data organization and easier user navigation. Furthermore, clicking on any row within this table allows the system to open a detailed page dedicated to the chosen annotation.

An added feature of this interface is the provision to incorporate a new annotation. This is made possible through an interactive component, which presumably is a

button, that when activated prompts the user to add a new annotation to the system.

4.6.4 Selected annotation wireframe

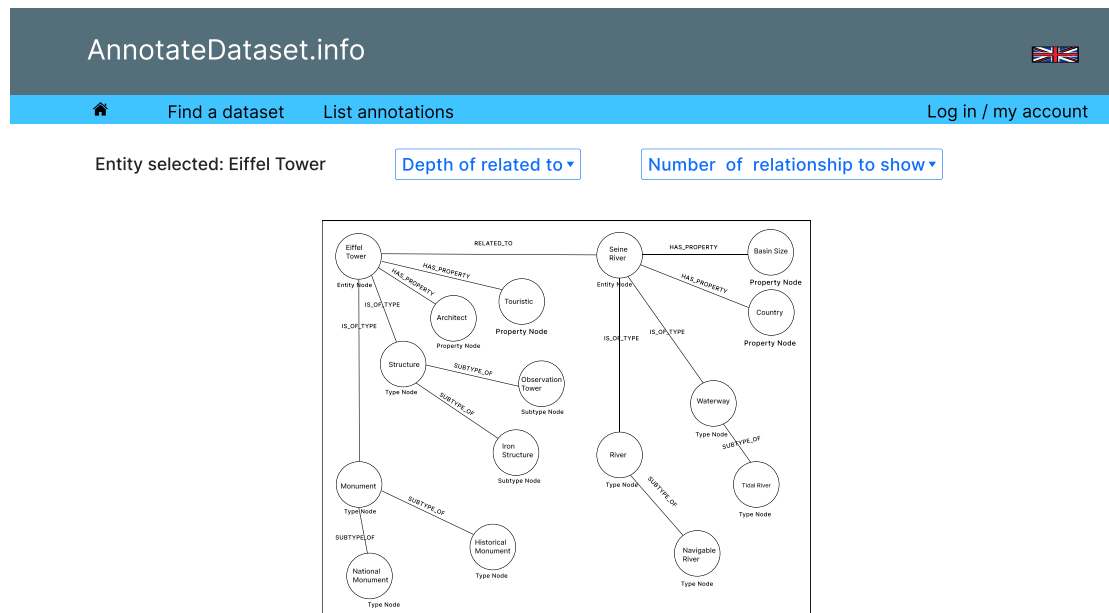


Figure 4.10: Selected annotation figma wireframe

In (see 4.10), illustrates a wireframe of the “Selected Annotation“ page. This graphical representation conveys vital aspects of the page, including the name of the entity associated with the selected annotation.

An essential component of this page is the 'depth of related-to' feature. This interactive mechanism allows users to adjust the level of relationships they wish to visualize, extending from the selected entity.

The 'number of relationships to show' is another user-controlled feature that determines the quantity of relationships the system presents in its output. This degree

of customization caters to diverse user requirements and enhances interpretability of the data.

Corresponding to these selected attributes, the page generates a graphical representation, displayed prominently for intuitive and insightful comprehension. This visualization aids in understanding the intricate relationships and the overall structure surrounding the selected annotation.

4.6.5 Add annotation wireframe

AnnotateDataset.info

Find a dataset List annotations Log in / my account

Annotate

Name: Eiffel Tower

Description: Tower in France

Unit: Structure

Related to

Entity name
Eiffel Tower
Seine River

Is of type

Type	Subtype
Monument	Historical Monument
Structure	Iron Structure

Has property

Property name
Architect
Touristic

Add annotation

Figure 4.11: Add annotation figma wireframe

In (see 4.11), we examine a wireframe of the webpage “Add Annotation,” which offers a graphical illustration of the annotation-adding process. Users have the capacity to input comprehensive details about the annotation, which includes attributes such as the name, description, and unit of the annotation.

One of the key components of this interface is the ability to establish relationships with the annotation. The user can denote the entities related to the annotation by interacting with the plus icon. This action results in the relationship's inclusion in the 'related-to' table displayed on the page. Users are granted the ability to view, add, or remove relationships from this table, thus providing a dynamic and adaptable user interface.

The 'has property' function operates similarly to the 'related-to' feature, enabling users to manage the properties associated with the annotation in a similar tabular format.

A slight distinction is found in the 'is of type' function, which additionally requires the input of a subtype field. Despite this minor difference, it largely operates akin to the aforementioned features, demonstrating a consistent user interface.

5 Implementation

In this section, I will discuss the practical implementation of the coding aspect of my diploma thesis and show how the current application looks like.

5.1 App implementation steps

At the start I initiated creation of a Git repository, resulting in the creation of the .git folder and a .gitignore file to exclude specific files from publication. Git, being a distributed version control system, is a pivotal tool designed to enhance the efficiency and manageability of projects, ranging from small-scale to extensive endeavors.

The repository serves as a safeguard during development, offering the capability to revert to a functional version in case of issues, thus saving valuable time on debugging. Within the Git folder, I organized a text file for tracking TODOs and personal notes, facilitating project management.

For an optimized file structure, I divided the project into frontend and backend directories. In the frontend, adhering to the specifications outlined in the design/technologies section of my diploma thesis, I instantiated a Vue.js 3 app with TypeScript, integrating Vue Router, Vite as the build tool, and employing the Composition API. For state management, I utilized the Pinia library, while Tailwind CSS was incorporated for styling. Additionally, I adopted the Vue Apollo Client for seamless communication with the backend GraphQL API.

In the backend, I initialized Firebase Functions and established a server folder.

This server will host my application, which will ultimately be deployed to Firebase Cloud Functions, akin to Google Cloud Functions. This structuring ensures a coherent organization of the project, aligning with best practices and optimizing the development workflow.

Within the server folder, I initiated the technologies specified in the design/technologies section. Specifically, I employed Nest.js, a JavaScript framework, which utilizes the Cypher query builder for seamless communication with the Neo4j database. Additionally, I created an Apollo Server to facilitate communication between the Vue Apollo Client and the backend.

While I previously mentioned the use of the Cypher query builder for Neo4j communication, I hadn't delved into the database setup. The database was created using Neo4j Desktop, configuring it through the UI. To enable successful backend connection, I specified the credentials in the .env file.

In the backend, a GraphQL schema was established, detailing all nodes, relations, queries, and mutations that the application would implement. Subsequently, a command was created to generate TypeScript definitions from the .graphql file, enhancing ease of use within the application.

During this phase of my coding, I established means for communication between the frontend and backend using Apollo Client and using standard API. Following discussions with my diploma thesis supervisor, several enhancements were suggested. Firstly, the Apollo Server was removed to align with the serverless nature of the application. Additionally, the registration and login processes were revamped to leverage Firebase authentication, a more robust and standardized solution.

Given the serverless architecture, I integrated Firebase Cloud Functions. Emulators were employed during development to simulate Google Cloud Functions locally,

streamlining testing without the need for continuous cloud deployment.

Following my supervisor's guidance, I opted for free templates from Tailwind UI, with PrimeVue as a backup option for more specific requirements, as it aligns with the project's architecture and is complimentary to Tailwind. Xicons was chosen as the icon library, and for notifications, the kyvg/vue3-notification library was selected. It's essential to note that these additions, Tailwind UI, PrimeVue, Xicons, and the notification library, were not initially specified in the design/technologies section but have been incorporated based on current project needs.

In the section on backend/adaptor pattern, I previously mentioned the adoption of the adaptor design pattern. However, due to the division of the application into frontend and backend, this pattern is inherently in use, with the backend acting as the adaptor. It facilitates requests from the frontend, allowing for potential changes in backend logic while maintaining consistent routes.

Looking ahead, I plan to implement the adaptor pattern within the backend as well. I've established interfaces and adaptors specifically for Neo4j. This modular approach ensures that if there's a decision to transition from, say, Neo4j to PostgreSQL in future developments, minimal code adjustments will be required. In this phase of my coding i streamlined the application by removing unused components and in the next phase I will incorporate Firebase login.

I successfully developed the entire login process utilizing Firebase, including the establishment of a Firestore database for storing user roles—currently defined as 'user' and 'admin.' The login logic incorporates Google Sign-In for enhanced user authentication.

To streamline administrative tasks, an admin page was introduced. This facilitates role adjustments for users, eliminating the need for direct database access by

administrators. In the optimization phase, I conducted a code cleanup to remove unused elements, enhancing code clarity.

Moving forward, I am set to implement functionalities for listing all available datasets and adding new datasets. The admin page is crafted using Flowbite and Prime, both open-source libraries offering a collection of interactive UI components.

Today's focus was on crafting the "Find Dataset" page, leveraging PrimeVue for the UI, notably employing the DataTable component for a feature-rich table presentation. A FileUpload component, also sourced from PrimeVue, posed a slight challenge as its default behavior conflicted with the desired custom processing logic. To address this, I reconfigured the component to invoke my upload function on the select event, circumventing the default upload behavior.

Encountering a persistent issue with clearing the file input after a successful upload, I introduced a solution by adding a key to the FileUpload component. Upon successful upload, updating the key triggered a forced refresh, effectively removing the file.

The upload process begins with extracting essential information from the selected file, such as name, size, uploader, and timestamp. Utilizing the Papaparse library for parsing, the information is then sent to the Firestore database collection "datasets." Notably, the Firestore system autonomously handles the assignment of unique identifiers (UIDs).

The implemented table boasts comprehensive functionality, featuring sortable columns and pagination. The search capability spans across various criteria, allowing users to explore datasets based on keywords, text, date, and more.

For testing purposes, I incorporated three datasets downloaded from Kaggle, accessible through the following links:

- Electric Vehicle Dataset [62]
- Small COVID-19 Dataset [63]
- Used Fiat 500 Dataset [64]

The subsequent step involved the creation of the “Selected Dataset” page, where a tabular representation of dataset fields and values is dynamically generated. Notably, the page header encountered an issue, which was promptly addressed by incorporating the Tailwind UI header component. I adapted the component to align with the specific requirements, addressing issues with button styling and resolving other encountered bugs.

Additionally, the responsiveness of the select button in the admin page posed a challenge. This was mitigated by implementing a solution using min-width adjustments in the CSS.

A complication emerged on the “Find Dataset” page regarding the responsiveness of the table. The issue stemmed from a conflict between the display flex property of the surrounding div and the styling of the datatable component. This was rectified by introducing the “mx-auto” class for centering, ensuring proper display alignment.

The “Selected Dataset” page offers users a comprehensive view of the selected dataset, presenting their name and all dataset values in a paginated table. Users have the flexibility to export a CSV file from the currently viewed table on the page or opt for a CSV export of the entire dataset. The inclusion of a multi-select feature allows users to choose which columns to display. Handling edge cases where no columns are selected or when the dataset is nonexistent has been implemented for seamless user experience.

In the process of selecting a visualization tool for my project, I explored various

options listed on the Neo4j blog [65]. I choose neovis. Given the limited number of data sources, I encountered challenges during the visualization phase and had to resort to searching GitHub issues for solutions. Fortunately, I found the necessary information, enabling me to display all values from Neo4j and visualize the path of my annotations with different colors for enhanced visibility. To improve clarity, I also implemented arrow indicators.

Despite encountering difficulties with the outdated TypeScript implementation of Neovis, I opted for an alternative npm package that lacks TypeScript compatibility. However, this substitution posed no issues in practice.

The current functionality allows users to click “Show,” triggering a dialog where they can add details such as name, shortcut, and a description for the annotation. A multiselect field facilitates the specification of relationships between annotations. Users can visualize the graph representation of the annotation relationships, providing a comprehensive view from start to finish. Hovering over the graph representation reveals detailed information.

Below this, a table displays all available annotations, equipped with built-in searching and sorting capabilities for user convenience. Upon selecting an annotation, users are presented with a visualization using the Neovis.js library, illustrating all connected annotations with a distinct color for each path. The entire graph path of the annotation is showcased. Upon clicking “Add Annotation,” the annotation is successfully added to the system.

In the subsequent development phase, I focused on enhancing the functionality of the “Selected Dataset“ page. Users can now select a column from the multiselect menu, specifying the column they wish to annotate. The addition of a “Column Description“ field allows users to provide additional information about the column. Upon clicking “Change Description,” a small dialog prompts the user to

enter the new column description. Clicking “Save Description” updates the displayed value, although the database remains unchanged until the user clicks “Save Changes.”

Furthermore, the option to select an annotation for annotating a column has been implemented. Users can choose an annotation from the multiselect menu, displaying the entire graph structure as text. Hovering over the graph structure reveals the complete annotation. Upon selecting an annotation and clicking “Save Changes,” several actions may occur. If it is the first annotation for this dataset, a dataset node is created with `datasetName`, `firebaseDatasetID`, and `firebaseUserID`. Additionally, a column node is created with `columnDescription` and `columnName`. The relationship “HAS_COLUMN” is established from the dataset to the column. If the user selected an annotation, the “ANNOTATED_WITH” relationship is formed from the column to the annotation.

To ensure data integrity and prevent duplication, checks are implemented. If the dataset or column nodes already exist, they are updated instead of being recreated. Similarly, existing relationships are not recreated. This strategy helps maintain a clean and efficient database structure.

In terms of the coding approach, I initially prioritized ease of development, deferring the comprehensive implementation of TypeScript. Subsequently, after achieving the desired functionalities, I incorporated interfaces and types to enhance code readability and maintainability.

I encountered an issue with uploading datasets larger than 1 MB to Firestore due to formatting constraints. To address this, I eliminated the use of Papaparse, opting for raw CSV values that occupy less space as they lack formatting elements such as spaces and indentations. However, the problem persists, and addressing it in future work might involve implementing sharding.

Additionally, I focused on seeder implementation to facilitate testing. Two seeders were created, one for Firestore and the other for Neo4j. If the environment variable `SEED_DATABASE` is set to true, the seeder populates the database. The dataset seeder adds nodes for datasets, columns, column descriptions, annotations, and related relationships. The seeded data comprises 40 nodes (1 Dataset, 29 Columns, 10 Annotations) and 47 relationships (29 `HAS_COLUMN`, 10 `ANNOTATED_WITH`, 8 `RELATED_TO`).

During consultations with my supervisor, a shift in the project's focus was recommended, transitioning from marketing datasets to business intelligence datasets. Consequently, I revamped the entire database structure and associated operations.

The first modification to the front end involved relocating the navigation bar from the top to the left side as a sidebar, aligning with my supervisor's preference. This adjustment was made using the PrimeVue sidebar property and incorporating the functionality from the existing header.

Today's focus was on enhancing user experience and code organization. I implemented a sticky header at the top, ensuring that the "Save Changes" button and annotations float with the user as they scroll down. Additionally, I created cards for each column, featuring options to add a column description and annotation. Neovis visualizations were incorporated into each card to allow users to view the graphs; however, the line visualization remains unchanged. I utilized components, child-parent relationships, and emits to manage data. Presently, the ability to add annotations is not functioning, but this will be addressed in future iterations, along with a broader restructuring of the functionality.

A significant part of today's work involved creating and structuring components. The `SelectedDatasetPage`, which initially exceeded 600+ lines of code, underwent

a restructuring process. Through division and organization, I split the code into multiple child components. This not only improves the code's usability but also enhances readability and maintainability.

Building upon the valuable insights from the articles [66, 67, 68]. I am investing time to delve deeper into the realm of hierarchical data in Neo4j and formulate a robust data model.

Inspired by the rules of location trees outlined in the articles, I've decided to adhere to the following principles:

- All relationships are directed from children to parents, ascending the hierarchy.
- A single relationship type (named `related_to`) is used for all relationships.
- Each node has a sole outgoing relationship to its parent.
- Nodes can have one or multiple incoming relationships from their children.

Implementing these rules, I've structured my nodes to have only one relationship named `related_to`. This design ensures a hierarchical tree structure where each node has only one relationship going to its parent, while multiple nodes can connect to the same parent, indicating sibling relationships.

To put this knowledge into practice, my supervisor provided a dataset. I invested a significant amount of time generating annotation data and defining their structures. This data generation process involved collaboration with ChatGPT and my personal input. After thorough verification to ensure accuracy, the generated data was seamlessly incorporated into the seed Neo4j database using a dedicated seeder. This strategic approach ensures a robust foundation for working with hierarchical data in Neo4j and aligns with the best practices gleaned from the referenced

articles.

When attempting to view annotations after adding one, I encountered an issue where they weren't being redrawn. This problem stemmed from the asynchronous nature of the `getAnnotations` function. I addressed this by adjusting the emit placement, moving it inside the `.then` block. Additionally, I resolved the issue by using a key on the `annotationsResponse` div.

Currently, I'm in the process of refining the `annotateColumn` function to accommodate annotating multiple columns simultaneously, as per my supervisor's guidance. I'm wrapping up these modifications today and will continue working on the completion tomorrow.

As part of these modifications, I introduced a "Delete Annotation" button, designed to remove the selected annotation along with all its relationships. Additionally, I implemented a button for creating relationships. Notably, this function restricts connections to nodes without any existing relationships, ensuring that a child is only connected to a parent without any prior connections.

Upon clicking "Add Annotation" and inspecting the available annotations for selection, I noticed that the newly added one wasn't visible. I identified an issue with asynchronicity and resolved it by relocating the emit statement inside the `.then` block. Additionally, I employed a key on the `annotationsResponse` div to address this problem.

Presently, I'm in the process of enhancing the `annotateColumn` functionality to handle multiple annotations for annotation. I'm actively engaged in restructuring this feature. Furthermore, I revised the code to support annotating multiple columns simultaneously, following my supervisor's instructions.

I introduced a "Delete Annotation" button, designed to remove the selected an-

notation along with all its relationships. Additionally, I implemented a button for creating relationships. Notably, this function restricts connections to nodes without any existing relationships, ensuring that a child is only connected to a parent without any prior connections. I also worked on making the page more responsive for smaller devices.

Following a consultation with my supervisor, it came to my attention that there were issues with the annotation data for the dataset. The error arose from annotating columns based on their specific values, whereas the correct approach involves defining annotations based on underlying concepts and generalizing these concepts. To rectify this, I revisited the concepts of knowledge bases and semantic annotation, delving into various papers and videos for a more comprehensive understanding.

In light of this newfound knowledge, I am now in the process of remaking the datasets and their annotations. This time around, I am aiming to adhere to the correct methodology, ensuring that annotations are rooted in conceptual understanding rather than being tied to specific values. The datasets I am currently annotating include:

- mkt_synthetic [69]
- KAG_conversion_data [70]
- Ecommerce Customers [71]

I made several updates to the application to ensure its functionality and security. Initially, I realized that I hadn't implemented authentication, so I am currently working on rectifying this by adding a header with the Firebase token. On the backend, I introduced a guard to verify if a user is logged in before allowing access to routes and routes that need admin access check if user is admin.

I modified the “getexport” route, which initially returned an array with one object. Now, it only returns the object. Additionally, I addressed an issue in the “deleteAnnotation” endpoint where the response was the same whether a node was deleted or not. I adjusted it to check explicitly if something was deleted before responding.

To streamline testing, I created a Postman collection with all available endpoints. During testing, I noticed that even when required information was missing from the request body, there were no errors. Recognizing the need for a validation pipeline, I implemented one to ensure proper validation.

Regarding the frontend, I encountered an issue with the “datasetCardComponent” not refreshing after saving changes. I resolved this by incorporating a “ref” key that updates whenever the user presses “save changes” so it force refreshes the card components.

I addressed the issue in “getColumnDescription” by switching the route from POST to GET to adhere to REST API principles. I eliminated an unused DTO and resolved a problem where dialogs were inconsistently displayed despite being the same on each page. The fix involved removing the “scoped” property from the CSS and making adjustments to the dialog styles.

I’ve rectified the connection of relationships, ensuring users cannot link an annotation to itself. Furthermore, I’ve tackled responsiveness issues and fixed problems with the updating of values. Lastly, I’ve prepared the code for submission.

5.2 Api endpoints

In this section, I will showcase and explain all the endpoints employed in my application. You can find all available endpoints also in file `diploma-thesis.postman_collection.json`

inside my diploma thesis folder but I will also list them here.

5.2.1 NestJS API endpoints

Within the /api cloud functions, the NestJS application encompasses several routes, each dedicated to distinct functionalities. Let's delve into the specifics of each route:

/annotation

- Post /createAnnotation - Creates a new annotation.
- Get /getAnnotations - Retrieves all annotations.
- Post /annotateColumn - Annotates a column with an annotation.
- Get /getExport/:id - Provides an export of a dataset and its corresponding annotations.
- Get /getColumnAnnotations/?datasetId=<value>&columnName=<value> - Gets the annotations from the specified datasets column.
- Delete /deleteAnnotation/:id - Deletes an annotation and its relations.
- Post /connectAnnotations - Creates a relation of related_to between two annotations.

/column

- Get /getColumnDescription/?datasetId=<value>&columnName=<value> - Retrieves the corresponding description of a column.
- Post /updateOrCreate - Updates or creates a description for a column.

/dataset

- Post `/createDataset` - Creates a new dataset.
- Get `/getDatasetDescription/:datasetId` - Retrieves the description of a dataset.
- Post `/updateDatasetDescription` - Updates the description of a dataset.

5.2.2 Firestore cloud functions endpoints

Additionally, Firestore cloud functions contribute to the application's functionality with the following endpoints:

- `/AddUserRole` - Adds a user role to a registered user.
- `/SetUserRole` - Sets the role of a user.

5.3 Screens

In this section, I will present the current appearance of my application. While my diploma thesis in section 4.6, User Interface, featured wireframes illustrating the anticipated look of the application, these served as templates. The final application deviates from these initial designs due to modifications made to the proposed views. Consequently, I will now showcase the updated and actual views of the application.

5.3.1 Find a dataset page

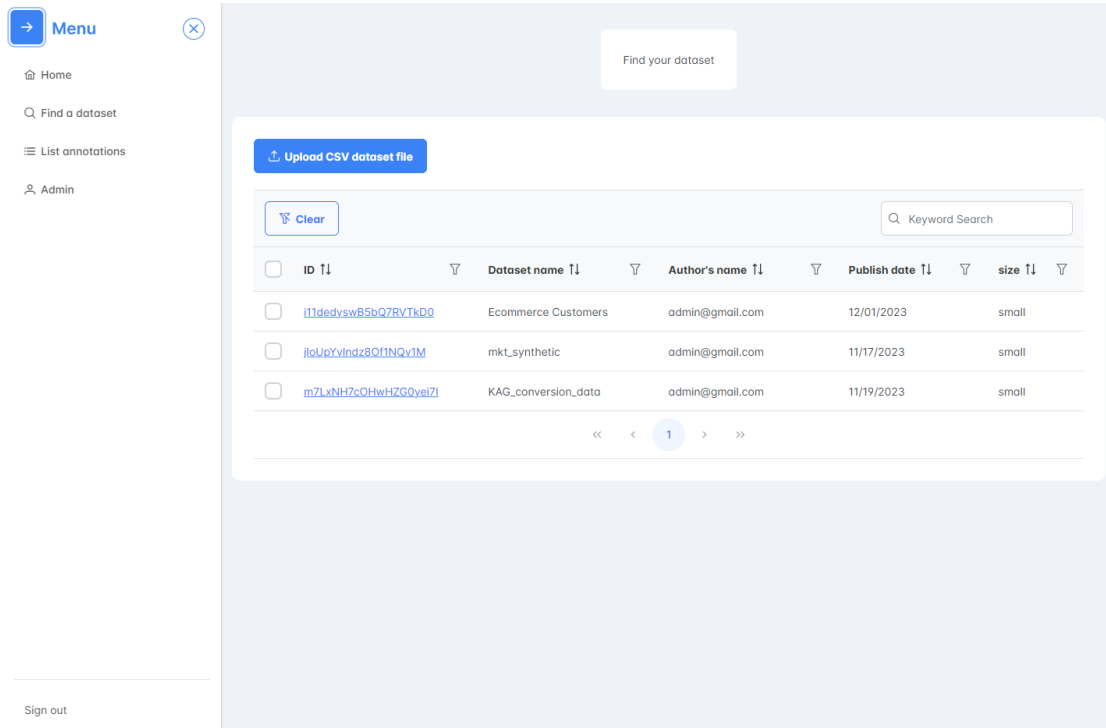


Figure 5.1: Find your dataset page

This webpage, titled “Find a Dataset,” serves as a platform for users to seamlessly add datasets to database. Upon selecting the “Upload CSV Dataset“ option, users are prompted to choose a dataset in .csv format. The page features a table displaying all available datasets within the application, equipped with a built-in sorting mechanism for each column. Additionally, users can employ a text search functionality to easily locate specific datasets. The table incorporates pagination for user-friendly navigation through multiple datasets. Furthermore, the page includes a convenient “Clear“ button that resets all applied filters, ensuring a streamlined user experience.

On the left side, a collapsible sidebar enhances navigation, featuring an “x“ icon

for closure and an arrow button for expansion. This sidebar not only aids in page navigation but also displays the user's login status. If the user is not logged in, they are provided with the option to log in. So this sidebar functions for navigating on the page and redirection.

5.3.2 Selected dataset page

Save changes **Add annotation**

Dataset name
KAG_conversion_data

Dataset description
KAG_conversion_data dataset description

Change description

ad_id xyz_campaign_id fb_campaign_id 3 of 11 columns selected **Export dataset** **Export annotations**

Clear

ad_id ↑↓	xyz_campaign_id ↑↓	fb_campaign_id ↑↓
708746	916	103916
708749	916	103917
708771	916	103920
708815	916	103928
708818	916	103928

« < 1 2 3 4 5 > »

Annotation part

Figure 5.2: Selected dataset page 1

On the “Selected Dataset“ page, users encounter comprehensive details about their chosen dataset, including its name and description. A table exhibits all the columns within the CSV dataset, allowing users the flexibility to choose specific columns for display. Notably, users can initiate a description change by clicking the “Change Description“ button, triggering a popup where they can input a new description for the dataset.

The page is equipped with two export features: the “Export Dataset“ button facilitates the export of the dataset in CSV format, while the “Export Annotations“ button exports the dataset’s annotations in JSON format. These functionalities significantly empower users to interact with and extract valuable insights from the selected dataset.

Additionally, a sticky header features two buttons: “Save All,“ which preserves all the provided information, and “Add Annotation,“ which opens a popup for users to add annotations. This latter feature will be elaborated upon in subsequent sections of this thesis.

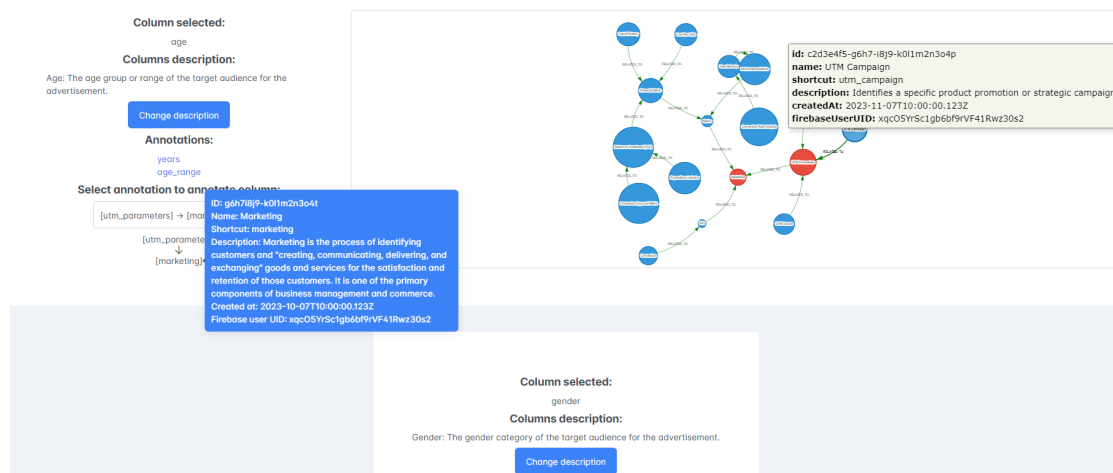


Figure 5.3: Selected dataset page 2

This section guides the user through the annotation process for each column. Each column is represented by a card that the user can annotate, and changes are only finalized upon clicking the “Save All“ button. On the left side, the user can provide a description for the column. Then user can see with which annotations has this column already been annotated with. On hover he sees all information about this annotation and after clicking on it he is redirected to page list annotations that I will talk about later. Additionally, the user has the option to select an annotation, accompanied by a visual preview displayed through arrows and text on the left side. Each text item features tooltips offering comprehensive information.

On the right side, a NeoVis graph visualization depicts the relationships. The red circle represents the current annotation, showcasing its hierarchical representation from the bottom to the top. Blue circles denote relationships that are not part of the current annotation hierarchy but are part of a broader hierarchy. The visualization selectively displays relevant relationships, offering the user a contextual understanding of the annotation’s broader scope.

5.3.3 List annotations page

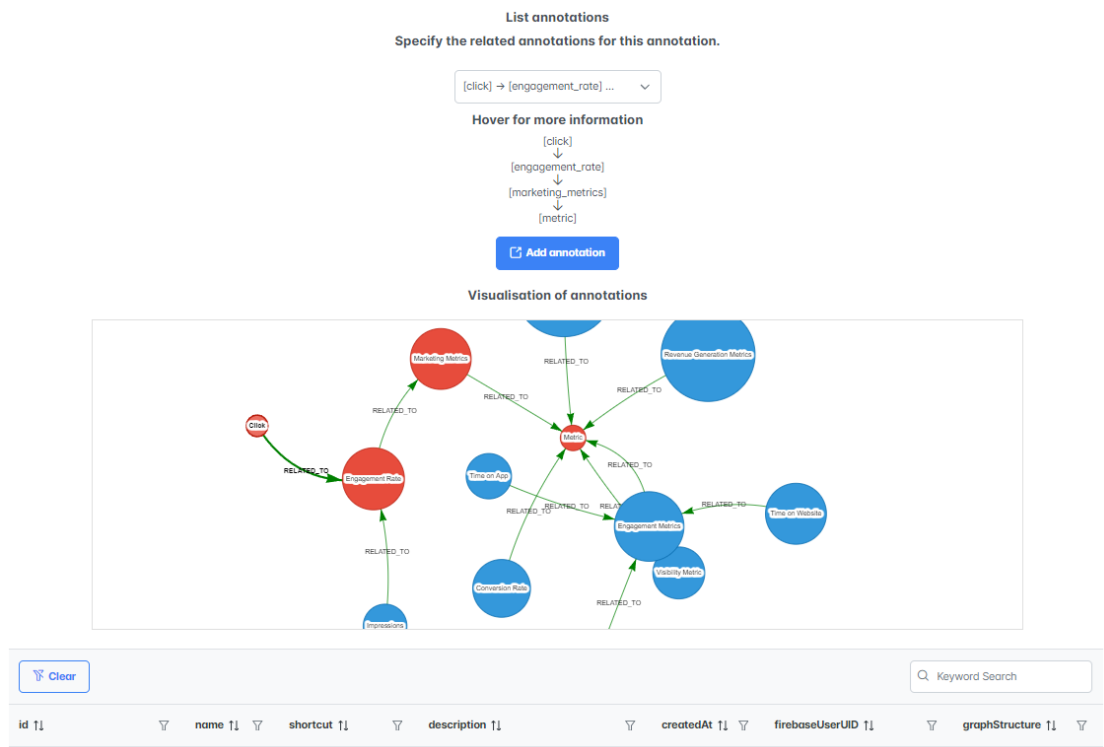


Figure 5.4: List annotations page

This page seamlessly combines elements from both the “Find an Annotation” and “Selected Annotation” sections outlined in my diploma thesis design. Users enjoy comprehensive access to all annotations in our knowledge base, thanks to a user-friendly table at the bottom that supports sorting and searching based on strings. Furthermore, users have the option to select a specific annotation, prompting the presentation of a dedicated NeoVis graph visualization. Additionally, there is a “Add Annotation” button that, when clicked, opens a popup for users to add annotations. This feature will be explored in greater detail in subsequent sections of this thesis.

In the visualization, the red circle signifies the selected annotation, revealing its hierarchical representation from the bottom to the top. Blue circles highlight relationships beyond the immediate annotation hierarchy, contributing to a more extensive structure. This strategic presentation of relationships provides users with a contextual understanding of the annotation's broader scope. Concurrently, users are presented with a text interpretation, akin to the format observed on the "Selected Dataset" page. This comprehensive approach enhances user understanding of annotations and their interconnected relationships.

There are two scenarios for this page: In one variant, the user navigated here through the left sidebar, and as a result, no annotation is pre-selected. In the other variant, the user arrived here via a redirect from the "Selected Dataset" page, where a specific annotation is already chosen.

5.3.4 Add annotation page

Add annotation

Name

Enter name of annotation

Shortcut

Enter the shortcut of an annotation

Description

Enter the description of an annotation

Specify the related annotations for this annotation (parent).

[marketing]

▼

🗑️ Delete annotation

Hover for more information

[marketing]

Add annotation

Specify the related annotations for this annotation (child).

[demographics]

▼

↔️ Connect annotation

Hover for more information

Figure 5.5: Add annotation page

On this page, users can add a new annotation by providing its name, shortcut, and description. Additionally, users have the option to establish connections with related annotations. The page includes a table displaying existing annotations, allowing users to search for the correct annotation to connect with the newly added one. Users can also delete annotations or establish connections between two annotations. Upon selecting an annotation from the list, a NeoVis graph representation, similar to other instances, is displayed along with a text representation. This dual visualization approach enhances user understanding of the relationships between annotations.

5.3.5 Admin page

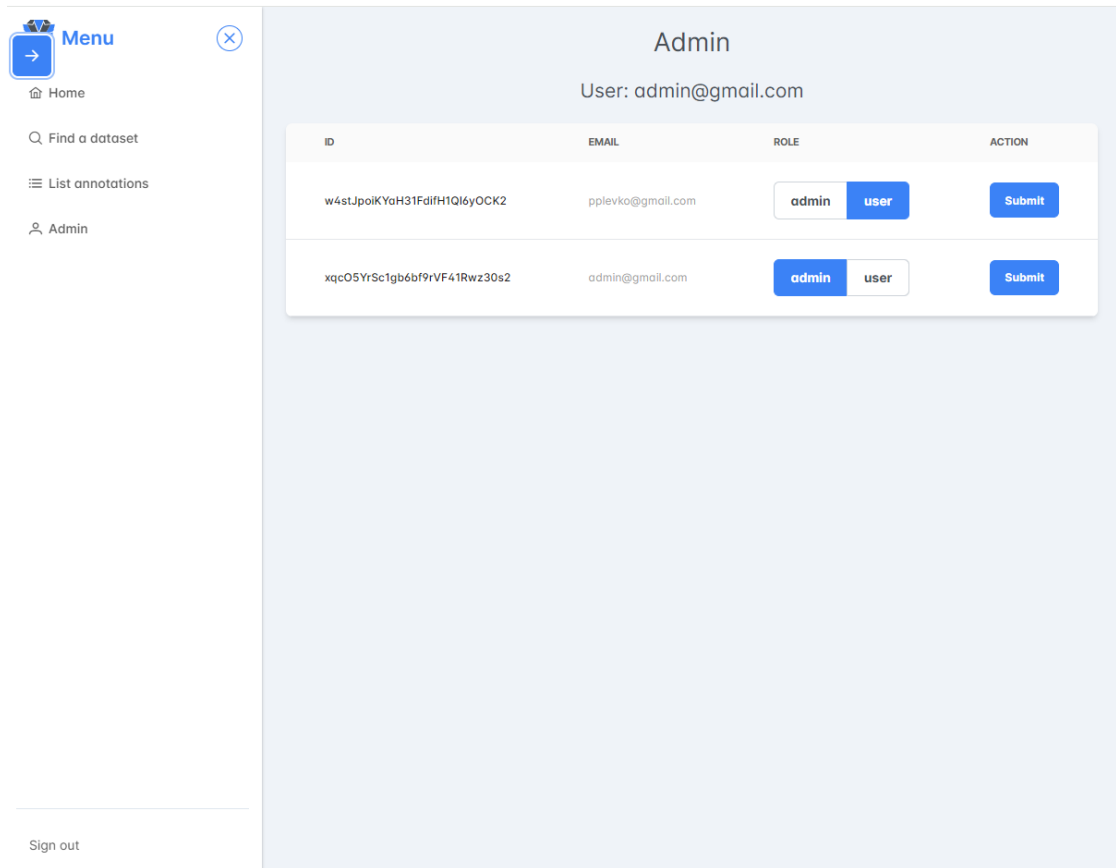


Figure 5.6: Admin page

This page allows users to modify the roles of registered users, toggling between admin and user roles. Once the user submits the changes, the updates are seamlessly reflected in the Firebase Firestore 'roles' collection. Additionally, users have the ability to view a list of all registered users who have utilized this page.

6 Future work

In this section, I will explore forthcoming tasks and potential enhancements that could be implemented to enhance the functionality of my application.

6.1 Deployment

Currently, my application's development cycle takes place on my local machine with Neo4j installed. When working on code, I run the frontend using the command **"npm run dev"** in the terminal. This command is specified in the package.json file under the scripts section, where **"npm run"** means to run from the package.json file, and **"dev"** is the corresponding script. In this case, it runs the **"vite"** command. If I don't want to use **"npm run dev"**, I can directly write vite in the terminal, but since Vite is installed locally in my project, I have to use **"npx vite"**.

To start my app, I open Neo4j Desktop, find my database, click start so now I have my Neo4j database running. Then I start my backend nestjs using **"nest start --watch"**. So now I have database backend and frontend running so I have all the things needed to run my application. Last step is to copy the url where my application resides on localhost from the frontend application, open Google Chrome, paste it, and wait for it to load. This process is quite lengthy, especially for user testing as users need to either have access to my repository, download all files, install Neo4j, and run it, or I need to visit them with my setup.

To simplify this, I aim to deploy my application to the web, where users only

need to find my webpage through Google or by pasting the URL. This deployment involves several components:

1. **Frontend:** Prepare the frontend for production by specifying the correct values in `.env` to connect to cloud functions and the database. I will use Firebase hosting for the Vue application.
2. **Backend:** Host the backend on Firebase Cloud Functions. This step may encounter challenges since I currently develop the app using “**nest start --watch**” because it is easier to develop this way. Deploying it as an app is the same as using “**firebase emulators:start --only functions**”. I need to ensure the statelessness of functions, which may involve fixing potential errors in my codebase.
3. **Database:** Utilize Neo4j Aura as the database.

6.2 Microservices

Presently, my application is a singular entity hosted on Google Cloud Functions, encountering challenges associated with cold start issues. It is a singular entity but there are login and change roles which are independent cloud functions. To address this, I am contemplating the division of this NestJS application into smaller, more modular components, each dedicated to specific functionalities.

The proposed split would be organized based on distinct modules within the application, aligning with their respective responsibilities:

1. **Annotation Module:** This component would exclusively handle tasks related to annotations, encompassing creation, retrieval, and other annotation-related operations.

2. **Column Module:** Responsible for managing the creation of columns, mapping them to datasets, and handling updates to their values.
3. **Dataset Module:** Focused on the creation and updating of datasets, this module would encapsulate all functionalities pertaining to dataset management.
4. **Seeder Module:** This module, intended for seeding the database, could potentially be split further. Given that it may be invoked only once during the initial setup to populate the database, it occupies space but does not require ongoing invocation.

This modular restructuring aims to enhance the efficiency and performance of the application, particularly addressing the challenges posed by cold starts. The division based on functionality ensures a more streamlined and manageable architecture, fostering better scalability and maintainability.

6.3 Annotation suggestions

Considering the existing datasets and annotations within the database, a valuable enhancement would involve the mapping of annotations. This mapping mechanism aims to leverage insights gained from prior datasets, facilitating a more intuitive annotation process.

For instance, if a column named “name” is annotated as “person” in one dataset, this mapping would provide a helpful suggestion. When users proceed to annotate a different dataset, the initial recommendation for a column with the same name, such as “name,” would be “person.”

This approach streamlines the annotation process, harnessing the knowledge gai-

ned from previous datasets to offer informed suggestions for similar column annotations.

6.4 User testing

I intend to conduct testing scenarios where individuals will evaluate my application. This process aims to refine the user interface and address any potential bugs identified during testing. I will develop specific testing scenarios, such as a user creating an account, uploading a dataset, annotating the dataset, and exporting it. I will gather participants and generate graphs to depict key metrics, including whether users required hints, the success or failure of completing the scenario, and the time taken to accomplish each task. This data will provide valuable insights for further improving the application's usability and performance.

6.5 Using a different database with the repository pattern

Presently, my application relies on a Neo4j database. However, envisioning a scenario where a switch to PostgreSQL is deemed necessary, the transition might pose challenges for developers as extensive changes would be required. To address this, the repository pattern proves beneficial. The process unfolds as follows:

1. **Define Repository Interfaces:** Establish repository interfaces for each method in the backend application, ensuring a consistent contract for data access.
2. **Model Creation:** Develop models for each method, encompassing Data Transfer Objects (DTOs) and response structures.
3. **Repository Implementations:** Create repository implementations for each

database type, such as `PostgresRepository` and `Neo4jRepository`, tailoring them to their respective databases.

4. **Service Integration:** Within the service, import and instantiate the relevant repository based on the chosen database. Invoke methods like `“this.userRepository.functionName(prop)”` to maintain database-agnostic service logic.
5. **Dependency Injection Configuration:** Configure dependency injection to dynamically select the appropriate repository, based on a specified database choice. This choice can be managed through configuration values, such as those specified in an `.env` file.

This systematic approach, leveraging the repository pattern, facilitates a smoother transition between databases. Developers can work within a consistent structure, abstracting away the intricacies of specific database interactions. The configurability introduced by dependency injection ensures adaptability and ease of maintenance in the face of evolving database requirements.

6.6 Removing restriction for dataset size

When working with Firestore, it's essential to be aware of the platform's constraints, especially regarding dataset size. Firestore imposes a maximum document size limit of 1 MiB (1,048,576 bytes), making it impossible to upload datasets larger than 1 MB. To overcome this limitation, there are the following solutions:

1. **Data Sharding:** When dataset surpasses the capacity of a single Firestore document, adopt data sharding. This involves breaking down the dataset into smaller, manageable pieces, distributing them across multiple documents. Establish a structured system, employing either a naming convention or an

identifier, to seamlessly link these documents to the same overarching dataset.

2. **Compression:** In cases where sharding isn't practical, and it is needed to store the data as a single document, consider compression techniques. This involves compressing the data before saving it in Firestore using libraries like zlib or gzip. Compression effectively minimizes the overall data size, ensuring compliance with Firestore's document size limits.

7 Evaluation

In this phase of my diploma thesis, I have successfully developed a functional prototype for an application designed to streamline dataset annotation processes. The application empowers users to perform essential operations, such as creating, deleting, updating, connecting, and visualizing annotations. Users can upload datasets for storage, and these datasets are downloadable as CSV files. Moreover, users have the option to augment the dataset with descriptions, providing detailed information about the dataset as a whole and individual columns.

A noteworthy feature is the system's support for exporting these descriptions along with annotations in JSON format. This ensures that comprehensive information about the dataset and its columns is included in the downloaded JSON file, thereby enhancing the application's utility for robust data management.

As said above, during this phase, my primary focus was on crafting a web prototype tailored for dataset storage and annotation. This involved a meticulous remodeling of the database structure and strategic adjustments to the overall architecture. The resultant product is an operational web application accompanied by exemplary datasets and annotations generated through seeders, alongside Postman endpoints for testing purposes. The application leverages a serverless architecture seamlessly integrated with Firebase services.

The ensuing segment of my thesis, designated as DP3, will address several critical aspects outlined briefly here, with a more exhaustive exploration presented in the "Future Work" section. These forthcoming endeavors include deploying the application, transitioning from a monolithic architecture to microservices, conduc-

ting user testing, and implementing a recommendation system based on insights derived from prior datasets.

As the concluding section of my diploma thesis, this recapitulation underscores the successful realization of a functional prototype, outlining the core features and advancements made in database design and architectural considerations. The forthcoming steps, articulated in the DP3 phase, promise to further elevate the application's functionality, usability, and scalability.

References

- [1] Shenai Krishna. *Introduction to database and knowledge-base systems*. Zv. 28. World scientific, 1992.
- [2] Lawrence Reeve a Hyoil Han. „Survey of semantic annotation platforms“. In: *Proceedings of the 2005 ACM symposium on Applied computing*. 2005, s. 1634–1638.
- [3] Rudi Studer, V Richard Benjamins a Dieter Fensel. „Knowledge engineering: Principles and methods“. In: *Data & knowledge engineering* 25.1-2 (1998), s. 161–197.
- [4] Thaddeus J. Kowalski a Donald E. Thomas. „The VLSI Design Automation Assistant: What’s in a Knowledge Base“. In: *22nd ACM/IEEE Design Automation Conference* (1985), s. 252–258. URL: <https://api.semanticscholar.org/CorpusID:7374804>.
- [5] Michael R. Douglas et al. „What Is Learned in Knowledge Graph Embeddings?“. In: *Complex Networks & Their Applications X*. Ed. Rosa Maria Benito et al. Cham: Springer International Publishing, 2022, s. 587–602. ISBN: 978-3-030-93413-2.
- [6] Ruben Dedecker et al. „What’s in a Pod? A Knowledge Graph Interpretation For The Solid Ecosystem“. In: *QuWeDa@ISWC*. 2022. URL: <https://api.semanticscholar.org/CorpusID:254248012>.
- [7] Samira Babalou et al. „What is needed in a Knowledge Graph Management Platform? A survey and a proposal“. In: 2022. URL: <https://api.semanticscholar.org/CorpusID:248980771>.
- [8] URL: <https://www.indeed.com/career-advice/career-development/static-vs-dynamic-website>.

- [9] *Static vs dynamic websites: Key differences*. Nov. 2022. URL: <https://www.pluralsight.com/blog/creative-professional/static-dynamic-websites-theres-difference>.
- [10] *What is open source?* URL: <https://www.redhat.com/en/topics/open-source/what-is-open-source>.
- [11] *The open source definition*. Feb. 2023. URL: <https://opensource.org/osd/>.
- [12] Ritesh Ranjan. *What is a framework in Programming and Why You Should Use one*. Jan. 2023. URL: <https://www.netsolutions.com/insights/what-is-a-framework-in-programming/>.
- [13] Babak Bashari Rad, Harrison John Bhatti a Mohammad Ahmadi. „An introduction to docker and analysis of its performance“. In: *International Journal of Computer Science and Network Security (IJCSNS)* 17.3 (2017), s. 228.
- [14] David Gourley et al. *HTTP: the definitive guide*. O'Reilly Media, Inc., 2002.
- [15] Qinwen Hu, Muhammad Rizwan Asghar a Nevil Brownlee. „A Large-Scale Analysis of HTTPS Deployments: Challenges, Solutions, and Recommendations“. In: *J. Comput. Secur.* 29.1 (jan. 2021), s. 25–50. ISSN: 0926-227X. DOI: 10.3233/JCS-200070. URL: <https://doi.org/10.3233/JCS-200070>.
- [16] Rushank Shah a Stevina Correia. „Encryption of Data over HTTP (Hypertext Transfer Protocol)/HTTPS (Hypertext Transfer Protocol Secure) Requests for Secure Data transfers over the Internet“. In: aug. 2021, s. 587–590. DOI: 10.1109/RTEICT52294.2021.9573978.
- [17] Audun Jøsang. „A consistent definition of authorization“. In: *Security and Trust Management: 13th International Workshop, STM 2017, Oslo, Norway, September 14–15, 2017, Proceedings 13*. Springer. 2017, s. 134–144.
- [18] auth0.com. *JSON web tokens introduction*. URL: <https://jwt.io/introduction>.
- [19] *Stack overflow developer survey 2022*. Jún 2022.

- [20] Dasari Hermitha Curie et al. „Analysis on Web Frameworks“. In: *Journal of Physics: Conference Series*. Zv. 1362. 1. IOP Publishing. 2019, s. 012114.
- [21] Chuck Musciano a Bill Kennedy. *HTML & XHTML: The Definitive Guide: The Definitive Guide*. O'Reilly Media, Inc., 2002.
- [22] Jennifer Kyrnin. „What is HTML 5“. In: *Saatavissa: http://webdesign. about. com/od/html5/qt/what_is_html5. htm [viitattu 2.2. 2011]* ().
- [23] Eric A Meyer. *CSS: The Definitive Guide: The Definitive Guide*. O'Reilly Media, Inc., 2006.
- [24] Paul Wilton. *Beginning JavaScript*. John Wiley & Sons, 2004.
- [25] Remo H Jansen. *Learning TypeScript*. Packt Publishing Ltd, 2015.
- [26] Basarat Syed. *Beginning Node. js*. Apress, 2014.
- [27] Elar Saks. „JavaScript Frameworks: Angular vs React vs Vue.“ In: (2019).
- [28] URL: <https://vitejs.dev/guide/why.html>.
- [29] Michael Roberts a John Chapin. *What is Serverless?* O'Reilly Media, Incorporated, 2017.
- [30] URL: <https://docs.nestjs.com/>.
- [31] Hassan B Hassan, Saman A Barakat a Qusay I Sarhan. „Survey on serverless computing“. In: *Journal of Cloud Computing* 10.1 (2021), s. 1–29.
- [32] Coralogix. *Aws Lambda vs Azure Functions vs google cloud functions*. Mar. 2023. URL: <https://coralogix.com/blog/aws-lambda-vs-azure-functions-vs-google-cloud-functions/>.
- [33] Chris Tozzi. *Compare aws lambda vs. Azure Functions vs. Google Cloud Functions: TechTarget*. Júl 2021. URL: <https://www.techtarget.com/searchcloudcomputing/tip/Compare-AWS-Lambda-vs-Azure-Functions-vs-Google-Cloud-Functions>.
- [34] Andrew Pavlo a Matthew Aslett. „What's really new with NewSQL?“ In: *ACM Sigmod Record* 45.2 (2016), s. 45–55.

- [35] Santiago Timón-Reina, Mariano Rincón a Rafael Martínez-Tomás. „An overview of graph databases and their applications in the biomedical domain“. In: *Database 2021* (2021).
- [36] *System properties comparison Amazon Neptune vs. Graphdb vs. Neo4j*. URL: <https://db-engines.com/en/system/Amazon+Neptune%5C%3BGraphDB%5C%3BNeo4j>.
- [37] Apr. 2023. URL: <https://neo4j.com/cloud/platform/aura-graph-database/>.
- [38] Ross Harmes a Dustin Diaz. „The Adapter Pattern“. In: *Pro JavaScript Design Patterns* (2008), s. 149–158.
- [39] John Hunt a John Hunt. „Adapter Pattern“. In: *Scala Design Patterns: Patterns for Practical Reuse and Design* (2013), s. 169–181.
- [40] Oludare Isaac Abiodun et al. „State-of-the-art in artificial neural network applications: A survey“. In: *Heliyon* 4.11 (2018), e00938. ISSN: 2405-8440. DOI: <https://doi.org/10.1016/j.heliyon.2018.e00938>. URL: <https://www.sciencedirect.com/science/article/pii/S2405844018332067>.
- [41] Mustafa Ergen et al. „What is artificial intelligence? Technical considerations and future perception“. In: *Anatolian J. Cardiol* 22.2 (2019), s. 5–7.
- [42] Mahalakshmi Neelam. „Neelam MahaLakshmi (2021) Aspects of Artificial Intelligence In Karthikeyan. J, Su-Hie Ting and Yu-Jin Ng (eds),“Learning Outcomes of Classroom Research” p: 250-256, L’Ordine Nuovo Publication, India“. In: (2022).
- [43] Batta Mahesh. „Machine learning algorithms-a review“. In: *International Journal of Science and Research (IJSR).[Internet]* 9 (2020), s. 381–386.
- [44] Bing Liu a Bing Liu. *Supervised learning*. Springer, 2011.
- [45] Peter Dayan, Maneesh Sahani a Grégoire Deback. „Unsupervised learning“. In: *The MIT encyclopedia of the cognitive sciences* (1999), s. 857–859.

- [46] Xiaojin Zhu a Andrew B Goldberg. „Introduction to semi-supervised learning“. In: *Synthesis lectures on artificial intelligence and machine learning* 3.1 (2009), s. 1–130.
- [47] Neha Gupta et al. „Artificial neural network“. In: *Network and Complex Systems* 3.1 (2013), s. 24–28.
- [48] Xing Wu, Paweł Różycki a Bogdan M. Wilamowski. „A Hybrid Constructive Algorithm for Single-Layer Feedforward Networks Learning“. In: *IEEE Transactions on Neural Networks and Learning Systems* 26.8 (2015), s. 1659–1668. DOI: 10.1109/TNNLS.2014.2350957.
- [49] Zhihua Zhang a Zhihua Zhang. „Artificial neural network“. In: *Multivariate time series analysis in climate and environmental research* (2018), s. 1–35.
- [50] Moshe Leshno et al. „Multilayer feedforward networks with a nonpolynomial activation function can approximate any function“. In: *Neural Networks* 6.6 (1993), s. 861–867. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/S0893-6080\(05\)80131-5](https://doi.org/10.1016/S0893-6080(05)80131-5). URL: <https://www.sciencedirect.com/science/article/pii/S0893608005801315>.
- [51] Elizabeth D Liddy. „Natural language processing“. In: (2001).
- [52] KR1442 Chowdhary a KR Chowdhary. „Natural language processing“. In: *Fundamentals of artificial intelligence* (2020), s. 603–649.
- [53] Murray Shanahan. „Talking About Large Language Models“. In: *arXiv pre-print arXiv:2212.03551* (2022).
- [54] Christian Janiesch, Patrick Zschech a Kai Heinrich. „Machine learning and deep learning“. In: *Electronic Markets* 31.3 (2021), s. 685–695.
- [55] Saman Razavi. „Deep learning, explained: Fundamentals, explainability, and bridgeability to process-based modelling“. In: *Environmental Modelling and Software* 144 (2021), s. 105159. ISSN: 1364-8152. DOI: <https://doi.org/>

- 10.1016/j.envsoft.2021.105159. URL: <https://www.sciencedirect.com/science/article/pii/S1364815221002024>.
- [56] Andreas C Neves et al. „A new approach to damage detection in bridges using machine learning“. In: *Experimental Vibration Analysis for Civil Structures: Testing, Sensing, Monitoring, and Control* 7. Springer. 2018, s. 73–84.
- [57] Joseph Awoamim Yacim a Douw Gert Brand Boshoff. „Impact of artificial neural networks training algorithms on accurate prediction of property values“. In: *Journal of Real Estate Research* 40.3 (2018), s. 375–418.
- [58] *Welcome to schema.org*. URL: <https://schema.org/>.
- [59] *Your machine learning and Data Science Community*. Feb. 2010. URL: <https://www.kaggle.com/>.
- [60] Kenza Kellou-Menouer et al. „A survey on semantic schema discovery“. In: *The VLDB Journal* 31.4 (2022), s. 675–710.
- [61] Apr. 2023. URL: <https://research.com/software/mobile-vs-desktop-usage>.
- [62] Geoff839. *EVS - one electric vehicle dataset - smaller*. Aug. 2020. URL: <https://www.kaggle.com/datasets/geoffnel/evs-one-electric-vehicle-dataset>.
- [63] C-3PO. *A small covid-19 dataset*. Máj 2021. URL: <https://www.kaggle.com/datasets/aditeloo/a-small-covid19-dataset>.
- [64] pc1975. *Small dataset about used Fiat 500 sold in Italy*. Apr. 2020. URL: <https://www.kaggle.com/datasets/paolocons/small-dataset-about-used-fiat-500-sold-in-italy>.
- [65] Niels de Jong. *15 tools for visualizing your neo4j graph database*. Jún 2023. URL: <https://neo4j.com/developer-blog/15-tools-for-visualizing-your-neo4j-graph-database/>.
- [66] URL: <https://neo4j.com/graphgists/my-bea/>.

- [67] URL: <https://neo4j.com/graphgists/product-hierarchy-graphgist/>.
- [68] Enzo. *What is a knowledge graph?* Nov. 2023. URL: <https://neo4j.com/blog/what-is-knowledge-graph/>.
- [69] URL: <https://openai.com/chatgpt>.
- [70] URL: https://github.com/feiqi9047/Project-Conversion/blob/master/KAG_conversion_data.csv.
- [71] URL: <https://github.com/araj2/customer-database/blob/master/Ecommerce%20Customers.csv>.

Attachment A: Diploma thesis evaluation

A.1 Work plan for phase DP1

Week of the semester	Activity plan
1.week	Familiarisation with the topic
2.week	Study of the problem
3.week	Reading articles on the topic and collecting literature
4.week	Reading articles on the topic and collecting literature
5.week	Reading articles on the topic and collecting literature
6.week	Reading articles on the topic and collecting literature
7.week	Thesis writing
8.week	Thesis writing
9.week	Thesis writing
10.week	Thesis writing
11.week	Thesis writing
12.week	Final touches

A.2 Evaluation of the work plan for the DP1 phase

Evaluation of the first part of the thesis DP1 in the summer semester

In the summer semester, I worked on the first steps of my thesis project. The aim of this part was to thoroughly familiarize myself with the topic and gain the necessary knowledge to solve the problem. This was followed by a schedule of activities, which I followed gradually:

In the first week I devoted myself to familiarizing myself with the topic. I studied

the relevant information about the problem, its context and the goals I had to achieve. This phase was important to gain a broader overview and basic understanding of the issue.

In the second week, I devoted myself intensively to the study of the problem. I analysed its nature in more detail, identified the main challenges and explored the related areas in depth. This step allowed me to gain a deeper knowledge and a basis for further procedures.

Weeks 3 to 6 were focused on reading articles on the topic and collecting literature. I thoroughly studied relevant studies, scientific articles and publications in the relevant fields. The aim was to gain a more comprehensive overview of the problem and to identify the most up-to-date knowledge.

In the seventh week, I worked on writing the thesis. Based on my researched materials and knowledge, I began to formulate the content of the thesis. I gradually developed the structure and wrote parts of the document in order to follow a logical progression and to ensure coherence and coherence of the text.

Weeks 8 to 11 were devoted to writing the document and gradually expanding it. At this stage I included all relevant information, analysis and results of my studies in the thesis. I tried to achieve a higher level of detailed analysis and to present my findings and results clearly.

In the twelfth week, I devoted myself to the final editing of the document. I reread the whole text and made the necessary proofreading and editing. I ensured that the document was complete, clear and grammatically correct.

Throughout the work, I consulted regularly with my supervisor. Together we went over my progress, discussing my results and evaluating the shortcomings and benefits of my work. His feedback provided guidance for my next steps and helped

me to better navigate the problem.

Overall, I can conclude that I have successfully met the objectives set out in the schedule. I familiarised myself with the topic, analysed the problem, studied the relevant literature and wrote a coherent document. These steps provided me with a solid foundation for the next stages of the thesis and allowed me to better understand and plan a solution to the problem I set out to solve.

A.3 Work plan for phase DP2

Week of the semester	Activity plan
1.week	Setup of frontend, backend and database
2.week	Creating login and register logic and pages
3.week	Creating admin page for role changes
4.week	Creating find you dataset page
5.week	Creating Selected dataset page
6.week	Creating annotations page
7.week	Creating export
8.week	Creating testing datasets
9.week	Creating database seeders
10.week	Thesis writing
11.week	Thesis writing
12.week	Final touches

A.4 Evaluation of the work plan for the DP2 phase

Evaluation of the second part of the thesis, DP2, during the winter semester primarily centered on the development of a application prototype. Over the course of the semester:

During the first week, attention was devoted to configuring the frontend, backend, and database. This involved the creation of a Firebase project, incorporating Firebase authentication, Firestore for data storage, and Firebase functions. Also the .env file was used for database credentials.

The second week was dedicated to formulating the logic and pages for the login and registration processes. Firebase authentication was employed, and Firestore was used to manage user roles. Access to specific pages was restricted based on roles, ensuring, for example, that only administrators could access the admin page, and only logged-in users could access the “find a dataset“ page.

Progressing to the third week, the focus shifted to the creation of an admin page with the capability to modify user roles. This feature streamlined the role-changing process, eliminating the need for direct interactions with the Firestore database.

In the fourth week, the “find your dataset“ page was established. Adding the function for adding datasets and storing them within Firestore. Subsequent weeks involved the creation of the selected dataset page, enabling users to add descriptions to datasets and their respective columns, while also providing a managed view of the dataset.

The sixth week saw the development of an annotations page, allowing users to view annotations. Additionally, functionality was added to annotate columns within the

selected dataset page.

In the seventh week, the implementation of dataset export functionality as both CSV and JSON formats, including annotated columns, was completed.

Week eight was dedicated to the creation of testing datasets, while week nine focused on the development of functional seeders for Firestore and Neo4j, incorporating previously generated datasets and annotations.

The final three weeks (weeks 10 to 12 included) were devoted to updating this Overleaf document, which forms an integral part of my diploma thesis.

Throughout the work, I consulted regularly with my supervisor. Together we went over my progress, discussing my results and evaluating the shortcomings and benefits of my work. His feedback provided guidance for my next steps and helped me to better navigate the problem.

Overall, I can conclude that I have successfully met the objectives set out in the schedule. I created a working prototype of my page. Which provides me with a solid foundation for the next and final stage of my thesis.

A.5 Work plan for phase DP3

Week of the semester	Activity plan
1.week	Deployment of the frontend
2.week	Deployment of cloud functions
3.week	Integrating neo4j aura
4.week	Final deployment integration
5.week	User testing
6.week	Evaluation of user testing
7.week	Fixing bugs found during user testing
8.week	Fixing UI problems found during user testing
9.week	Thesis writing
10.week	Thesis writing
11.week	Thesis writing
12.week	Final touches

A.6 Evaluation of the work plan for the DP3 phase

Work that has not been completed yet

Attachment B: User Guide

B.1 Program function

I developed a web-based data science environment designed to seamlessly handle the integration, annotation, and distribution of datasets to meet the specific needs of data scientists. The primary objective is to provide a comprehensive infrastructure for efficient data manipulation and AI model evaluation. This application supports dynamic addition, annotation, and transmission of datasets on demand.

To facilitate user-friendly data interaction, I devised a method for annotating datasets and sending them. The chosen annotation format ensures clarity and accessibility for both human users through the interface and machine interpretation through the API.

The application features a robust database model for storing annotated datasets, allowing users to explore the data representation and understand the relationships between different elements. This dual functionality, both human-centric through the user interface and machine-readable through the API, ensures versatility in data manipulation.

I've set up an admin user for testing purposes with the following credentials:

1. Username: admin@gmail.com
2. Password: admin@gmail.com

To facilitate testing, I've also provided a Postman export that contains all available endpoints. You can find the export file in the folder of my diploma thesis, named "diploma-thesis.postman_collection.json." This will assist you in efficiently testing

and exploring the various functionalities outlined in the project.

Some endpoints require user to be logged in, mainly the ones that change the data. To enable the functionality of the Postman collection, in other words to log in follow these steps:

1. Initiate a call to either `firebase/login` or `firebase/register`.
2. Extract the “idToken” value from the response.
3. Assign this “idToken” value to a global variable named “firebaseToken.”

Locate the “Environment quick look” at the top right corner or refer to this tutorial. for detailed instructions. By completing these steps, you’ll set up the necessary authentication token for Firebase API requests within the Postman environment.

B.2 Installing the program

In later stages this website will be available online without the need to install anything but Currently for the program to run you have to install Neo4j desktop create Project: diploma-thesis-project with database: diploma-thesis-database with password: 123456789 and version 5.12.0. Then you need to click start on this database. Next step is to go into the project folder which I will upload together with my thesis. Go into folder back-end/server and run commands: “`npm i`” to install all the required dependencies and then run command “`npm run dev`” to run the back-end of my application. Then go to front-end and run command “`npm i`” to again install the required dependencies and run “`npm run dev`” to run the front-end of my application. Now you are able to visit “`http://localhost:5173/`” where my application resides.

Attachment C: Description of the digital part

Registration number of the work in the information system: XXXXXXXXXXXX

Content of the digital part of the thesis (ZIP archive):

Folder/File	Description
\back-end	firebase initialization files
\back-end\server	NestJS application
\back-end\server\.env	environment variables
\back-end\server\package.json	contains necessary libraries to run the application
\back-end\server\src	application source files
\back-end\server\src\seeders	seeders for firebase and neo4j
\back-end\server\src\api	api of my application
\front-end	Vue.js application and all config files
\front-end\package.json	contains necessary libraries to run the application
\front-end\src	application source files
\front-end\src\main.ts	main application file
\front-end\src\views	views of my application
\front-end\src\components	components of my application
\front-end\src\assets	styles of my application
\front-end\src\router	routes of my application
\datasets	example datasets
diploma-thesis.postman_collection.json	postman JSON with all available endpoints
\assets	assets of my diploma thesis

Archive name: Plevko_Peter_DP2.zip