

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

June 2023

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

June 2023

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

V Bratislave, Jún 2023

.....

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

June 2023

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

June 2023

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	Web page	5
2.2	Open source	6
2.3	Framework	6
2.4	Docker	8
2.5	Security	9
2.5.1	HTTP	9
2.5.2	Login	10
2.5.3	User types	11
2.5.4	JSON Web Token	12
2.6	Front-end	13
2.6.1	HTML	14
2.6.2	CSS	15
2.6.3	JavaScript	16
2.7	Back-end	18
2.7.1	Database	20
2.7.2	Adapter pattern	21
2.8	Data model	24
2.8.1	Artificial Intelligence	26
2.8.2	Machine learning	27
2.8.3	Artificial Neural Networks	29

2.8.4	Natural language processing	31
2.8.5	Large language model	33
2.8.6	Deep learning	34
2.9	Data format	36
3	Related Works	40
4	Design	43
4.1	Technologies	44
4.2	Web application specification and requirements	46
4.2.1	Usability	47
4.2.2	Awareness	47
4.2.3	Accessibility	47
4.2.4	Accessibility	48
4.2.5	Ease of Use	48
4.3	Target group	48
4.3.1	Target Audience Characteristics	49
4.3.2	Device Usage Statistics	49
4.4	Database model	50
4.5	Diagrams	54
4.5.1	Login diagram flowchart	54
4.5.2	Add and annotate dataset flowchart	56
4.5.3	Google cloud infrastructure diagram	58
4.5.4	Google cloud CI/CD diagram	60
4.6	User interface	62
4.6.1	Find a dataset	63
4.6.2	Selected dataset	64
4.6.3	Find annotation	65

4.6.4	Selected annotation	66
4.6.5	Add annotation	67
A	Harmonogram práce v ls	A-1
A.1	Zhodnotenie	A-1

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 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 [4, 5]. 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 [4]. 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 [4]. 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.

2.2 Open source

As per the information provided by sources such as [6, 7], 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.

2.3 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 [8]. 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.

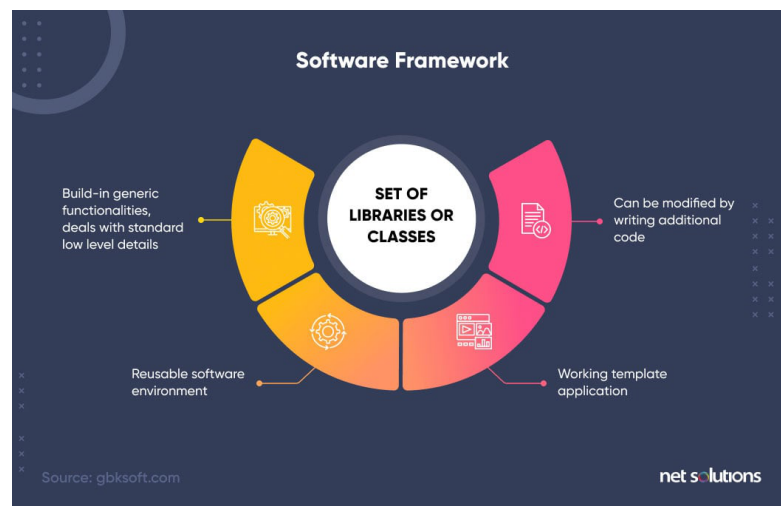


Figure 2.1: Software Framework [8]

2.4 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 [9].

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.

2.5 Security

2.5.1 HTTP

As described in the book “HTTP: The Definitive Guide“ [10], 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 [11, 12]. It operates as an application-specific implementation that encapsulates HTTP within SSL (Secure Socket 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 [11]. 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.

2.5.2 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 [13].

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 [13].

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.

2.5.3 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.

2.5.4 JSON Web Token

All the information provided in this section was taken from jwt.io website [14]. 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.

2.6 Front-end

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 [15]. 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 [16].

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.

2.6.1 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 [17]. 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 [18]. 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.

2.6.2 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 [19].

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.

2.6.3 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 [20].

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 over-

head 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 [21].

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 [22].

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 [23] 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.

2.7 Back-end

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 [24].

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, leveraging a framework like NestJS, which is built on top of Node.js and TypeScript, affords a modular and scalable architecture while providing support for diverse databases and APIs.

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 [25].

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 [26, 27].

2.7.1 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 [28]. 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 [29].

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 [29, 30].

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 [29, 30].

2.7.2 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 [31].



Figure 2.2: Adapter design pattern [32]

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 [31].

Let's examine an example that illustrates the effectiveness of the adapter design pattern in our project. Currently, we are utilizing Neo4j as our database system. However, it is important to consider the possibility of future changes in the choice of database, such as a transition to PostgreSQL or some other database.

In this scenario, the adapter design pattern proves to be a suitable solution. By implementing adapters, we can seamlessly switch between different database systems while minimizing the need for extensive modifications to the existing codebase. The adapters act as intermediaries between the application code and the underlying database, providing a consistent interface regardless of the chosen database technology.

As seen in figure 2.3 to implement the adapter pattern, we introduce a class called "ChooseAdapter" within each controller. This class serves as the decision-making component that selects the appropriate adapter based on an environment variable specifying the current database in use. By encapsulating the logic for adapter selection within the "ChooseAdapter" class, we decouple the code from the specific database implementation.

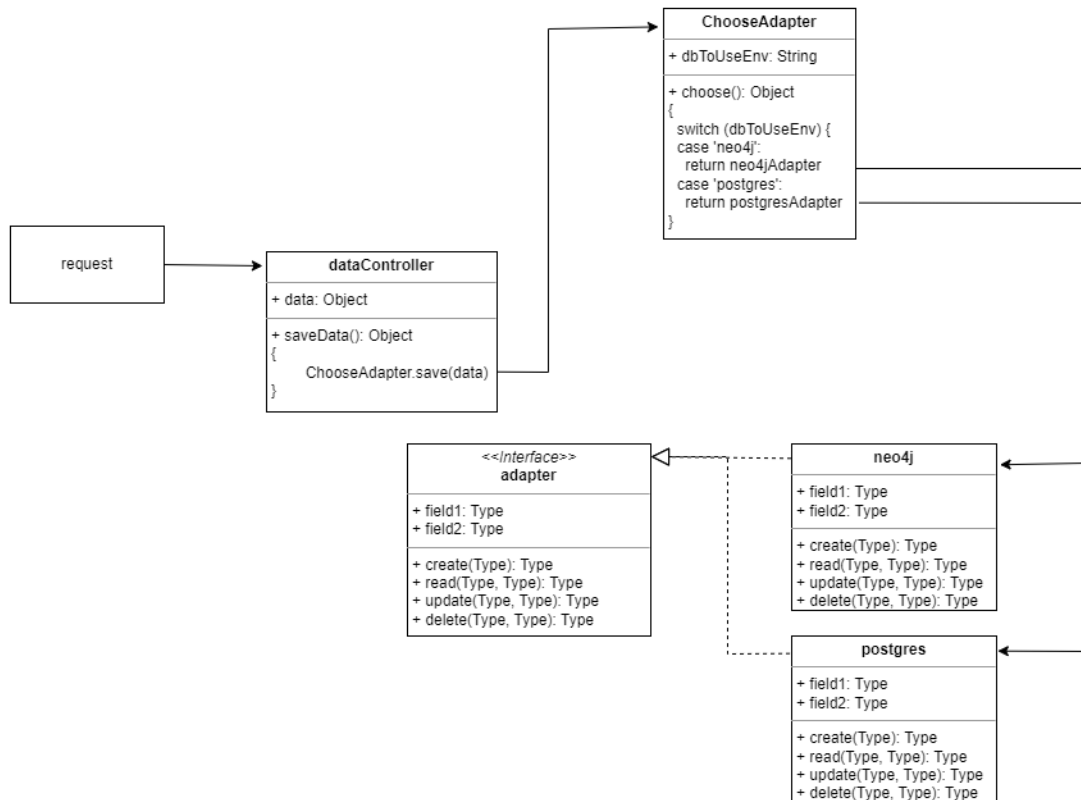


Figure 2.3: Adapter design pattern in project

Each adapter implements an interface that defines a standardized set of methods, ensuring compatibility between the adapter and the codebase. This interface establishes a contract that specifies the required behavior for interacting with the database. By adhering to this interface, the adapters provide a consistent API for the controllers, regardless of the underlying database technology.

In the context of transitioning from Neo4j to PostgreSQL, the necessary changes for the user are limited to adding a switch option in the “ChooseAdapter” class and implementing the PostgreSQL adapter. This approach significantly reduces the effort required to adapt the codebase to the new database system, as it eliminates the need for modifications in every controller and method.

The adapter design pattern provides flexibility and maintainability by separating the concerns of the application code and the database system. It allows for easy integration of different database technologies, accommodating potential future changes in the choice of the database system.

By utilizing the adapter design pattern, we can ensure a smooth transition from Neo4j to PostgreSQL, while keeping our codebase adaptable to future changes in the database technology.

2.8 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 Figure 2.4 [33]. 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.

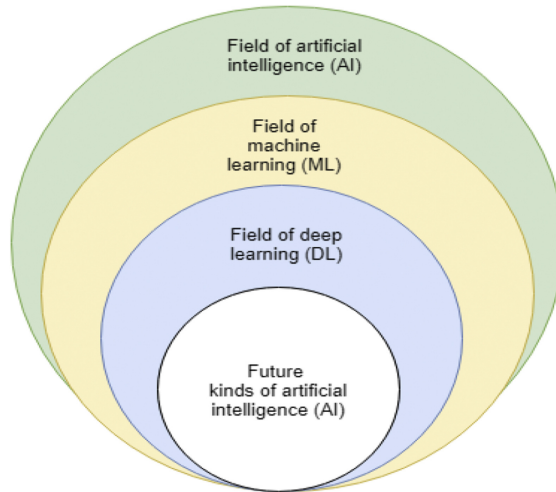


Figure 2.4: Artificial intelligence development and expansion. [33]

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 meaning-

ful 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.

2.8.1 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 [34].

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 cata-

lyst, 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 [35].

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.

2.8.2 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 [36].

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 [37].

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 [38].

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 [39].

2.8.3 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 [40].

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 [41].

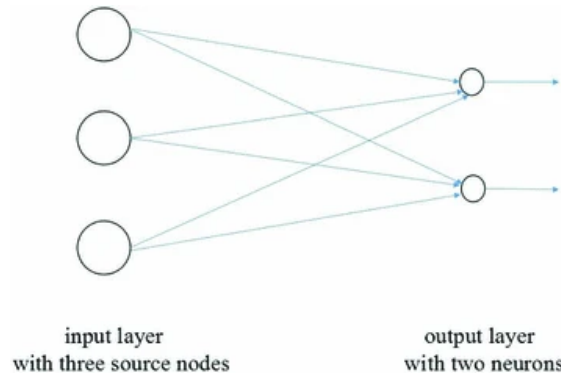


Figure 2.5: Single-layer feedforward networks [42]

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 corres-

ponding 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 [43].

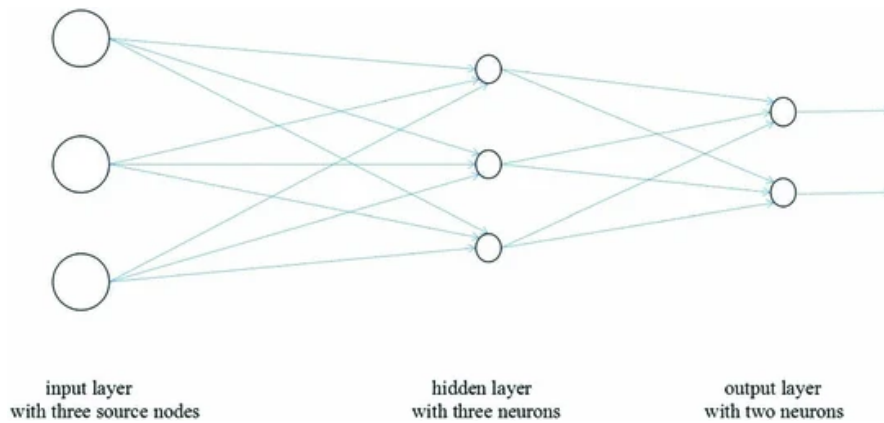


Figure 2.6: Multilayer feedforward network with a single hidden layer [42]

2.8.4 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 natu-

ral 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 [44].

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 sentence 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 [45].

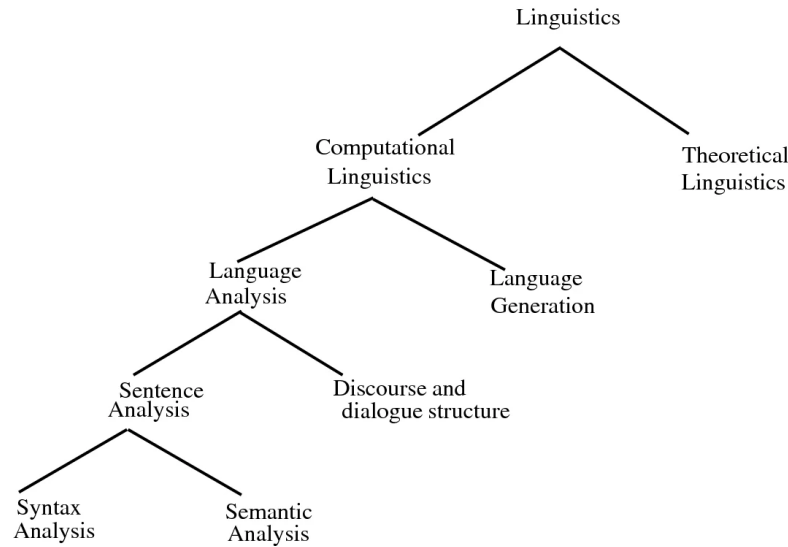


Figure 2.7: Components of Natural Language Processing [45]

2.8.5 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 mathematical 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 [46].

2.8.6 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 interpretability. 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 unders-

tanding 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 [47].

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 Figure 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 [48].

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

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,

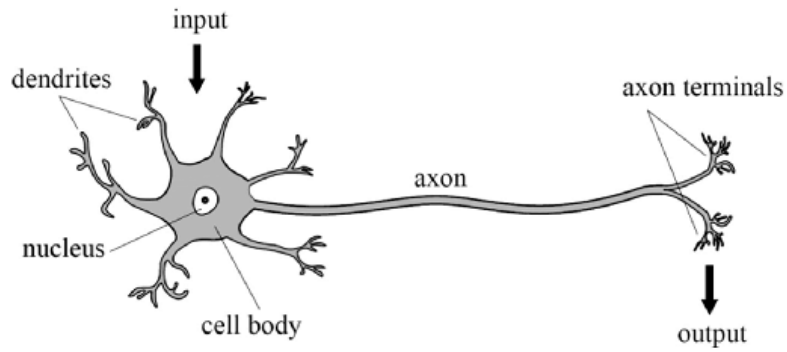


Figure 2.8: Biological neuron [49]

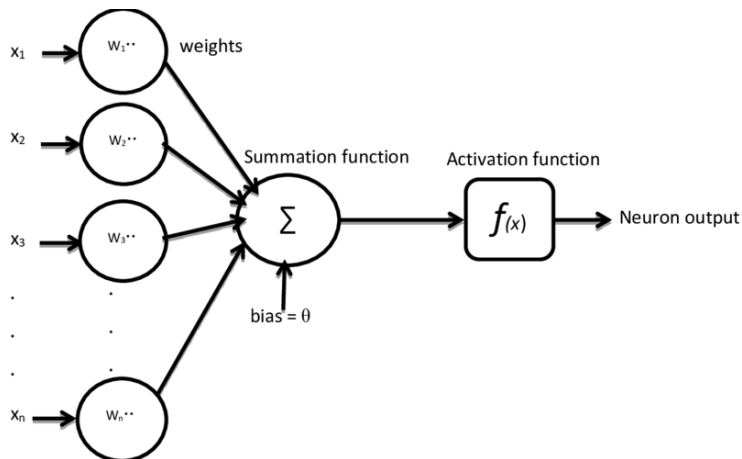


Figure 2.9: Artificial neuron - perceptron [50]

2.9 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 [51] 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 [52] 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 [53]. 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 in Neo4j. Furthermore, the sections showcase diagrams that depict the functionality of the web page, and conclude with the design of both low-fidelity and 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.

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 improvement.

Additionally, both low-fidelity and high-fidelity wireframes will be designed to showcase the visual layout and user interface of the web page. Low-fidelity wireframes provide a rough representation of the page's structure and content placement, while high-fidelity wireframes offer a more refined and detailed representation, closely resembling the final product.

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.

To ensure easy deployment and scalability, the front-end of the application will be packaged into a Docker image, which will be subsequently published into the Google App Engine. This cloud-based hosting service provides a reliable infrastructure for running web applications with minimal configuration and maintenance overhead.

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.

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 switch between different database implementations with ease. This design pattern promotes code reusability, scalability, and adaptability, enhancing the overall functionality of the project.

Data will be stored in two distinct ways. Nodes representing CSV dataset will be utilized to store the dataset, ensuring efficient storage and retrieval of large volumes of data. Additionally nodes representing metadata will be employed to store annotated information related to the dataset. This relational connection between the dataset and metadata provides a comprehensive and structured approach to data management and organization.

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 Accessibility

To suit to users with varying technical proficiencies, a clear and brief user manual will be provided as an appendix in the master's thesis. The manual will serve as a comprehensive guide, elucidating the functionality of the web application and providing step-by-step instructions for dataset browsing, downloading, and annotation. This documentation will ensure that even less tech-savvy users can easily comprehend and utilize the web application's features.

4.2.5 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 [54] 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 Figure 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.

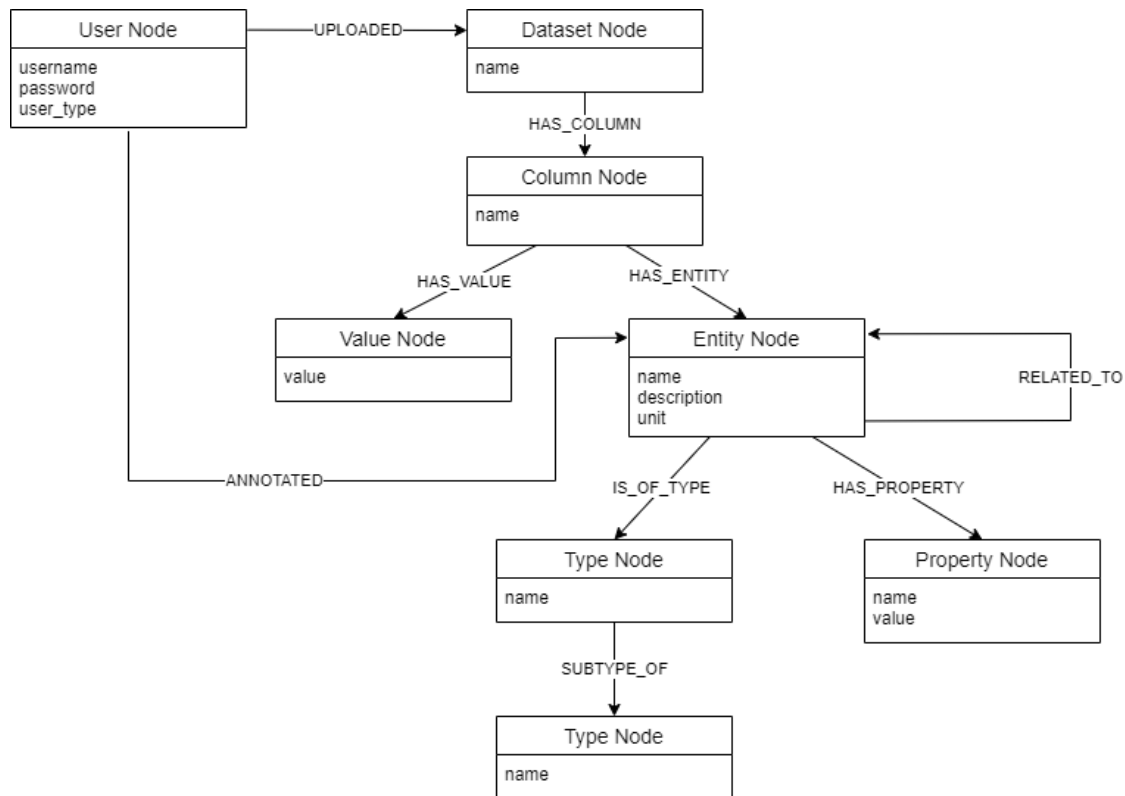


Figure 4.1: Neo4j database model

The subsequent section showcases the Neo4j code representation of the aforementioned database model showed earlier. This Neo4j code articulates the structural blueprint of the database, capturing its entities, attributes, and the intricate interconnections that govern the data relationships within the system.

Node: User

```

Properties: {username: 'data_scientist', password: '
    hashed_password'}
Relationships: UPLOADED -> Dataset {name: 'housing_data'},
    ANNOTATED -> Entity {name: 'Area', description: 'A
    physical quantity denoting area', unit: 'meters'}

Node: Dataset
Properties: {name: 'housing_data'}
Relationships: CONTAINS -> Column {name: 'area'}

Node: Column
Properties: {name: 'area'}
Relationships: HAS_ENTITY -> Entity {name: 'Area',
    description: 'A physical quantity denoting area', unit:
    'meters'}

Node: Entity
Properties: {name: 'Area', description: 'A physical
    quantity denoting area', unit: 'meters'}
Relationships: IS_OF_TYPE -> Type {name: 'Area Annotation
    '}, HAS_PROPERTY -> Property {name: 'Unit', value: '
    meters'}, IS_RELATED_TO -> Entity {name: 'Population',
    description: 'The number of individuals in a given area
    ', unit: 'count'}

Node: Type
Properties: {name: 'Area Annotation'}
Relationships: SUBTYPE_OF -> Type {name: 'Physical Quantity
    Annotation'}

```

Node: Type

Properties: {name: 'Physical Quantity Annotation'}

Node: Property

Properties: {name: 'Unit', value: 'meters'}

Node: Entity

Properties: {name: 'Population', description: 'The number
of individuals in a given area', unit: 'count'}

This code describes a graph data model in Neo4j, 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:

- **Node: User:** Represents a user of the system. Properties like 'username' and 'password' are stored in this node.
- **Relationship: UPLOADED -> Dataset:** Indicates that the user has uploaded a dataset to the system. The dataset is represented as another node in the graph.
- **Node: Dataset:** Represents a dataset that the user has uploaded. The name of the dataset is stored in this node.
- **Relationship: CONTAINS -> Column:** Indicates that the dataset contains a column. Each column in the dataset is represented as another node in the graph.
- **Node: Column:** Represents a column in the dataset. The name of the column is stored in this node.

- **Relationship: HAS_ENTITY -> Entity:** Indicates that the column has been annotated with a certain entity. The entity, which can be any concept or thing that the system knows about, is represented as another node in the graph.
- **Node: Entity:** Represents an entity that the column has been annotated with. This node stores properties like the name, description, and unit of the entity.
- **Relationship: IS_OF_TYPE -> Type:** Indicates the type of the entity. The type is represented as another node in the graph.
- **Node: Type:** Represents the type of an entity. In this case, there are two **Type** nodes: one for “Area Annotation” and one for “Physical Quantity Annotation”. The “Area Annotation” type is a subtype of the “Physical Quantity Annotation” type, as indicated by the **SUBTYPE_OF** relationship.
- **Relationship: HAS_PROPERTY -> Property:** Indicates that the entity has a property. The property is represented as another node in the graph.
- **Node: Property:** Represents a property of an entity. This node stores properties like the name and value of the property.
- **Relationship: ANNOTATED -> Entity:** Indicates that the user has annotated a column with a certain entity.
- **Relationship: RELATED_TO -> Entity:** Indicates a relationship between two entities, representing their association or relevance within the given context.

4.5 Diagrams

4.5.1 Login diagram flowchart

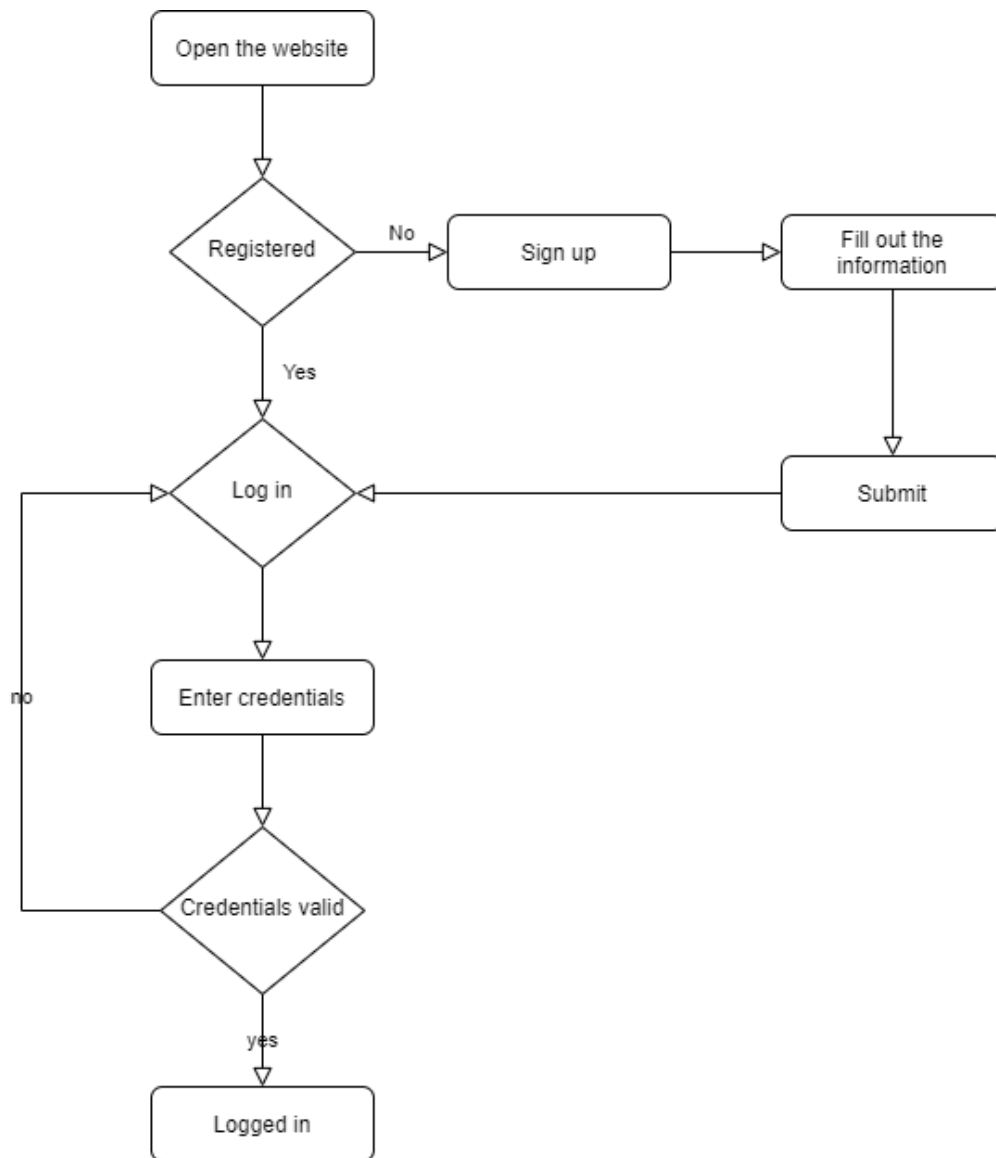


Figure 4.2: Login flowchart diagram

The flowchart presents a sequential representation of the user journey starting from accessing the website. Upon opening the website, the flowchart checks whether the user is registered or not. If the user is not registered, the flowchart directs them to the sign-up process, where they are prompted to provide necessary information. After filling out the required information, the user submits the form.

Following the submission, the flowchart proceeds to the login process. If the user is already registered, they are directed straight to the login step. At this point, the user is prompted to enter their credentials. The flowchart then evaluates the validity of the provided credentials. If the credentials are determined to be valid, the flowchart signifies successful authentication, and the user is logged into the system. Conversely, if the credentials are deemed invalid, the flowchart redirects the user back to the login step for them to re-enter their information.

4.5.2 Add and annotate dataset flowchart

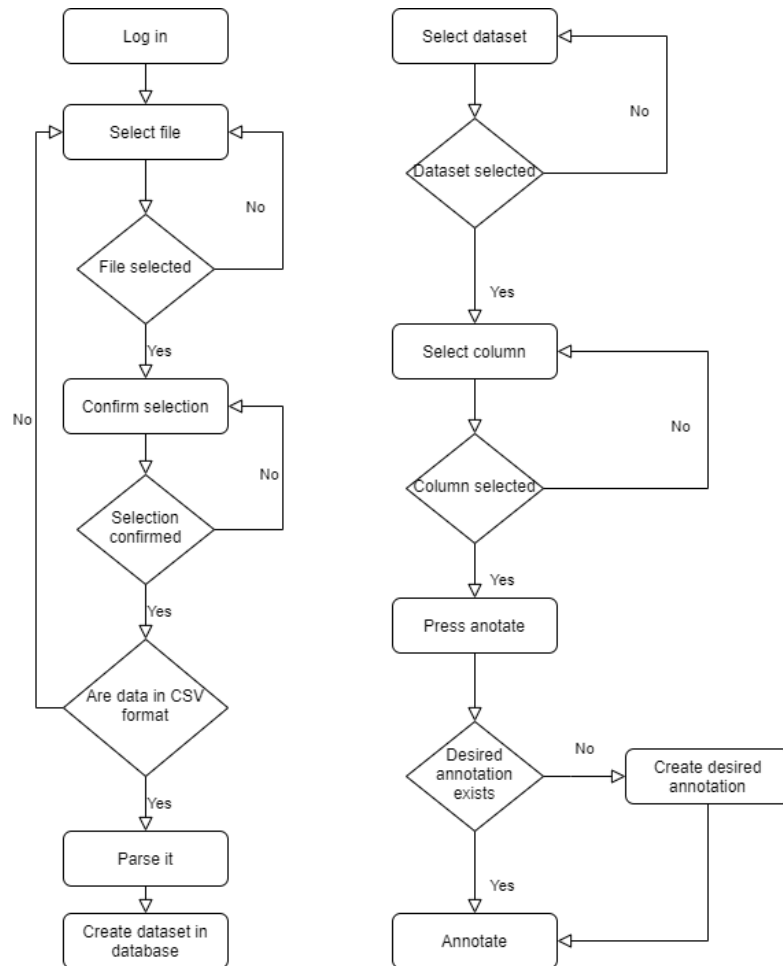


Figure 4.3: Add and annotate dataset flowchart diagrams

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.

Once the user selects a file, the flowchart advances to the “Confirm Selection” stage. If the file selection is not confirmed, the flowchart redirects the user back to the “Select File” stage to choose a different file. However if the selection is confirmed,

the flowchart moves forward to validate whether the data in the selected file is in CSV format.

If the data is not in the CSV format, the flowchart loops back to the “Select File“ stage, allowing the user to either select a different file or confirm the selection of a valid CSV file. On the other hand, if the data is indeed in CSV format, the flowchart proceeds to the next step, which involves parsing the data from the CSV file.

After successful parsing, the flowchart leads to the creation of a dataset in the database. This step involves storing the parsed data into the appropriate format within the database.

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 “Select Column“ 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 “Press Annotate“ 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 Annotation“ stage, where the user defines the annotation. Then, it transitions to the “Annotate“ step for annotation.

4.5.3 Google cloud infrastructure diagram

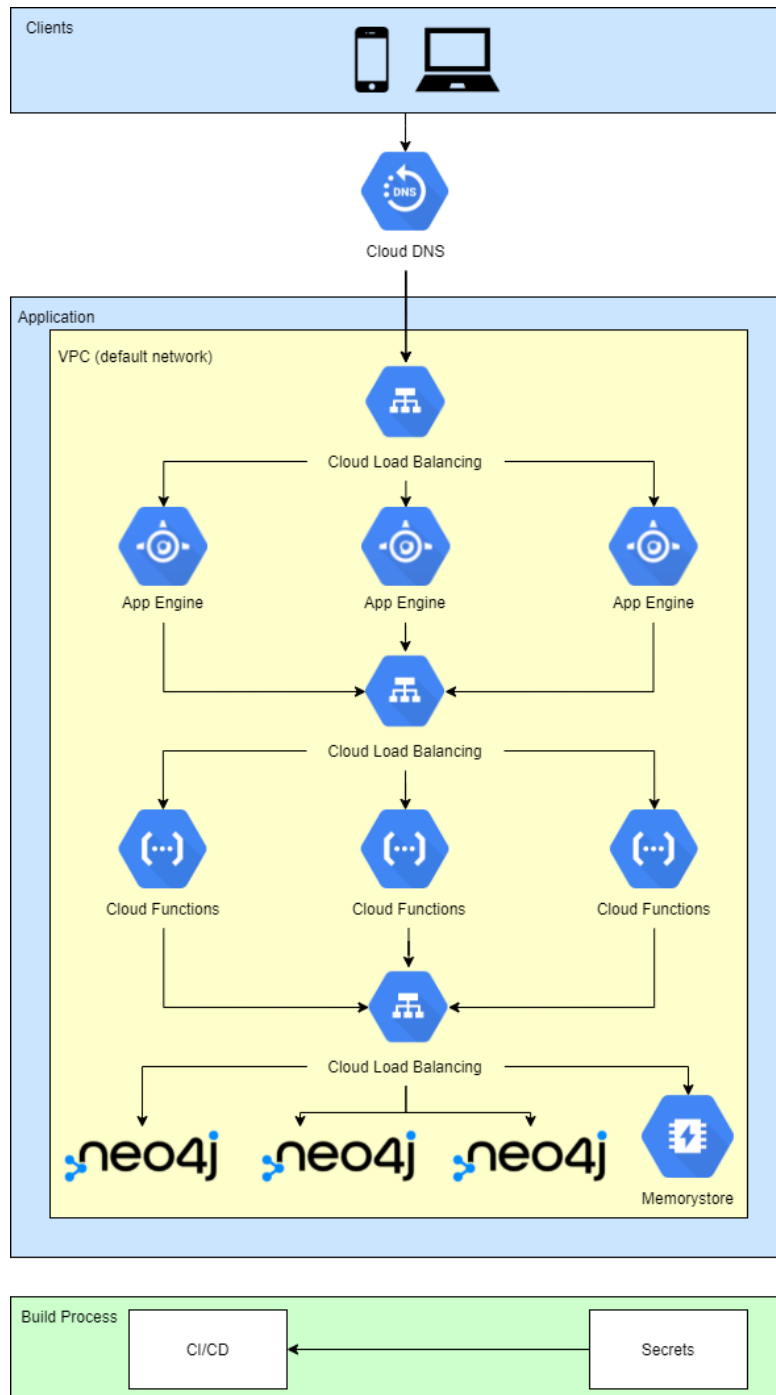


Figure 4.4: Google cloud infrastructure diagram

The diagram in Figure 4.4 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 App Engine. 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 and one instance of Memory Store. Neo4j Aura is a fully-managed, always-on graph database with horizontal scalability and high availability, while Memory Store provides in-memory data storage capabilities.

4.5.4 Google cloud CI/CD diagram

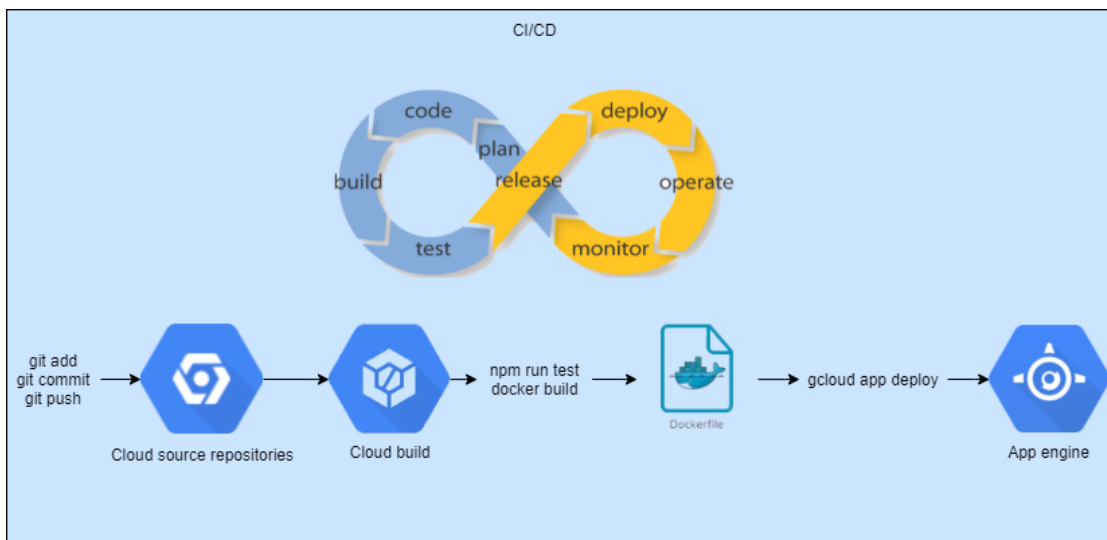


Figure 4.5: Google cloud CI/CD diagram

The Continuous Integration and Continuous Deployment (CI/CD) workflow for the frontend 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.

The role of Google Cloud Build in the CI/CD pipeline is twofold. Primarily, it functions as the automated build service for the application. It is configured to execute a series of instructions outlined in a `Dockerfile`, which essentially involves testing the code and if there are no errors building a Docker image of the frontend application. Secondly, Google Cloud Build is responsible for the deployment of the built Docker image to Google App Engine.

Google App Engine is a fully managed, serverless platform provided by Google Cloud, which is designed to host and run applications. In this workflow, it acts as the final destination for the Docker image, thereby realizing the deployment of the updated frontend application.

In the context of backend development, Google Cloud Functions simplifies the process significantly. It allows developers to author code on their local machine and subsequently deploy these functions directly to the Google Cloud environment via the `'gcloud'` command-line tool. This streamlined process mitigates the need for detailed deployment diagrams, which are often crucial for more complex deployment strategies. Nevertheless, despite the absence of a strict necessity for a git repository in this scenario, it's important to adhere to good development practices. Employing a version control system, like git, for instance, is recognized universally as a best practice in the software development industry. Version control systems provide several benefits, including but not limited to, maintaining a history of

code changes, facilitating code reviews, and fostering collaborative work on shared codebases. Furthermore, they enable developers to experiment with different features and versions of their code without jeopardizing the functionality of their main application.

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 appendix of this thesis, within the Figma project file.

4.6.1 Find a dataset

AnnotateDataset.info

Find a dataset List annotations Log in / my account

Find your dataset

Dataset name Author name Added after Added before Dataset size

Search Add new 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 4.6: Find a dataset figma wireframe

In Figure 4.6, we have a sketch or 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

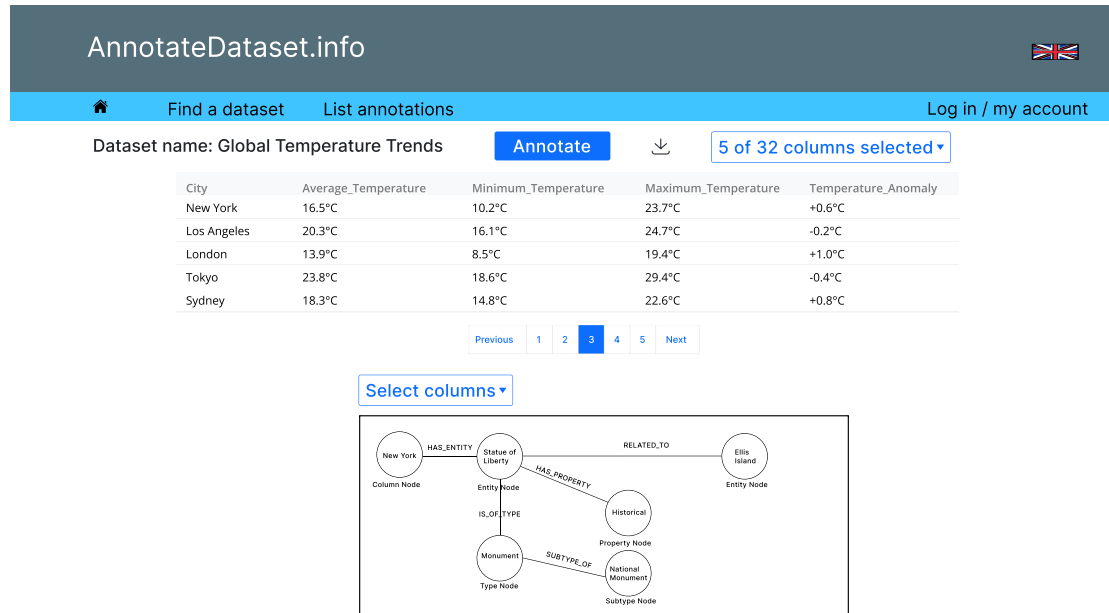


Figure 4.7: Selected dataset figma wireframe

In Figure 4.7, 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

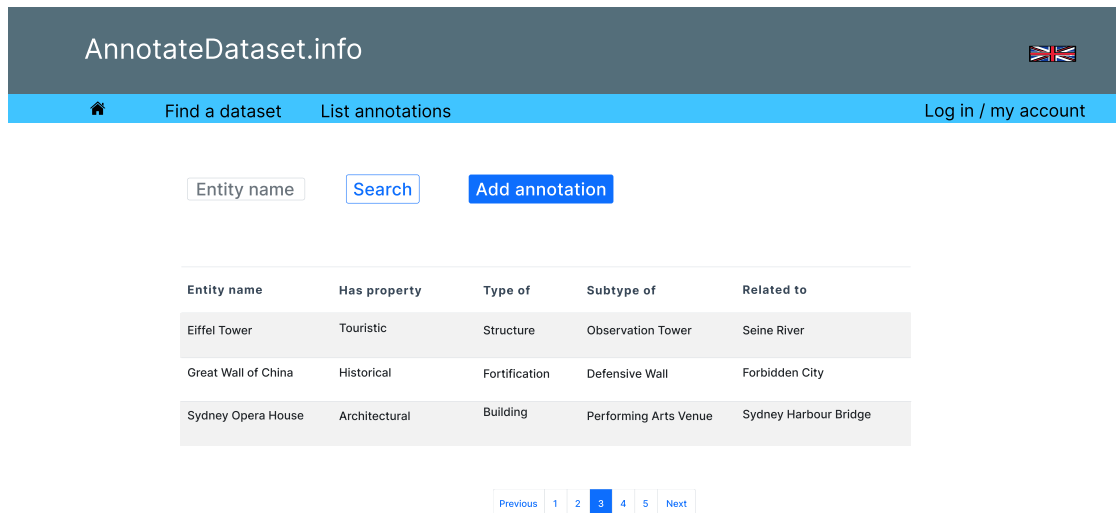


Figure 4.8: Find annotation figma wireframe

In Figure 4.8, 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

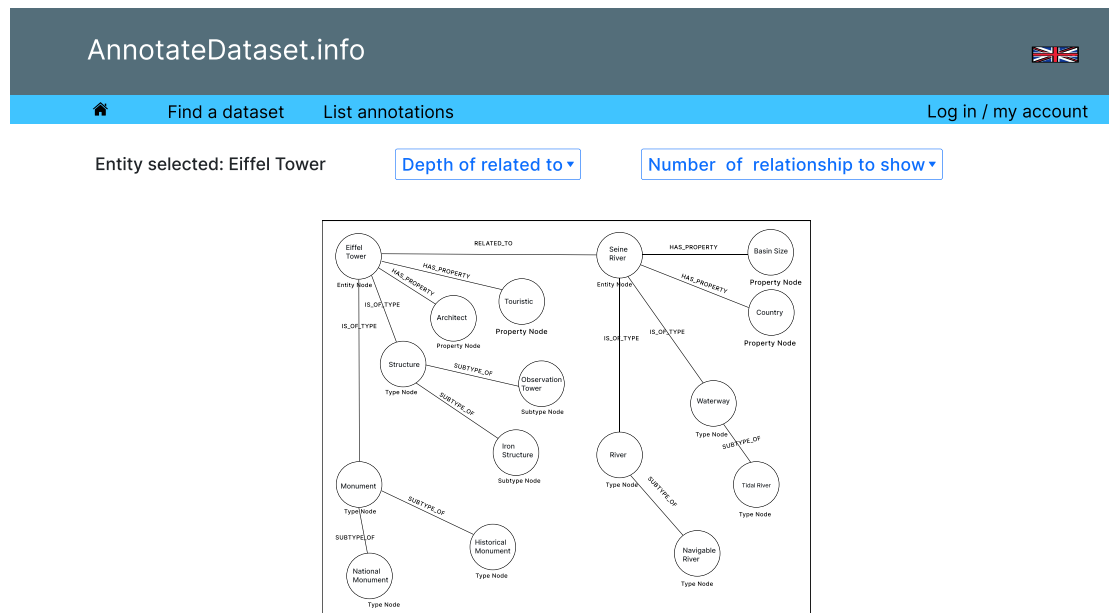


Figure 4.9: Selected annotation figma wireframe

Figure 4.9 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.

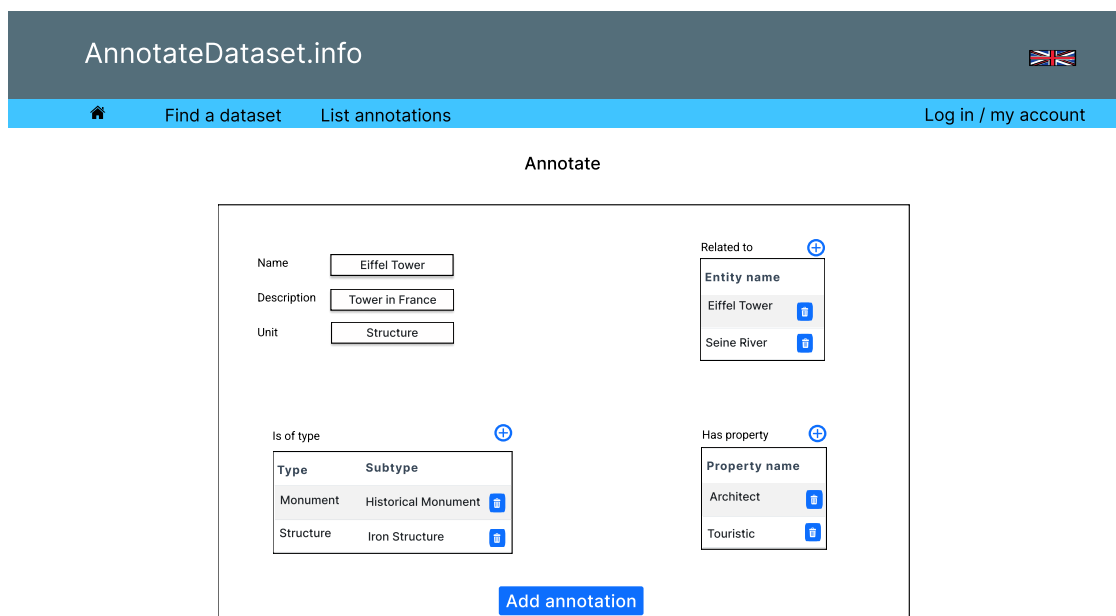
An essential component of this page is the 'depth of related-to' feature. This interactive mechanism allows users to adjust the level of relationships they wish to visualize, extending from the selected entity.

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



The wireframe shows a web page titled "AnnotateDataset.info" with a navigation bar containing "Find a dataset", "List annotations", and "Log in / my account". The main section is titled "Annotate" and contains a form for adding a new annotation. The form is divided into several sections:

- Name:** A text input field containing "Eiffel Tower".
- Description:** A text input field containing "Tower in France".
- Unit:** A text input field containing "Structure".
- Related to:** A section with a dropdown arrow, containing a list of entities: "Eiffel Tower" and "Seine River", each with a small blue icon.
- Is of type:** A section with a dropdown arrow, containing a table with two columns: "Type" and "Subtype". The table has two rows: "Monument" with "Historical Monument" and "Structure" with "Iron Structure". Each row has a small blue icon.
- Has property:** A section with a dropdown arrow, containing a table with two columns: "Property name" and "Value". The table has two rows: "Architect" and "Touristic", each with a small blue icon.

At the bottom of the form is a blue button labeled "Add annotation".

Figure 4.10: Add annotation figma wireframe

In Figure 4.10, 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.

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] URL: <https://www.indeed.com/career-advice/career-development/static-vs-dynamic-website>.
- [5] *Static vs dynamic websites: Key differences*. Nov. 2022. URL: <https://www.pluralsight.com/blog/creative-professional/static-dynamic-websites-theres-difference>.
- [6] *What is open source?* URL: <https://www.redhat.com/en/topics/open-source/what-is-open-source>.
- [7] *The open source definition*. Feb. 2023. URL: <https://opensource.org/osd/>.
- [8] 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/>.
- [9] 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.

- [10] David Gourley et al. *HTTP: the definitive guide*. Ö'Reilly Media, Inc.", 2002.
- [11] 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>.
- [12] 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.
- [13] 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.
- [14] auth0.com. *JSON web tokens introduction*. URL: <https://jwt.io/introduction>.
- [15] *Stack overflow developer survey 2022*. Jún 2022.
- [16] Dasari Hermitha Curie et al. „Analysis on Web Frameworks“. In: *Journal of Physics: Conference Series*. Zv. 1362. 1. IOP Publishing. 2019, s. 012114.
- [17] Chuck Musciano a Bill Kennedy. *HTML & XHTML: The Definitive Guide: The Definitive Guide*. Ö'Reilly Media, Inc.", 2002.
- [18] Jennifer Kyrnin. „What is HTML 5“. In: *Saatavissa: http://webdesign. about.com/od/html5/qt/what_is_html5. htm [viitattu 2.2. 2011]* ().
- [19] Eric A Meyer. *CSS: The Definitive Guide: The Definitive Guide*. Ö'Reilly Media, Inc.", 2006.
- [20] Paul Wilton. *Beginning JavaScript*. John Wiley & Sons, 2004.
- [21] Remo H Jansen. *Learning TypeScript*. Packt Publishing Ltd, 2015.
- [22] Basarat Syed. *Beginning Node. js*. Apress, 2014.
- [23] Elar Saks. „JavaScript Frameworks: Angular vs React vs Vue.“ In: (2019).

- [24] Michael Roberts a John Chapin. *What is Serverless?* O'Reilly Media, Incorporated, 2017.
- [25] 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.
- [26] 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/>.
- [27] 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>.
- [28] Andrew Pavlo a Matthew Aslett. „What’s really new with NewSQL?“ In: *ACM Sigmod Record* 45.2 (2016), s. 45–55.
- [29] 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).
- [30] *System properties comparison Amazon Neptune vs. Graphdb vs. Neo4j*. URL: <https://db-engines.com/en/system/Amazon+Neptune%5C%3BGraphDB%5C%3BNeo4j>.
- [31] Ross Harmes a Dustin Diaz. „The Adapter Pattern“. In: *Pro JavaScript Design Patterns* (2008), s. 149–158.
- [32] John Hunt a John Hunt. „Adapter Pattern“. In: *Scala Design Patterns: Patterns for Practical Reuse and Design* (2013), s. 169–181.
- [33] 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>.

- [34] Mustafa Ergen et al. „What is artificial intelligence? Technical considerations and future perception“. In: *Anatolian J. Cardiol* 22.2 (2019), s. 5–7.
- [35] 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).
- [36] Batta Mahesh. „Machine learning algorithms-a review“. In: *International Journal of Science and Research (IJSR).[Internet]* 9 (2020), s. 381–386.
- [37] Bing Liu a Bing Liu. *Supervised learning*. Springer, 2011.
- [38] Peter Dayan, Maneesh Sahani a Grégoire Deback. „Unsupervised learning“. In: *The MIT encyclopedia of the cognitive sciences* (1999), s. 857–859.
- [39] 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.
- [40] Neha Gupta et al. „Artificial neural network“. In: *Network and Complex Systems* 3.1 (2013), s. 24–28.
- [41] 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.
- [42] Zhihua Zhang a Zhihua Zhang. „Artificial neural network“. In: *Multivariate time series analysis in climate and environmental research* (2018), s. 1–35.
- [43] 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>.

- [44] Elizabeth D Liddy. „Natural language processing“. In: (2001).
- [45] KR1442 Chowdhary a KR Chowdhary. „Natural language processing“. In: *Fundamentals of artificial intelligence* (2020), s. 603–649.
- [46] Murray Shanahan. „Talking About Large Language Models“. In: *arXiv pre-print arXiv:2212.03551* (2022).
- [47] Christian Janiesch, Patrick Zschech a Kai Heinrich. „Machine learning and deep learning“. In: *Electronic Markets* 31.3 (2021), s. 685–695.
- [48] 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>.
- [49] 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.
- [50] 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.
- [51] *Welcome to schema.org*. URL: <https://schema.org/>.
- [52] *Your machine learning and Data Science Community*. Feb. 2010. URL: <https://www.kaggle.com/>.
- [53] Kenza Kellou-Menouer et al. „A survey on semantic schema discovery“. In: *The VLDB Journal* 31.4 (2022), s. 675–710.
- [54] Apr. 2023. URL: <https://research.com/software/mobile-vs-desktop-usage>.

Príloha A: Harmonogram práce v ls

Týždeň semestra	Plán aktivity
1.týždeň	Oboznámenie sa s témou
2.týždeň	Štúdium problému
3.týždeň	Čítanie článkov o téme a zber literatúry
4.týždeň	Čítanie článkov o téme a zber literatúry
5.týždeň	Čítanie článkov o téme a zber literatúry
6.týždeň	Čítanie článkov o téme a zber literatúry
7.týždeň	Písanie dokumentu
8.týždeň	Písanie dokumentu
9.týždeň	Písanie dokumentu
10.týždeň	Písanie dokumentu
11.týždeň	Písanie dokumentu
12.týždeň	Upravovanie finálneho dokumentu

A.1 Zhodnotenie

Zhodnotenie prvej časti diplomovej práce v letnom semestri

V letnom semestri som sa venoval prvým krokom v rámci svojho diplomového projektu. Cieľom tejto časti bolo dôkladne sa oboznámiť s témou a získať potrebné znalosti na riešenie problému. Nasledoval harmonogram aktivít, ktorý som postupne dodržiaval:

V prvom týždni som sa venoval oboznámeniu sa s témou. Preštudoval som relevantné informácie o probléme, jeho kontexte a cieľoch, ktoré som mal dosiahnuť.

Táto fáza bola dôležitá na získanie širšieho prehľadu a základného porozumenia problematike.

V druhom týždni som sa intenzívne venoval štúdiu problému. Podrobnejšie som analyzoval jeho podstatu, identifikoval hlavné výzvy a hlbkovo preskúmal súvisiace oblasti. Tento krok mi umožnil získať hlbšie znalosti a základ pre ďalšie postupy.

Týždne 3 až 6 boli zamerané na čítanie článkov o danej téme a zber literatúry. Dôkladne som preštudoval relevantné štúdie, vedecké články a publikácie z príslušných oblastí. Cieľom bolo získanie ucelenejšieho prehľadu o probléme a identifikácia najaktuálnejších poznatkov.

V siedmom týždni som sa venoval písaniu dokumentu. Na základe mojich preštudovaných materiálov a poznatkov som začal formulovať obsah diplomovej práce. Postupne som rozvíjal štruktúru a písal časti dokumentu, aby som postupoval logicky a zabezpečil súdržnosť a súvislosť textu.

Týždne 8 až 11 som venoval písaniu dokumentu a jeho postupnému rozširovaniu. V tejto fáze som zahrnul všetky relevantné informácie, analýzy a výsledky svojich štúdií do diplomovej práce. Snažil som sa dosiahnuť vyššiu úroveň podrobnej analýzy a jasne prezentovať svoje poznatky a výsledky.

V dvanástom týždni som sa venoval finálnemu upravovaniu dokumentu. Znovu som si prečítal celý text a vykonal potrebné korektúry a úpravy. Zabezpečil som, aby bol dokument kompletný, jasný a gramaticky správny.

Počas celej práce som pravidelne konzultoval so svojím školiteľom. Spoločne sme prechádzali môj postup, diskutovali o mojich výsledkoch a zhodnocovaní nedostatkov a prínosov mojej práce. Jeho spätná väzba mi poskytla usmernenie pre ďalšie kroky a pomohla mi lepšie orientovať sa v riešení problému.

Celkovo môžem konštatovať, že som úspešne splnil ciele stanovené v harmonograme. Oboznámil som sa s témou, analyzoval som problém, preštudoval relevantnú literatúru a napísal súvislý dokument. Tieto kroky mi poskytli pevný základ pre ďalšie etapy diplomovej práce a umožnili mi lepšie pochopiť a naplánovať riešenie problému, ktorý som si stanovil.