metadata

Protokol o kontrole originality

webprotokol

Kontrolovaná práca

Citácia	Percento*
Development of web application components for semantic annotation of datasets / autor Plevko Peter, Bc. - školiteľ Rác Miroslav, Ing. - oponent Khylenko Volodymyr, prof., Ing., PhD. - FIIT / UPAI (FIIT). - Bratislava, 2024 *plagID: 1823019 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	1,39% ⬜🟥🟥🟥🟥

* Číslo vyjadruje percentuálny podiel textu, ktorý má prekryv s indexom prác korpusu CRZP. Intervaly grafického zvýraznenia prekryvu sú nastavené na [0-20, 21-40, 41-60, 61-80, 81-100].

Zhoda v korpusoch: Korpus CRZP: 1,16% (180), Internet: 0,45% (5), Wiki: 0,00% (0), Slov-Lex: 0,00% (0)

Informácie o extrahovanom texte dodanom na kontrolu

Dĺžka extrahovaného textu v znakoch: 196784
Počet slov v slovníku (SK, CZ, EN, HU, DE): 16193
Súčet dĺžky slov v slovníku (SK, CZ, EN, HU, DE): 121201

Celkový počet slov textu: 26955
Pomer počtu slovníkových slov: 60,1%
Pomer dĺžky slovníkových slov: 61,6%

Interval	100%-70%	70%-60%	60%-50%	40%-30%	30%-0%
Vplyv na KO*	žiadny	malý	stredný	veľký	zásadný

* Kontrola originality je výrazne oplyvnená kvalitou dodaného textu. Slovníkový test vyjadruje mieru zhody slov kontrolovanej práce so slovníkom referenčných slov podporovaných jazykov. Nízka zhoda môže byť spôsobená: nepodporovaný jazyk, chyba prevodu PDF alebo úmyselná manipulácia textu. Text práce na vizuálnu kontrolu je na konci protokolu.

Početnosť slov - histogram

Dĺžka slova	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Indik. odchylka	=	>>	=	=	=	=	<<	=	=	=	=	=	=	<<	=	=	=	=	=	=	=	=	=

* Odchýlky od priemerných hodnôt početnosti slov. Profil početností slov je počítaný pre korpus slovenských prác. Značka ">>" indikuje výrazne viac slov danej dĺžky ako priemer a značka "<<" výrazne menej slov danej dĺžky ako priemer. Výrazné odchylky môžu indikovať manipuláciu textu. Je potrebné skontrolovať "plaintext" ! Priveľa krátkych slov indikuje vkladanie oddelovačov, alebo znakov netradičného kódovania. Privela dlhých slov indikuje vkladanie bielych znakov, prípadne iný jazyk práce.

Práce s nadprahovou hodnotou podobnosti

Dok.	Citácia	Percento*
1	**Generating sequence diagrams from code** / autor Delinčák Matej, Bc. - školiteľ Lang Ján, doc., Ing., PhD. - oponent Dakić Pavle, Mgr. - FIIT / UISI (FIIT). - Bratislava, 2024 *plagID: 1822283 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	0,42% ⬜🟥🟥🟥🟥
2	**Application of federated learning in classification of medical images** / autor Slonskyi Roman, Bc. - školiteľ Kavec Martin, Ing., PhD. - oponent Ahmadzai Mirwais - FIIT / UPAI (FIIT). - Bratislava, 2023 *plagID: 1784084 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	0,42% ⬜🟥🟥🟥🟥
3	**Segmentation and Classification of Histopathological Images using Deep Learning** / autor Franczel Michal, Bc. - školiteľ Hudec Lukáš, Ing., PhD. - oponent Jakab Marek, Ing., PhD. - FIIT / UPAI (FIIT). - Bratislava, 2023 *plagID: 1784478 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	0,36% ⬜🟥🟥🟥🟥

4	**Automated segmentation in histopathology images using deep neural networks** / autor Zhan Soňa, Bc. - školiteľ Hudec Lukáš, Ing., PhD. - oponent Jakab Marek, Ing., PhD. - FIIT / UPAI (FIIT). - Bratislava, 2023 *plagID: 1784505 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	**0,36%** ▢▢▢▢🟥
5	**Cross-chain sharing unification on Polkadot Network** / autor Morháč Dušan, Bc. - školiteľ Valaštín Viktor, Ing. - oponent Ries Michal, doc., Ing. - FIIT / UPAI (FIIT). - Bratislava, 2024 *plagID: 1816424 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	**0,35%** ▢▢▢▢🟥
18	**Tracking temporal seasonality with intelligent methods** / autor Čambál Sebastián - školiteľ Nguyen Thu Giang, doc., Ing., PhD. - oponent Ahmadzai Mirwais - FIIT / UISI (FIIT). - Bratislava, 2023. - 40. s *plagID: 1785576 typ práce: bakalárska zdroj: STU.Bratislava*	**0,32%** ▢▢▢▢🟥
36	**Detekcia bežných objektov pomocou konvolučných neurónových sietí** / autor Válka Peter, Bc. - školiteľ Malík Peter, Ing., PhD. - oponent Jakab Marek, Ing. - FIIT / UPAI (FIIT). - Bratislava, 2019 *plagID: 1611983 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	**0,21%** ▢▢▢▢🟥
44	**http://www.ijarcsse.com/docs/papers/Volume_3/9_September2013/V3I9-0299.pdf** / Stiahnuté: 15.02.2015; Veľkosť: 23,73kB. *plagID: 15414894 zdroj: internet/intranet*	**0,19%** ▢▢▢▢🟥
97	**Sémantické vyhľadávanie v obsahu kultúrnych inštitúcií** / autor Habdák Martin, Bc. - školiteľ Andrejčíková Nadežda, Ing., PhD. - oponent Šimko Marián, Ing., PhD. - FIIT / UISI (FIIT). - Bratislava, 2013 *plagID: 1276707 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	**0,15%** ▢▢▢▢🟥
99	**https://dip.felk.cvut.cz/browse/pdfcache/trepktom_2012bach.pdf** / Stiahnuté: 05.11.2012; Veľkosť: 86,76kB. *plagID: 2920591 zdroj: internet/intranet*	**0,15%** ▢▢▢▢🟥
102	**Komponent pre editovanie textu s grafickými prvkami** / autor Mikuda Šimon, Bc. - školiteľ Drahoš Peter, Ing., PhD. - oponent Kapec Peter, Ing., PhD. - FIIT / UAI (FIIT). - Bratislava, 2014 *plagID: 1348384 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	**0,12%** ▢▢▢▢🟥
108	**Využitie frameworkov Apache Kafka a Spark pre spracovanie údajov a detekciu bezpečnostných incidentov** / autor Kukumberg Tomáš, Bc. - školiteľ Balogh Štefan, Ing., PhD. - oponent Kossaczký Igor, RNDr., CSc. - FEI / ÚIM (FEI). - Bratislava, 2023 *plagID: 1780839 typ práce: magisterská_inžinierska zdroj: STU.Bratislava*	**0,11%** ▢▢▢▢🟥

* Číslo vyjadruje percentuálny prekryv testovaného dokumentu len s dokumentom uvedeným v príslušnom riadku.

: Dokument má prekryv s veľkým počtom dokumentov. Zoznam dokumentov je krátený a usporiadaný podľa percenta zostupne. Celkový počet dokumentov je [**185**]. V prípade veľkého počtu je často príčinou zhoda v texte, ktorý je predpísaný pre daný typ práce (položky tabuliek, záhlavia, poďakovania). Vo výpise dokumentov sa preferujú dokumenty, ktoré do výsledku prinášajú nový odsek (teda dokumenty ktoré sú plne pokryté podobnosťami iných dokumentov sa v zozname nenachádzajú. Pri prekročení maxima počtu prezentovateľných dokumentov sa v zarážke zobrazuje znak ∞.

Detaily - zistené podobnosti

1. odsek : **spoľahlivosť [88%]**

Slovak University of Technology in Bratislava Ú[2»]stav počítačového inžinierstva a aplikovanej informatiky Faculty of Informatics and Information Technologies Academic year: 2023/2024 Reg. No.: FIIT-182905-103097 MASTER THESIS TOPIC Student: Student's ID: Study programme: Study field: Thesis supervisor: Head of department: Bc. Peter Plevko 103097 Intelligent Software Systems Computer Science Ing. Miroslav Rác Ing. Katarína Jelemenská, PhD. Topic: Development of web application components for semantic annotation of datasets Language of thesis: English Specification of Assignment: V oblasti dátovej vedy sa využíva štatistika, procesy a algoritmy na extrakciu poznatkov a znalostí zo štruktúrovaných aj neštruktúrovaných dát. Dátoví vedci potrebujú infraštruktúru na zjednodušen[«2]ie práce s da

2. odsek : **spoľahlivosť [72%]**

[1»] vyhodnoťte prostredníctvom testov. Deadline for submission of Master thesis: Approval of assignment of Master thesis: Assignment of Master thesis approved by: 17. 05. 2024 18. 04. 2024 prof. Ing. Vanda Benešová, PhD. – study programme supervisor I hereby declare that this thesis is the result of my own work, based on my own knowledge, consultations with my supervisor, and using the listed literature. Bratislava, 16. 5. 2024 .... Peter Plevko Acknowledgement My gratitude goes[«1] to Ing.

3. odsek : **spoľahlivosť [96% - 100%]**

[102»] scholarly[97»] pursuit genuinely fulfilling. Anotácia Slovenská technická univerzita v Bratislave FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ Študijný program: Inteligentné softvérové systémy Autor: Bc. Peter Plevko Diplomová práca: Vývoj súčastí webovej aplikácie na sémantickú anotáciu datasetov Vedúci[«102] diplomovej práce: Ing. Miroslav Rác 2024, Máj Táto práca opisuje vývoj[«97] webového

4. odsek : **spoľahlivosť [96%]**

[36»] do GCP a anotovaní suboru udajov. Annotation Slovak University of Technology Bratislava FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES Course: Intelligent Software Systems Author: Bc. Peter Plevko Master's thesis: Development of web application components for semantic annotation[«36]

5. odsek : **spoľahlivosť [99%]**

[99»] Access Layer Front-end Presentation Layer API Application Programming Interface HTML HyperText Markup Language CSS Cascading Style Sheets DNS Domain Name system HTTP Hypertext Transfer Protocol HTTPS Hypertext Transfer Protocol Secure JS Javascript JSON JavaScript Object Notation URL Uniform Resource Locator OOP Object-Oriented Programming IP Internet[«99] Protocol SSL

6. odsek : **spoľahlivosť [94%]**

[44»] represents a notable example of a fully connected, three-layer feedforward network, which includes 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 that encompass function approximation, time series prediction, and system control [47]. Feedforward neural networks (FNNs) are renowned for their exceptional performance[«44] in

7. odsek : **spoľahlivosť [82%]**

[108»] / what-is-a-framework-in-programming/. [13] Babak Bashari Rad, Harrison John Bhatti a Mohammad Ahmadi. „An introduction to docker and analysis of its performance“. In: International Journal of Computer Science and Network Security (IJCSNS) 17.3 (2017), s. 228. [14] David Gourley et al. HTTP: the definitive guide. O'Reilly [«108]

8. odsek :	**spoľahlivosť [67%]**
[18»] Reuse and Design (2013), s. 169–181. [40] Oludare Isaac Abiodun et al. „State-of-the-art in artificial neural network applications: A survey“. In: Heliyon 4.11 (2018), e00938. issn: 2405-8440. doi: https://doi.org/10.1016/j.heliyon.2018.e00938. url: https: //www.sciencedirect.com/science/article/pii/S2405844018332067. [41] Mustafa Ergen et al. „What is artificial intelligence?**[«18]** Technical	

Plain text dokumentu na kontrolu

Skontroluje extrahovaný text práce na konci protokolu! Plain text (čistý text - extrahovaný text) dokumentu je základom pre textový analyzátor. Tento text môže byť poškodený úmyselne (vkladaním znakov, používaním neštandardných znakových sád, ...) alebo neúmyselne (napr. pri konverzii na PDF nekvalitným programom). Nepoškodený text je čitateľný, slová sú správne oddelené, diakritické znaky sú správne, množstvo textu je primeraný rozsahu práce. Pri podozrení na poškodený text (väčšieho rozsahu), je potrebné prácu na kontrolu originality zaslať opakovane pod rovnakým CRZPID.

[2»]Slovak University of Technology in Bratislava Ústav počítačového inžinierstva a aplikovanej informatiky
Faculty of Informatics and Information Technologies Academic year: 2023/2024 Reg. No.: FIIT-182905-103097
MASTER THESIS TOPIC
Student: Student's ID: Study programme: Study field: Thesis supervisor: Head of department:
Bc. Peter Plevko
103097 Intelligent Software Systems Computer Science Ing. Miroslav Rác Ing. Katarína Jelemenská, PhD.
Topic: Development of web application components for semantic annotation of datasets
Language of thesis: English
Specification of Assignment:
V oblasti dátovej vedy sa využíva štatistika, procesy[«2]a algoritmy na extrakciu poznatkov a znalostí zo štruktúrovaných aj neštruktúrovaných dát. Dátoví vedci potrebujú infraštruktúru na zjednodušenie práce s datasetmi a vyhodnocovania modelov umelej inteligencie. Ich potreby zahŕňajú uchovávanie, organizovanie a anotáciu datasetov, a tiež ich distribúciu cez sieť. Cieľom dátovej vedy je zjednotiť štatistiku, informatiku a súvisiace metódy s cieľom pochopiť a analyzovať skutočné javy pomocou dát. Cieľom práce je vytvoriť webové prostredie, ktoré bude podporovať pridávanie, anotáciu a odosielanie súboru dát na požiadanie, pričom výsledný webový nástroj umožní efektívne pracovať s dátami a bude podporovať vyhodnocovanie modelov umelej inteligencie. Analýzou možností vkladania, odosielania a anotáciou dát, navrhnite vhodnú metódu a formát pre anotáciu súboru dát a ich následné odosielanie. Vytvorte databázový model pre ukladanie týchto dát a umožnite používateľovi pozrieť sa na vytvorenú reprezentáciu dát a na vzťahy medzi nimi. Implementované webové prostredie by malo umožňovať pracovať s dátami vo forme zrozumiteľnej pre človeka (user interface) aj pre stroj (API). Navrhnuté riešenie, ktoré bude schopné na základe analytických vstupov ukladať a odosielať anotovaný súbor dát, implementujte ako online verziu. Samotnú funkčnosť a korektnosť webového nástroja overte a[1»]vyhodnoťte prostredníctvom testov.
Deadline for submission of Master thesis: Approval of assignment of Master thesis: Assignment of Master thesis approved by:
17. 05. 2024 18. 04. 2024 prof. Ing. Vanda Benešová, PhD. – study programme supervisor
I hereby declare that this thesis is the result of my own work, based on my own knowledge, consultations with my supervisor, and using the listed literature.
Bratislava, 16. 5. 2024
.... Peter Plevko
Acknowledgement
My gratitude goes[«1]to Ing. Miroslav Rác, my thesis advisor, for his exceptional mentorship during my master's thesis journey. His constant availability, even for minor inquiries, significantly enhanced my work. I value his proficiency and his genuine desire for my success.
I would also like to express my gratitude to my classmates for their assistance and motivation. Our collaboration was effective and we mutually improved our academic performance. I appreciate their companionship and our shared determination to excel in our studies.
I wish to express my profound gratitude to my family for their persistent backing. Their motivation and steadfast faith in me have been my propelling force. I consider myself extremely lucky to have such a supportive and affectionate family by my side.
Lastly, I want to give a shoutout to my buddy and roommate, Matej Delinčák. He has been a rock for me, especially when things get tough. His positive vibes always lift me up and I really appreciate our friendship.
To all those acknowledged, my deepest thanks. Your support and encouragement were indispensable to my success. You have made my[102»] scholarly[97»]pursuit genuinely fulfilling.
Anotácia
Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program:
Inteligentné softvérové systémy
Autor:
Bc. Peter Plevko
Diplomová práca:
Vývoj súčastí webovej aplikácie na sémantickú
anotáciu datasetov
Vedúci[«102]diplomovej práce: Ing. Miroslav Rác
2024, Máj
Táto práca opisuje vývoj[«97]webového nástroja, ktorého cieľom je uľahčiť správu, anotáciu a nahrávanie súborov údajov so zameraním na podporu efektívneho spracovania údajov a hodnotenia modelov umelej inteligencie. Prostredníctvom dôkladnej analýzy boli navrhnuté vhodné metódy a formáty na anotovanie súborov údajov spolu s vytvorením databázového modelu na ukladanie a vizualizáciu vzťahov medzi údajmi. Aplikácia je užívateľsky prívetivá a vyhovuje potrebám ľudských používateľov cez svoje rozhranie, ako aj strojovým interakciám prostredníctvom rozhrania API. Medzi kľúčové použité technológie patrí Vue.js na vývoj frontendu, NestJS na back-end a integrácia databáz Neo4j a PostgreSQL a využitie platformy Google Cloud Platform (GCP). Fáza návrhu zahŕňala vytvorenie vývojových diagramov, wireframov a databázových diagramov na zmapovanie štruktúry aplikácie. Pri prechode do fázy implementácie bola vyvinutá celá aplikácia. Hodnotenie zahŕňalo analýzu optimálnych oblastí nasadenia,

porovnanie monolitickej a mikroslužbovej architektúry a posúdenie výkonnosti databáz Neo4j a PostgreSQL. Výsledná aplikácia bola úspešne nasadená, splnila ciele diplomovej práce a ponúkla cenné poznatky o nasadení[36»]do GCP a anotovaní suboru udajov.
Annotation
Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Course:
Intelligent Software Systems
Author:
Bc. Peter Plevko
Master's thesis: Development of web application components for semantic
annotation[«36]of datasets
Supervisor:
Ing. Miroslav Rác
2024, May

This thesis outlines the development of a web-based tool that aims to facilitate data set management, annotation, and upload, with a focus on supporting efficient data handling and AI model evaluation. Through careful analysis, appropriate methods and formats were devised to annotate datasets, alongside the creation of a database model to store and visualize data relationships. The application was designed to be user-friendly, catering to both human users through its interface and machine interactions through APIs. The key technologies used include Vue.js for frontend development, NestJS for back-end, and integration of Neo4j and PostgreSQL databases and the use of the Google Cloud Platform (GCP). The design phase involved creating flow diagrams, wireframes, and database diagrams to map out the application structure. In transitioning to the implementation phase, the entire application was developed. The evaluation included analysis of optimal deployment regions, comparison of monolithic and microservice architectures,
and performance assessments of the Neo4j and PostgreSQL databases. The resulting application has been successfully deployed, meeting the thesis objectives and offering valuable insights into deployment to GCP and dataset annotation.

Contents

Abbreviations list
SQL Structured Query Language
Back-end Data**[99»]**Access Layer
Front-end Presentation Layer
API Application Programming Interface
HTML HyperText Markup Language
CSS Cascading Style Sheets
DNS
Domain Name system
HTTP Hypertext Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure
JS Javascript
JSON
JavaScript Object Notation
URL
Uniform Resource Locator
OOP
Object-Oriented Programming
IP Internet**[«99]**Protocol
SSL Secure Sockets Layer
AJAX
Asynchronous JavaScript and XML
xv
1 Introduction
Data science has emerged as a crucial field that encompasses 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 data sets becomes imperative for their work. In addition, data science seeks to integrate statistics, computer science, and related methodologies to understand and analyze real-world phenomena through data.
This work aims to develop a web-based environment specifically tailored to support the needs of data scientists. The environment will allow seamless addition, semantic annotation, and transmission of datasets, enabling efficient data manipulation and facilitating the evaluation of AI models. Semantic annotation enriches the data by adding descriptive text, clarifying its content, structure, and purpose. This aids data scientists in understanding and utilizing datasets for semantic analysis, which involves extracting meaning and relationships within data for various applications.
Through a comprehensive analysis of various methods and formats for data insertion, transmission, and annotation, this study will determine the most appropriate approach. In addition, 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
1
a user-friendly interface and 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 use in their daily work. When we familiarize ourselves with these concepts, we can design a tool that is in line with their requirements. In the following 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. In addition, 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 seeks 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 work, our goal is to facilitate improved data manipulation, support the evaluation of AI models, and ultimately empower data scientists in their pursuit of knowledge extraction and analysis from diverse datasets.
2
2 Analysis
Our website serves as a comprehensive knowledge base specifically tailored 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 after each user-annotated data set. For example, 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 preexisting annotations when annotating tables with similar names. However, if a required value has not been previously added, the users have the freedom to contribute and introduce new annotations. This approach showcases a semiautomatic nature, as it necessitates user input while effectively managing the storage aspect. It is important to note that complete automation of annotation creation remains an ongoing
challenge in the field [2].
Knowledge engineering, an integral field of artificial intelligence (AI), seeks to emulate the judgment and reasoning capabilities of human experts within specific domains [3]. Our platform offers a variety of APIs that enable data researchers to access pertinent data for training AI models or evaluating their performance after training. Activating in a passive manner, our website requires users to initiate API requests to provide the desired information.
Developing an efficient and robust web page prompts critical considerations surrounding the selection of programming languages for both the client and server
3
components, the identification of an optimal framework, architectural decisions between server-based or serverless models, the choice between relational, nonrelational, or graph databases, and determining the ideal format for transmitting annotated dataset metadata. In the subsequent sections, we will dive into the possibilities surrounding these questions while offering academic perspectives and insights.
2.1 Knowledge base
A knowledge base is a centralized repository of information that can be accessed and used by individuals or systems. It can contain a wide range of information, including facts, rules, procedures, and best practices. Knowledge bases can be used in various fields, such as computer science, medicine, and economics, to support decision making, problem solving, and learning. Examples of knowledge bases include expert systems, databases, and Wikis [4].
2.2 Knowledge graph
A knowledge graph is a graph-structured data model used to store descriptions of entities such as people, places, and events. It can be defined as a collection of ordered triples (h, r, t), where h and t are the head and tail entities, and r is the relation between them. Knowledge graphs play an important and growing role in semantics-based support of a wide variety of applications, and are useful for tasks such as answering and reasoning questions [5]. They are also being used in the Solid ecosystem to improve pod access through different Web APIs that act as views on the knowledge graph, providing improved opportunities for storage, publication, and querying of decentralized data in more flexible and sustainable
4
ways [6]. Knowledge graph management has evolved, and there are now KG management platforms that support the lifecycle of KGs from their creation to their maintenance and use [7].
2.3 Web page
According to the information collected from the sources cited, a website represents a collection of interconnected web pages that are accessible through the Internet [8, 9]. There are two primary classifications of websites: static and dynamic. Static websites are constructed using “fixed code“ and employ languages such as HTML, JavaScript, and CSS [8]. The content of static web pages remains unchanged until manual modifications are made. In contrast, dynamic websites offer interactivity and are implemented using languages such as CGI, AJAX, ASP, and ASP.NET [8]. Dynamic Web pages exhibit variability in content depending on visitor interactions, though at the expense of increased loading times compared to static Web pages. This dynamism is particularly useful in scenarios where frequent information updates are required, such as when displaying real-time stock prices or weather information.
2.4 Open source
As per the information provided by sources such as [10, 11], the concept of open source refers to computer software that is released under a licensing framework, granting users specific rights to use, study, modify, and distribute software and its corresponding source code. This licensing approach enables individuals to access and use the software freely, without restrictions, while fostering an environment of collaboration and community-driven development. Open source software often
5
undergoes public and collaborative development processes, involving multiple contributors working together to enhance its functionality and address issues.
In addition to technical aspects, the term “open source“ encompasses a broader set of values that promote principles such as open exchange, participatory collaboration, transparent development, rapid prototyping, meritocracy and communityoriented participation. These values support an inclusive and collaborative environment in which individuals can openly contribute to the improvement and evolution of the software.
2.5 Framework
In figure 2.1 we can see a brief description of what a framework is, but let us look at it in more detail. 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 low-level standard functionality, allowing developers to focus on the distinctive aspects of their projects. The underlying principle that governs frameworks is 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 improving reliability, accelerating 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 that provides developers with access to a community of fellow practitioners who can provide guidance and assis-
6

tance. This collective knowledge pool can prove invaluable when facing challenges during the development process. Additionally, frameworks often incorporate security measures and best practices, ensuring that developers can build applications with robust security foundations. In general, the adoption of frameworks contributes to improved efficiency, reduced development time, enhanced software reliability, and cost savings [12]. Recognizing the significance of frameworks in programming can help make informed decisions when selecting the appropriate framework for developing a knowledge base website that targets data scientists.
Figure 2.1: Software Framework [12]
2.6 Docker
Docker, an open source containerization platform, has gained widespread recognition as a powerful tool for packaging applications into portable and self-contained units known as containers. These containers encapsulate the application's source code, along with the necessary operating system libraries and dependencies, enab-
7
ling consistent and reliable execution across diverse computing environments. The rise in popularity of cloud systems has further propelled Docker's adoption, as it facilitates the seamless deployment and execution of applications on various host machines [13].
One of Docker's key advantages lies in its ability to ensure application consistency and portability. By encapsulating the application and its dependencies within a container, Docker enables developers to build and test applications in isolation, independent of the underlying host system. This containerized approach provides a standardized environment, eliminating compatibility issues, and enabling the application to run consistently across different computing environments.
Furthermore, Docker simplifies the process of sharing and deploying applications. The containerized nature of Docker allows developers to package an application along with its dependencies into a single artifact, ensuring that the application runs reliably and predictably on any compatible Docker host. This portability significantly reduces the complexities associated with configuring and setting up an application on different computers, enhancing development efficiency and facilitating seamless collaboration.
Moreover, Docker promotes scalability and efficiency through its lightweight and resource-efficient containerization model. Containers provide a lightweight alternative to traditional virtualization, enabling faster start 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.
8
2.7 Security
2.7.1 HTTP
As described in the book "HTTP: The Definitive Guide" [14], HTTP serves as the universal language for communication between web browsers, servers, and related web applications on the Internet. It facilitates the reliable transmission of web content from servers to clients, ensuring the integrity of the data exchanged. Web content is stored on HTTP servers, which respond to requests from HTTP clients by providing the requested data along with relevant information such as object type and length. This interaction between HTTP clients and servers forms the fundamental basis of the World Wide Web.
Building on the foundation of HTTP, HTTPS (HyperText Transfer Protocol Secure) serves as an enhanced and more secure version, as indicated in the reference articles [15, 16]. It operates as an application-specific implementation that encapsulates HTTP within SSL (Secure Sockets Layer) or TLS (Transport Layer Security) protocols. HTTPS plays a vital role in ensuring secure data transmission over the Internet. By encrypting data exchanged between servers and clients, HTTPS mitigates the risk of interception and unauthorized access by malicious entities. It serves as a crucial method for securing web servers and ensuring network security.
A comprehensive analysis of HTTPS deployments and associated challenges was conducted, as mentioned in one of the cited articles [15]. This analysis highlights the effectiveness of HTTPS in establishing end-to-end secure connections, even when modifications are introduced during the communication between clients and web servers. The use of HTTPS in the design of the webpage in question will significantly enhance its security measures.
9
2.7.2 Login
Authentication is a fundamental aspect of user login in web applications, serving the purpose of verifying the claimed identity of a user. There are three primary types of authentication: single-factor, two-factor, and three-factor authentication [17].
Single-factor authentication, which relies on knowledge-based factors, is commonly used in web applications. This method typically involves the use of a username and password combination for user verification.
The second type, two-factor authentication, requires the user to provide two independent pieces of information for authentication. For example, 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, requires 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. Ensures that each user type is assigned appropriate access privileges according to their role or level of authorization [17].
In our proposed implementation, we will incorporate a model that encompasses authentication through single-factor authentication using a username and password. Additionally, we will address authorization by assigning different access levels and permissions to distinct user types within the application.
10
2.7.3 User types
The application will cater to three distinct user types, each possessing varying levels of authority and privileges. The most common user category is the "normal user", while the least common is the "admin". Furthermore, users will be categorized as either "registered" or "unregistered."
Registration will grant access to additional features, but it is essential to acknowledge that the requirement to register can discourage 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 content, 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 non-sensical data sets.
The "user" category enjoys the same privileges as a visitor but gains the additional capability to add data sets 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, which grants them complete 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 the corresponding privileges to ensure a structured and controlled environment within the application, promoting user engagement and content quality.
11
2.7.4 JSON web token
All the information provided in this section was taken from the jwt.io website [18]. JSON Web Token (JWT) is a widely adopted open standard (RFC 7519) that establishes a way to securely transfer data between parties using a JSON object. These data can be trusted because it is digitally signed, either by a secret key using the HMAC algorithm or a public/private key using RSA or ECDSA.
Generally, 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 to securely transfer 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 was not 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 is valid, access is granted. It is 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.
12
2.8 Frontend
According to the Stack Overflow survey conducted in 2022, the prevailing languages utilized in web development include JavaScript, HTML/CSS, Python, Java, C#, and PHP [19]. These languages are widely employed by developers to create robust and interactive user interfaces for web applications. The incorporation of frameworks further enhances development efficiency by providing libraries and pre-built components that reduce the amount of code required.
Frameworks offer numerous advantages, particularly in simplifying the implementation of common tasks encountered during web application development. Tasks such as form validation, data sanitization, and CRUD operations often involve repetitive processes. Instead of reinventing solutions for these tasks, frameworks provide developers with pre-existing functions and modules specifically designed to handle them, streamlining the development process. Prominent frameworks widely used in the industry include PHP Laravel, Symfony, JavaScript React, Vue.js, Python Django, and others [20].
The front-end of a web application typically comprises three primary languages: HTML, CSS, and JavaScript. Each language serves a distinct purpose in shaping the visual and interactive aspects of the user interface. HTML (HyperText Markup Language) defines the structure and organization of web content, specifying the elements and layout of the page. CSS (Cascading Style Sheets) governs the presentation and styling of HTML elements, controlling visual aspects such as color, typography, and layout. JavaScript, on the other hand, adds interactivity and dynamic behavior to the web page, facilitating client-side scripting and enhancing user engagement.
Considering the widespread popularity of JavaScript, employing it as the primary
13
client-side language is a good choice. JavaScript boasts a substantial user base, ample resources available on platforms like Stack Overflow, and a plethora of JavaScript implementations and libraries. These factors contribute to a vibrant development ecosystem that supports the language’s continuous growth and adaptation to evolving industry needs.
2.8.1 HTML
HTML, an acronym for Hypertext Markup Language, serves as a fundamental markup language extensively employed in crafting web pages and various forms of online content for rendering in web browsers. It operates by using 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. Currently, the World Wide Web Consortium (W3C) assumes 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 HTML page on the user’s screen [21]. The accessibility and ease of use associated with HTML contribute to its popularity and widespread adoption, enabling a diverse range of people to engage with the language. Moreover, HTML offers significant customizability, enabling developers to design web pages that exhibit a broad spectrum of visual styles and functional attributes.
14
In recent times, HTML has undergone notable advances, 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 such as WebGL and WebSockets, facilitating the development of immersive web applications and captivating games [22]. The continuous evolution of HTML reinforces its pivotal role in shaping the modern Web and underpins its enduring significance in facilitating captivating and interactive online experiences.
2.8.2 CSS
During the early stages of the web, HTML faced limitations in terms of structural markup, leading to the introduction of presentational elements like and <BIG> to cater to the growing demand for new elements. However, this approach resulted in a conflation of structural and presentational concerns that impeded effective content indexing, accessibility, and maintainability. Consequently, a solution that would disentangle presentation from structure would be necessary, allowing greater flexibility and sophistication in styling. As a response to these challenges, Cascading Style Sheets (CSS) were developed.

CSS was designed to separate the presentation layer from the structural markup, providing improved control over styling. Unlike HTML, CSS offers a comprehensive range of styling options that include the ability to define colors, create borders, regulate spacing, and incorporate various other features. By employing CSS for presentation control, developers are empowered to structure content in a manner that optimizes accessibility and indexing while simultaneously achieving the desired visual effects [23].
The introduction of CSS brought about a paradigm shift in web design, fostering
15
improved modularity, maintainability, and extensibility. By decoupling presentation concerns from the underlying structure, CSS facilitates a more structured and sustainable approach to web development, ensuring the longevity and adaptability of web content.
2.8.3 JavaScript
JavaScript is a versatile programming language utilized to provide instructions for various computational tasks that are processed sequentially by computers. As an interpreted language, JavaScript requires 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 context of the Web, web servers store web pages identified by IP addresses, while domain name servers perform the crucial task of converting these addresses into more human-readable domain names [24].
TypeScript, on the other hand, is a statically typed programming language that incorporates static type checking during compilation to mitigate potential run-time 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 TypeScript type-checking capabilities operate exclusively during design time, ensuring that no additional runtime overhead is incurred. The resulting code is highly compatible with web browsers and can target ECMAScript 3, 5, and 6. TypeScript is designed to align with existing and future
16
ECMAScript proposals and is freely available as a cross-platform development tool under the open source Apache license [25].
Node.js, in turn, empowers developers to execute JavaScript outside the confines of web browsers. Within the Node.js environment, all JavaScript code is processed by the V8 engine, originally developed for the Google Chrome browser. The widespread adoption of Google Chrome reflects the importance of V8 in web development, due to its exceptional speed and seamless integration between platforms [26].
The adoption of frameworks has become indispensable in modern JavaScript development. In particular, the primary frameworks in the JavaScript ecosystem encompass Angular, React, and Vue. By comparing these frameworks, a study conducted by Elar Saks [27] provides valuable information. 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 option, which makes it advantageous for job seekers. Vue shines in terms of simplicity and performance, and Angular excels in building large-scale applications.
It should be noted that Angular and React default to TypeScript, whereas Vue grants users the flexibility to choose their preferred language. Angular, React, and Vue are open-source frameworks, which ensure active community maintenance and offer unrestricted download, use, and modification rights.
Taking into account the comparison above, Angular emerges as the most suitable
17
framework for constructing large-scale applications. React, which is highly sought after by employers, is important for job seekers. Meanwhile, Vue stands out for its user-friendly learning curve and exceptional performance. However, the selection of an appropriate front-end framework should be guided by various factors, such as project scale, complexity, scalability, and team expertise. In my case, as a non-job seeker and considering the size and requirements of my webpage, Vue.js presents itself as the optimal choice.
Vue.js will be used together with the vitebuild tool. As said in [28]. Vite is a frontend development tool created by Evan You, the founder of Vue.js. Offers a faster and more efficient development experience for modern web projects. Key features include native ES module support, faster dev server startup, quick updates with Hot Module Replacement (HMR), and an optimized production build. Vite leverages Rollup's flexible plugin API for better performance and flexibility. It is compatible with Vue.js, TezJS, and React, and ongoing development efforts aim to enhance its performance further. In general, Vite stands out for its speed, streamlined workflow, and commitment to modern web development practices.
2.9 Backend
When deliberating on the architectural approach to developing a web application, two primary options come to the forefront: server-based architecture and serverless architecture. The serverless architecture can be classified into two domains: Backend as a Service (BaaS) and Functions as a Service (FaaS). BaaS involves transferring the back-end 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 the underlying infrastructure. It should be noted that BaaS encompasses the entirety of server
18
functionality, while FaaS focuses solely on event-driven functions. An inherent advantage of serverless architecture lies in its pay-per-use model, facilitating cost optimization. However, in the traditional serverless approach, charges may still apply for idle time [29].
On the contrary, server architecture refers to the conventional approach to building 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. This decision offers the advantages of streamlined development processes, reduced compatibility issues, and seamless data exchange between the front-end and back-end using JavaScript-based JSON. Furthermore, by employing NestJS, a framework constructed on top of Node.js and TypeScript, developers can build modular and scalable architectures. NestJS is tailored for efficient and scalable server-side applications, supporting TypeScript and incorporating elements of Object-Oriented Programming (OOP), Functional Programming (FP), and Functional Reactive Programming (FRP). It utilizes Express as the default HTTP server framework and can optionally be configured to use Fastify.
Nest abstracts the underlying frameworks while exposing their APIs, allowing developers to utilize third-party modules for enhanced adaptability[30].

Ultimately, the choice between server and serverless architecture is based on the specific requirements of the application at hand. However, serverless architecture presents several compelling advantages. First, it enables developers to primarily concentrate on application logic, obviating the need for extensive server-side management and configurations. Second, serverless computing promotes improved scalability by automatically allocating resources based on demand, culminating in improved performance. Moreover, serverless architecture has the potential to
19
produce cost savings and reduced latency compared to traditional server-based computing [31].
There are multiple serverless platforms, 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, including $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 along with 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. Taking into account the benefits and features offered by Google Cloud Functions, particularly its attractive free tier, it emerges as a suitable choice
for the present project [32, 33].
2.9.1 Database
When considering the selection of a database for the Web application, it is crucial to store data and their relationships efficiently. Various types of databases are available, including relational databases like PostgreSQL, graph databases such as Neo4j, nonrelational databases like MongoDB, and NewSQL databases like SurrealDB. NewSQL databases belong to the relational database class and aim to provide scalability comparable to non-relational databases while preserving ACID (Atomicity, Consistency, Isolation, Durability) properties [34]. ACID represents a set of principles that ensure reliable and robust data operations. Careful conside-
20
ration of the 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 such as Amazon Neptune and Microsoft Azure Cosmos DB [35].
Neo4j and Amazon Neptune are both graph databases suitable for knowledge graphs. Neo4j is a scalable ACID-compliant graph database that has been extensively tested for performance and scalability. It enables users to establish connections between text and data in large knowledge graphs. 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. On the contrary, Neo4j has a proven track record of delivering high performance and scalability [35, 36].
Ultimately, the choice between Neo4j and Amazon Neptune depends on the specific needs and use cases of the project. Neo4j can be deployed locally on a personal computer or on AWS using a Partner Solution developed by AWS and Neo4j. On the other hand, Amazon Neptune is a cloud service and does not offer local deployment. Taking into account these arguments, it is evident that Neo4j is the preferred choice, given its battle-tested performance and scalability and the flexibility of deployment options [35, 36].
In the deployment phase, Neo4j serves as the chosen graph database solution. The deployment options include setting up a virtual machine and installing Neo4j locally or opting for Neo4j Aura. As detailed in the source [37], Neo4j Aura stands out as Neo4j's fully managed cloud service. It provides a highly efficient, reliable,
21
and scalable graph database solution delivered entirely as a cloud service. Neo4j Aura is characterized by its exceptional speed, reliability, and scalability, offering a seamless and automated experience for managing graph databases in the cloud environment.
2.9.2 Adapter pattern
The adapter pattern, as described in Harmes et al. (2008), is a design pattern that facilitates the adaptation of incompatible interfaces between classes. This pattern involves the use of objects, often referred to as wrappers, that encapsulate another object within a new interface. Adapters play a crucial role for programmers and interface designers in situations where existing APIs cannot be used directly with certain interfaces. Using adapters, it becomes possible to incorporate these classes into the system without the need for direct modifications. In figure 2.2 we can see that illustrates the various components that need to be implemented to effectively utilize the adapter design pattern [38].
Figure 2.2: Adapter design pattern [39]
In the context of software engineering, design patterns like the adapter pattern serve as valuable tools to address interface compatibility challenges. They provide a structured and reusable approach to enable communication and interaction between classes that would otherwise be incompatible. Using 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 22 that can accommodate future changes in interface requirements [38].
In our project, currently utilizing Neo4j as the database system, the significance of the adapter design pattern becomes apparent. Anticipating potential shifts in the choice of databases or even a transition from the NestJS framework to another, the adapter design pattern emerges as a fitting solution. This pattern allows us to seamlessly switch between different frameworks while minimizing the need for extensive modifications to the existing codebase.
Consider the scenario where a change in the database system is contemplated, perhaps a transition to PostgreSQL or another database. In such cases, the repository design pattern proves to be a better match for handling database changes. However, in the broader context of adapting to various back-end modifications, including a change in the choice of databases or the framework, the adapter design pattern remains a robust solution.
The implementation of adapters acts as intermediaries between the application front-end and the back-end, shielding the front-end from the intricacies of changes at the back-end, such as alterations in the choice of databases.
The adapter design pattern introduces a layer of abstraction that enables the front-end to remain agnostic to the technologies employed in the back-end. This means that even if there are changes in the backend, be it the choice of database or the framework, the frontend can remain untouched, provided the API stays the same.

In the provided example 2.3, the front-end serves as the client, the back-end functions as the adapter, and the database acts as the adapter. The front-end communicates with the back-end through a well-defined API, maintaining loose coupling with the back-end and insulating itself from the underlying changes. The backend,
23
acting as an adapter, translates and adapts requests between the front end and the database, ensuring a seamless interaction. This modular and adaptable architecture is aligned with the principles of loose coupling and abstraction. The separation between the front-end, back-end, and database creates a system where changes to the back-end, such as a switch in databases or modifications in the internal implementation, do not necessitate alterations in the front-end as long as the API remains consistent.
Figure 2.3: Adapter design pattern in project
2.10 Data model
To create an effective data model, it is crucial to acquire a comprehensive understanding of the terminology 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 complex layers of neural networks. The key distinction between DL and other neural networks lies in its ability to accommodate a greater number of layers, neurons, and computational power, 24
facilitating the representation of complex models and automatic feature extraction. Therefore, DL is considered a subset of ML, which, in turn, falls under the broader umbrella of AI. Over time, the field of artificial intelligence has witnessed significant technological advancements, as demonstrated in figure 2.4. These advances have contributed to the development and adoption of sophisticated algorithms, increased computational capabilities, and the availability of vast amounts of data. This progress has propelled AI and ML to the forefront of various industries, enabling organizations to take advantage of data-driven insights for decision making, automation, and problem solving. [40]
Figure 2.4: Artificial intelligence development and expansion. [40]
To ensure the adequacy and effectiveness of our data model, it is advisable to actively engage with professionals and experts in the field of AI and Data Science. Collaborating with individuals specializing in these domains will provide invaluable guidance and expertise on 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 first-hand insights into best practices, and
25
make informed decisions when building our data model.
By incorporating fundamental AI concepts, exploring cutting-edge technologies, and tapping into the expertise of domain specialists, we can create a robust data model that not only adheres to the principles of AI and data science, but also enables accurate analysis, efficient processing, and the extraction of meaningful insights.
In the following 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, providing the reader with a comprehensive understanding of AI and its associated lexicon within a broader context.
2.10.1 Artificial intelligence
Artificial Intelligence (AI) represents a rapidly evolving technology that enables machines to participate in cognitive functions that include perception, reasoning, learning, and interaction. This transformative paradigm is expected to revolutionize industries across the world, with projected revenue increments exceeding ten trillion dollars. The realization of AI hinges on the orchestration of algorithms, large 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 is of paramount importance, as they form the bedrock upon which AI models are constructed. Mathematical constructs underpin the decisionmaking process in AI, involving the multiplication of incoming data with weighted vectors, thereby enabling informed judgments and responses. Over the years, the field of artificial intelligence has seen significant
breakthroughs, fostering advance-
26
ments in algorithms and nurturing the growth of this transformative technology [41].
The origins of AI can be traced back to the imaginative realms of philosophers and science fiction writers. The emergence of early AI research was fueled by notable developments, such as the introduction of “The Turk“, a chess-playing automaton, during the 18th and 19th centuries. This invention served as a catalyst, stimulating further exploration into the realms of artificial intelligence. 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 surprised the world by defeating the reigning Go champion, underscoring the remarkable progress achieved in the realm of AI [42].
It is imperative to recognize that AI’s transformative potential is contingent upon a multidisciplinary approach, engaging scholars and experts across diverse domains. A scholarly exploration of the underlying principles, advancements and historical context of AI, coupled with a steadfast commitment to ongoing research and collaboration, shall foster a comprehensive understanding of this important field. By embracing academic discourse and continuously monitoring the latest developments, stakeholders can position themselves at the vanguard of AI, capitalizing on
27
its immense potential to redefine industries and drive innovation.
2.10.2 Machine learning

Throughout history, humans have harnessed a diverse array of tools and machines to streamline tasks and fulfill various needs, spanning domains such as transportation, industrial processes, and computing. Within the realm of computing, machine learning has emerged as a pivotal field, enabling machines to acquire knowledge and enhance their performance without explicit programming. Coined by Arthur Samuel, machine learning involves equipping computers with the ability to learn from data and improve their efficiency in handling information. Given the abundance of datasets available today, the demand for machine learning methodologies has surged, with industries leveraging its power to extract meaningful insights from vast volumes of data. At the core of machine learning lies the objective of learning from data, which requires the use of various algorithms tailored to specific problem domains, the number of variables involved, and the optimal model for the given context [43].
Supervised learning serves as a prominent paradigm within machine learning, finding applications across various domains, including text mining and web applications. It revolves around leveraging past data to enhance performance in real-world tasks, with a primary focus on learning a function that predicts discrete class attributes. Supervised learning has witnessed extensive research and practical adoption, particularly in the realm of web mining applications [44].
In contrast, unsupervised learning represents a methodology in which 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
28
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, in which statistical models are constructed, and feature extraction, which directly extracts statistical regularities from input data [45].
Furthermore, semisupervised learning presents a viable approach that leverages both labeled and unlabeled data to train models. Particularly in scenarios involving large datasets, the process of labeling data, i.e., assigning outcomes to instances, often proves time-consuming and resource-intensive. In an effort to enhance model performance, semi-supervised 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 selection of models remains paramount in achieving optimal results within the realm of semi-supervised learning [46].
2.10.3 Artificial neural networks
Artificial Neural Networks (ANNs) represent computational systems inspired by the intricate structure and functionality of biological neural networks. Comprising numerous interconnected processors, ANNs serve as powerful 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 through 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 learning or training process aimed at minimizing error. ANNs are particularly effective at tackling complex problems such as pattern recogni-
29
tion, clustering, categorization, and prediction. Within the domain of ANNs, two primary categories exist: feedforward 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**[44»]**represents a notable example of a fully connected, three-layer feedforward network, which includes 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 that encompass function approximation, time series prediction, and system control [47].
Feedforward neural networks (FNNs) are renowned for their exceptional performance**[«44]**in supervised learning scenarios and find application across diverse fields. Among the simplest variants of FNNs is the single-layer feedforward network (SLFN), as seen in Figure 2.5. which includes 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. The tunable parameters within the SLFNs encompass nonlinear parameters of the hidden layer and linear parameters of the output layer, collectively shaping the behavior and performance of the network [48].
A multilayer feedforward network, a prominent variant of artificial neural networks (ANNs), as seen in figure 2.6, constitutes a sophisticated architecture comprising an input layer, an output layer, and one or more hidden layers interleaved between them. This network structure is used to establish a mapping between the input patterns and the corresponding output representations. The underlying behavior
30
Figure 2.5: Single-layer feedforward networks [49]
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 the connections, and the choice of activation function used. 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 involves characterizing the closure, which pertains to the collection of limit points, and encompassing the functions that can be computed by the network. By unraveling the boundaries and capabilities of such networks, researchers and practitioners can gain insight into the expressive power and limitations of multilayer feedforward architectures, further advancing their understanding and utilization in diverse applications [50].
2.10.4 Natural language processing
Natural Language Processing (NLP) is an interdisciplinary field, as seen in Figure 2.7, that combines the realms of linguistics, computer science, and artificial intelligence to investigate, comprehend, and generate human language. It encompasses a broad spectrum of computational techniques and theories aimed at representing and processing natural language across various levels of linguistic analysis.
31
Figure 2.6: Multilayer feedforward network with a single hidden layer [49]

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, natural texts that span 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-answer systems, and many others [51].
The field of computational linguistics encompasses two primary branches: computational linguistics and theoretical linguistics. Computational linguistics focuses on the development of algorithms and methodologies for analyzing and generating
32
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 divided into syntax and semantic analysis. Sentence analysis, as a crucial component, attempts 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, helping to determine structural relationships and dependencies within a sentence [52].
Figure 2.7: Components of Natural Language Processing [52]
2.10.5 Large language model Large language models (LLMs) such as Bert and GPT-2 have ushered in a paradigm shift in the realms of natural language processing (NLP) and machine learning (ML). These sophisticated models have been specifically designed to generate coherent and contextually appropriate responses to given prompts, harnessing the
33
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.
Although LLMs exhibit remarkable capabilities in a variety of tasks, they lack the shared "form of life" that underpins the foundation of mutual understanding and trust between human beings. Consequently, LLMs are susceptible to generating language that can be deemed inappropriate or biased, particularly when confronted with unfamiliar or sensitive subject matter. Instances have arisen where 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, and their responses are solely based on statistical patterns embedded within the training data. Hence, an understanding of the limitations of LLMs is essential, deploying them as tools rather than surrogates for human intelligence [53].
2.10.6 Deep learning
Deep neural networks represent a prominent class of machine learning algorithms that encompass multiple hidden layers, allowing 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, most advan-
34
ced machine learning algorithms, including deep neural networks, are classified 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 produce superior and more interpretable outcomes. It is crucial to note that deep neural networks do not possess the capacity to address challenges requiring strong artificial intelligence capabilities, such as literal understanding and intentionality. In essence, deep learning represents a powerful machine learning approach that provides advanced functionality to process large and high-dimensional data sets, while acknowledging that shallow machine learning algorithms can outperform and offer greater interpretability for low-dimensional data and limited training data scenarios [54].
The roots of this field date back to 1957 when Frank Rosenblatt formulated the perceptron algorithm, a foundational component of deep learning. The perceptron, similar to a neuron as shown in figure 2.8, mirrors the fundamental functional unit of the brain and can be expressed through an equation, as shown 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. Although 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 sum of the weighted input vector, together with the bias term, passed through the activation function f [55].
35
Where: y f D wi xi b
D
y = f (wixi + b)
i=1
is output value, is activation function is the dimension of input space is a set of weights corresponding to the input vector x is the input vector is bias,
(1)
Figure 2.8: Biological neuron [56]
2.11 Data format

The question of data format arises when considering the appropriate representation for data transmission and storage. Different formats, such as JSON and CSV, have their own advantages and considerations. For instance, JSON (JavaScript Object Notation) is a widely used format for structuring data that is human-readable 36
Figure 2.9: Artificial neuron - perceptron [57]
and easily processed by various programming languages. On the other hand, CSV (Comma-Separated Values) is a tabular format that is often used to represent 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 a 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 data set 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 data sets in different formats as they may have varying column structures and sizes. This requires us to address
37
the problem of data format consistency and compatibility. Data pre-processing 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 have different 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 help to convey the intended meaning and ensure an accurate understanding and interpretation of the annotated data. In general, addressing data format considerations, ensuring compatibility
between storage systems, handling variations in data set formats, and resolving ambiguity in annotations are essential aspects of data management and analysis in the realm of information technology and data science.
38
3 Related Works
The schema.org website [58] serves as a scholarly compendium, jointly established by renowned entities such as Google, Microsoft, and Yahoo. It provides an authoritative guide for the systematic development of structured data schemas. This platform offers valuable insights into the use of 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 preexisting types, thus expediting the initial stages of schema development.
In the realm of machine learning classification models, the Kaggle platform [59] assumes a pivotal role in the access to relevant data sets. It offers a large amount of resources that inspire researchers and developers alike. Users can access downloadable datasets accompanied by comprehensive descriptions that shed light on the data contained therein. Furthermore, Kaggle provides specific statistics, revealing the cardinality and distribution of unique values. This information enables researchers to comprehend the nature and scope of each data set column, including its associated data type. Additionally, metadata pertaining to dataset origins and contributors can be found, along with usage statistics such as views and downloads, which offer valuable insights into the dataset's popularity and engagement.
In scholarly discourse, the creation of comprehensive schemas assumes significance, particularly in cases where a system is being defined for the first time or remains partially articulated [60]. With an increasing influx of weakly structured and irregular data sources, the extraction of schema information is vital for diverse tasks
39
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.
40
4 Design
Designing a project involves careful consideration of the technologies to be used, the database model, and the overall structure of the Web page. This section provides an overview of the technologies that will be utilized, followed by a comprehensive database model showing firebase, Neo4j, and postgreSQL. Furthermore, the sections showcase diagrams that depict the functionality of the web page, and conclude with the design of high-fidelity wireframes, illustrating the envisioned appearance of the page.
The selection of appropriate technologies is crucial for the success of any project, as it ensures robustness and efficiency. Various modern technologies will be employed, leveraging their unique features and capabilities.
A well-designed database model serves as the foundation for organizing and managing data efficiently. In this project, the Neo4j graph database model will be utilized due to its advantages in handling complex relationships and interconnected data. Neo4j represents entities as nodes and relationships as edges, enabling seamless data traversal and facilitating data-driven decision making. PostgreSQL database will be implemented for comparison purposes. Firebase will primarily be utilized for its authentication functionality, with additional utilization of its Firestore database.
To provide a comprehensive understanding of the functionality of the web page, a set of diagrams will be presented. These diagrams will illustrate the various components and interactions within the system, including user interfaces, data flows, and system architecture. The use of visual representations helps convey the conceptual
41
design and identify potential bottlenecks and areas for improvement.
4.1 Technologies

In the culmination of the comprehensive analysis conducted, we carefully examined various technologies and evaluated their respective merits and drawbacks. Based on this assessment, we have made informed decisions about 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 used, 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. In addition, a DNS record will be configured to improve 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, which offers fast and optimized builds. Furthermore, the Composition API will be used 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.
42
In terms of styling, the Tailwind CSS framework will be employed, offering a utilityfirst approach that facilitates rapid development and consistent styling throughout the application. By leveraging Tailwind CSS, the design process becomes more efficient, enabling customization and responsiveness. PrimeVue is the project's key component library, providing a versatile toolkit for seamless UI development. Its reliability and community support ensure a strong foundation for long-term adaptability to evolving design trends and technology.
For streamlined deployment and scalability, the application's front-end will be hosted on Firebase Hosting.
In 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 to manage server infrastructure. To streamline development and enhance code organization, the Nest.js framework will be used, 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. Another database, postgreSQL, will be utilized for comparison. This project will also utilize Firebase authentication for the login/registration process and Firestore to save the roles of a user and the information about the dataset.
To ensure code maintainability and minimize redundancy, various design patterns will be used throughout the development process. Notably, the adapter design pattern will be utilized, enabling the flexibility to change database and back-end
43
without the need of changing the front-end if we dont change the endpoints. This design pattern promotes reusability, scalability, and adaptability of the code, enhancing the overall functionality of the project.
The entire application will be hosted on the Google Cloud Platform (GCP), a cloud computing service that provides a robust and scalable infrastructure. Using GCP ensures reliable performance, scalability, and availability of the web page to users around the world.
4.2 Web application specification and requirements
This section presents the specifications and requirements for the development of a web application that targets data scientists who seek annotated datasets to train their artificial intelligence models. The application will also provide a platform for users to upload and annotate their own data sets. Users will have the flexibility to access the web application from any device connected to the Internet. The application will be structured into multiple web pages, categorized based on user preferences. Furthermore, the system will incorporate multiple types of users, each with different rights and privileges.
4.2.1 Usability
The primary objective of the Web application is to ensure usability, allowing 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 available datasets. Intuitive search and filtering mechanisms will be implemented to streamline the discovery of data sets, enhancing the overall user experience.
44
4.2.2 Awareness
The web application will serve as a comprehensive repository of information, encompassing all relevant aspects related to the topic. Detailed descriptions and metadata for each dataset will be provided, equipping users with the knowledge necessary to make informed decisions regarding data set selection. In addition, the application will offer information on the annotation process of the dataset, 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 data sets without mandatory registration. However, to contribute or annotate datasets, users will be required to register an account. It is important to recognize that registration may pose a potential barrier to user participation, as it might discourage individuals from contributing information to the platform.
4.2.4 Ease of use
The web application will prioritize ease of use, in order to provide a seamless and intuitive user experience. The navigation system will be designed thoughtfully, allowing users to quickly 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 easily perform tasks and navigate between pages. Additionally, responsive design principles will be imple-
45
mented to ensure an optimal user experience across various devices and screen sizes.
4.3 Target group

This chapter focuses on the analysis of the target group for the web application, considering the individuals 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 look for 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 have 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 cover 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 46
[61] reveals that mobile devices, including mobile phones, represented approximately 51. 3% of Internet usage in 2016 and this figure increased to 53% in 2019. Although 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
Drawing from the insightful articles [62, 63, 64], I am constructing a hierarchical data structure in Neo4j and formulating a resilient data model. Guided by the principles of location trees elucidated in these articles, I have chosen to abide by the following guidelines.
• All relationships are directed from children to parents, ascending the hierarchy.
• A single relationship type (named related_to) is used for all relationships.
• Each node has a sole outgoing relationship with its parent.
• Nodes can have one or multiple incoming relationships from their children.
Applying these guidelines, I have organized my nodes to incorporate a single relationship named " related_to." This approach guarantees a hierarchical tree structure, where each node possesses only one relationship leading to its parent. Meanwhile, multiple nodes can establish connections with the same parent, signifying
47
siblings' relationships. Using this Neo4j model as a base, I subsequently devised a model for a PostgreSQL database. In the following text, I will elaborate on the final database models for both systems.
In Figure 4.1, we present an illustrative depiction of the database model created using the diagram.io platform. This diagram provides a visual representation of the data structure and relationships within the database system. It showcases the integration of Neo4j, Firebase, and PostgreSQL databases within the application architecture.
The following text describes a graph data model in Neo4j and firebase, with a structure suitable for a knowledge base that stores data and user annotations about datasets. Here is a breakdown of what each part does:
Firebase:
• Firebase Authentication Users: Represents a user of the system.
• Firebase Firestore Database Roles: Represents the role of a user, the role is mapped to the user by UserUID.
• Firebase Firestore Database Datasets: Represents a data set that the user has uploaded.
Neo4j:
• Node Dataset: Represents a dataset that the user has uploaded. Inside of neo4j database. This dataset is mapped to the one in firestore by firebaseDatasetID.
• Node Column: Represents a column in the dataset.
• Relationship HAS_COLUMN: Indicates that the column belongs to the dataset.
48
Figure 4.1: Database model
49
• Node Annotation: Represents an annotation that user has added to our knowledge base.
• Relationship ANNOTATED_WITH: Indicates that the column has been annotated by this annotation.
• Relationship RELATED_TO: Indicates that this annotation is related to another annotation.
In the context of PostgreSQL, the database schema comprises several tables, each serving a specific purpose:
• Annotation Table: This table stores information about annotations. It includes a relationship where each annotation can optionally refer to another annotation through a "0 to 1" relationship from the parent_annotation_id column to the id column within the same table. Additionally, there is a "1 to many" relationship from the id column of the annotation table to the foreign key column_id in the column_annotation table.
• Dataset_Column Table: Responsible for storing information about columns within a dataset, this table establishes a "many to one" relationship with the dataset_id column of the dataset table. Furthermore, there is a "one to many" relationship from the id column of the dataset_column table to the column_id column in the column_annotation table.
• Dataset Table: Stores information pertaining to data sets. It maintains a relationship from its id column to the dataset_id column in the dataset_column table. Additionally, it is mapped to Firebase datasets via its id.
• Column_Annotation Table: This table serves as a mapping between columns and their corresponding annotations. It establishes a relationship from
50
the column_id column to the id column in the annotation table, facilitating the association between annotations and columns. Additionally, it includes an annotation_id column, establishing a relationship with the id column in the annotation table, further enhancing the linkage between columns and their annotations.
In general, this schema design facilitates the organization and retrieval of data related to annotations, datasets, and their respective columns within the PostgreSQL database.

4.5 Diagrams
4.5.1 Login diagram flowchart
The flowchart on figure 4.2 outlines the user journey upon accessing the website. It begins by checking the user's registration status. If the user is not registered, they are directed to the sign-up process, where they provide the necessary information and submit the form. Alternatively, users can opt for a streamlined sign-up process using their Google account, facilitated by Firebase authentication.
After successful registration or if the user is already registered, the flow chart proceeds to the login process. Users enter their credentials, and the flow chart validates the information. If the credentials are valid, the user is successfully authenticated and logged into the system. In the event of invalid credentials, users are redirected to the log-in step for correction. Additionally, users have the option to log into their Google account as an alternative method.
51
Figure 4.2: Login diagram flowchart 52
4.5.2 Add dataset diagram flowchart
The flowchart on figure 4.3 begins with log-in then the flowchart proceeds to the next step, which involves selecting a file. If the user does not select a file, the flow chart loops back to the "Select File" stage, prompting the user to choose a file.
If the file is not in the CSV format, the flow chart loops back to the "Select File" stage, allowing the user to select a different file. On the other hand, if the data are indeed in CSV format, the flowchart proceeds to the next step, which involves saving the csv data inside firestore and also saving other information such as who added this data set, when, and how big it is.
Figure 4.3: Add dataset diagram flowchart
53
4.5.3 Annotate dataset diagram flowchart
The flow chart on Figure 4.4 starts with the "Select Dataset" stage, where the user is asked to choose a dataset for annotation. If the user cannot select a dataset, the flow chart redirects them back to the "Select Dataset" stage, allowing them to select a dataset.
Once a data set is chosen, the flow chart progresses to the "Input dataset description" step. This step is optional and involves adding a description for the dataset.
Once a data set is chosen, the flow chart progresses to the "Select Column for annotation" step. In the event that the user does not select a column within the dataset, the flow chart redirects them to the "Select a Column" stage, enabling them to choose a column.
Upon selecting a column, the flow chart proceeds to the "Input column description" stage. Here, the user can add a description for the column to better describe it.
Upon selecting a column, the flow chart proceeds to the "Add annotation" stage. Here, the user starts the annotation process. If the desired annotation already exists for the selected column, the flow chart transitions into the "Annotate" step, allowing the user to proceed with the annotation.
If the desired annotation doesn't exist, the flowchart moves to the "Create desired annotation"stage, where the user defines the annotation. Then, it transitions to the "Annotate" step for annotation.
If the user did not annotate all the desired columns he is moved back to "Select column for annotation". If all the descriptions and annotations are added, the user
54
then presses "Save changes",which saves all the data into the database. Here, the user can export the csv dataset or export annotations with which he has annotated the dataset.
Figure 4.4: Annotate dataset diagram flowchart
55
4.5.4 Google cloud infrastructure diagram
The diagram in Figure 4.5 can be effectively conceptualized as a hierarchical representation 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 humanreadable hostnames into IP addresses, ensuring that the client can communicate effectively with the GCP resources.
Once DNS resolution is performed, the client is routed to firebase hosting. This service acts as the host for the front-end Vue.js application, providing built-in security and a developer-friendly environment for managing and deploying web applications.
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 the microservice architecture.
Each of the cloud function connects to the terminal tier, which includes three instances of Neo4j Aura. Neo4j Aura is a fully managed, always-on graph database with horizontal scalability and high availability. 56
Figure 4.5: Google cloud infrastructure diagram 57
4.5.5 Google cloud CI/CD diagram
The Continuous Integration and Continuous Deployment (CI/CD) workflow as seen in figure 4.6 for the application unfolds as follows:
Initially, the developer executes modifications to the application's source code in 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.
The GitHub repository serves as the version-controlled storage mechanism for the application's source code, tracking all changes. It is intricately linked to GitHub Actions, so that any changes pushed to the repository initiate an automatic trigger in GitHub Actions.
In the CI/CD pipeline, GitHub Actions manages the end-to-end deployment of both the Vue.js front-end application and Cloud Functions. Orchestrates the build, test, and deployment processes, ensuring a streamlined workflow for the entire application stack. 58
Figure 4.6: Google cloud CI/CD diagram
4.6 User interface
In this segment of my Master's thesis, we 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 is important to note that not all wireframes developed during this process will be included in this section. Instead, I will 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 have included these additional wireframes in the project folder of my thesis, within the Figma project file.
59
4.6.1 Find a dataset wireframe
AnnotateDataset.info
Find a dataset
List annotations
Find your dataset
Dataset name
Author name
Added after
Added before
Search
Add new dataset
Log in / my account Dataset size
Dataset name Global Temperature Trends COVID-19 Worldwide Cases European Bird Migration Patterns
Author name Dr. John Smith Prof. Emma Johnson Dr. Sofia Martinez
Publish date 2023-05-31 2023-03-20 2023-04-10
Size Medium Big Small
Previous
1 2 3 4 5 Next
Figure 4.7: Find a dataset figma wireframe
In figure 4.7, we have a wireframe of a website called "Find Your Dataset". This page has search boxes where we can type in a data set name or author name to look for specific data. We can also search by the date the dataset was published or how large 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 is a button labeled "Add New Dataset". If we press this button, we are asked to choose a dataset to add to the system.
60
4.6.2 Selected dataset wireframe
AnnotateDataset.info
Find a dataset List annotations Dataset name: Global Temperature Trends
Annotate
Log in / my account 5 of 32 columns selected
City New York Los Angeles London Tokyo Sydney
Average_Temperature 16.5°C 20.3°C 13.9°C 23.8°C 18.3°C
Minimum_Temperature 10.2°C 16.1°C 8.5°C 18.6°C 14.8°C
Maximum_Temperature 23.7°C 24.7°C 19.4°C 29.4°C 22.6°C
Temperature_Anomaly +0.6°C -0.2°C +1.0°C -0.4°C +0.8°C
Previous 1 2 3 4 5 Next
Select columns
New York
HAS_ENTITY
Statue of Liberty
Column Node
Entity Node
HAS_PROPERTY
IS_OF_TYPE
RELATED_TO Historical
Monument Type Node
SUBTYPE_OF
Property Node National Monument
Subtype Node
Ellis Island Entity Node
Figure 4.8: Selected dataset figma wireframe
In figure 4.8, we have a sketch or wireframe of a website called "Selected Dataset". This page shows the name of the dataset that we have chosen. We can also download this data set in a format called CSV by clicking on a download icon.
An interesting feature of this page is that we can choose the information or columns that we want to see. We can select all of them, a few, or even none. The dataset is shown in a table that is 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 on 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 important details.
61
4.6.3 Find annotation wireframe
AnnotateDataset.info
Find a dataset List annotations
Entity name

Search
Add annotation
Log in / my account
Entity name Eiffel Tower Great Wall of China Sydney Opera House
Has property Touristic Historical Architectural
Type of Structure Fortification Building
Subtype of Observation Tower
Related to Seine River
Defensive Wall
Forbidden City
Performing Arts Venue Sydney Harbour Bridge
Previous 1 2 3 4 5 Next
Figure 4.9: Find annotation figma wireframe
In figure 4.9, an illustrative wireframe representation of the page, "Find annotation" is displayed. This page allows for searching for 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, enabling 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 ability to incorporate a new annotation. This is made possible through an interactive component, which presumably is a button, which when activated prompts the user to add a new annotation to the system.
62
4.6.4 Selected annotation wireframe
AnnotateDataset.info
Find a dataset
List annotations
Entity selected: Eiffel Tower
Depth of related to
Log in / my account Number of relationship to show
Eiffel
REL ATED_TO
Tower
Entity Node
H A S_PRO
HAS_PROPERTY
PERTY
I S _O F_T Y P E
TToouurirsitsictic
I S _O F_T Y P E
Architect
Property Node
Property Node
Structure Type Node
S U B T Y P E _O F SU
BTY PE_O F
Observation Tower
Subtype Node
Monument Type Node
S U B T Y P E _O F
SUBTYPE_OF
National Monument
Type Node
Iron Structure Subtype Node
Historical Monument Type Node
Seine River
Entity Node
HAS_PROPERTY HAS_PROPERTY
Basin Size Property Node
I S _O F_T Y P E
I S _O F_T Y P E
Country Property Node
River Type Node
F
_O E P Y T B U S
Waterway
Type Node S U B T Y P E _O F
Tidal River Type Node
Navigable River Type Node
Figure 4.10: Selected annotation figma wireframe

In figure 4.10, a wireframe of the "Selected Annotation"page is illustrated. 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 the interpretability of the data.
Depending on these selected attributes, the page generates a graphical representation, prominently displayed for intuitive and insightful comprehension. This visu-
63
alization helps to understand the intricate relationships and the overall structure surrounding the selected annotation.
4.6.5 Add annotation wireframe
AnnotateDataset.info
Find a dataset List annotations
Annotate
Name Description Unit
Eiffel Tower Tower in France
Structure
Related to Entity name Eiffel Tower
Seine River
Log in / my account
Is of type Type Monument Structure
Subtype Historical Monument Iron Structure
Has property Property name Architect
Touristic
Add annotation
Figure 4.11: Add annotation figma wireframe
In figure 4.11, we examine a wireframe of the webpage "Add Annotation," which offers a graphical illustration of the annotation-adding process. Users have the ability to input comprehensive details about the annotation, including 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 given the ability to view, add, or remove relationships from this table, thus providing a dynamic and adaptable user interface.
64
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 operates akin to the aforementioned features, demonstrating a consistent user interface.
65
66
5 Implementation
In this implementation section, I will discuss the practical application of the coding aspect of my thesis. I will cover essential implementation steps, such as different approaches I took at the beginninging, annotation suggestions, removing dataset size restrictions, adopting a different database with the repository pattern, integrating visualization tools, establishing a seeder, and deploying the application. Following these crucial steps, I will showcase the API endpoints and final screens of my thesis, along with considerations on costs, application testing, and user testing to conclude.
5.1 Essential Implementation Steps
In this segment, I will discuss the significant implementation steps, decisions, and functionalities I developed during the creation of my application, as well as the various approaches I adopted.
5.1.1 Different approaches
Initially, I integrated the Vue Apollo Client in the front-end to seamlessly interact with the GraphQL API in the back-end. To facilitate this communication, I developed an Apollo server. This setup allowed for smooth data exchange between Vue Apollo Client and the backend. On the backend, I constructed a GraphQL schema that outlines all entities, relationships, queries, and mutations pertinent to the application. Furthermore, a
67
command was implemented to generate TypeScript definitions from the.graphql file, which streamlined usage within the application.
As the development progressed, discussions with my diploma thesis supervisor led to several improvements. Primarily, to align with the serverless architecture of the application, we decided to remove the Apollo Server.
In trying to integrate TypeORM with my PostgreSQL database, I encountered a significant issue where my requests stopped functioning properly. Strangely, the body within all URLs became undefined, despite being expected to have a value. Despite attempts to troubleshoot and solve the problem, I was unable to find a solution. As a workaround, I decided to resort to using raw SQL queries instead.
5.1.2 Annotation suggestions
The mapping of annotations has been implemented to enhance the annotation process based on insights gained from prior datasets. With this enhancement, when users annotate a dataset, they receive helpful suggestions based on annotations assigned to similar columns in previous datasets.
For example, let us say a column called "name" was labeled as "person" in one dataset. So, if users later work on another dataset and find a column named "name," the system will automatically suggest "person" as the initial label.
Also, if there are multiple labels available for a column, users can see all of them. This is helpful, especially if different datasets have used the same label. Users can also see how relevant each label is for the current data set by checking its count.
Using insights from previous datasets, this approach significantly streamlines the

68
annotation process, providing users with informed suggestions to annotate similar columns.
5.1.3 Removing restriction for dataset size
When utilizing Firestore, it is crucial to understand its limitations, particularly regarding dataset size. Firestore imposes a maximum document size limit of 1 MiB (1,048,576 bytes), rendering it impractical to upload datasets larger than 1 MB directly into the database.
To circumvent this constraint, I adopted an alternative approach: instead of uploading the entire CSV file to Firestore, I stored it within Firebase Storage. Subsequently, I inserted the download link for the CSV file into Firestore documents. This solution enables me to store CSV files of any size, overcoming Firestore's size limitation.
5.1.4 Using a different database with the repository pattern
In the development of the application, the repository design pattern played a pivotal role in achieving flexibility and scalability, particularly when dealing with multiple database options. The use of this pattern facilitated seamless integration with different database technologies, allowing for smooth transitions between database systems without significant changes to the application's codebase.
The repository pattern abstracts the data access layer, providing a unified interface to interact with the underlying data storage, irrespective of the specific database technology employed. This abstraction layer protects the higher-level components, such as controllers, from the complexities of the underlying database operations.
69
To illustrate this, consider the scenario of my application, which needs to support both Neo4j and PostgreSQL databases. For each module within the application, a similar approach to database interaction was employed. However, for demonstration purposes, we will focus on the annotation module as an example.
Within the annotation module of the application, the repository interface is represented by the AnnotationService. Concrete implementations of this repository interface are AnntoationServiceNeo4j and AnnotationServicePostgresql, each tailored to interact with the respective database technology.
These repository implementations encapsulated the database-specific operations required for managing annotations, such as creation, retrieval, updating, and deletion.
At the heart of this implementation, the AnnotationService interface (or abstract class) defined the contract for interacting with annotation data. The controller components of the application interacted with this interface, relying on its methods to perform annotation-related tasks.
To determine which repository implementation to use at runtime, the application leveraged environment variables. By configuring the DB_TYPE environment variable, the application dynamically selected the appropriate repository implementation to interact with the chosen database technology.
This modular approach to database interaction not only simplified the development process but also enhanced the application's maintainability and extensibility. It allowed easy swapping of database technologies, facilitated unit testing through dependency injection, and promoted a clean separation of concerns within the application architecture.
Furthermore, the flexibility of the repository pattern allowed users to easily incor-
70
porate additional databases with minimal code changes. By modifying the module to choose the correct database based on environment variables and implementing the service for the new database, users could seamlessly integrate additional database options into the application without updating the controller logic. This extensibility underscores the power and versatility of the repository pattern in enabling scalable and adaptable database management solutions.
To provide a clearer visualization, let us examine the diagram depicted in Figure 5.1. Within the annotation module, we have the AnnotationService interface, which serves as a contract to which both the AnnotationServicePostgreSQL and AnnotationServiceNeo4j classes are bound. These concrete classes use the implement keyword to ensure that they have the same function declarations as the interface.
Each function within the AnnotationService utilizes Data Transfer Objects (DTOs) to transfer data. While not all DTOs are shown in the diagram to avoid clutter, they play a crucial role in the data exchange process.
Moving to the AnnotationController, it exclusively utilizes the AnnotationService, which means that it interacts only with this service and is unaware of the underlying database implementations. This dependency between the AnnotationController and AnnotationService is illustrated by an arrow indicating usage.
Lastly, a note in the diagram highlights that the choice of the correct database implementation is determined by an environment variable, ensuring the flexibility to switch between database technologies seamlessly.
Overall, this diagram provides a comprehensive overview of how the repository design pattern is implemented within the annotation module, showcasing the interactions between different components and their dependencies.
71
Figure 5.1: Repository design pattern 72
5.1.5 Visualization tools
In the process of selecting a visualization tool for my project, I explored various options listed on the Neo4j blog [65]. I chose neovis because it was the most suitable for my needs.Neovis.js, as it says on its github page, is a combination of Neo4j + vis.js. It is a graph visualizations in the browser with data from neo4j. Neovis connects to the neo4j database and displays the real-time state of the database. Use rcan to start query for it meaning set what data will be shown. I made some adjustments to the default visual look of neovis. To visualize the path of my annotations with different colors for enhanced visibility. To improve clarity, I also implemented arrow indicators.
Despite encountering difficulties with the outdated Neovis TypeScript implementation, I opted for an alternative npm package that lacks TypeScript compatibility. I opted for the one without typescript as it had more features than the typescript one was not up to date. However, this substitution posed no issues in practice.
For the visualization of my postgresql i unfortunately could not use neovis as it is only for neo4j but I could use viss.js concretely vis-network, but it posed challenges because now I had to style it the same way as neovis and also prepare data in the correct form for it to work.
Link to the visualization tools:
• neovis.js [66]
• vis-network [67]
73
5.1.6 Seeder data timeline

Initially, for testing purposes, I integrated three datasets obtained from Kaggle. These datasets are accessible through the following links:
• Electric Vehicle Dataset [68]
• Small COVID-19 Dataset [69]
• Used Fiat 500 Dataset [70]
Additionally, I prioritized the implementation of seeders to streamline testing procedures. Three seeders were developed: one for Firestore, one for Neo4j, and one for PostgreSQL. Seeders function as follows: an endpoint ”/seedNeo4jDatabase” is responsible for seeding Neo4j, while an endpoint ”/seedPostgreSQLDatabase” is in charge of seeding PostgreSQL.
During consultations with my supervisor, a shift in the project’s focus was recommended, transitioning from marketing datasets to business intelligence datasets.
Following a consultation with my supervisor, it came to my attention that there were issues with the annotation data for the dataset. The error arose from annotating columns based on their specific values, whereas the correct approach involves defining annotations based on the underlying concepts and generalizing these concepts. To rectify this, I revisited the concepts of knowledge bases and semantic annotation, delving into various papers and videos for a more comprehensive understanding.
In light of this newly found knowledge, I remade the datasets and their annotations. This time around, I am ensuring adherence to the correct methodology, where annotations are rooted in conceptual understanding rather than being tied
74
to specific values. The datasets I am annotating include:
• mkt_synthetic [71]
• KAG_conversion_data [72]
• Ecommerce Customers [73]
I implemented annotation recommendation logic, which required testing with multiple datasets containing the same column names. By adding the following datasets, I ensured the presence of the ”age” column across three different datasets (KAG_conversion_data, customer_conversion_training_dataset, and conversion_data), the ”gender” column across two datasets (KAG_conversion_data and customer_conversion_training_dataset), and the ”converted” column across two datasets (mkt_synthetic and conversion_data). Here are the newly added data sets:
• customer_conversion_traing_dataset [74]
• conversion_data [75]
5.1.7 Deployment
The deployment was carried out according to the diagrams provided during the design phase of my project. You can access the entire infrastructure diagram here. To ensure consistent synchronization between the web app version and the one on my localhost, I implemented CI/CD. You can refer to the CI/CD diagram here for a detailed overview of the process.
To optimize performance and overcome the cold start issue, I have restructured the application into microservices. This reorganization aims to streamline the architecture, making it more scalable and manageable. As illustrated in Figure 5.2,
75
the Google Cloud Functions deployment now includes various microservices, replacing the previous monolithic API. By breaking down the monolithic structure, each microservice focuses on specific functionalities, leading to improved efficiency and maintainability. Now, let us dive deeper into the current division. The following cloud functions, and hence microservices, are employed:
1. AddUserRole This service takes care of setting the default role of user when he first registers.
2. AnnotationService This service handles tasks related to annotations, including creation, retrieval, and other annotation-related operations.
3. API This service contains all the functionality within it, making it a monolith. It serves primarily for testing purposes.
4. ColumnService This service is used to manage the creation of columns, mapping them to datasets, and handling updates to their values.
5. DatasetService This service focuses on the creation and updating of datasets; this module encapsulates all functionalities pertaining to dataset management.
6. SeederService This service takes care of seeding the databases either Neo4j or PostgreSQL
7. SetUserRole This service handles the setting of a role for a specified user.
Figure 5.2: Google cloud microservices
76
5.2 API endpoints
In this section, I will cover the endpoints utilized in my application.
5.2.1 NestJS API endpoints
My NestJS application encompasses several routes, each dedicated to different functionalities. Let us delve into the specifics of each route:
/annotation
• Post /createAnnotation - Creates a new annotation. • Get /getAnnotations - Retrieves all annotations. • Post /annotateColumn - Annotates a column with an annotation. • Get /getExport/:id - Provides an export of a dataset and its corresponding
annotations. • Get /getColumnsAnnotations/:datasetId - Get the annotations for all co-
lumns in specified dataset. • Delete /deleteAnnotation/:id - Delete an annotation and its relations. • Post /connectAnnotations - Creates a relation of related_to between two
annotations. • Post /getAnnotationsSuggestions - Returns annotations suggestions for all
columns in specified dataset. • Post /postImport - Imports dataset and its annotation from .csv and .json
file. /column
77
• Get /getColumnDescription/:datasetId - Retrieves the corresponding description of all columns of a specified dataset.
• Post /updateOrCreate - Updates or creates a description for all columns. /dataset
• Post /createDataset - Creates a new dataset. • Get /getDatasetDescription/:datasetId - Retrieves the description of a da-
taset. • Post /updateDatasetDescription - Updates the description of a dataset. • Post /renameDataset - renames the specified dataset. • Post /deleteDatasets - Deletes all selected datasets. /seeder • Get /seedNeo4jDatabase - Seeds Neo4j database • Get /seedPostgreSQLDatabase - Seeds PostgreSQL database • Get /testingNeo4j/:number - Generates test data for Neo4j • Get /testingPostgreSQL/:number - Generates test data for PostgreSQL

5.2.2 Firestore cloud functions endpoints
Additionally, Firestore cloud functions contribute to the application's functionality with the following endpoints:
• /AddUserRole - Adds a user role to a registered user. • /SetUserRole - Sets the role of a user. 78
5.3 Screens
In this section, I will present the current appearance of my application. While my diploma thesis in section 4.6, User Interface, featured wireframes illustrating the anticipated look of the application, these served as templates. The final application deviates from these initial designs due to modifications made to the proposed views. Consequently, I will now present the updated and actual views of the application. 5.3.1 Find a dataset page
Figure 5.3: Find a dataset page Webpage in figure 5.3, titled "Find a Dataset," serves as a platform for users to seamlessly add data sets to the database. By clicking the "Upload CSV Dataset" option, users are prompted to choose the a.csv file format. Upon clicking on the
79
"Import dataset and annotations" a pop-up appears to allow the user to add a CSV file of the data set and a JSON file with annotations. The page features a table that displays all available datasets within the application, equipped with a built-in sorting mechanism for each column. In addition, users can use the text search functionality to easily locate specific datasets. The table incorporates pagination for user-friendly navigation through multiple datasets. Furthermore, the page includes a convenient "Clear" button that resets all applied filters, ensuring a smooth user experience. There is also an option to select datasets after clicking on the delete button to be deleted.
On the left side, a collapsible sidebar enhances navigation, featuring an "x" icon for closure and an arrow button for expansion. This sidebar not only aids in page navigation, but also displays the user's log-in status. If the user is not logged in, they are provided with the option to log in. So, this sidebar functions for navigating on the page and redirection.
5.3.2 Selected dataset page
On the "Selected Dataset" page as seen in figure 5.4, users encounter comprehensive details about their chosen dataset, including its name and description. A table shows all columns within the CSV data set, allowing users to choose specific columns for display. In particular, users can initiate a name change by clicking the "Change dataset name" button, which triggers a pop-up where they can input a new name for the data set. The same thing happens for "Change description" popup opens, and the user can put the name dataset description.
The page is equipped with two export features: the "Export Dataset" button facilitates the export of the dataset in CSV format, while the "Export Annotations" button exports the dataset's annotations in JSON format. These functionalities
80
Figure 5.4: Selected dataset page 1 81
significantly enable users to interact with and extract valuable insights from the selected dataset. Additionally, a sticky header features two buttons: "Save All", which preserves all the information provided, and "Add Annotation," which opens a pop-up for users to add annotations. This latter feature will be elaborated upon in subsequent sections of this thesis.
Figure 5.5: Selected dataset page 2 This section, as seen in figure 5.5, guides the user through the annotation process for each column. Each column is represented by a card that the user can annotate, and the changes are only finalized after clicking the "Save All" button. On the left side, the user can provide a description for the column using "Change description" button which opens a pop-up for the new description. Then the user can see with which annotations this column has already been annotated. On hover, he sees all information about this annotation, and after clicking on it, he is redirected to page list annotations with the annotation he clicked on being selected; I will talk about this page later. Additionally, the user has the option to select an annotation, accompanied by a visual tree structure preview displayed through arrows and text on the left side. Each text item features tooltips that provide comprehensive 82
information about the selected annotation. The user also has the option of clicking on "Suggest annotations", which will open a pop-up and suggest the most suitable annotation for this column based on existing datasets.
On the right side, a NeoVis graph visualization depicts the relationships. The red circles represent the current annotation, showcasing its hierarchical representation from the bottom to the top using the RELATED_TO relationship. Blue circles denote relationships that are not part of the current annotation hierarchy, but are part of a broader hierarchy. The visualization selectively displays relevant relationships, offering the user a contextual understanding of the annotation's broader scope.
5.3.3 List annotations page
This page, as seen in figure 5.6, seamlessly combines elements from both the "Find an annotation "and "Selected annotation "sections outlined in my diploma thesis design. Users enjoy comprehensive access to all annotations in our knowledge base, thanks to a user-friendly table at the bottom that supports sorting and searching based on strings. Furthermore, users have the option to select a specific annotation, prompting the presentation of a dedicated NeoVis graph visualization and tree-like visualization as explained above. Additionally, there is a "Add Annotation" button that, when clicked, opens a pop-up for users to add annotations. This feature will be explored in greater detail in subsequent sections of this thesis.
In the visualization, the red circle signifies the selected annotation, revealing its hierarchical representation from the bottom to the top. Blue circles highlight relationships beyond the immediate annotation hierarchy, contributing to a more extensive structure. This strategic presentation of relationships provides users with a contextual understanding of the annotation's broader scope. Currently, users are
83
Figure 5.6: List annotations page 84
presented with a text interpretation, similar to the format observed on the "Selected Dataset" page. This comprehensive approach enhances the user's understanding of annotations and their interconnected relationships.
There are two scenarios for this page: In one variant, the user navigated here through the left sidebar, and as a result, no annotation is pre-selected. In the other variant, the user arrived here via a redirect from the "Selected Dataset" page, where a specific annotation is already chosen.
5.3.4 Add annotation page
On this page, which we can see in figure 5.7, users can add a new annotation by providing its name, shortcut, and description. In addition, users have the option to establish connections with related annotations. The page includes a table displaying existing annotations the same as the one on the list annotations page, allowing users to search for the correct annotation to connect with the newly added one. Users can also delete annotations or establish connections between two annotations. Upon selecting an annotation from the list, a NeoVis graph representation, similar to other instances, is displayed along with a text representation. This dual visualization approach improves the user's understanding of the relationships between annotations.

85
Figure 5.7: Add annotation page 86
5.3.5 Admin page
Figure 5.8: Admin page This page, which we can see in figure 5.8, allows the admin to modify the roles of registered users, toggling between admin and user roles. Once the user submits the changes, the updates are seamlessly reflected in the Firebase Firestore 'roles' collection. In addition, the administrator has the ability to view a list of all registered users who have used this page.
87
88
6 Evaluation
In this section, I will assess various aspects of my application. This includes evaluating cost considerations to determine the most cost-effective database option, analyzing the optimal deployment region, comparing monolithic versus microservice architectures, examining Neo4j versus PostgreSQL databases, reviewing cloud function memory allocation, and finally, conducting user testing.
6.1 Cost considerations
In this section, we will delve into cost considerations, specifically focusing on identifying the most economical options for Neo4j and PostgreSQL databases in the Google Cloud.
One notable option for cost-conscious users is Neo4j Aura, which offers a free tier without any charges. However, it is important to note that this free tier comes with certain limitations, such as the inability to utilize Neovis functionality, so I cannot use this free tier for my application.
Here is a table illustrating the most economical option:

Database	RAM	CPU	Storage	Hourly rate	Daily rate	Monthly rate
Neo4j	1GB	1 CPU	2GB	0.09$	2.16$	64.80$
PostgreSQL	0.614GB	1 vCPU	10GB	0.01$	0.34$	7.20$

Below is a table comparing instances with equivalent computing capabilities su-
itable for production environments, ensuring high availability across multiple zo-
89
nes:

Database	RAM	CPU	Storage	Hourly rate	Daily rate	Monthly rate
Neo4j	4GB	1 CPU	8GB	0.26$	8.64$	259.20$
PostgreSQL	3.75GB	1 vCPU	10GB	0.17$	4.03$	120.90$

In conclusion, it is evident that PostgreSQL emerges as a significantly more cost-
effective option for deployment, regardless of the size of the instance.
6.2 Performance Analysis and Comparative Testing
In this section of the diploma thesis, we will perform a comprehensive analysis of performance and perform various comparative tests to evaluate the efficiency of different configurations and sets of systems.
6.2.1 Analyzing Optimal Regions
In this section, we will explore the testing process aimed at identifying the most
favorable region. We conducted testing using pre-defined configurations, presented
in the table below.

Database	Application style	Cold start	Seeded
Neo4j	Monolithic	no	yes

In Figure 6.1, we observe a graph showing the optimal region. The vertical axis
represents the time taken in milliseconds for each endpoint to complete, while the
horizontal axis lists all endpoints utilized during testing. In addition, the graph
illustrates the various regions tested.
Analysis of the graph reveals that the deployment of cloud functions in the uscentral1 region resulted in the most significant delay, likely due to its distance from
90
our location. In contrast, deployment in Europe exhibited minimal differences in time, indicating relative proximity. In particular, Europe-west3 emerges as the best region for deployment due to its proximity to us and the presence of the hosted Neo4j Aura instance. Since this region emerged as the optimal choice, all subsequent tests will be carried out within it.
Figure 6.1: Best region line chart 6.2.2 Comparing Monolit and Microservices In this section, we will look at the testing process to compare the time taken to complete a request between APIs and microservices. We conducted tests using pre-defined configurations, which are outlined in the table below.

91
Database Application style Cold start Seeded
Neo4j
both
no yes
In Figure 6.2, we observe a graph showing the time taken in milliseconds. The horizontal axis represents the endpoints used, while the vertical axis indicates the time taken. Two lines in the graph illustrate the comparison between API requests and microservices.
Upon analysis, we find that API requests generally took longer than microservices across most endpoints, although the difference was not as significant as expected. However, the graph indicates that requests to APIs are slower than those to microservices.
Figure 6.2: Api vs microservices line chart 92
6.2.3 Comparing Request Speed: Neo4j versus PostgreSQL
In this section, we will examine how the speed of requests varies when using Neo4j versus PostgreSQL. We conducted tests using pre-defined configurations, as outlined in the table below.
Database Application style Cold start Seeded
Both
Monolithic
no yes
In Figure 6.3, the graph illustrates the time it takes for requests to be completed in milliseconds, with endpoints listed along the horizontal axis. Below each endpoint, the graph shows two lines representing Neo4j versus PostgreSQL.
Analysis of the graph reveals that Neo4j generally took longer to complete requests compared to PostgreSQL. This result is somewhat unexpected considering the complexity of the data set with numerous nodes and relationships. It is plausible that Neo4j's larger and more robust nature may have influenced this outcome, with PostgreSQL proving faster on smaller datasets. However, further tests are necessary to definitively determine these findings.
As mentioned above, further tests were necessary to compare the performance of the two databases. The testing involved using a single dataset with 7 columns, each having a maximum depth of related annotations set to 4. The parameter that varied during the testing was the number of annotations, ranging from 100 to 100,000, and the number of relationships from columns to these annotations.
The creation process followed a sequential pattern, in which each annotation was linked to the next in sequence. For example, Annotation 1 was linked to Annotation 2, which was linked to Annotation 3, and so on, until Annotation 5. Then, the sequence restarted, with Annotation 6 linked to Annotation 7, and so on. This sequential pattern was also applied to the relationships between columns and an-
93
Figure 6.3: Neo4j vs PostgreSQL line chart 94
notations, with the first column linked to Annotation 1, the second column linked to Annotation 2, and so on.
Figure 6.4: Neo4j vs PostgreSQL clustered column chart 100 annotations In Figure 6.4, the results of the 100 annotation test demonstrate PostgreSQL's superior performance over Neo4j on all endpoints except for deleteAnnotation and deleteDataset. Therefore, PostgreSQL emerges as the preferred option. In Figure 6.5, we observe the results of the tests with 1000 annotations. Here, I have selected to show only the endpoints with the heaviest annotation workload due to small differences in times. Once more, PostgreSQL emerges as the superior option, outperforming Neo4j across all endpoints except for getAnnotations where they had same time. In Figure 6.6, we look at the results of the test with 10000 annotations. Here, I have only shown the parts where annotations are used the most, as there were only small differences in times elsewhere. Neo4j is faster in getAnnotations, but in
95
Figure 6.5: Neo4j vs PostgreSQL clustered column chart 1000 annotations
Figure 6.6: Neo4j vs PostgreSQL clustered column chart 10000 annotations 96
almost everything else, it is much slower.
Figure 6.7: Neo4j vs PostgreSQL clustered column chart 100000 annotations In Figure 6.7, we examine the results of the test with 100000 annotations, our final trial. Here, we observe that in the most intensive operations involving annotations, Neo4j emerges as the winner. It is faster than PostgreSQL in the "getExport" task and significantly faster in " getAnnotations." However, in the "import" task, Neo4j is much slower compared to PostgreSQL. For better clarity, I organized all the results from these graphs into a table below. Here we could see the times more clearly. On the left, we can see the endpoints we tested. At the top, we could see the number of annotations used in the testing. Then, in each row, we saw two numbers: the first representing the time it took for Neo4j in seconds and the second representing the time it took for PostgreSQL in seconds.
97
100 1000 10000 100 000
getExport
0.85 5.15 121.08 327.46 0.02 0.056 3.94 3769.93
0.11 0.12 1.5
4.11
getAnnotations
0.01 0.13 6.15 1153.26
importTesting
2.06 19.2 269.56 14768.01 0.4 1.72 12.34 146.27
From these graphs and table, we can deduce the following findings: If there are fewer than 100,000 annotations, PostgreSQL emerges as the preferred database choice due to its overall superiority in terms of cost and speed. However, beyond the 100,000 annotations mark, Neo4j starts to exhibit faster performance, albeit at a higher cost.
The reason why PostgreSQL is emerging as the preferred choice for databases when there are fewer than 100,000 annotations is due to its efficiency in handling smaller datasets. PostgreSQL is known for its robustness and speed, particularly in the management of structured data and the efficient performance of standard SQL operations.

As the data set grows beyond the 100,000 annotations mark, Neo4j's graph database architecture starts to show its advantages. Neo4j excels at handling complex relationships and traversing interconnected data, making it more suitable for tasks like querying graph-like structures. This inherent advantage becomes more pronounced as the size of the dataset increases and the complexity of the relationships grows.
It should be noted that while Neo4j may be faster in database operations, such as ”getAnnotations,” it significantly lags behind PostgreSQL in the ”import” task. This indicates that while Neo4j may have a longer data processing time due to its
98
larger overhead, it excels at faster operations on more data.
Lastly, in the table below, we can observe how the size and number of rows of the annotation JSON file changed in relation to the number of annotations.
Number of annotations: File size:
Number of rows:
100 84 KB 2357
1000 792 KB 23057
10000 7.76 MB 230057
100000 78,4 MB 2300057
6.2.4 Memory Allocation Analysis
In this section, we will explore how the speed of requests varies depending on the memory allocated to the cloud function. We conducted tests using pre-defined configurations, as outlined in the table below.
Database Application style Cold start Seeded
Neo4j
Monolithic
no yes
In Figure 6.8, the graph shows the time it takes for requests to complete in mil-
liseconds, with the endpoints represented along the horizontal axis. Below the
endpoints, the available memory size is displayed.
The graph illustrates that increasing the available memory results in faster endpoint completion times, which is not surprising. However, significant differences are observed between 128MB, 256MB, and 512MB allocations. Interestingly, around the 1GB mark, the completion times show minimal variance regardless of whether the allocation is 1GB, 2GB, 4GB or 8GB. Consequently, beyond 1GB, increasing memory allocation does not significantly improve performance, with the bottleneck shifting to the database.
Taking into account cost considerations, the optimal price-performance ratio corresponds to approximately 1GB of available memory, equal to 0.583 vCPU. The
99
reason behind the increase in computing power with memory increments is that as memory increases, Google Cloud also adjusts the CPU power allocated to the respective cloud function.
Figure 6.8: Available memory line chart
6.3 User testing
In this part, I tested the website with 10 people who were my classmates. Their assignment was to complete two tasks that I specifically designed. The first task focused on testing the management of annotations and all related functionalities, while the second task focused on data set management. The times were rounded to the nearest second. The tests were carried out using predefined configurations, 100
as shown in the table below.
Database Application style Cold start Seeded
Neo4j
microservices
no
yes
6.3.1 Management of annotations
The objective of this test is to add two annotations, connect them, and subsequently delete them. To accomplish this task, users must navigate through the following pages: starting from the homepage, click on the left menu, then proceed to ”List Annotations,” followed by ”Add Annotation.” On this page, users are required to add two annotations, connect them, and finally delete both annotations.
Figure 6.9: Management of annotations
In Figure 6.9, the completion times recorded for users are sorted from highest to lowest: 260, 215, 212, 207, 180, 178, 157, 123, 94, 82, resulting in an average completion time of 169 seconds. Furthermore, the median completion time is 179 seconds
101
and the standard deviation of the completion times is approximately 49. 6.3.2 Dataset management The aim of this test is to perform a sequence of actions with a data set. Users are tasked with starting from the homepage and then navigating to the data set section to upload a CSV dataset. Once uploaded, users should open the data set and go to the selected data set page. On this page, you need to change the default name of the data set and provide a description of it. Users are also instructed to add descriptions to two specific columns within the dataset. Following this, users must annotate each of the designated columns with two annotations. Subsequently, users are required to export both the dataset and the annotations. Finally, users should return to the dataset section and delete the dataset.
Figure 6.10: Dataset management In Figure 6.10, the completion times recorded for users are sorted from highest 102
to lowest: 259, 232, 220, 204, 198, 187, 175, 162, 161, 89 resulting in an average completion time of 188 seconds. Furthermore, the median completion time is 192 seconds and the standard deviation of the completion times is approximately 40.
6.3.3 Identified Issues and Bugs
• The issue of annotations persisting in the selection after deletion was resolved by implementing a refetch mechanism, triggered whenever the user clicks the delete button, thereby ensuring that the page list reflects the updated status accurately.
• File uploading functioned flawlessly in the development environment. However, upon using the emulator, an error occurred due to malformed form data. As a result, an alternative approach to file uploading had to be devised.

• An issue was arising with the ”datasetCardComponent” not refreshing after saving changes. To address this, I implemented a solution by incorporating a ”ref” key. This key is updated whenever the user presses ”save changes,” effectively forcing a refresh of the card components.
• I have rectified the connection of relationships, ensuring that users cannot link an annotation to itself.
• Occasionally, the column annotation failed to save due to the asynchronous nature of the request. This issue was resolved by including the “await“ keyword. 103
6.3.4 Suggestions for improvement
In this section, I take a look at the feedback gathered from the tested participants. The insights provided by the subjects during testing is valuable. Certain suggestions aimed at enhancing the user experience of this website have been noted and may be considered for implementation in the future. Below are the improvements:
• Adjust the placement of the neovis visualization after selecting the parent annotation so that it appears below the second select field for improved visibility. Alternatively, consider resizing the neovis visualization to enhance visibility.
• Change the name of the “Add Annotation“ button that opens the popup to “Manage Annotations“.
• Make a tutorial on the home page showing basic use cases of my website.
• Modify the visualization of annotations so that they are not displayed in the selection but instead linked to the table action button “Show Annotation Detail “Additionally, include action buttons for “Delete‘ and “Connect Annotations,“ which would open a dialog with a PrimeFaces picklist for user selection.
• Add another field next to the one for selecting the child that would select the parent.
• Make it possible to annotate the column with two annotations at the same time.
• Change the sticky header to the sticky footer.
• Change the arrangement showing the cards for each column and make it not
104
vertical but horizontal, so there is less empty space. • Add the delete button of a dataset to the selected dataset page. • Put the delete button inside of a table not above it. • Change the dialog of change name to an input that you can write into. • Put export buttons into sticky header • Change the layout from utilizing cards for each column to a table format. In
the last column, incorporate an “Add Action“ button to prompt the selection of annotations, revealing an annotation list table. The final column should also include an action button for selecting the annotation. • Make the cards explaining which page I am on bigger.
105
106
7 Conclusion
The purpose of the work was to create a Web environment that supported the management, annotation, and upload of a data set on demand. The resulting webbased tool enabled efficient work with the data and supported the evaluation of artificial intelligence models. When the options for inserting, sending and annotating data were analyzed, an appropriate method and format were designed to annotate the data set for download. A database model was created to store these data and to allow the user to view the created data representation and the relationships between them. The implemented web environment allowed one to work with data in a form that is understandable to both humans (user interface) and machines (API). The proposed solution, which was able to store and annotate a data set based on analytical input, was implemented as an online version. The functionality and correctness of the web-based tool were verified and evaluated through tests.
During the analysis phase of the problem, we initially examined the most commonly used technologies for web pages, assessing their respective advantages and disadvantages. Subsequently, I selected the ones that best suited my application. In addition, I explored various cloud providers to host my application, carefully choosing the most suitable one for my needs.
Next came the design phase of my diploma thesis; after conducting the aforementioned analysis, I opted for the main technologies for my application, which include Vue.js for the front end and NestJS for the backend. I utilized two databases, Neo4j and PostgreSQL, and leveraged the GCP platform, specifically Firebase.
107
Additionally, I used Firebase Firestore to store user roles and dataset information, Firebase Storage to store CSV dataset files, Firebase Hosting for website hosting, and Cloud Functions for hosting the backend application.
To visualize the functionality and structure of my application, I created flow diagrams. In addition, I developed low-fidelity wireframes using Figma to outline the appearance of my web pages. Additionally, I crafted a database diagram and diagrams for the Google Cloud Platform to illustrate the deployment architecture of my application, along with diagrams depicting the structure of my CI/CD pipeline.
Following the design phase of my diploma thesis, I transitioned to the implementation stage, where I executed everything according to the design plan. Using the technologies, diagrams, and databases mentioned above, I developed the entire application, transforming wireframes into functional screens.
In conclusion, I ensured the functionality of the application through user testing and various other testing methods. I analyzed optimal deployment regions and compared request speeds between monolithic application and microservices. Furthermore, I assessed the performance of the Neo4j and PostgreSQL databases to determine which was the most suitable. Furthermore, I examined how memory allocation for cloud functions affected their computing performance.
In summary, the application for managing annotations and datasets has been fully developed, rigorously tested, and successfully deployed for user access, meeting the objectives of the diploma thesis. Through testing, Europe-West3 was identified as the optimal deployment region, which is aligned with its proximity to me and the location of the Neo4j Aura instance. Furthermore, the use of microservices proved to be faster compared to a monolithic architecture, as anticipated. This efficiency was achieved by breaking down the application into smaller components.
108
Performance comparison between PostgreSQL and Neo4j revealed that PostgreSQL outperforms Neo4j until the dataset reaches at least 100,000 annotations. However, beyond this threshold, Neo4j surpasses PostgreSQL in annotation operations, including endpoints such as getExport and getAnnotations. This finding was surprising to me, as I initially expected Neo4j to be the faster option. Neo4j’s superiority in these operations becomes more pronounced as the dataset grows larger, indicating that Neo4j performs better with larger datasets. This is because Neo4j’s graph database model is particularly well suited for handling complex relationships and traversing large volumes of interconnected data efficiently. As the dataset size increases, the advantages of Neo4j’s graph-based approach become more apparent, leading to improved performance compared to traditional relational databases like PostgreSQL. Nevertheless, it is important to note that Neo4j’s longer seeding time is attributed to its larger overhead compared to PostgreSQL,

which may affect the overall efficiency of the system during the initial data loading phase.
Moreover, it was discovered that the ideal memory-to-cost ratio for computing power is 1 GB. This finding was unexpected, as it was initially thought that higher memory would equate to better performance. However, the values plateau after 1 GB, suggesting diminishing returns. Additionally, it is worth noting that higher memory allocations in Google Cloud Functions also result in the allocation of more CPU resources, which further influences performance. According to the test results, for basic applications, PostgreSQL was found to be faster and more costeffective.
109
110
8 Future work
In this part, I will discuss potential future directions for my diploma thesis.
To expand the project's capabilities, users can incorporate another database type along with PostgreSQL and Neo4j. Currently, PostgreSQL serves as a relational database, while Neo4j operates as a graph database. To further explore options, users can introduce additional database types for testing and integration. For instance, integrating a document database like MongoDB or incorporating a Column Family Store such as Apache Cassandra would offer greater diversity in database management. Furthermore, incorporating a NewSQL database like SurrealDB would be beneficial. Implementing the repository design pattern would facilitate the seamless utilization of this new database type within the system back-end. Subsequently, developing back-end endpoints to interact with SurrealDB would enable users to take advantage of its functionalities effectively. Finally, rigorous testing would provide insight into its performance and capabilities compared to existing databases.
In the current testing phase, the system is evaluating annotation relationships up to a depth of 5, where the maximum depth represents the sequence "annotation1 -> annotation2 -> annotation3 -> annotation4 -> annotation5", resulting in a maximum depth of 4 annotations for a total of 5 annotations. This depth configuration has been applied consistently across all testing scenarios. To further explore the system's capabilities, additional testing with increased maximum depths, such as 10, 50, and 100, could provide valuable insights. This expanded testing approach would involve altering the annotation sequence, thereby varying the number of annotations assigned to each column. Unlike the current setup where each column receives the same number of annotations in sequence, this testing would introduce
111
variations in the annotation distribution across columns. Make some adjustments to the UI of the application as users required hints to successfully complete the user testing scenarios. In addition, incorporate some of the suggestions gathered from the user testing sessions into the app.
112
Resumé
Úvod
Dátová veda sa stala kľúčovou oblasťou, ktorá zahŕňa využívanie štatistiky, procesov a algoritmov na získavanie cenných poznatkov a vedomostí zo štruktúrovaných aj neštruktúrovaných údajov. S neustále rastúcou dostupnosťou údajov si dátoví vedci vyžadujú robustné infraštruktúry, ktoré uľahčujú efektívne spracovanie súborov údajov a vyhodnocovanie modelov umelej inteligencie (AI). Schopnosť ukladať, organizovať, anotovať a distribuovať súbory údajov sa stáva pre ich prácu nevyhnutnosťou. Okrem toho sa dátová veda snaží integrovať štatistiku, informatiku a súvisiace metodiky s cieľom pochopiť a analyzovať javy reálneho sveta prostredníctvom údajov.
Cieľom tejto práce je vyvinúť webové prostredie špeciálne prispôsobené na podporu potrieb dátových vedcov. Prostredie umožní bezproblémové pridávanie, anotovanie a prenos súborov údajov, čím umožní efektívnu manipuláciu s údajmi a uľahčí vyhodnocovanie modelov umelej inteligencie. Prostredníctvom komplexnej analýzy rôznych metód a formátov na vkladanie, prenos a anotovanie údajov sa v tejto práci určí najvhodnejší prístup. Okrem toho sa navrhne databázový model na ukladanie údajov, ktorý umožní používateľom skúmať vytvorené reprezentácie údajov a vzťahy medzi nimi. Webové prostredie napokon poskytne dátovým vedcom užívateľsky prívetivé rozhranie, ako aj strojovo prívetivé rozhranie (API).
Na dosiahnutie týchto cieľov je nevyhnutné hlboko pochopiť pojmy a terminológiu,
113
ktoré dátoví vedci používajú pri svojej každodennej práci. Dôkladným preskúmaním týchto pojmov a ich vzájomných súvislostí chceme získať cenné poznatky, ktoré budú podkladom pre návrh a vývoj účinného nástroja pre dátových vedcov. Táto práca sa snaží prispieť k oblasti dátovej vedy tým, že rieši naliehavé potreby dátových vedcov a rozvíja možnosti ich infraštruktúry. Prostredníctvom tohto výskumu sa usilujeme uľahčiť lepšiu manipuláciu s údajmi, podporiť vyhodnocovanie modelov umelej inteligencie a v konečnom dôsledku posilniť postavenie dátových vedcov v ich snahe o získavanie a analýzu znalostí z rôznych súborov údajov.
Znalostná báza
Znalostná databáza je centralizované úložisko informácií, ku ktorým majú prístup a ktoré môžu používať jednotlivci alebo systémy. Môže obsahovať širokú škálu informácií vrátane faktov, pravidiel, postupov a osvedčených postupov. Znalostné databázy sa môžu používať v rôznych oblastiach, napríklad v informatike, medicíne a ekonomike, na podporu rozhodovania, riešenia problémov a učenia. Medzi príklady znalostných báz patria expertné systémy, databázy a wiki [4].
Znalostný graf
Znalostný graf je graficky štruktúrovaný dátový model, ktorý sa používa na ukladanie opisov entít, ako sú ľudia, miesta a udalosti. Možno ho definovať ako súbor usporiadaných trojíc (h, r, t), kde h a t sú predná a zadná entita a r je vzťah medzi nimi. Znalostné grafy zohrávajú významnú a rastúcu úlohu v sémantickej podpore širokej škály aplikácií a sú užitočné pri úlohách, ako sú odpovede na otázky a zdôvodňovanie [5]. Využívajú sa aj v ekosystéme Solid na zlepšenie prístupu k podom prostredníctvom rôznych webových rozhraní API, ktoré fungujú ako pohľady do
114
grafu znalostí a poskytujú lepšie možnosti na ukladanie, publikovanie a vyhľadávanie decentralizovaných údajov flexibilnejšími a udržateľnejšími spôsobmi [6]. Správa znalostných grafov sa vyvinula a v súčasnosti existujú platformy na správu KG, ktoré podporujú životný cyklus KG od ich vytvorenia až po ich údržbu a používanie [7].
Klientská časť aplikácie
Pri vývoji webových stránok sa na vytváranie interaktívnych rozhraní používajú populárne jazyky, ako sú JavaScript, HTML/CSS, Python, Java, C# a PHP. Rámce ako PHP Laravel, Symfony, JavaScript React, Vue.js, Python Django atď. zjednodušujú vývoj pomocou vopred pripravených komponentov na úlohy, ako je napríklad validácia formulárov. HTML štruktúruje obsah, CSS ho štylizuje a JavaScript pridáva interaktivitu. Široké využitie a robustný ekosystém JavaScriptu z neho robia najlepšiu voľbu pre skriptovanie. [19]

TypeScript, staticky typovaný jazyk, je v súlade s návrhmi ECMAScriptu, čo zmierňuje problémy pri behu. Node.js vykonáva JavaScript mimo prehliadačov pomocou enginu V8, ktorý je známy svojou rýchlosťou. [24].
Vo vývoji JavaScriptu dominujú frameworky ako Angular, React a Vue. Vue exceluje pre jednoduchosť a výkon, Angular exceluje pri vývoji rozsiahlych aplikácii a React pre svoju všestrannosť a popularitu. [27].
Vue.js sa bude používať spolu s nástrojom vitebuild. Ako je uvedené v [28]. Vite je nástroj na vývoj frontendov, ktorý vytvoril Evan You, zakladateľ Vue.js. Ponúka rýchlejší a efektívnejší vývoj moderných webových projektov. Medzi hlavné funkcie patrí natívna podpora modulov ES, rýchlejšie spustenie vývojového servera, rýchle aktualizácie pomocou funkcie Hot Module Replacement (HMR) a optimali-
115
zované zostavenie pre produkciu. Vite využíva flexibilné rozhranie API zásuvných modulov Rollup pre lepší výkon a flexibilitu. Je kompatibilný s Vue.js, TezJS a React a prebiehajúce vývojové úsilie má za cieľ ďalej zvýšiť jeho výkon. Celkovo Vite vyniká svojou rýchlosťou, zjednodušeným pracovným postupom a záväzkom k moderným postupom vývoja webových stránok.
Serverová čast aplikácie
Pri vývoji webovej aplikácie sa architektonický prístup často zužuje na serverovú alebo bezserverovú architektúru. Bezserverová architektúra zahŕňa Backend ako službu (BaaS) a Funkcie ako službu (FaaS). BaaS outsourcuje funkcie backendu na servery tretích strán, zatiaľ čo FaaS umožňuje písať funkcie špecifické pre aplikáciu bez správy infraštruktúry. Bezserverové modely ponúkajú optimalizáciu nákladov prostredníctvom platby za používanie, hoci sa môžu uplatňovať poplatky za čas nečinnosti [29].
Naproti tomu serverová architektúra nasadzuje aplikácie na vyhradené servery, pričom backend v jazyku JavaScript/TypeScript sa ukazuje ako výhodný z dôvodu kompatibility s frontendom. NestJS, postavený na Node.js a TypeScripte, uľahčuje modulárne a škálovateľné architektúry a podporuje objektovo orientované a funkčné reaktívne programovanie [30].
Výber medzi serverovou a bezserverovou architektúrou závisí od konkrétnych požiadaviek aplikácie. Bezserverové modely sa zameriavajú na aplikačnú logiku, ponúkajú automatické škálovanie, potenciálnu úsporu nákladov a zníženú latenciu [31]. Cloudové funkcie Google s bezplatnými kreditmi a bezproblémovou integráciou s ostatnými službami Google sa ukazujú ako výhodná možnosť [32, 33].
116
Databáza
Výber databázy pre webovú aplikáciu zahŕňa efektívne ukladanie údajov a vzťahov. Medzi možnosti patria relačné databázy ako PostgreSQL, grafové databázy ako Neo4j, nerelačné databázy ako MongoDB a databázy NewSQL ako SurrealDB. Cieľom databáz NewSQL je poskytnúť škálovateľnosť podobnú nerelačným databázam pri zachovaní vlastností ACID [34].
V prípade databáz znalostných grafov sú na výber Neo4j, trojité úložiská RDF, Amazon Neptune a Microsoft Azure Cosmos DB [35]. Neo4j a Amazon Neptune sú vhodné pre znalostné grafy. Neo4j ponúka škálovateľnosť, zhodu s ACID a osvedčený výkon. Amazon Neptune, cloudová databáza grafov, podporuje modely grafov RDF a Gremlin [35, 36].
Výber závisí od potrieb projektu. Neo4j možno nasadiť lokálne alebo na AWS, zatiaľ čo Amazon Neptune je založený na cloude. Vďaka overenému výkonu a flexibilite nasadenia je Neo4j preferovanou voľbou [35, 36].
Neo4j Aura poskytuje plne spravovanú cloudovú službu pre grafové databázy, ktorá ponúka efektívnosť, spoľahlivosť a škálovateľnosť pri nasadení v cloude.[37].
Technológie
Po analýze sme vybrali nasledujúce služby, technológie, databázy a poskytovateľov cloudových služieb:
Front-end Vue.js verzia 3 s TypeScriptom, Vue Router, Vite, Composition API a Pinia. Pre CSS použijeme Tailwind CSS. Tento framework ponúka rýchly vývoj a konzistentné štylizovanie, zatiaľ čo PrimeVue poskytuje všestranné komponenty používateľského rozhrania. Firebase Hosting zabezpečuje zjednodušené nasadenie
117
a škálovateľnosť.
Google Cloud Functions zabezpečujú bezserverové nasadenie backendu, pričom backend je implementovaný v Nest.js. Grafová databáza Neo4j účinne spravuje zložité vzťahy a prechádzanie dát. Na porovnanie výkonu bude použitá aj PostgreSQL.
Firebase Authentication sa stará o zabezpečenie prihlasovania a registrácie, zatiaľ čo Firestore ukladá údaje o rolách používateľov a informácie o datasetoch. Návrhový vzor Adapter zlepšuje udržiavateľnosť a škálovateľnosť kódu, zatiaľ čo vzor Repository umožňuje pridávať iné databázy bez veľkých zmien v kóde.
Hosťovanie na platforme Google Cloud Platform zaručuje spoľahlivosť, škálovateľnosť a celosvetovú dostupnosť.
Model databázy
Na základe článkov [62, 63, 64] vytváram hierarchickú štruktúru údajov v Neo4j a formulujem dátový model. Riadiac sa princípmi lokačných stromov objasnenými v týchto článkoch som sa rozhodol dodržiavať nasledujúce zásady:
• Všetky vzťahy smerujú od detí k rodičom, a to vzostupne v hierarchii.
• Pre všetky vzťahy sa používa jeden typ vzťahu (s názvom related_to).
• Každý uzol má jediný odchádzajúci vzťah k svojmu rodičovi.
• Uzly môžu mať jeden alebo viacero prichádzajúcich vzťahov od svojich potomkov.
Na základe týchto pokynov som usporiadal svoje uzly tak, aby zahŕňali jeden vzťah s názvom ”related_to”. Tento prístup zaručuje hierarchickú stromovú štruktúru, 118
v ktorej má každý uzol len jeden vzťah vedúci k jeho rodičovi. Pritom viacero uzlov môže nadviazať spojenie s tým istým rodičom, čo znamená súrodenecké vzťahy. Na základe tohto modelu Neo4j som následne navrhol model pre databázu PostgreSQL. V nasledujúcom texte sa budem venovať konečným databázovým modelom pre oba systémy.
Na obrázku 8.1 uvádzame názorné zobrazenie modelu databázy vytvoreného pomocou programu diagram.io. Tento diagram poskytuje vizuálne znázornenie štruktúry údajov a vzťahov v rámci databázového systému. Ukazuje integráciu databáz Neo4j, Firebase a PostgreSQL v rámci architektúry aplikácie.
Nasledujúci text opisuje dátový model v Neo4j a firebase so štruktúrou vhodnou pre znalostnú bázu, ktorá uchováva údaje a anotácie používateľov o súboroch údajov. Tu je rozpis toho, čo robia jednotlivé časti:
Firebase:
• Firebase Authentication Users: Reprezentuje používateľa systému.
• Firebase Firestore Database Roles: Predstavuje rolu používateľa, ktorá je priradená k používateľovi pomocou UserUID.
• Firebase Firestore Database Datasets: Predstavuje súbor údajov, ktorý nahral používateľ.

Neo4j:
• Node Dataset: Predstavuje informácie o datasete, ktorý nahral používateľ. Tento dataset je mapovaný na dataset vo firestore pomocou firebaseDatasetID.
• Node Column: Predstavuje informácie o stĺpci datasetu.
• Vzťah HAS_COLUMN: Označuje, že stĺpec patrí ku konkrétnemu data-
119
Figure 8.1: Databázový model 120
setu.
• Node Annotation: Predstavuje anotáciu, ktorú používateľ pridal do našej databázy znalostí.
• Vzťah ANNOTATED_WITH: Označuje, že stĺpec bol anotovaný touto anotáciou.
• Vzťah RELATED_TO: Označuje, že táto anotácia súvisí s inou anotáciou.
V kontexte PostgreSQL sa databázová schéma skladá z niekoľkých tabuliek, z ktorých každá slúži na určitý účel:
• Annotation Table: V tejto tabuľke sú uložené informácie o anotáciách. Obsahuje vzťah, v ktorom sa každá anotácia môže voliteľne odkazovať na inú anotáciu prostredníctvom vzťahu "0 to 1"zo stĺpca parent_annotation_id do stĺpca id v rámci tej istej tabuľky. Okrem toho existuje vzťah "1 to many"zo stĺpca id tabuľky anotácií na cudzí kľúč column_id v tabuľke stĺpec_anotácie.
• Dataset_Column Table: Táto tabuľka je zodpovedná za ukladanie informácií o stĺpcoch datasetu a vytvára vzťah "many to one"so stĺpcom dataset_id na tabuľku dataset. Okrem toho existuje vzťah öne to many"zo stĺpca id tabuľky dataset_column k column_id v tabuľke column_annotation.
• Dataset Table: Ukladá informácie týkajúce sa súborov údajov. Udržiava vzťah zo svojho stĺpca id k stĺpcu dataset_id v tabuľke dataset_column. Okrem toho je mapovaný na datasety v databáze Firebase prostredníctvom svojho id.
• Column_Annotation Table: Táto tabuľka slúži ako mapovanie medzi
121
stĺpcami a im zodpovedajúcimi anotáciami. Vytvára vzťah medzi column_id a stĺpcom id v tabuľke anotácií, čím uľahčuje prepojenie medzi anotáciami a stĺpcami. Okrem toho obsahuje stĺpec annotation_id, ktorý vytvára vzťah so stĺpcom id v tabuľke anotácií, čím sa ďalej zlepšuje prepojenie medzi stĺpcami a ich anotáciami.
Celkovo tento návrh schémy uľahčuje organizáciu a vyhľadávanie údajov týkajúcich sa anotácií, súborov údajov a ich príslušných stĺpcov v databáze PostgreSQL.
Diagram cloudovej infraštruktúry
Diagram na 8.2 možno efektívne predstaviť ako hierarchické zobrazenie architektúry platformy Google Cloud Platform (GCP), ktorá znázorňuje cestu od koncového bodu klienta k databáze Neo4j Aura pomocou viacerých distribuovaných služieb.
Vrchol hierarchie sa začína na strane klienta, z ktorého sa uskutočňuje pripojenie k infraštruktúre GCP prostredníctvom služby Cloud DNS. Tá predstavuje systém doménových mien, základnú internetovú službu, ktorá transformuje ľudsky čitateľné názvy hostiteľov na adresy IP a zabezpečuje, aby klient mohol efektívne komunikovať so zdrojmi GCP.
Po vykonaní rozlíšenia DNS je klient presmerovaný na hosting firebase. Táto služba funguje ako hostiteľ pre front-endovú aplikáciu Vue.js, poskytuje zabudované zabezpečenie a prostredie vhodné pre vývojárov na správu a nasadzovanie webových aplikácií.
Následnou úrovňou je trojica Cloud Functions, prostredie spoločnosti Google na vykonávanie bez servera. Tieto funkcie predstavujú udalosťami riadené výpočtové
122
Figure 8.2: Diagram cloudovej infraštruktúry 123
riešenie, ktoré umožňuje vývojárom vykonávať svoj kód bez potreby spravovať alebo poskytovať servery, čo z neho robí optimálne riešenie pre architektúru mikroslužieb. Každá z cloudových funkcií sa pripája k terminálovej vrstve, ktorá obsahuje tri inštancie Neo4j Aura. Neo4j Aura je plne spravovaná, vždy zapnutá grafová databáza s horizontálnou škálovateľnosťou a vysokou dostupnosťou.
GCP CI/CD diagram
Figure 8.3: GCP CI/CD diagram Na diagrame 8.3 môžeme vidieť pracovný postup kontinuálnej integrácie a kontinuálneho nasadenia (CI/CD) pre aplikáciu, prebieha nasledovne: Na začiatku vývojár vykoná úpravy zdrojového kódu aplikácie vo svojom lokálnom vývojovom prostredí. Po dokončení úprav vývojár vykoná pridanie zmien pomocou príkazu git add v systéme Git a následne zapúzdri zmeny do revízie pomocou príkazu git commit. Následne sú tieto revízie odoslané do vzdialeného úložiska 124
umiestneného v službe Google Cloud Source Repositories pomocou príkazu git push.
Úložisko GitHub slúži na ukladanie zdrojového kódu aplikácie s riadením verzií a sleduje všetky zmeny. Je prepojený s GitHub Actions, takže všetky zmeny v úložisku spúšťajú automatický trigger v GitHub Actions.
V CI/CD pipeline riadi GitHub Actions komplexné nasadenie frontendovej aplikácie Vue.js aj cloudových funkcií. Organizuje procesy zostavovania, testovania a nasadenia, čím zabezpečuje zjednodušený pracovný postup pre celý zásobník aplikácií.
Porovnanie rýchlosti požiadaviek: Neo4j vs. PostgreSQL
V tejto časti budeme skúmať, ako sa líši rýchlosť požiadaviek pri použití Neo4j v porovnaní s PostgreSQL. Testovanie zahŕňalo použitie jedného datasetu so 7 stĺpcami, pričom každý z nich mal maximálnu hĺbku súvisiacich anotácií nastavenú na 4. Parametrom, ktorý sa počas testovania menil, bol počet anotácií v rozsahu od 100 do 100 000 a počet vzťahov zo stĺpcov k týmto anotáciám.
Proces vytvárania prebiehal podľa sekvenčného vzoru, kde každá anotácia bola prepojená s ďalšou v poradí. Napríklad anotácia 1 bola prepojená s anotáciou 2, ktorá bola prepojená s anotáciou 3 a tak ďalej až po anotáciu 5. Potom sa postupnosť začala odznova, pričom anotácia 6 bola prepojená s anotáciou 7 atď. Táto postupnosť sa uplatnila aj na vzťahy medzi stĺpcami a anotáciami, pričom prvý stĺpec bol prepojený s anotáciou 1, druhý stĺpec s anotáciou 2 atď.
Na obrázku 8.4 sú výsledky testovania so 100 anotáciami, ktoré dokazujú vyšší výkon PostgreSQL oproti Neo4j vo všetkých koncových bodoch okrem deleteAnnotation a deleteDataset. Preto sa PostgreSQL ukazuje ako preferovaná možnosť.
125
Figure 8.4: Neo4j vs PostgreSQL zhlukový stĺpcový graf 100 anotácií
Figure 8.5: Neo4j vs PostgreSQL zhlukový stĺpcový graf 1000 anotácií 126
Na obrázku 8.5 sledujeme výsledky testovania s 1000 anotáciami. Tu som sa rozhodol zobraziť len endpointy s najväčším anotačným zaťažením kvôli malým rozdielom v časoch. Opäť sa ukazuje, že PostgreSQL je lepšia možnosť, ktorá prekonáva Neo4j vo všetkých koncových bodoch okrem getAnnotations, kde mali rovnaký čas.

Figure 8.6: Neo4j vs PostgreSQL zhlukový stĺpcový graf 10000 anotácií Na obrázku 8.6 sa pozeráme na výsledky testu s 10000 anotáciami. Tu som zobrazil len tie časti, kde sa anotácie používajú najviac, pretože inde boli len malé rozdiely v časoch. Neo4j je rýchlejší v getAnnotations, ale takmer vo všetkom ostatnom je oveľa pomalší. Na obrázku 8.7 skúmame výsledky testu so 100 000 anotáciami, nášho posledného pokusu. Tu pozorujeme, že pri najintenzívnejších operáciách s anotáciami sa Neo4j ukazuje ako víťaz. Je rýchlejší ako PostgreSQL v úlohe "getExport" a výrazne rýchlejší v úlohe "getAnnotations". V úlohe "import" je však Neo4j v porovnaní s
127
Figure 8.7: Neo4j vs PostgreSQL zhlukový stĺpcový graf 100000 anotácií
PostgreSQL oveľa pomalší.
Pre lepšiu prehľadnosť som všetky výstupy z týchto grafov usporiadali do tabuľky nižšie, kde môžeme vidieť časy. Vľavo môžeme vidieť endpointy, ktoré sme testovali. V hornej časti môžeme vidieť počet anotácií použitých pri testovaní. Potom sme v každom riadku videli dve čísla: prvé predstavovalo čas potrebný pre Neo4j v sekundách a druhé čas potrebný pre PostgreSQL v sekundách.
getExport getAnnotations importTesting
100 1000 10000 100 000
0.85 5.15 121.08 327.46
0.02 0.056 3.94 3769.93
0.11 0.12 1.5
4.11
0.01 0.13 6.15 1153.26
2.06 19.2 269.56 14768.01
0.4 1.72 12.34 146.27
128
Z týchto grafov a tabuľky môžeme vyvodiť nasledujúce zistenia: Ak je anotácií menej ako 100 000, PostgreSQL sa stáva preferovanou databázou vďaka svojej celkovej výhodnosti z hľadiska nákladov a rýchlosti. Po prekročení hranice 100 000 anotácií však Neo4j začína vykazovať vyšší výkon, aj keď za cenu vyšších nákladov.
Dôvodom, prečo sa PostgreSQL stáva preferovanou voľbou pre databázu pri počte anotácií menšom ako 100 000, je jej efektívnosť pri spracovaní menších súborov údajov. PostgreSQL je známa svojou robustnosťou a rýchlosťou, najmä pri správe štruktúrovaných údajov a efektívnom vykonávaní štandardných operácií SQL.
Keď súbor údajov narastie nad hranicu 100 000 anotácií, architektúra grafovej databázy Neo4j začne ukazovať svoje výhody. Neo4j vyniká v spracovaní zložitých vzťahov a prechádzaní prepojených údajov, vďaka čomu je vhodnejšia na úlohy, ako je dopytovanie grafových štruktúr. Táto prirodzená výhoda sa stáva výraznejšou s rastúcou veľkosťou súboru údajov a zložitosťou vzťahov.
Treba poznamenať, že hoci Neo4j môže byť rýchlejší v databázových operáciách, ako je napríklad "getAnnotations", výrazne zaostáva za PostgreSQL v úlohe "import". To naznačuje, že hoci Neo4j môže mať dlhší čas spracovania údajov kvôli väčšej réžii, vyniká v rýchlejších operáciách s väčším množstvom údajov.
Zhodnotenie
Cieľom práce bolo vytvoriť webové prostredie, ktoré by podporovalo správu, anotovanie a nahrávanie súboru údajov na požiadanie. Výsledný webový nástroj umožňoval efektívnu prácu s údajmi a podporoval vyhodnotenie modelov umelej inteligencie. Analýzou možností vkladania, odosielania a anotovania údajov bol navr-
129
hnutý vhodný spôsob a formát anotovania súboru údajov a jeho následného odosielania. Bol vytvorený databázový model na ukladanie týchto údajov a umožňujúci používateľovi zobraziť vytvorenú reprezentáciu údajov a vzťahy medzi nimi. Implementované webové prostredie umožnilo pracovať s údajmi vo forme zrozumiteľnej pre človeka (používateľské rozhranie) aj pre stroj (API). Navrhnuté riešenie, ktoré bolo schopné ukladať a anotovať súbor údajov na základe analytických vstupov, bolo implementované ako online verzia. Funkčnosť a správnosť webového nástroja sa overila a vyhodnotila prostredníctvom testov.
Vo fáze analýzy problému som najprv skúmal najčastejšie používané technológie pre webové stránky a posudzoval ich výhody a nevýhody. Následne som vybral tie, ktoré sú pre moju aplikáciu najvhodnejšie. Okrem toho som preskúmal rôznych poskytovateľov cloudových služieb na umiestnenie mojej aplikácie a starostlivo som vybral najvhodnejšieho pre moje potreby.
Potom prišla fáza návrhu mojej diplomovej práce, po vykonaní vyššie uvedenej analýzy som sa rozhodol pre hlavné technológie mojej aplikácie, medzi ktoré patrí Vue.js pre frontend a NestJS pre backend. Využil som dve databázy, konkrétne Neo4j a PostgreSQL, a využil som platformu GCP, konkrétne Firebase. Okrem toho som použil Firebase Firestore na ukladanie používateľských rolí a informácií o súboroch údajov, Firebase Storage na ukladanie súborov súborov údajov CSV, Firebase Hosting na hosting webových stránok a Cloud Functions na hosting backendovej aplikácie.
Na vizualizáciu funkčnosti a štruktúry mojej aplikácie som vytvoril viaceré diagramy. Okrem toho som vytvoril low-fidelity wireframy pomocou programu Figma, aby som načrtol vzhľad svojich webových stránok. Okrem toho som vytvoril databázový diagram a diagramy pre platformu Google Cloud Platform na znázornenie architektúry nasadenia mojej aplikácie spolu s diagramami znázorňujúcimi štruk-
130
túru mojej CI/CD pipeline.
Po fáze návrhu mojej diplomovej práce som prešiel do fázy implementácie, kde som všetko vykonal podľa navrhnutého plánu. S využitím už spomínaných technológií, diagramov a databáz som vyvinul celú aplikáciu, pričom som transformoval wireframy na funkčné obrazovky.
Na záver som zabezpečil funkčnosť aplikácie prostredníctvom používateľského testovania a rôznych ďalších testovacích metód. Analyzoval som optimálne oblasti nasadenia a porovnal som rýchlosť požiadaviek medzi monolitickou aplikáciou a mikroslužbami. Okrem toho som posúdil výkonnosť databáz Neo4j a PostgreSQL, aby som určil, ktorá z nich je vhodnejšia. Okrem toho som skúmal, ako prideľovanie pamäte pre cloudové funkcie ovplyvňuje ich výpočtový výkon.
Zhrnutie, možno konštatovať, že aplikácia na správu anotácií a súborov údajov bola kompletne vyvinutá, dôkladne otestovaná a úspešne nasadená na prístup používateľov, čím boli splnené ciele diplomovej práce. Prostredníctvom testovania bola ako optimálna oblasť nasadenia identifikovaná eurpe-west3, čo je v súlade s jej blízkosťou ku mne a umiestnením inštancie Neo4j Aura. Okrem toho sa ukázalo, že použitie mikroslužieb je rýchlejšie v porovnaní s monolitickou architektúrou, ako sa predpokladalo. Táto efektívnosť sa dosiahla rozdelením aplikácie na menšie komponenty.

Porovnanie výkonu PostgreSQL a Neo4j ukázalo, že PostgreSQL prekonáva Neo4j, kým súbor údajov nedosiahne aspoň 100 000 anotácií. Po prekročení tejto hranice však Neo4j prekonáva PostgreSQL v anotačných operáciách vrátane endpointov, ako sú getExport a getAnnotations. Toto zistenie ma prekvapilo, pretože som pôvodne očakával, že Neo4j bude rýchlejšia možnosť. Prevaha Neo4j v týchto operáciách sa stáva výraznejšou s rastúcim súborom údajov, čo naznačuje, že Neo4j dosahuje lepšie výsledky pri väčších súboroch údajov. Je to preto, že model gra-
131
fovej databázy Neo4j je mimoriadne vhodný na spracovanie zložitých vzťahov a efektívne prechádzanie veľkých objemov prepojených údajov. S rastúcou veľkosťou súboru údajov sa výhody prístupu Neo4j založeného na grafoch stávajú zjavnejšími, čo vedie k lepšiemu výkonu v porovnaní s tradičnými relačnými databázami, ako je PostgreSQL. Napriek tomu je dôležité poznamenať, že dlhší čas načítavania Neo4j sa pripisuje jeho väčšej réžii v porovnaní s PostgreSQL, čo môže ovplyvniť celkovú efektívnosť systému počas počiatočnej fázy načítavania údajov. Okrem toho sa zistilo, že ideálny pomer pamäte a nákladov na výpočtový výkon je 1 GB. Toto zistenie bolo neočakávané, pretože pôvodne sa predpokladalo, že väčšia pamäť sa bude rovnať vyššiemu výkonu. Hodnoty sa však po 1 GB ustálili, čo naznačuje klesajúcu návratnosť. Okrem toho stojí za zmienku, že vyššie alokácie pamäte v Google Cloud Functions vedú aj k alokácii väčšieho množstva zdrojov CPU, čo ďalej ovplyvňuje výkon. Podľa výsledkov testov sa v prípade bežného používania
ukázalo, že PostgreSQL je rýchlejšia a úspornejšia voľba.
132
Bibliography
[1] Shenai Krishna. Introduction to database and knowledge-base systems. Zv. 28. World scientific, 1992.
[2] Lawrence Reeve a Hyoil Han. „Survey of semantic annotation platforms“. In: Proceedings of the 2005 ACM symposium on Applied computing. 2005, s. 1634–1638.
[3] Rudi Studer, V Richard Benjamins a Dieter Fensel. „Knowledge engineering: Principles and methods“. In: Data & knowledge engineering 25.1-2 (1998), s. 161–197.
[4] Thaddeus J. Kowalski a Donald E. Thomas. „The VLSI Design Automation Assistant: What's in a Knowledge Base“. In: 22nd ACM/IEEE Design Automation Conference (1985), s. 252–258. url: https://api.semanticscholar. org/CorpusID:7374804.
[5] Michael R. Douglas et al. „What Is Learned in Knowledge Graph Embeddings?“ In: Complex Networks & Their Applications X. Ed. Rosa Maria Benito et al. Cham: Springer International Publishing, 2022, s. 587–602. isbn: 978-3-030-93413-2.
[6] Ruben Dedecker et al. „What's in a Pod? A Knowledge Graph Interpretation For The Solid Ecosystem“. In: QuWeDa@ISWC. 2022. url: https://api. semanticscholar.org/CorpusID:254248012.
[7] Samira Babalou et al. „What is needed in a Knowledge Graph Management Platform? A survey and a proposal“. In: 2022. url: https://api. semanticscholar.org/CorpusID:248980771.
[8] url: https://www.indeed.com/career- advice/career- development/ static-vs-dynamic-website.
133
[9] Static vs dynamic websites: Key differences. Nov. 2022. url: https : / / www.pluralsight.com/blog/creative-professional/static-dynamicwebsites-theres-difference. [10] What is open source? url: https://www.redhat.com/en/topics/opensource/what-is-open-source. [11] The open source definition. Feb. 2023. url: https : / / opensource . org / osd/. [12] Ritesh Ranjan. What is a framework in Programming and Why You Should Use one. Jan. 2023. url: https : / / www . netsolutions . com / insights[108»]/ what-is-a-framework-in-programming/. [13] Babak Bashari Rad, Harrison John Bhatti a Mohammad Ahmadi. „An introduction to docker and analysis of its performance“. In: International Journal of Computer Science and Network Security (IJCSNS) 17.3 (2017), s. 228. [14] David Gourley et al. HTTP: the definitive guide. O'Reilly[«108]Media, Inc., 2002. [15] Qinwen Hu, Muhammad Rizwan Asghar a Nevil Brownlee. „A Large-Scale Analysis of HTTPS Deployments: Challenges, Solutions, and Recommendations“. In: J. Comput. Secur. 29.1
(jan. 2021), s. 25–50. issn: 0926-227X. doi: 10.3233/JCS-200070. url: https://doi.org/10.3233/JCS-200070. [16] Rushank Shah a Stevina Correia. „Encryption of Data over HTTP (Hypertext Transfer Protocol)/HTTPS (Hypertext Transfer Protocol Secure) Requests for Secure Data transfers over the Internet“. In: aug. 2021, s. 587–590. doi: 10.1109/RTEICT52294.2021.9573978. [17] Audun Jøsang. „A consistent definition of authorization“. In: Security and
Trust Management: 13th International Workshop, STM 2017, Oslo, Norway, September 14–15, 2017, Proceedings 13. Springer. 2017, s. 134–144. [18] auth0.com. JSON web tokens introduction. url: https://jwt.io/introduction. [19] Stack overflow developer survey 2022. Jún 2022.
134
[20] Dasari Hermitha Curie et al. „Analysis on Web Frameworks“. In: Journal of Physics: Conference Series. Zv. 1362. 1. IOP Publishing. 2019, s. 012114.
[21] Chuck Musciano a Bill Kennedy. HTML & XHTML: The Definitive Guide: The Definitive Guide. O'Reilly Media, Inc., 2002.
[22] Jennifer Kyrnin. „What is HTML 5“. In: Saatavissa: http://webdesign. about. com/od/html5/qt/what_is_html5. htm [viitattu 2.2. 2011] ().
[23] Eric A Meyer. CSS: The Definitive Guide: The Definitive Guide. O'Reilly Media, Inc., 2006.
[24] Paul Wilton. Beginning JavaScript. John Wiley & Sons, 2004. [25] Remo H Jansen. Learning TypeScript. Packt Publishing Ltd, 2015. [26] Basarat Syed. Beginning Node. js. Apress, 2014. [27] Elar Saks. „JavaScript Frameworks: Angular vs React vs Vue.“ In: (2019). [28] url: https://vitejs.dev/guide/why.html. [29] Michael Roberts a John Chapin. What is Serverless? O'Reilly Media, Incor-
porated, 2017. [30] url: https://docs.nestjs.com/. [31] Hassan B Hassan, Saman A Barakat a Qusay I Sarhan. „Survey on serverless computing“. In: Journal of Cloud Computing 10.1 (2021), s. 1–29. [32] Coralogix. Aws Lambda vs Azure Functions vs google cloud functions. Mar. 2023. url: https : / / coralogix . com / blog / aws - lambda - vs - azure functions-vs-google-cloud-functions/. [33] Chris Tozzi. Compare aws lambda vs. Azure Functions vs. Google Cloud Functions: TechTarget. Júl 2021. url: https : / / www . techtarget . com / searchcloudcomputing/tip/Compare-AWS-Lambda-vs-Azure-Functionsvs-Google-Cloud-Functions. [34] Andrew Pavlo a Matthew Aslett. „What's really new with NewSQL?“ In: ACM Sigmod Record 45.2 (2016), s. 45–55.
135
[35] Santiago Timón-Reina, Mariano Rincón a Rafael Martínez-Tomás. „An overview of graph databases and their applications in the biomedical domain“. In: Database 2021 (2021).
[36] System properties comparison Amazon Neptune vs. Graphdb vs. Neo4j. url:

https : / / db - engines . com / en / system / Amazon + Neptune % 5C % 3BGraphDB % 5C%3BNeo4j. [37] Apr. 2023. url: https : / / neo4j . com / cloud / platform / aura - graph database/. [38] Ross Harmes a Dustin Diaz. „The Adapter Pattern“. In: Pro JavaScript Design Patterns (2008), s. 149–158. [39] John Hunt a John Hunt. „Adapter Pattern“. In: Scala Design Patterns: Patterns for Practical**[18»]**Reuse and Design (2013), s. 169–181. [40] Oludare Isaac Abiodun et al. „State-of-the-art in artificial neural network applications: A survey“. In: Heliyon 4.11 (2018), e00938. issn: 2405-8440. doi: https://doi.org/10.1016/j.heliyon.2018.e00938. url: https: //www.sciencedirect.com/science/article/pii/S2405844018332067. [41] Mustafa Ergen et al. „What is artificial intelligence?**[«18]**Technical considerations and future perception“. In: Anatolian J. Cardiol 22.2 (2019), s. 5–7. [42] Mahalakshmi Neelam. „Neelam MahaLakshmi (2021) Aspects of Artificial
Intelligence In Karthikeyan. J, Su-Hie Ting and Yu-Jin Ng (eds),"Learning Outcomes of Classroom Research" p: 250-256, L'Ordine Nuovo Publication, India“. In: (2022). [43] Batta Mahesh. „Machine learning algorithms-a review“. In: International Journal of Science and Research (IJSR).[Internet] 9 (2020), s. 381–386. [44] Bing Liu a Bing Liu. Supervised learning. Springer, 2011. [45] Peter Dayan, Maneesh Sahani a Grégoire Deback. „Unsupervised learning“. In: The MIT encyclopedia of the cognitive sciences (1999), s. 857–859.
136
[46] Xiaojin Zhu a Andrew B Goldberg. „Introduction to semi-supervised learning“. In: Synthesis lectures on artificial intelligence and machine learning 3.1 (2009), s. 1–130.
[47] Neha Gupta et al. „Artificial neural network“. In: Network and Complex Systems 3.1 (2013), s. 24–28.
[48] Xing Wu, Paweł Różycki a Bogdan M. Wilamowski. „A Hybrid Constructive Algorithm for Single-Layer Feedforward Networks Learning“. In: IEEE Transactions on Neural Networks and Learning Systems 26.8 (2015), s. 1659– 1668. doi: 10.1109/TNNLS.2014.2350957.
[49] Zhihua Zhang a Zhihua Zhang. „Artificial neural network“. In: Multivariate time series analysis in climate and environmental research (2018), s. 1–35.
[50] Moshe Leshno et al. „Multilayer feedforward networks with a nonpolynomial activation function can approximate any function“. In: Neural Networks 6.6 (1993), s. 861–867. issn: 0893-6080. doi: https : / / doi . org / 10 . 1016 / S0893 - 6080(05) 80131 - 5. url: https : / / www . sciencedirect . com / science/article/pii/S0893608005801315.
[51] Elizabeth D Liddy. „Natural language processing“. In: (2001). [52] KR1442 Chowdhary a KR Chowdhary. „Natural language processing“. In: Fundamentals of artificial intelligence (2020), s. 603–649. [53] Murray Shanahan. „Talking About Large Language Models“. In: arXiv pre-print arXiv:2212.03551 (2022). [54] Christian Janiesch, Patrick Zschech a Kai Heinrich. „Machine learning and deep learning“. In: Electronic Markets 31.3 (2021), s. 685–695. [55] Saman Razavi. „Deep learning, explained: Fundamentals, explainability, and bridgeability to process-based modelling“. In: Environmental Modelling and Software 144 (2021), s. 105159. issn: 1364-8152. doi: https://doi.org/
137
10.1016/j.envsoft.2021.105159. url: https://www.sciencedirect. com/science/article/pii/S1364815221002024. [56] Andreas C Neves et al. „A new approach to damage detection in bridges using machine learning“. In: Experimental Vibration Analysis for Civil Structures: Testing, Sensing, Monitoring, and Control 7. Springer. 2018, s. 73–84. [57] Joseph Awoamim Yacim a Douw Gert Brand Boshoff. „Impact of artificial neural networks training algorithms on accurate prediction of property values“. In: Journal of Real Estate Research 40.3 (2018), s. 375–418. [58] Welcome to schema.org. url: https://schema.org/. [59] Your machine learning and Data Science Community. Feb. 2010. url: https: //www.kaggle.com/. [60] Kenza Kellou-Menouer et al. „A survey on semantic schema discovery“. In: The VLDB Journal 31.4 (2022), s. 675–710. [61] Apr. 2023. url: https://research.com/software/mobile-vs-desktopusage. [62] url: https://neo4j.com/graphgists/my-bea/. [63] url: https://neo4j.com/graphgists/product-hierarchy-graphgist/. [64] Enzo.
What is a knowledge graph? Nov. 2023. url: https://neo4j.com/ blog/what-is-knowledge-graph/. [65] Niels de Jong. 15 tools for visualizing your neo4j graph database. Jún 2023. url: https://neo4j.com/developer-blog/15-tools-for-visualizingyour-neo4j-graph-database/. [66] neo4j-contrib. NEO4J-contrib/neovis.js: Neo4j + vis.js = neovis.js. graph visualizations in the browser with data from neo4j. url: https://github. com/neo4j-contrib/neovis.js. [67] Visjs. Visjs/vis-network:dizzy: Display dynamic, automatically organised, customizable network views. url: https://github.com/visjs/vis-network.
138
[68] Geoff839. EVS - one electric vehicle dataset - smaller. Aug. 2020. url: https : / / www . kaggle . com / datasets / geoffnel / evs - one - electric vehicle-dataset.
[69] C-3PO. A small covid-19 dataset. Máj 2021. url: https://www.kaggle. com/datasets/aditeloo/a-small-covid19-dataset.
[70] pc1975. Small dataset about used Fiat 500 sold in Italy. Apr. 2020. url: https://www.kaggle.com/datasets/paolocons/small-dataset-aboutused-fiat-500-sold-in-italy.
[71] url: https://openai.com/chatgpt. [72] url: https : / / github . com / feiqi9047 / Project - Conversion / blob / master/KAG_conversion_data.csv. [73] url: https://github.com/araj2/customer- database/blob/master/ Ecommerce%20Customers.csv. [74] Muhammad Shahid Azeem. Customer conversion dataset for Stuffmart.com. Okt. 2023. url: https://www.kaggle.com/datasets/muhammadshahidazeem/ customer-conversion-dataset-for-stuffmart-com?select=customer_ conversion_traing_dataset%2B.csv. [75] Jainds. Eda-for-conversion-rate-dataset. url: https://github.com/jainds/ eda- for- conversion- rate- dataset/blob/master/data/conversion_ data.csv.
139