UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2024 Makarenko Volodymyr

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Mikroslužba pro správu multimediální dat pro trénování neuronových sítí

Bakalářská práce

2024 Makarenko Volodymyr

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Volodymyr Makarenko**
Osobní číslo:	**I20123**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Mikroslužba pro správu multimediální dat pro trénování neuronových sítí**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem této práce je vytvoření mikroslužby do systému pro přípravu multimediálních data setů, které budou sloužit pro trénování neuronových sítí. Mikroslužba bude umožňovat správu multimediálních dat a metadat doplněných uživatelem.

Rozsah pracovní zprávy: **30**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Joseph Ingeno, Software Architect's Handbook, Birmingham Packt Publishing Ltd., ISBN 978-1-78862--406-0
Eric J. Braude, Michael E. Bernstein, Software Engeneering Modern Approaches, Boston University, Metropolitan College, ISBN 978-1-4786-3230-6
KROENKE, David a AUER, David J. Databáze. Přeložil Jakub GONER. Brno: Computer Press, 2015. ISBN 978-80-251-4352-0.
POKORNÝ, Jaroslav a VALENTA, Michal. Databázové systémy. 2. přepracované vydání. Praha: Česká technika – nakladatelství ČVUT, 2020. ISBN 978-80-01-06696-6.

Vedoucí bakalářské práce: **Ing. Martin Pozdílek, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Práci s názvem „Mikroslužba pro správu multimediální dat pro trénování neuronových sítí“ jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne XX. XX. 20XX

Makarenko Volodymyr.

PODĚKOVÁNÍ

Děkuji vedoucímu mojí bakalářské práce Ing. Martinovi Pozdilku, Ing., za pomoc, cenné rady, odborné vedení, poradenství a trpělivost při psáni bakalářské práce. Dále děkuji rodině a přátelům za podporu a pomoc během studia.

ANOTACE

Cílem této práce je vytvořit funkční aplikaci, která bude obsahovat nástroje sloužící pro zajištění činnosti služby správu multimediálních dat pro trénování neuronových sítí. Aplikace poskytne uživatelům přístup k systému podle oprávnění, správu projektů, datových sad a mediálních souborů.

Teoretická část bude obsahovat rešerši, analýzu a srovnání s systémy zabývajícími se danou problematikou.

Praktická část bude zahrnovat popis použitých technologií, návrh databáze a ER diagram. Pro vytvoření aplikace bude použit programovací jazyk Java a databáze MongoDB.

KLÍČOVÁ SLOVA

Webová aplikace, mikroslužba, multimediální služba, Spring framework, Spring Boot, MongoDB, Docker.

TITLE

Microservice for multimedia data management for neural network training.

ANNOTATION

The goal of this work is to create a functional application that will contain tools used to ensure the operation of the multimedia data collection service for training neural networks. The application will provide users with permission-based system access, management of projects, datasets and media files. The theoretical part will contain research, analysis and comparison with systems dealing with the given issue. The practical part will include a description of the technologies used, a database design and an ER diagram. The Java programming language and the MongoDB database will be used to create the application.

KEYWORDS

Web application, microservice, multimedia service, Spring framework, Spring Boot, MongoDB, Docker.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

AI	Artifical intelligence
BSON	Binary-encoded serialization of JSON-like documents
CNN	Convolutional neural network
DDL	Data Definition Language
DML	Data Manipulation Language
ETL	Extract, transform and load
HOG	Histogram of Oriented Gradients
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IoC	Inversion of control
JSON	JavaScript Object Notation
ML	Machine Learning
MQL	MongoDB Query Language
MVC	Model-View-Controller
NoSQL	Non-SQL
OpenCV	Open source computer vision library
PAC	Probably Approximately Correct
REST	Representational state transfer
SQL	Structured Query Language
SVB	Support vector machine
SVM	Support vector machine
VM	Virtual Machine
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

ÚVOD

V oblasti strojového učení a neuronových sítí je správa a příprava dat klíčová pro úspěšný vývoj a trénování modelů. Tato bakalářská práce se zaměřuje na vývoj mikroslužby, která by sloužila pro správu multimediálních datových sad potřebných pro trénování neuronových sítí.

Hlavním cílem této práce je návrh a implementace systému, který umožní efektivní manipulaci s multimediálními daty a souvisejícími metadaty. Tento systém bude integrován do širšího ekosystému pro trénování neuronových sítí, což uživatelům umožní efektivně spravovat a připravovat data potřebná pro jejich výzkumné a vývojové projekty. Systém navržen tak, aby podporoval rozličné formáty dat a byl schopen zvládat velké objemy informací bez újmy na výkonu.

Pro realizaci tohoto projektu bude použit moderní software a technologie, včetně programovacího jazyka Java, databáze MongoDB a platformy Docker, které společně přispějí k robustní a škálovatelné architektuře mikroslužeb.

1 Úvod do umělé inteligence

1.1 Inteligence

Otázky týkající se myšlení, inteligence a existence lidského života jsou fascinující a historicky diskutované. Existuje touha vytvořit umělý život nebo inteligenci, což se odráží nejen v literatuře, ale i ve snahách alchymistů či vynálezců jako Leonardo da Vinci.

Průzkum biologické inteligence začal zkoumáním jejího fyzického substrátu, jako mozku a nervového systému, což vedlo k postupnému posunu představy o sídle "duše" k mozku. Umělá inteligence se začala vyvíjet s rozvojem počítačů, které se ukázaly schopné řešit různé problémy. Existují různé přístupy k vytváření umělé inteligence, od symbolických modelů a expertních systémů po modelování fungování lidského mozku pomocí umělých neuronových sítí.

Hodnocení úrovně inteligence umělého systému je složité, ať už se jedná o testování pomocí Turingova testu nebo zkoumání jeho schopnosti řešit problémy, učit se, adaptovat se a předvídat. Umělá inteligence může být rozdělena na úzkou, specifickou inteligenci, která je omezena na řešení konkrétních problémů, a na obecnou, silnou inteligenci, která je schopná myslit a jednat samostatně.[13]

1.2 Umělá inteligence

Umělá inteligence čerpá z různých oborů, jakými jsou pravděpodobnost a statistika, metody rozhodování, teorie fuzzy množin, umělé neuronové sítě, umělé formální jazyky, teorie rozhodování a teorie her, a také teorie grafů. Tyto oblasti spolu úzce souvisejí a vzájemně se ovlivňují, což umožňuje vytváření sofistikovaných systémů umělé inteligence. [13][28]

2 Strojové učení

Strojové učení (ML) je oblast umělé inteligence, která se zabývá vývojem a studiem statistických algoritmů, které se mohou učit z dat a generalizovat na neviděná data, a tak vykonávat úkoly bez explicitních instrukcí. Nedávno umělé neuronové sítě dokázaly překonat mnoho předchozích přístupů ve výkonu.

ML nachází uplatnění v mnoha oblastech, včetně zpracování přirozeného jazyka, počítačového vidění, rozpoznávání řeči, filtrování e-mailů, zemědělství a medicíny. Když je aplikováno na obchodní problémy, je známé pod názvem prediktivní analytika. I když ne všechno strojové učení je založeno na statistice, výpočetní statistika je důležitým zdrojem metod oboru.

Matematické základy ML poskytují metody matematické optimalizace (matematického programování). Dolování dat je související (paralelní) oblast studia, která se zaměřuje na průzkumnou analýzu dat prostřednictvím učení bez učitele.

Z teoretického hlediska poskytuje pravděpodobně přibližně správné (PAC) učení rámec pro popis strojového učení.[28]

2.1 Metody strojového učení

2.1.1 Učení s učitelem

Učení s učitelem (supervised learning), využívá označené datové sady k trénování algoritmů klasifikace dat nebo přesné predikce výsledků. Během tohoto procesu model upravuje své parametry tak, aby byl co nejlépe přizpůsoben vstupním datům. To probíhá jako součást křížové validace, aby se zajistilo, že model byl natrénován kvalitně. Řízené učení pomáhá organizacím řešit různé reálné problémy na velké škále, například třídění spamu do samostatné složky ve vaší e-mailové schránce. Některé z metod používaných v řízeném učení zahrnují neuronové sítě, naivní Bayesův přístup, lineární regresi, logistickou regresi, náhodný les a podpůrné vektorové stroje (SVM)

2.1.2 Učení bez učitele

Učení bez učitele (unsupervised learning), využívá algoritmů strojového učení k analýze a seskupování nenávratných sad dat (podskupin nazývaných shluky). Tyto algoritmy odhalují skryté vzory nebo skupiny dat bez zásahu člověka. Schopnost tohoto přístupu rozpoznávat podobnosti a rozdíly v datech ho činí ideálním pro průzkumnou analýzu dat, segmentaci a také rozpoznávání obrazů a vzorů.

2.1.3 Posilované učení

Posilované učení (reinforcement learning) je metoda strojového učení, která se liší od učení s učitelem tím, že algoritmus nepoužívá předem označená data k učení. Místo toho se model učí na základě výsledků svých akcí a získané zkušenosti. Postupem času tímto způsobem model postupně vytváří strategie a rozhoduje o řešení konkrétních úkolů.[29][27]

2.2 Počítačové vidění

Počítačové vidění je multidisciplinární oblast, která se zabývá technikami a metodami zpracování a interpretace digitálních obrazů a videí pomocí počítačů. Jeho hlavním cílem je umožnit počítačům porozumět okolnímu světu prostřednictvím vizuálních dat z různých zdrojů, jako jsou kamery, skenery nebo mikroskopy. Tato oblast má široké využití v mnoha odvětvích, včetně medicíny, robotiky, bezpečnosti, průmyslu, autonomních vozidel, rozpoznávání obličejů a dalších. Existuje několik běžných úkolů v počítačovém vidění, včetně detekce hran, klasifikace, segmentace obrazu a rozpoznávání vzorů. Tyto úkoly lze řešit tradičními algoritmy, ale moderní přístupy často využívají strojového učení a hlubokých neuronových sítí, což umožňuje automatické učení z dat a dosažení vysoké přesnosti. Tato kapitola se zaměřuje na využití neuronových sítí k řešení některých typických úkolů v počítačovém vidění.[2]

2.3 Jaké úkoly řeší?

Časté úkoly umělé inteligence jsou klasifikace, regrese, segmentace, detekce, shluková analýza, optimalizace.

Účelem **klasifikace** je přenos vstupních dat do jedné skupiny z předem definovaných skupin.

Regrese se zabývá předpovídáním výsledku na základě vstupních hodnot pomocí předpovědního modelu vyškoleného na základě výukových dat.

Segmentace je rozdělení vstupních dat do skupin nebo segmentů na základě jejich charakteristik.

Detekce se zabývá určením objektů v datech

Shluková analýza se zabývá rozdělením měření různých objektů do tříd.

Optimalizace se zabývá hledáním řešení, která odpovídají maximálním nebo minimálním hodnotám vyhodnocovací funkce.[8]

2.3.1 Klasifikace obrazu

Modely hlubokých neuronových sít mají schopnost dosahovat lepší přesnosti a obecnosti ve srovnání s tradičními metodami klasifikace, zejména pokud jsou dostatečně trénovány na relevantních datech. Při identifikaci například ručně psaných číslic jsme využili plně propojenou neuronovou síť. Tato architektura sítě však není nejvhodnější pro práci s rozměrnějšími obrazovými daty, především kvůli velikosti jednotlivých vrstev a nedostatečnému zohlednění prostorového uspořádání dat. Plně propojené sítě, když zkoumají vztahy mezi pixely, obvykle nedokáží rozlišit mezi sousedními pixely a těmi v opačných částech obrázku. V této kapitole se zaměříme na konvoluční neuronové sítě (CNN), které lépe zvládají práci s prostorově uspořádanými daty a jsou tak efektivnější při učení i v reálném provozu. Typickou úlohou je identifikace druhů, chorob, nebo vad, jako například rozpoznávání chorob rostlin.[30]

Když se člověk dívá na obrázek, jeho mozek se soustředí na klíčové charakteristiky a často ignoruje mnoho detailů. Například při pohledu na obrázek domu si povšimneme důležitých prvků, jako jsou okna, dveře, střecha a stěny, přičemž barva a okolí nejsou tak důležité. Pro rychlé a efektivní rozpoznání jsou klíčové vlastnosti a jejich vzájemná poloha důležitější než ostatní detaily. Nicméně u jiných obrázků může být barva a tvar klíčovými faktory pro identifikaci, zatímco vzájemná poloha nemusí hrát takovou roli. Naše neuronová síť se naučila rozpoznávat objekty podle těchto důležitých charakteristik.[10]

2.3.2 Detekce

Další úlohou, která následuje po klasifikaci obrázků, je **lokalizace**. Představme si například, že máme obrázek pláže a chceme identifikovat místo, kde se nachází slunečník. Neuronová síť nám nejen řekne, že na obrázku je slunečník, ale také nám poskytne informaci o tom, kde se nachází slunečník v obraze, obvykle v podobě souřadnic určujících jeho polohu ve formě obdélníku.

Obrázek 1: Lokalizace (zdroj – vlastní)

Pokročilejší úlohou je detekce objektů, kde neuronová síť nejen identifikuje objekty na obrázku, ale také určuje jejich umístění a typ. Tato schopnost má široké uplatnění v autonomních vozidlech, bezpečnostních systémech, monitorování osob a průmyslových procesech. Dříve se k této úloze využívaly tradiční metody jako HOG a kaskádové klasifikátory.[3]

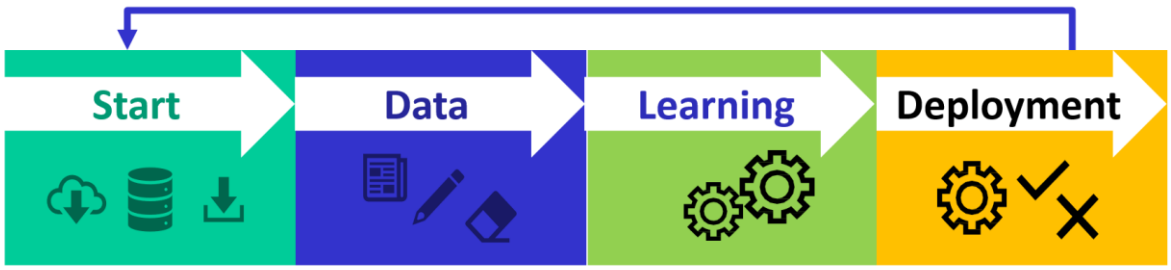
2.3.3 Segmentace

Segmentace obrazu je proces rozdělení digitálního obrázku na různé části nebo oblasti s podobnými vlastnostmi. Cílem tohoto procesu je identifikovat a oddělit jednotlivé objekty nebo regiony v obrázku, což umožňuje další analýzu a porozumění jeho obsahu. Tato technika má široké uplatnění v mnoha odvětvích, jako je medicína, průmysl, bezpečnost a počítačová grafika. Kvalita segmentace je klíčová pro úspěšné zpracování a manipulaci s obrázkem, ačkoliv může být ovlivněna různými faktory, jako je osvětlení, šum nebo překážky v obraze.

Zatímco při detekci objektů je výstupem klasifikovaný obdélník, při segmentaci je každý pixel přiřazen do určité třídy, která může zahrnovat i pozadí. Výstupní data jsou tedy obvykle podobně velká jako vstupní data. Často se výstupem segmentace stává matice příznaků jednotlivých pixelů nebo množina polygonů, které vymezují hranice jednotlivých segmentů.[23]

2.4 Projekt umělé inteligence

Projekt v oblasti umělé inteligence prochází podobnými etapami jako běžný softwarový projekt, ale s určitými odlišnostmi v pojmenování. Začíná se definicí projektu, jeho cílů, požadavků a dopadů na firemní procesy. Financování projektu je klíčové, stejně jako zajištění přístupu k relevantním a kvalitním datům z různých zdrojů.



Obrázek 2: fáze AI projektu (zdroj – vlastní)

Příprava dat je důležitou fází, která připomíná proces ETL (extrakce, transformace, načtení) známý z datových skladů. Během extrakce se zajišťuje přístup k datům a jejich přenos do pracovního prostoru. Zdrojová data často obsahují chyby a nekonzistence, a proto je důležité je čistit a standardizovat.

Pokud vyžaduje zvolený algoritmus trénovací data, je třeba je připravit. Tato část je nejnáročnější a klíčová pro úspěch projektu. Trénovací data musí obsahovat různé případy, které se vyskytují v reálném světě, včetně nestandardních situací.

Je-li potřeba, aby umělá inteligence pracovala za různých podmínek, je nutné doplnit odpovídající data. V některých případech můžeme nahradit chybějící nebo nedostupná data, a to například průměrnými hodnotami.

Augmentace dat je další možností pro rozšíření trénovací sady, kde se vytvářejí nová data na základě stávajících. Některá data nelze použít z důvodu citlivosti, a proto se používají metody anonymizace. Model je také vhodné upravit pro konkrétní algoritmus a typ dat.

Rozdělení trénovacích dat je důležité, zejména pro algoritmy učení s učitelem, kde se data dělí na tréninkovou, testovací a validační skupinu. Další fáze zahrnuje samotné učení, kdy se algoritmus trénuje na trénovacích datech a testuje na testovacích datech.

Pokud model nesplňuje požadavky, je třeba se vrátit k předchozím fázím a hledat zlepšení. Nakonec dochází k nasazení modelu do produkce, kde je důležité sledovat jeho výsledky a zpětnou vazbu a případně se vracet k úpravám.[1][15][14]

3 Značení obrázků

Značení obrázků (Image labeling) spočívá v identifikaci celkového obrázku, ale může zahrnovat také identifikaci různých aspektů obrázku. Například tento proces je jednoduchý pro obrázky obsahující jediný motiv, jako je portrétní fotografie. Nicméně značení obrázků může být složitější pro fotografie s více detaily, jako jsou širokoúhlé snímky pořízené na veřejných místech.

Některé aspekty obrázku, které lze označit a identifikovat, zahrnují:

- Obličeje
- Zvířata
- Detaily krajiny
- Rozpoznávání textu
- Barvy
- Aktivity
- Fiktivní postavy
- Profese
- A další

Čím více detailů je zahrnuto v popisu obrázku, tím více informací může poskytnout. Například zatímco označení obrázku jako horské krajiny může být užitečné, ještě lepší je, když lze identifikovat konkrétní horský hřeben.

V závislosti na detailech obrázku a úrovni specifičnosti v označování může být značení obrázků časově náročný proces, který zahrnuje bohatství informací o tom, co lze na obrázku vidět.[11]

3.1 Použití značení obrázků

Existují mnohé účely označování obrázků, které budou nadále zásadní, s ohledem na budoucí vývoj internetu.

Dnešní software pro označování obrázků a rozpoznávání obličejů výrazně usnadňuje používání sociálních médií a organizaci obrázků v online albech. Fotografie mohou být automaticky tříděny podle obsahu a označovány bez potřeby identifikace každého obrázku samostatně.

To je také prospěšné pro podniky a tvůrce obsahu. Metadata lze automaticky generovat z obrázků a obsahu na stránkách, což eliminuje potřebu ručního vytváření. Takové informace mohou být využity k popisu obrázků na webových stránkách, což nejen zlepšuje optimalizaci pro vyhledávače, ale také zvyšuje atraktivitu stránek pro zrakově postižené.

V budoucnosti může označování obrázků také usnadnit cílenou reklamu na základě informací získaných z obrázků zákazníků.[11]

3.2 Proč je to důležité?

Značkování obrázků je klíčovou součástí vývoje řízených modelů s možnostmi počítačového vidění. Tento proces pomáhá trénovat modely strojového učení k označování celých obrázků nebo identifikaci tříd objektů uvnitř obrázku. Zde jsou některé způsoby, jak může značkování obrázků pomoci:

Rozvoj funkčních modelů umělé inteligence (AI) - Nástroje a metody označování obrázků pomáhají vyznačit nebo zachytit určité objekty na obrázku. Tyto značky zpřístupňují obrázky pro stroje a vyznačené obrázky často slouží jako sady trénovacích dat pro modely umělé inteligence a strojového učení.

Zlepšování počítačového vidění - označování a anotace obrázků pomáhají zvýšit přesnost počítačového vidění prostřednictvím rozpoznávání objektů. Trénování AI a strojového učení pomocí těchto značek pomáhá těmto modelům odhalovat vzory, dokud nejsou schopny rozpoznávat objekty samostatně.[11]

4 Rešerže

V rešeržesi zaměříme na současné webové služby, které umožňují ukládání a označování obrázků, což je oblast, ve které se naše aplikace pohybuje.

4.1 LabelMe

Labelme je nástroj pro anotaci obrazů založený na jazyce Python, který je open-source a umožňuje ruční anotaci obrázků pro detekci objektů, segmentaci a klasifikaci. Je to offline verze online nástroje LabelMe, který nedávno ukončil možnost registrace nových uživatelů. Labelme je lehká grafická aplikace s intuitivním uživatelským rozhraním, která umožňuje vytvářet polygonální oblasti, obdélníky, kruhy, čáry, body nebo úseky čar.

Se jedná o poměrně spolehlivou aplikaci s jednoduchou funkcionalitou pro ruční označování obrázků a pro širokou škálu úloh v oblasti počítačového vidění.

4.2 CVAT

CVAT je open-source nástroj počítačového vidění pro anotaci obrázků vyvinutý společností Intel. Jedná se o webový nástroj umožňující anotaci obrázků, který podporuje čtyři typy anotací: body, mnohoúhelníky, ohraničující rámečky a polylinie. CVAT podporuje také segmentaci obrázků, detekci objektů a klasifikaci obrázků, což jsou některé úkoly v oblasti počítačového vidění. V roce 2022 byly data, obsah a repozitář CVAT přeneseny do OpenCV, kde zůstává s otevřeným zdrojovým kódem. CVAT je také možné použít k anotaci QR kódů uvnitř obrázků, což usnadňuje integraci rozpoznávání QR kódů do konvejorů a aplikací počítačového vidění.

Modul dané práce je základní modul a se zaměřuje na správu dat ale do budoucna je v plánu tak moduly pro učení neuronových sítí a jejich nasazování do provozu.[9]

4.3 Porovnaní s vytvořeným systémem

LabelMe a CVAT patří mezi nejpopulárnější a nejvyspělejší aplikace pro práci s obrázky, zejména pro jejich anotaci. Nicméně vyvíjený projekt se zaměřuje na budoucnosti na přidání modulu nebo modulů pro trénování neuronových sítí na základě uživatelských datových sad.

5 NÁVRH SYSTÉMU

5.1 Architektura mikroslužeb

Architektura mikroslužb (Microservice architecture) jsou způsob, jak strukturovat aplikaci, který ji rozděluje na malé, nezávislé a volně propojené služby. Tyto služby jsou často vytvářeny kolem konkrétních obchodních funkcí a mají schopnost být nasazeny nezávisle na sobě. Každá mikroslužba je typicky vyvíjena a spravována malým týmem.

Pro lepší pochopení stylů mikroslužeb, je užitečné jej porovnat s monolitním stylem: monolitní aplikace je postavena jako jedno celé. Korporátní aplikace obvykle skládají ze tří hlavních částí: uživatelského rozhraní na straně klienta (tzv. front-end), serverové aplikace (tzv. back-end) a databáze. Serverová aplikace zpracovává HTTP požadavky, provádí doménovou logiku, získává a aktualizuje data z databáze a vybírá a vyplňuje HTML zobrazení pro odeslání do prohlížeče. Tato serverová aplikace představuje monolit - jediný logický spustitelný soubor. Každá změna v systému vyžaduje vytvoření a nasazení nové verze serverové aplikace.

Monolitické aplikace mohou být úspěšné, ale s rozvojem cloudových technologií jejich obliba postupně upadá, i když nevymírá úplně. Jak aplikace postupem času roste, je stále obtížnější provádět změny v aplikačních modulech kvůli obtížím spojeným s potřebou provádět změny v jiných modulech.

Tyto výzvy vedly k popularizaci architektonického stylu mikroslužeb: rozdělení aplikace na sadu služeb, které jsou nasazovány a škálovány nezávisle na sobě.[12]

5.2 Analyza

5.2.1 Funkční požadavky

Funkční požadavky jsou funkce, které musí systém minimálně zajišťovat. Jedná se o požadavky, které lze vidět ve finálním produktu, na rozdíl od nefunkčních požadavků.[6]

Tabulka 1: Funkční požadavky. Zdroj vlastní

ID	Požadavek
F1	Systém umožní registraci a autorizaci uživatelů.
F2	Systém umožní správu uživatelských projektů.
F3	Systém umožní správu uživatelů a jejich rolí, kteří mohou mít přístup k projektům.

F4	Systém umožní správu uživatelských datasetů.
F5	Systém umožní správu uživatelů a jejich rolí, kteří mohou mít přístup k datasetům.
F6	Systém umožní správu uživatelských obrázků.
F7	Systém umožní správu uživatelských objektů propojených s obrázky.

5.2.2 Nefunkční požadavky

Nefunkční požadavky jsou požadavky na kvalitu, které musí systém splňovat a jsou evidovány jako kvalitativní atributy. Tyto požadavky obvykle řeší problémy, jako je přenositelnost, zabezpečení, škálovatelnost, výkon atd.[6]

Tabulka 2: Nefunkční požadavky. Zdroj vlastní

ID	Požadavek
F1	Systém bude naprogramován ve programovacím jazyce Java
F2	Systém bude spouštěn ve Docker kontejneru
F3	Systém bude jako databázi používat databázi MongoDB
F4	Systém zajistí bezpečnost dat, včetně šifrování dat, a také implementaci systému řízení přístupu založeného na rolích, aby se odlišil přístup ke zdrojům.

5.3 Role a oprávněni

Každý prvek v systému možné přiřadit konkrétní roli pro určitého uživatele.

5.3.1 Systém

Pro ovládání systému následující role

Admin – hlavní admin, který může vše. Spravovat uživatele, projekt apod.

Researcher – uživatel, který je schopen založit projekt

User – běžný uživatel, který může otevírat projekty, ke kterým má přístup

5.3.2 Projekt

Pro projekt následující role

Owner – uživatel, který vytvořil projekt. Má právo ho upravovat, mazat projekt. Upravovat oprávnění

Dataset – uživatel, který má právo administrovat datasety – vytvářet, měnit a nastavovat práva, kopírovat datasety

5.3.3 Dataset

Pro dataset následující role

Owner – uživatel, který administruje daný dataset. V rámci datasetu vytváří kategorie, názvy objektů, exportuje a importuje datasety, vkládá a maže obrázky. apod.

Label – uživatel, který může v rámci projektu u obrázků označovat kategorie a označovat objekty

Viewer – pouze nahlíží

6 Použité technologie

6.1 Java

Hlavní aplikační služba je napsána v Javě verze 17. Java je vysokoúrovňový objektově orientovaný programovací jazyk.

Syntakticky je Java přímým potomkem jazyků C/C++, mnoho funkcí je z těchto jazyků zděděno, ale s mnoha inovacemi se stala jedním z nejčastěji používaných programovacích jazyků.[24]

Projekt také používá knihovnu **Lombok** k automatickému generování getterů, setterů, konstruktorů a dalších částí nezbytných komponent třídy, aby se zabránilo opakování kódu.[21]

6.2 Spring Framework

Spring Framework je populární platforma pro vývoj aplikací v jazyce Java, která poskytuje komplexní sadu nástrojů a funkcí pro rychlý a efektivní vývoj robustních a spolehlivých aplikací. Spring implementuje princip Inverze řízení (Inversion of Control - IoC), což znamená, že objekty nejsou vytvářeny přímo v kódu aplikace, ale jsou spravovány kontejnerem IoC. Tento kontejner řídí životní cyklus objektů a jejich závislosti, což umožňuje snadnější správu a testování aplikace.

Samotný Spring Framework je velký a obsahuje mnoho modulů, které usnadňují vývoj aplikací v jazyce Java. Níže jsou uvedeny stručné popisy několika klíčových modulů:

Spring Core: Tento modul poskytuje základní funkčnost frameworku Spring, zajišťuje klíčové mechanismy, které umožňují správu objektů a jejich vzájemné propojování v rámci aplikace pomocí konceptů dependency inje

Spring Boot: Tento modul je široce využíván pro tvorbu samostatných, produkčně nasaditelných aplikací postavených na technologii Spring. Zjednodušuje nasazování a konfiguraci Spring aplikací, umožňuje snadnější vytváření a správu aplikací, které jsou postaveny na Springu, a zjednodušuje jejich nasazení a konfiguraci pro produkční prostředí.

Spring Data: Tento modul poskytuje konzistentní přístup k práci s daty a integruje se s různými datovými zdroji, včetně relačních databází, NoSQL databází a dalších.

Spring JDBC je modul pro práci s relačními databázemi v prostředí Spring. Poskytuje jednoduché a elegantní API pro připojení k databázím a provádění SQL dotazů. Abstrahuje detaily spojené s obsluhou databází a umožňuje vývojářům psát odolný a čistý kód.

Spring Data MongoDB usnadňuje práci s MongoDB pomocí známých konceptů Spring. Poskytuje šablonové třídy pro práci s hlavním API a zjednodušený přístup k datům ve stylu repozitáře, což usnadňuje vývoj aplikací pomocí konzistentního programovacího modelu.

Spring MVC je webový framework pro vytváření robustních a modulárních webových aplikací v jazyce Java.Založen na návrhovém vzoru Model-View-Controller (MVC), který odděluje logiku aplikace od prezentační vrstvy.Poskytuje mnoho funkcí pro zpracování HTTP požadavků, správu stavu a generování dynamických HTML stránek.

Spring Security je modul pro zajištění bezpečnosti aplikací postavených na platformě Spring. Poskytuje autentizaci, autorizaci a další bezpečnostní funkce, jako jsou ochrana před útoky, správa uživatelských rolí a ochrana dat. Umožňuje konfiguraci bezpečnostních pravidel a strategií pomocí anotací nebo XML konfigurace.

Spring ORM je modul pro integraci objektově-relačního mapování (ORM) do aplikací postavených na platformě Spring.Umožňuje přístup k relačním databázím z objektově orientovaného jazyka pomocí různých ORM frameworků, jako je například Hibernate. Poskytuje podporu pro deklarativní transakční správu, mapování objektů na tabulky a správu databázových spojení. [25][26]

Spring Bean:V kontextu Spring Frameworku se pojem "Spring Bean" odkazuje na spravovaný objekt v kontejneru IoC (Inversion of Control). Bean je jednoduše Java objekt, který je vytvořen, spravován a nakonfigurován kontejnerem Springu. Spring Bean může být jakákoliv třída nebo objekt, který je definován v konfiguraci Spring kontejneru a je schopen poskytovat určitou funkcionalitu v rámci aplikace.

Komponenty Spring aplikace jsou definovány pomocí anotací, kterých je v rámci tématu frameworku poměrně hodně, nejčastěji používané jsou:

@Configuration — Anotace @Configuration označuje třídu jako definici konfigurace Springu, často používanou společně s anotací @Bean. Metody označené anotací @Bean uvnitř třídy s anotací @Configuration definují Spring beany.

@Repository — Třída anotovaná jako Repository je součástí datové vrstvy a poskytuje API pro interakci, což umožňuje vývojáři psát méně kódu. Anotace @Repository je značkou pro třídu, která plní roli objektu pro přístup k datům.

@Service — anotace pro třídy servisní vrstvy aplikace, obvykle se používá k anotaci tříd definujících obchodní logiku aplikace.

@Controller — Třída označená anotací @Controller v aplikaci Spring MVC je použita k definování kontroléru. Kontroléry zpracovávají příchozí HTTP požadavky a vracejí odpovědi. Metody uvnitř třídy označené jako @Controller jsou často anotovány pomocí @RequestMapping nebo jiných anotací pro mapování požadavků, které určují URL koncové body, které zpracovávají.

@GetMapping, @PostMapping, @PutMapping, @DeleteMapping — jsou anotaci pro zprávu http požadavků.

6.3 MongoDB

6.3.1 Co je to relační databáze?

Relační databáze jsou typem databází, které ukládají data do tabulek a umožňují přístup k nim pomocí jazyka SQL. SQL (Structured Query Language) je standardizovaný jazyk pro práci s daty v relačních databázích.

Principy relační databáze spočívají v ukládání dat do tabulek, kde každý řádek představuje záznam a každý sloupec atribut tohoto záznamu. Tyto databáze využívají klíče ke spojení dat v různých tabulkách a udržování vztahů mezi nimi.

DDL (Data Definition Language) je sada příkazů v SQL pro definici struktury databáze, jako vytváření tabulek, indexů nebo pohledů. **DML (Data Manipulation Language)** se používá k manipulaci s daty v databázi, jako je vkládání, aktualizace nebo mazání záznamů.

Vztahy určují, jak jsou entity propojeny. Každá entita musí mít specifikováný vztah, který určuje celkovou strukturu pravidel. Typy vztahu vznikajících mezi tabulky:

- 1:1 — pokud jednomu záznamů v tabulce odpovídá jeden záznam v jiné tabulce.
- 1:N — pokud jednomu záznamů v tabulce odpovídá vice (N) záznamů v jiné tabulce.
- M:N — pokud několika záznamu (M) v tabulce odpovídá vice (N) záznamů v jiné tabulce.[20]

6.3.2 Co je to nerelační databáze?

NoSQL databáze se liší od relačních databází v tom, že ukládá data ve formátu, který se neřídí tabulkovou strukturou. Termín „NoSQL“ je často interpretován jako „nejen SQL“ a zahrnuje širokou škálu databází, které nabízejí alternativní modely ukládání dat. Některé z těchto modelů zahrnují dokumenty, páry klíč–hodnota, sloupce a grafy. Databáze NoSQL jsou obvykle navrženy tak, aby zpracovávaly velké objemy nestrukturovaných nebo polo-strukturovaných dat a poskytovaly flexibilitu při škálování a zpracování dat.[7]

MongoDB patří do rodiny databází NoSQL, které se používají k ukládání nestrukturovaných dokumentů ve formátu JSON. Poprvé byla spuštěna v roce 2009 a od té doby se stala jednou z předních databází v prostoru NoSQL.

V MongoDB se záznam nazývá dokument a je to datová struktura složená z párů klíč-hodnota, podobně jako struktura objektů v JSON formátu.

Pro zpravu dokumentů používá MongoDB Query Language (MQL) je jazyk speciálně navržený pro interakci s databázemi MongoDB. Umožňuje uživatelům provádět operace CRUD (vytvořit, číst, aktualizovat, mazat) i pokročilejší operace, jako je agregace. MQL poskytuje sadu funkcí a operátorů, které umožňují uživatelům filtrovat, třídit, seskupovat a manipulovat s daty podle jejich potřeb, čímž se zvyšuje výkon aplikací. Dotazy lze psát pomocí MongoDB Compass, MongoDB Atlas nebo přímo z příkazového řádku.[22]

Model „jedna-ke-mnoha“ s vloženými dokumenty

Model 1:N s vnořenými dokumenty využívá strukturu databáze, kde hlavní entita obsahuje vnořené informace o souvisejících podřízených entitách. Tímto způsobem se minimalizuje počet čtecích operací a zajišťuje se, že veškerá potřebná data jsou dostupná v jednom dokumentu.[16]

```
{
  "publishers": [
    {
      "_id": "pub1",
      "name": "O'Reilly Media",
      "founded": 1980,
      "location": "CA",
      "books": [
        {
          "_id": "book1",
          "title": "MongoDB: The Definitive Guide",
          "author": ["Kristina Chodorow", "Mike Dirolf"],
          "published_date": "2010-09-24",
          "pages": 216,
          "language": "English"
        }
      ]
    }
  ]
}
```


Obrázek 3: model „jedna-ke-mnoha“ s vloženými dokumenty (zdroj – vlastní)

Model „one-to-many“ s odkazy na dokumenty

Model 1:N s odkazy na dokumenty využívá referencí mezi dokumenty k popisu vztahů jedna-ke-mnoha mezi propojenými daty. Tento přístup umožňuje minimalizovat opakování informací a lépe organizovat data. V praxi se používá k oddělení informací o nadřazené entitě (např. vydavatel) od podřazených entit (např. kniha), což umožňuje efektivnější správu dat a snižuje riziko vzniku rostoucích a měnitelných polí v databázi.[16]


```
{
  "publishers": [
    {
      "_id": "pub1",
      "name": "O'Reilly Media",
      "founded": 1980,
      "location": "CA"
    }
  ],
  "books": [
    {
      "_id": "book1",
      "publisher_id": "pub1",
      "title": "MongoDB: The Definitive Guide",
      "author": ["Kristina Chodorow", "Mike Dirolf"],
      "published_date": "2010-09-24",
      "pages": 216,
      "language": "English"
    }
  ]
}
```


Obrázek 4: model „jedna-ke-mnoha“ s odkazy na dokumenty (zdroj – vlastní)

6.4 Docker

Docker zjednodušuje nasazování aplikací tím, že zapouzdřuje software a jeho závislosti do kontejnerů, což zajistí konzistenci napříč různými počítačovými prostředími. Když kód vyvíjený na jednom systému narazí na problémy na jiném kvůli odlišným konfiguracím, Docker eliminuje takové kompatibilitní obavy. Tím, že vše, co aplikace potřebuje k běhu, zabalí do jednoho balíčku včetně knihoven a konfiguračních souborů, Docker abstrahuje od rozdílů v operačních systémech a infrastruktuře. To zajišťuje, že aplikace běží spolehlivě bez ohledu na podkladové prostředí, ať už jde o počítač vývojáře, testovací prostředí nebo produkční server. Kontejnerizace řeší výzvy spojené s rozdíly ve verzích softwaru, konfigurací a síťových topologiích, poskytujíc spolehlivé řešení pro nasazování softwaru napříč různými prostředími

6.4.1 Kontejnery a virtuální stroje

Na rozdíl od virtuálních strojů (VM), kontejnery nepotřebují vestavěné operační systémy; volání o operační systémové prostředky jsou prováděny pomocí aplikačního programovacího rozhraní. Na rozdíl od toho server, který běží s třemi aplikacemi ve kontejnerech, jako je Docker, běží pouze na jednom operačním systému a každý z kontejnerů sdílí jádro tohoto operačního systému s ostatními kontejnery. Je třeba poznamenat, že sdílené části operačního systému jsou pouze pro čtení, zatímco každý kontejner má svůj vlastní způsob přístupu k operačnímu systému pro zápis. To znamená, že kontejnery jsou mnohem lehčí a využívají mnohem méně prostředků než virtuální stroje.[19]

6.4.2 Docker Compose

Docker Compose je nástroj pro definování a spouštění více-kontejnerových aplikací. Poskytuje snadný a efektivní způsob vývoje a nasazení aplikací. Usnadňuje správu více aplikací pomocí jednoduchého konfiguračního souboru YAML obsahujícího informace o službách, sítích a svazcích.

Následně jsou pomocí jediného příkazu spouštěny všechny služby definované v konfiguračním souboru.[4]

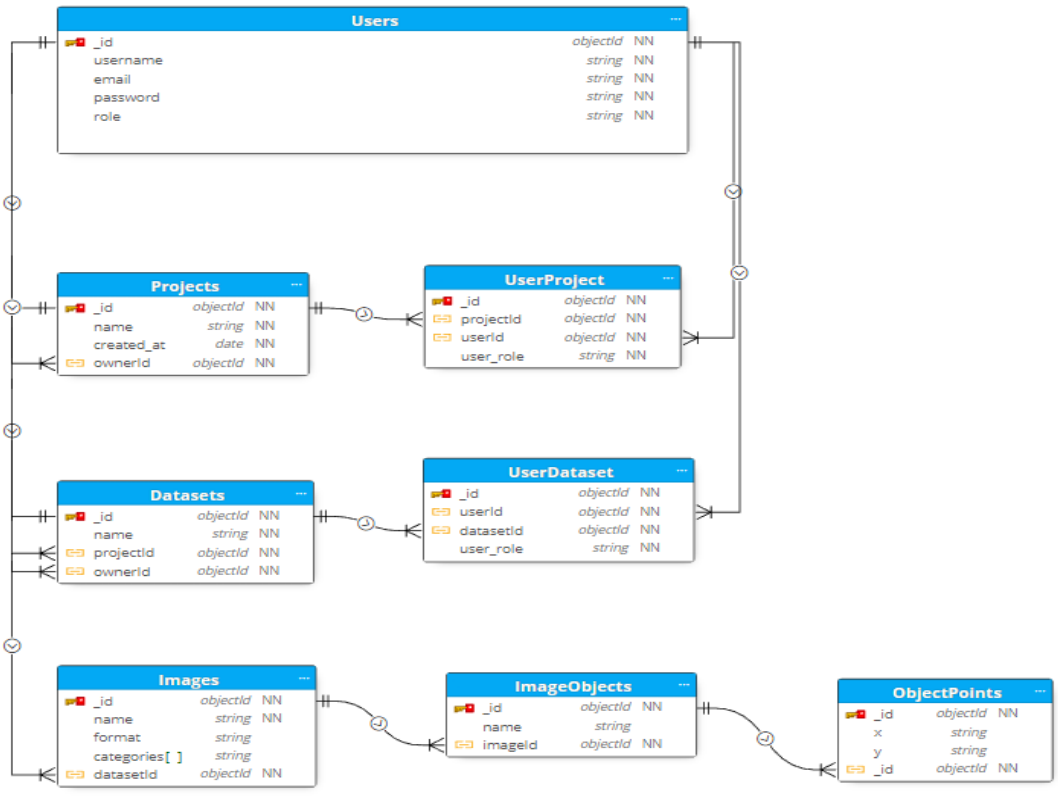
7 Datový model

Tato kapitola je věnovaná popisu databázového modelu práce, zahrnuje popis schema databáze a popis všech použitých tabulek.

7.1.1 Logický model

Logický datový model slouží ke strukturování a objasnění vztahů mezi daty v systému. Tento model zvýrazňuje jednotlivé entity, jako jsou autoři knih, aby se usnadnilo vyhledávání a zajistilo se lépe organizované ukládání dat. Vytvoření logického modelu umožňuje zajistit přehlednost a systematizaci vztahů mezi různými objekty, což je nezbytné pro efektivní a přesnou reprezentaci dat v databázi.[17]

Tento model byl postaven v aplikaci MoonLoader, ze které lze také vygenerovat skript pro vytvoření všech tabulek.



Obrázek 5: datový model (zdroj – vlastní)

7.1.2 Tabulky

Databáze obsahuje 8 tabulek:

Users:

Tabulka pro ukladani uživatele

- *id*: Primární klíč, identifikuje entitu.
- *username*: Unikátní uživatelské jméno pro přihlášení. Je unikátní.
- *email*: Email uživatele, je unikátní.
- *password*: Heslo uživatele, zašifrované pomocí hashovací funkce bcrypt.
- *role*: Role uživatele.

Relace tabulky:

- *project:* relace 1:N, jednomu uživateli muže byt přiraženo N projektu
- *userProject:* relace 1:N, jeden uživatel muže mít přistup k několika projekty

Projects:

- *id*: Primární klíč, identifikuje entitu.
- *name*: Název projektu.
- *created_*at: Datum a čas vytvoření projektu.
- *ownerId*: Odkaz na vlastníka projektu.

Relace tabulky:

- *Users:* relace 1:N, jednomu uživateli muže byt přiraženo N projektu.
- *UserProject*: relace 1:N, k jednomu projektu mohou mít přistup N uživatelů.
- *Datasets*: relace 1:N, k jednomu pojektu odpovída několik datasetu .

UserProject:

Tabulka pro zajisteni vazby N:M mezi tabulky *Users* a *Projects*

- *id*: Primární klíč, identifikuje entitu.
- *projectId*: Odkaz na projekt uživatele. Relace 1:N z tabulkou *Projects.*
- *userId*: Odkaz na uživatele.

- *user_role*: Role uživatele v projektu.

Relace tabulky:

- *Projects*: relace 1:N z tabulkou *Projects*.
- *Users*: relace 1:N z tabulkou *Users.*

Datasets:

- *name*: Název datasetu.
- *projectId*: Odkaz na projekt, ke kterému dataset patří.
- *ownerId*: Odkaz na vlastníka datasetu.

Relace tabulky:

- *Projects*: relace 1:N, nekolik datasetu odpovida jednomu projektu.
- *Users:* relace 1:N, jednomu uživateli muže byt přiraženo N datasetu.
- *UserDatasets*: relace 1:N, nekolik uzivatelu mohou mit pristup ke datasetu.

UserDataset:

- *id*: Primární klíč, identifikuje entitu.
- *userId*: Odkaz na uživatele.
- *datasetId*: Odkaz na dataset.
- *user*_role: Role uživatele v datasetu.

Relace tabulky:

- *Datasets*: relace 1:N z tabulkou *Datasets*.
- *Users*: relace 1:N z tabulkou *Users.*

Images:

- *id*: Primární klíč, identifikuje entitu.

- *name*: Název obrázku.
- *format*: Formát obrázku.
- *categories*: Kategorie obrázku (pole).
- *datasetId*: Odkaz na dataset, ke kterému obrázek patří.

Relace tabulky:

- *Datasets*: relace 1:N z tabulkou *Datasets*.

ImageObjects:

- *id*: Primární klíč, identifikuje entitu.
- *name*: Název objektu na obrázku.
- *imageId*: Odkaz na obrázek, ke kterému objekt patří.

Relace tabulky:

- *Images*: relace 1:N z tabulkou *Images*.

ObjectPoints:

- *id*: Primární klíč, identifikuje entitu.
- *objectId*: Odkaz na objekt, ke kterému bod patří.
- *x*: Souřadnice X objektu.
- *y*: Souřadnice Y objektu.

Relace tabulky:

- *Images*: relace 1:N z tabulkou *ObjectPoints*.

7.1.3 GridFS

GridFS je specifikace pro ukládání a získávání souborů, které překračují limit velikosti BSON dokumentu 16 MB.

Místo ukládání souboru v jednom dokumentu GridFS rozděluje soubor na části, nazývané **kusy** (chunk), a ukládá každý kus jako samostatný dokument. Výchozí velikost kusu je obvykle 255 kB, přičemž poslední kus je velikosti potřebné pro zbývající data. Soubory menší než velikost kusu mají pouze jeden kus, využívající jen potřebný prostor a navíc metadata.

GridFS používá dva kolekce k uložení souborů. Jedna kolekce uchovává kusy souborů a druhá ukládá metadata souborů.

Při dotazu na soubor v GridFS bude ovladač potřebné kusy souboru znovu složit dohromady. Lze provádět rozsahové dotazy na soubory uložené pomocí GridFS.[18]

V dané práci byl použit modul GridFsTemplate pro práci s Javou poskytnutý balíčkem Spring Data MongoDB. Spring Data poskytuje rozhraní GridFsOperations a jeho implementaci - GridFsTemplate - pro snadnou interakci s tímto souborovým systémem.

Když uživatel pošle požadavek na systém pro zaslání souboru, příslušná metoda spravuje práci objektu GridFsTemplate, který zajistí uložení souboru.

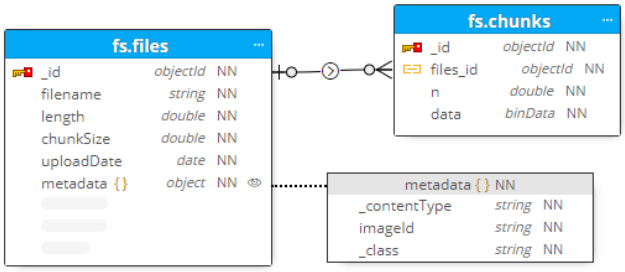
```
public Image saveImage(Image image, MultipartFile data) throws IOException {
   Image save = imageRepository.save(image);

    DBObject metadata = new BasicDBObject();
    metadata.put( key: "imageId", image.getId());

    gridFsTemplate.store(data.getInputStream(), data.getOriginalFilename(), data.getContentType(), metadata);
    return save;
}
```

Obrázek 6: příklad kódu pro použití GridFs (zdroj - vlastní)

Struktura tabulek odpovidajicih za prace z GridFs muze vypadat nasledovne:



Obrázek 7: datový model GridFs (zdroj - vlastní)

Tabulka fs.files odpovídá za ukládání dat o souboru jako název (filename), velikost části (chunkSize), datum nahrání (uploadDate) a metadata souboru. Tabulka fs.chunks odkazuje na

konkrétní soubor (files_id) a ukládá samostatné části každého souboru (data) spolu s číslem uložené části (n).

8 RestAPI Documentace

REST (Representational State Transfer) je architektonický styl používaný pro distribuované systémy, jako je web. Založený na souboru omezení, která zlepšují výkon, škálovatelnost a údržbu internetových aplikací, REST odděluje uživatelské rozhraní od datového skladování, což umožňuje nezávislý vývoj komponent. RESTful systémy komunikují přes státní protokoly a operace (např. HTTP) s využitím bezstavových požadavků bez ukládání informací o sezení na serveru, což zvyšuje viditelnost a spolehlivost a snižuje závislost na konkrétních softwarových technologiích.[5]

V rámci této kapitoly je popsán základní způsob komunikace s navrženou aplikací. Celkový popis API této aplikace je příliš rozsáhlý; podrobnější informace jsou uvedeny v příloze "REST API DOKUMENTACE". Zde je popsána základní struktura a funkce.

Pro interakci s aplikací jsou zodpovědné následující kontroléry:

AuthController — zajišťuje autorizaci a registraci nových uživatelů;

UserController — spravuje údaje uživatelů, jako jsou přihlašovací jméno, heslo a role;

ProjectController — spravuje údaje o projektech uživatelů;

DatasetController — spravuje údaje o datasetech uživatelů;

ImageController — spravuje uživatelské obrázky.

8.1.1 Auth Controller

Tento kontrolér zajišťuje autorizaci a registraci nových uživatelů. Pro to se využívá balíček Spring Security.

POST /api/auth/register — Registrace nového uživatele. Zkontroluje unikátnost údajů a při úspěchu vrací JWT token uživatele.

Příklad odpovědi:{ "jwtResponse": "string"}.

Vraci "400 BAD REQUEST" chybu v pripade kdyz zadana uzivatelske jmeno nebo email jiz existuji v systemu

POST /api/auth/login — Přihlášení uživatele.

Příklad odpovědi:{"jwtResponse": "string"}.

Vrací "403 FORBIDDEN" v případe spatné uvedeného helsa nebo uživatelského jména

Oba pozadavky se zpracovavaji pomoci objektu AuthService ktery zpravuje logiku obou autorizaci a registraci uživatelů.

8.1.2 User controller

User Controller poskytuje rozhraní pro správu uživatelů v systému. Každá metoda je zabezpečena pomocí anotace PreAuthorize, která omezuje přístup k určitým skupinám uživatelů.

PUT /api/users/updateUsername: Aktualizuje uživatelské jméno.

PUT /api/users/updateRole: Aktualizuje roli uživatele.

PUT /api/users/updatePassword: Aktualizuje heslo uživatele.

PUT /api/users/updateEmail: Aktualizuje emailovou adresu uživatele.

PUT /api/users/update/{id}: Aktualizuje všechny údaje uživatele podle zadaného ID.

POST /api/users/createUser: Vytváří nového uživatele.

GET /api/users/getByUsername/{username}: Získává uživatele podle uživatelského jména.

GET /api/users/getById/{id}: Získává uživatele podle identifikátoru.

GET /api/users/getAll: Získává seznam všech uživatelů.

DELETE /api/users/deleteById/{id}: Maže uživatele podle identifikátoru.

Každá z těchto metod může vrátit odpovědi ve formě JSON s odpovídajícími údaji o uživatelích nebo chybové kódy v případě, že operace selže.

8.1.3 Project controller

Project Controller zpracovává uživatelské projekty a spravuje přístup uživatelů k nim. Metody jsou zabezpečeny anotací PreAuthorize a využívají parametr UserDetails označený anotací @AuthenticationPrincipal pro získání dat o oprávněném uživateli.

PUT /api/projects/update/{id}: Aktualizuje projekt podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k projektu.

POST /api/projects/users/addUserToProject: Přidá uživatele do projektu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k projektu.

POST /api/projects/create: Vytváří nový projekt. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá oprávnění správce.

GET /api/projects/users/getByUserIdAndProjectId: Získává uživatele podle ID uživatele a ID projektu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k projektu.

GET /api/projects/users/getByProjectId/{projectId}: Získává uživatele podle ID projektu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k projektu.

GET /api/projects/getById/{projectId}: Získává projekt podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k projektu.

GET /api/projects/getAll: Získává seznam všech projektů. Vrací 403 Forbidden, pokud uživatel není oprávněný správce.

GET /api/projects/getAllByOwnerId/{id}: Získává seznam všech projektů podle ID vlastníka. Vrací 403 Forbidden, pokud uživatel není autorizován nebo má jiné ID nebo nemá oprávnění správce.

DELETE /api/projects/users/delete: Odstraní uživatele z projektu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k projektu.

DELETE /api/projects/delete/{id}: Odstraní projekt podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k projektu.

8.1.4 Dataset Controller

Dataset Controller zpracovává uživatelské datasety a spravuje přístup uživatelů k nim. Metody jsou zabezpečeny anotací PreAuthorize a využívají parametr UserDetails označený anotací @AuthenticationPrincipal pro získání dat o oprávněném uživateli pomocí frameworku SpringSecurity.

PUT /api/datasets/update/{id}: Aktualizuje dataset podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k datasetu.

POST /api/datasets/users/addUserToDataset: Přidá uživatele do datasetu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k datasetu.

POST /api/datasets/create: Vytvoří nový dataset. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá oprávnění správce.

GET /api/datasets/users/getByDatasetIdAndUserId: Získává uživatele podle ID uživatele a ID datasetu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k datasetu.

GET /api/datasets/users/getByDatasetId/{datasetId}: Získává uživatele podle ID datasetu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k datasetu.

GET /api/datasets/getById/{id}: Získává dataset podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k datasetu.

GET /api/datasets/getAll: Získává seznam všech datasetů. Vrací 403 Forbidden, pokud uživatel není oprávněný správce.

GET /api/datasets/getAllByProjectId/{projectId}: Získává seznam všech datasetů podle ID projektu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k projektu.

GET /api/datasets/getAllByOwnerId/{ownerId}: Získává seznam všech datasetů podle ID vlastníka. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá oprávnění správce.

GET /api/datasets/getAllByName/{name}: Získává seznam všech datasetů podle jména. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá oprávnění správce.

DELETE /api/datasets/users/deleteUserFromDataset: Odstraní uživatele z datasetu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo *nemá* přístup k datasetu.

DELETE /api/datasets/deleteAllByProjectId/{id}: Odstraní všechny datasety podle ID projektu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k projektu.

DELETE /api/datasets/delete/{id}: Odstraní dataset podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k datasetu.

8.1.5 Image Controller

Image Controller spravuje obrazy uložené uživatelem a objekty určené pro labelování obrazu. Každá metoda je zabezpečena anotací PreAuthorize, která omezuje skupinu uživatelů s přístupem k požadavku. Všechny metody v parametru požadavku mají parametr UserDetails označený anotací @AuthenticationPrincipal, což umožňuje získat data o oprávněném uživateli pomocí frameworku SpringSecurity.

PUT /api/images/objects/update/{id}: Aktualizuje objekt obrazu podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

POST /api/images/save: Ukládá obraz. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

POST /api/images/objects/save: Ukládá objekt obrazu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

POST /api/images/objects/points/save: Ukládá body objektu obrazu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

GET /api/images/objects/points/getAll: Získává všechny body objektů obrazu. Vrací 403 Forbidden, pokud uživatel nemá oprávnění správce.

GET /api/images/objects/points/getAllByObjectId/{id}: Získává všechny body objektů obrazu podle ID objektu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

GET /api/images/objects/findAll: Získává všechny objekty obrazu. Vrací 403 Forbidden, pokud uživatel nemá oprávnění správce.

GET /api/images/objects/findAllByImageId/{imageId}: Získává všechny objekty obrazu podle ID obrazu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

GET /api/images/getById/{imageId}: Získává obraz podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

GET /api/images/getAll: Získává seznam všech obrazů. Vrací 403 Forbidden, pokud uživatel nemá oprávnění správce.

GET /api/images/getAllByDatasetId/{datasetId}: Odstraní seznam všech obrazů podle ID datasetu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

GET /api/images/findAllByCategoryContains: Odstraní seznam všech obrazů obsahujících zadanou kategorii. Vrací 403 Forbidden, pokud uživatel nemá oprávnění správce.

GET /api/images/findAllByCategories: Odstraní seznam všech obrazů podle zadaných kategorií. Vrací 403 Forbidden, pokud uživatel nemá oprávnění správce.

GET /api/images/findAllByCategoriesAndDatasetId: Odstraní seznam všech obrazů podle zadaných kategorií a ID datasetu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

DELETE /api/images/objects/points/deleteAllByObjectId/{id}: Odstraní všechny body objektů obrazu podle ID objektu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

DELETE /api/images/objects/points/delete/{id}: Odstraní bod objektu obrazu podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

DELETE /api/images/objects/deleteById/{id}: Odstraní objekt obrazu podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

DELETE /api/images/objects/deleteAllByImageId/{id}: Odstraní všechny objekty obrazu podle ID obrazu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

DELETE /api/images/deleteById/{imageId}: smaže obraz podle ID. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

DELETE /api/images/deleteAllByDatasetId/{datasetId}: smaže všechny obrazy podle ID datasetu. Vrací 403 Forbidden, pokud uživatel není autorizován nebo nemá přístup k vybranému datasetu nebo nemá oprávnění správce.

ZÁVĚR

Cílem této bakalářské práce bylo vytvoření efektivní mikroslužby pro správu multimediálních dat, která by sloužila k trénování neuronových sítí. Klíčovým úkolem byla příprava a implementace systému, který by umožňoval efektivní manipulaci s velkými objemy dat a zároveň byl dostatečně flexibilní, aby podporoval různé typy multimediálních dat a metadat.

Byl navržen a vyvinut robustní systém s využitím moderních technologií, jako jsou Java, Spring Framework, MongoDB a Docker. Tyto technologie byly zvoleny pro svou spolehlivost, škálovatelnost a širokou podporu ve vývojářské komunitě.

Databázový model byl navržen tak, aby efektivně zvládal ukládání a správu dat s důrazem na integritu a dostupnost dat. Systém byl dále otestován, aby se ověřila jeho funkcionalita a splnění stanovených cílů.

Hlavním přínosem této práce je vývoj mikroslužby, která poskytuje uživatelům přístup k pokročilým nástrojům pro správu dat potřebných pro trénování neuronových sítí. Systém nejenže zefektivňuje proces přípravy dat, ale také umožňuje jejich systematické zpracování a analýzu.

Výsledný systém má potenciál pro další rozvoj a adaptaci na specifické požadavky různých výzkumných projektů a komerčních aplikací. Dále může sloužit jako základ pro rozšíření funkcionalit, včetně integrace s dalšími nástroji a platformami v oblasti strojového učení.

Celkově lze konstatovat, že výsledky této práce představují cenný příspěvek k řešení problémů spojených se správou a analýzou multimediálních datových sad a poskytují solidní základ pro další výzkum a vývoj v dynamicky se rozvíjející oblasti neuronových sítí a umělé inteligence.

POUŽITÁ LITERATURA

[1] AWAN, Abid Ali. The Machine Learning Life Cycle Explained[online] 2022. [cit. 2024-05-06] Dostupné z: https://www.datacamp.com/blog/machine-learning-lifecycle-explained

[2] BANDYOPADHYAY, Hmrishav. What Is Computer Vision? [Basic Tasks & Techniques]. What Is Computer Vision? [Basic Tasks & Techniques] [online]. 2022 [cit. 2024-05-06]. Dostupné z: https://www.v7labs.com/blog/what-is-computer-vision

[3] BANERJEE, Ananya. Different Computer Vision Tasks [online]. 2019 [cit. 2024-05-06]. Dostupné z: https://ananya-banerjee.medium.com/different-computer-vision-tasks-b3b49bbae891

[4] Docker. Compose Application Model [online]. [cit. 2024-05-13]. Dostupné z: https://docs.docker.com/compose/compose-application-model/

[5] FIELDING, Roy Thomas. REST Architectural Style [online]. 2021 [cit. 2024-05-13]. Dostupné z: https://web.archive.org/web/20210513160155/https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[6] Functional vs Non-Functional Requirements [online]. [cit. 2024-05-06]. Dostupné z: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/

[7] GROLINGER, K.; HIGASHINO, W. A.; TIWARI, A.; Capretz, M. A. M. (2013). "Data management in cloud environments: NoSQL and NewSQL data stores" (PDF). Aira, Springer. Retrieved 8 January 2014.

[8] HAMILTON Serena, JAKEMAN H, A.J. Artificial Intelligence techniques: An introduction to their use for modelling environmental systems.[online] 2008 [cit. 2024-05-06]. Dostupné z: https://www.researchgate.net/figure/Seven-categories-of-tasks-ANN-can-perform-83_fig4_257219838

[9] IAKUSHECHKIN, Dmitrii. The Best Labeling Tools for Computer Vision [online]. [cit. 2024-05-13]. Dostupné z: https://dida.do/blog/the-best-labeling-tools-for-computer-vision

[10] Image Classification [online] 2019 cit[2024-05-06] Dostupné z: https://www.sciencedirect.com/topics/engineering/image-classification

[11] Image labeling [online]. 2023-05-25. [cit. 2024-05-06] Dostupné z: https://medium.com/@saiwa.ai/image-labeling-fff892dc2f1e

[12] INGENO Joseph, Software Architect's Handbook, Birmingham Packt Publishing Ltd., ISBN 978-1-78862-406-0

[13] KOĎOUSKOVÁ, Barbora. Umělá inteligence (AI): historie a trendy pro rok 2024[online] 2024-04-03. [cit. 2024-05-06] Dostupné z: https://www.rascasone.com/cs/blog/umela-inteligence-ai-trendy

[14] KUMAR, Sanjay; REHMAN, Fasih Ur. What are some tips for ANN project management and workflow?[online] [cit. 2024-05-06] Dostupné z: https://www.linkedin.com/advice/3/what-some-tips-ann-project-management-workflow-dvqne

[15] LÓPEZ, Estrella Funes, et al. The various phases in an ANN development project. These stages are fundamental to right performance of ANN [online] 2015. [cit. 2024-05-06] Dostupné z: https://www.researchgate.net/figure/The-various-phases-in-an-ANN-development-project-These-stages-are-fundamental-to-right_fig1_273075306

[16] MongoDB. Applications Data Models Relationships [online]. [cit. 2024-05-13]. Dostupné z: https://www.mongodb.com/docs/manual/applications/data-models-relationships/
[17] MongoDB. Data Modeling [online]. [cit. 2024-05-13]. Dostupné z: https://www.mongodb.com/resources/basics/databases/data-modeling
[18] MongoDB. GridFS [online]. [cit. 2024-05-13]. Dostupné z: https://www.mongodb.com/docs/manual/core/gridfs/
[19] Open Source For You. Docker: The Favourite in DevOps World [online]. 2017. [cit. 2024-05-13]. Dostupné z: https://www.opensourceforu.com/2017/02/docker-favourite-devops-world/
[20] ORACLE. Define a relationship [online]. [cit. 2024-05-13]. Available from: https://docs.oracle.com/html/E79061_01/Content/Data%20model/Define_a_relationship.htm
[21] Project Lombok: Features [online]. [cit. 2024-05-06] Dostupné z: https://projectlombok.org/features/
[22] RABOY, Nic. Getting Started with Atlas and the MongoDB Query Language.[online] 2022-01-14 [cit. 2024-05-06]. Dostupné z: https://www.mongodb.com/developer/products/atlas/getting-started-atlas-mongodb-query-language-mql/
[23] SHARMA Pulkit. Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques (Part 1) [online] 2024 [cit. 2024-05-06]. Dostupné z: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/
[24] SCHILDT, Herbert. Java: The Complete Reference, Eleventh Edition. McGraw Hill, prosinec 2018. ISBN 978-1260440232.
[25] Spring Boot [online]. [cit. 2024-05-06] Dostupné z: https://spring.io/projects/spring-boot
[26] Spring IO Spring Data MongoDB Reference Documentation [online]. [cit. 2024-05-06] Dostupné z: https://docs.spring.io/spring-data/data-mongodb/reference/mongodb.html
[27] Supervised and Unsupervised. learning.[online] cit[2024-05-06] Dostupné z: https://www.geeksforgeeks.org/supervised-unsupervised-learning/
[28] TEGMARK, Max. Život 3.0: člověk v éře umělé inteligence. Přeložil Markéta IVÁNKOVÁ. Zip., svazek 69. Praha: Argo, 2020. ISBN 978-80-7363-948-8.
[29] Umělá inteligence (AI).[online] [cit. 2024-05-06] Dostupné z: https://www.lesensky.cz/umela-inteligence-ai#
[30] ZOUMANA, Keita. An Introduction to Convolutional Neural Networks (CNNs). An Introduction to Convolutional Neural Networks (CNNs) [online]. 2023 [cit. 2024-05-06]. Dostupné z: https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns