Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Inteligentní klasifikace obrazu v systému Kontent.ai

2024

Bc. Lenka Klimčíková

Vedoucí práce:
RNDr. Martin Trnečka, Ph.D.

Studijní program: Informatika,
Specializace: Umělá inteligence

Bibliografické údaje

Autor:	Bc. Lenka Klimčíková
Název práce:	Inteligentní klasifikace obrazu v systému Kontent.ai
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Informatika, Specializace: Umělá inteligence
Vedoucí práce:	RNDr. Martin Trnečka, Ph.D.
Počet stran:	47
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	slovenský

Bibliographic info

Author:	Bc. Lenka Klimčíková
Title:	Intelligent image classification in Kontent.ai system
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Computer Science, Specialization: Artificial Intelligence
Supervisor:	RNDr. Martin Trnečka, Ph.D.
Page count:	47
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Slovak

Anotácia

Táto práca, vypracovaná pod záštitou spoločnosti Kontent.ai, skúma možnosti použitia inteligentnej klasifikácie obrazu v cloudovom systéme Kontent.ai. Porovnané sú rôzne existujúce technológie a frameworky určené pre klasifikáciu rôzneho druhu obsahu. Vybraná bola technológia Cognitive Services poskytovateľa Microsoft Azure, pomocou ktorej bol následne implementovaný prototyp klasifikátoru obsahu. Funguje ako samostatná webová aplikácia v .NET MVC, ktorá pomocou SDK a REST API komunikuje s cloudovým systémom Kontent.ai a klasifikačným frameworkom Azure Cognitive Services. Umožňuje používateľom vykonať označovanie či generovanie popisu knižnice súborov ich projektu, a exportovať vygenerované výsledky priamo do ich projektu ako taxonómie pre jednoduchšiu orientáciu či filtrovanie vo veľkej sade dát. Súčasťou práce je tiež zhodnotenie možnosti reálneho nasadenia a škálovateľnosti vytvoreného prototypu.

Synopsis

This thesis, created under the support of Kontent.ai company, explores the possibilities of using intelligent image classification in cloud system Kontent.ai. Different existing technologies and frameworks used for classification of different types of content are investigated. Cognitive Services technology from Microsoft Azure provider was chosen, by means of which was consequently implemented a content classification prototype. It works as a stand-alone web application in .NET MVC, that communicates through SDK and REST API with cloud system Kontent.ai and classification framework Azure Cognitive Services. It enables its users to perform tagging or caption generating of their project's asset library, and export the generated results directly to their project as taxonomies for easier orientation or filtering in big data set. Part of the thesis is also an evaluation of the possibility of real application deployment and scalability of the created prototype.

Kľúčové slová: klasifikácia; umelá inteligencia; obraz; video; Kontent.ai; taxonómie; Azure Cognitive Services; webová aplikácia; .NET MVC; REST API

Keywords: classification; artificial intelligence; image; video; Kontent.ai; taxonomies; Azure Cognitive Services; web application; .NET MVC; REST API

Touto cestou by som sa chcela poďakovať konzultantovi tejto diplomovej práce za firmu Kontent.ai, pánovi Ing. Petrovi Pyszkovi za odborné vedenie a pravidelné konzultácie. Takisto by som chcela poďakovať vedúcemu práce pánovi RNDr. Martinovi Trnečkovi, Ph.D. za pomoc pri tvorbe práce a cenné rady.

Odovzdaním tohto textu jeho autor/ka čestne vyhlasuje, že celú prácu vrátane príloh vypracoval/a samostatne a za použitia iba zdrojov spomínaných v texte práce a uvedených v zozname literatúry.

Obsah

1 Úvod **5**

2 Klasifikácia obsahu v systéme Kontent.ai **6**
2.1 Definícia riešeného problému 6

3 Dostupní poskytovatelia technológií rozpoznávania obsahu **11**
3.1 Porovnanie poskytovateľov 15
3.2 Testovanie vybraných poskytovateľov 16
3.3 Konečný výber poskytovateľa a zdôvodnenie 20

4 Návrh riešenia, implementácia a použité technológie **21**
4.1 Návrh používateľského rozhrania 21
4.2 Kontent Management API 25
4.3 Azure AI Vision Image Analysis 25
4.4 Azure Video Indexer 27
4.5 ASP .NET Core MVC a Razor Pages 28
4.6 Verziovanie a nasadenie aplikácie 36

5 Testovanie a škálovanie aplikácie **38**
5.1 Testovanie aplikácie 38
5.2 Možné vylepšenia 39
5.3 Škálovanie a integrácia do produktu 40

Záver **43**

Conclusions **44**

A Obsah elektronických dat **45**

Literatúra **46**

1 Úvod

Za posledné roky je možné sledovať výrazný pokrok v technológiách oblasti umelej inteligencie. V obore informačných technológií aktuálne predstavuje výrazný trend a zahŕňa širokú škálu techník ako je strojové učenie, neurónové siete či spracovanie prirodzeného jazyka. Jednou z dôležitých aplikácií umelej inteligencie je bezpochyby jej využitie pri tvorbe a správe digitálneho obsahu. Hrá dôležitú úlohu pri vykonávaní úloh, ktoré boli predtým časovo náročné či vykonávané manuálne.

V spoločnosti Kontent.ai, ktorá sa zaoberá tvorbou cloudového systému správy obsahu, je práve obsah a jeho efektívna tvorba a správa na prvom mieste. Zákazníci majú v systéme v knižnici súborov uložené veľké množstvo dát, pričom jednoduchá orientácia v nich je pre ich každodennú prácu kľúčová. Táto práca má za účel preskúmať možnosti použitia klasifikácie obsahu v tomto systéme. Preskúmaní a porovnaní boli rôzni verejní poskytovatelia technológií spracovania obsahu a vybraný bol jeden najvhodnejší pre tento prípad použitia. Pomocou toho bol navrhnutý prototyp kvalifikátora ako samostatná webová aplikácia, ktorá komunikuje so systémom Kontent.ai a je schopná jednoduchým rozhraním previesť užívateľa výberom žiadaných vlastností a ich vykonaním na celej jeho existujúcej knižnici obsahu. Výsledok si môže exportovať do svojho projektu a využiť napríklad pri filtrovaní v knižnici.

Na začiatku práce v kapitole 2 je predstavený systém Kontent.ai a definícia riešeného problému spoločne s motiváciou k jeho riešeniu.

Kapitola 3 sa ďalej zaoberá prieskumom dostupných poskytovateľov technológií rozpoznávania obsahu. Jednotliví poskytovatelia boli porovnaní typmi a cenami analýzy, otestovaní na konkrétnych príkladoch a nakoniec bolo potrebné vybrať najvhodnejšie riešenie, ktoré je použité v prototype aplikácie.

Potom mohlo dôjsť k samotnému návrhu a implementácii webovej aplikácie. Kapitola 4 popisuje návrh aplikácie, využité technológie na jej tvorbu, podrobnejšie rozoberá využité služby a tiež sa zaoberá niektorými implementačnými detailami. Je v nej tiež spomenutý spôsob verziovania aplikácie i jej nasadenie.

Kapitola 5 nakoniec popisuje testovanie aplikácie, ďalšie možné vylepšenia a tiež možnosti škálovania a integrácie funkcionality priamo do produktu Kontent.ai.

2 Klasifikácia obsahu v systéme Kontent.ai

Pre vysvetlenie praktického výsledku, ktorý má táto práca dosiahnuť, je najskôr potrebné oboznámenie so samotným softvérom Kontent.ai a základnými využívanými pojmami. To bude predstavené v tejto kapitole spolu s definíciou riešeného problému.

Systém správy obsahu (anglicky Content Management System, skratka CMS) je softvér, ktorý má za úlohu pomáhať užívateľom vytvárať a spravovať digitálny obsah na webových stránkach bez potreby technickej znalosti tvorby webových stránok. Umožňuje teda vytvoriť webovú stránku bez potreby písania kódu. Pomáha podnikom byť efektívnejšími pri správe a distribúcii obsahu, čo je v dnešnej rýchlo sa meniacej digitálnej dobe veľmi dôležité. Najznámejším open-source CMS je WordPress.[1]

Headless CMS je typ systému správy obsahu, ktorý umožňuje poskytovať obsah na viacerých kanáloch súčasne a nestará sa pritom o správu prezenčnej vrstvy aplikácie. Tradičné systémy správy obsahu používajú šablóny a motívy a nedovoľujú využiť rovnaký obsah na rôznych kanáloch či zariadeniach súčasne. Práve headless CMS tento problém rieši a umožňuje zákazníkom spravovať prezentáciu ich webu pomocou vlastného kódu a nezávisle od konkrétnej platformy alebo technológie. [1]

Kontent.ai je cloudová headless CMS služba od rovnomennej firmy Kontent.ai, ktorá poskytuje nástroje na plánovanie, publikovanie a optimalizáciu životného cyklu obsahu. Umožňuje obsah publikovať kdekoľvek prostredníctvom API koncových bodov a je postavený na moderných vývojových štandardoch s proaktívnym monitorovaním a systémami upozornení a notifikácií.

Architektúra je založená na službách a platformách poskytovaných cez Microsoft Azure, ktoré zabezpečujú spoľahlivé prostredie pre autorov obsahu a ich koncových používateľov. Všetky dáta sú pravidelne zálohované a šifrované pre zabezpečenie ich ochrany. [1]

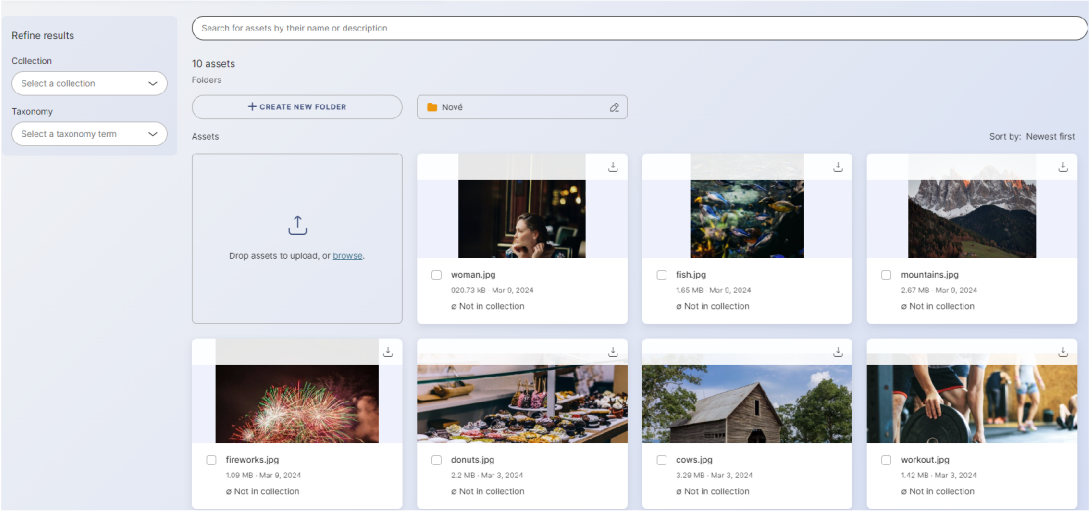
2.1 Definícia riešeného problému

Informácie v tejto kapitole vysvetľujúce fungovanie systému Kontent.ai sú napísané s pomocou dokumentácie systému [2]. Obrázky sú snímky obrazovky z testovacieho projektu vytvoreného pre účely tejto diplomovej práce.

Jedným z najčastejších typov obsahu, ktoré majú zákazníci v systéme Kontent.ai uložené, sú obrázky. V menšom počte sa vyskytujúce sú to ďalej videá a textové dokumenty. Všetky typy súborov konkrétneho projektu sú v systéme uložené v knižnici súborov (v systéme nazývaná *Asset Library*). Táto knižnica dáva zákazníkovi kompletný prehľad o všetkých súboroch nahraných do daného projektu. Príklad knižnice v testovacom projekte je možné vidieť

[1] https://sk.wordpress.org/

na obrázku 1.



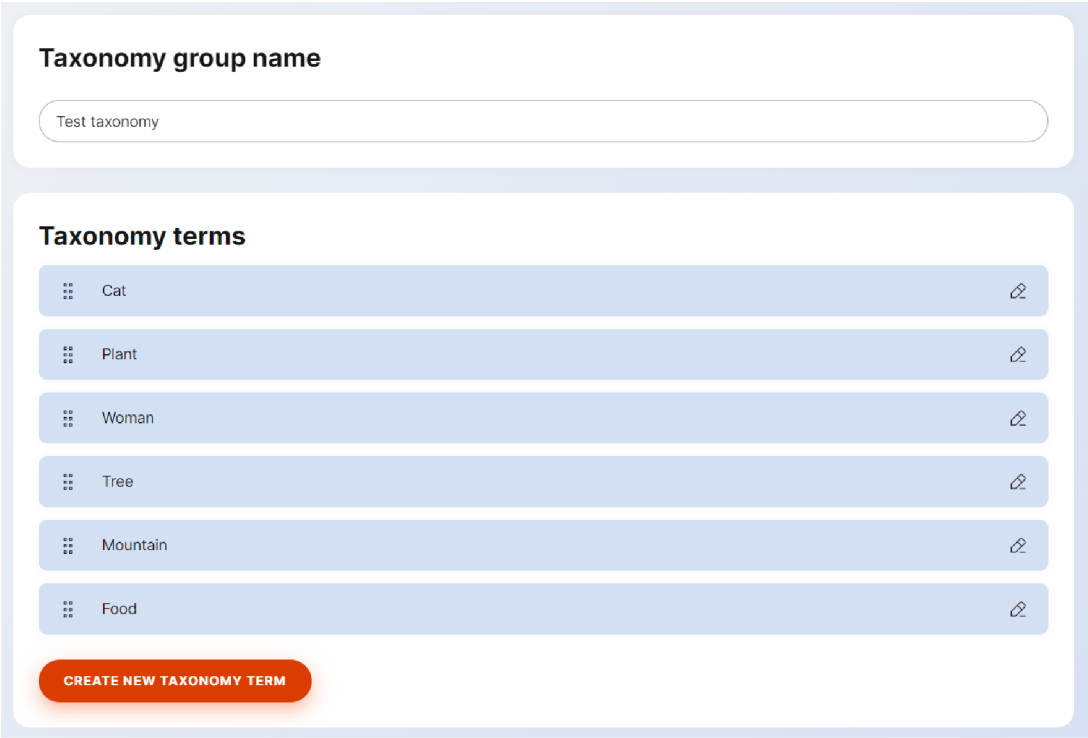
Obr. 1: Príklad knižnice súborov v softvéri Kontent.ai

Na obrázku 1 vľavo hore je možné vidieť vyhľadávaciu lištu. Vyhľadávanie v knižnici je možné cez názvy alebo popisy súborov. Existuje však aj ďalšia možnosť, práve dôležitá pre cieľ tejto diplomovej práce. Jedná sa o vyhľadávanie relevantných obrázkov podľa kľúčových slov popisujúcich obsah týchto obrázkov. Stačí do vyhľadávania napísať napríklad slovo „mačka“, a v knižnici sa zobrazia všetky obrázky, na ktorých sa nachádza mačka. To je možné pomocou taxonómií súborov.

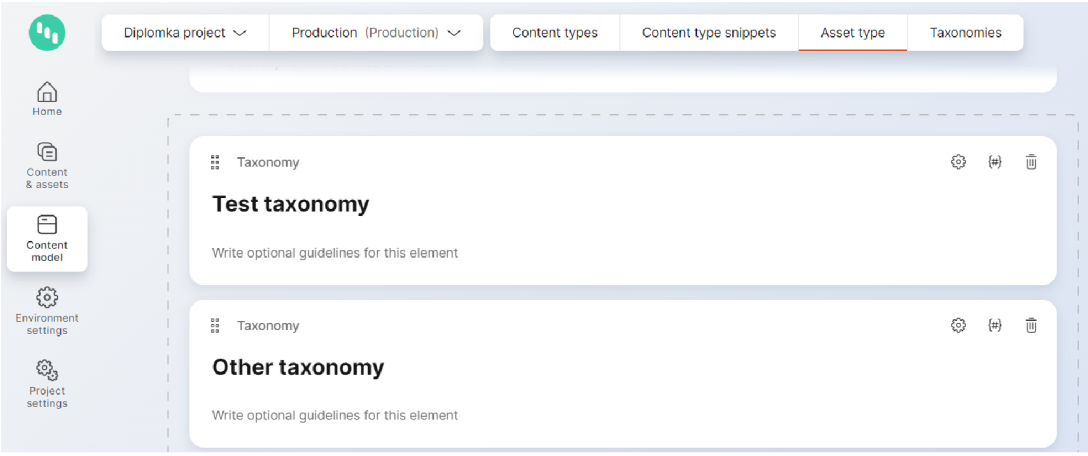
Taxonómia predstavuje označenie obrázka (či iného typu súboru) príslušnými pojmami napríklad podľa toho, čo sa na obrázku nachádza, podľa oddelenia či regiónu ku ktorému obrázky patria, či téma ku ktorej sa vzťahujú. Rozdelenie obrázkov podľa taxonómií je možné spraviť ľubovoľne podľa konkrétneho prípadu daného projektu. Pri použití taxonómií súborov na lepšiu organizáciu v knižnici je možné vyhľadávanie podľa priradených pojmov (anglicky *Taxonomy Terms*). Vďaka konzistentnému označovaniu súborov pomocou taxonómií je možné efektívne spravovať aj veľké knižnice.

Pre vytvorenie takýchto taxonómií je ako prvé potrebné vytvoriť skupinu taxonómií (anglicky *Taxonomy Group*) s ľubovoľným menom a pridať do nej jednotlivé pojmy (predstavujúce napríklad obsah obrázkov, viď obrázok 2).

Ďalším krokom je pridanie vytvorenej taxonómie do dostupných elementov súborov (v systéme anglicky nazvané *Asset Type*), aby výber pojmov z vytvorenej skupiny taxonómií bol pri obrázkoch k dispozícii. Tento krok je možné vidieť na obrázku 3.

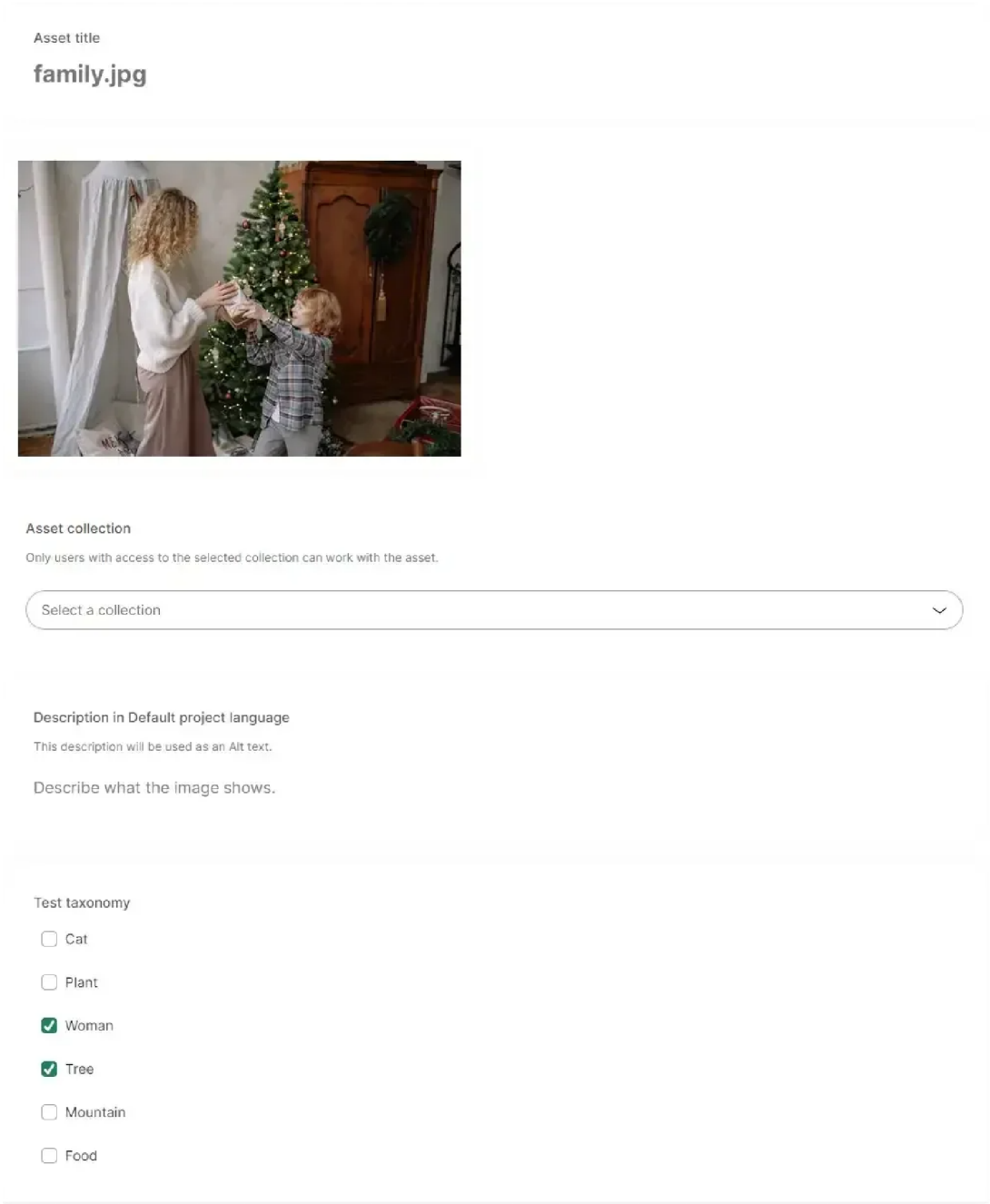


Obr. 2: Príklad skupiny taxonómií s jednotlivými pojmami v softvéri Kontent.ai



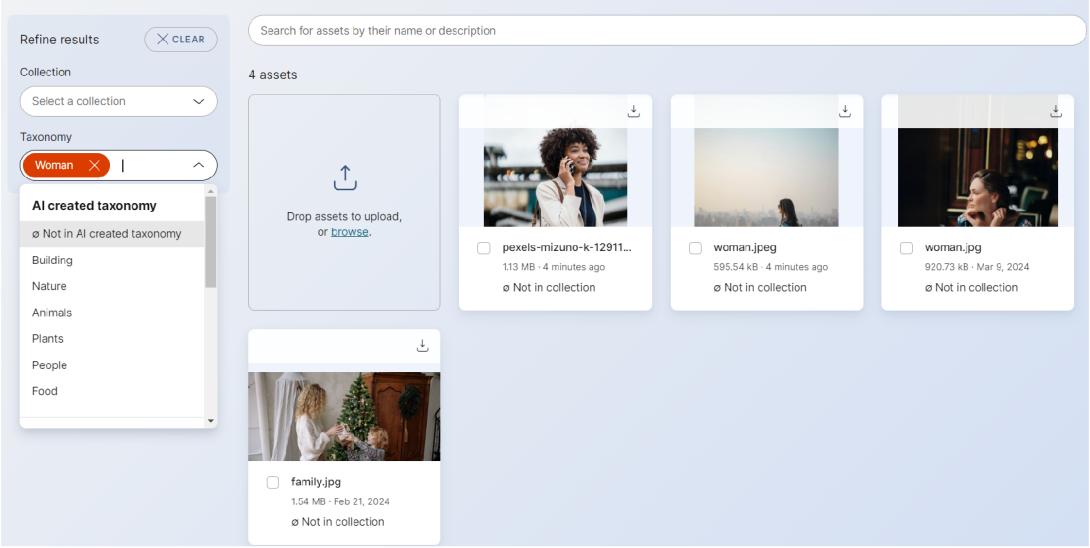
Obr. 3: Priradenie vytvorenej skupiny „Test taxonomy“ do elementov súborov v softvéri Kontent.ai

Samotné priradenie pojmov z taxonómie na obrázky je potom možné urobiť kliknutím na detail jednotlivých obrázkov v knižnici. Pri každom obrázku je potrebné označiť žiadané zaškrtávacie políčka (alebo v prípade väčšieho počtu pojmov ich vybrať z ponuky), ako je možné vidieť na obrázku 4.



Obr. 4: Príklad označenia obrázku príslušnými pojmami v softvéri Kontent.ai

Nakoniec po označení všetkých žiadaných súborov ich bude možné filtrovať podľa priradených pojmov, ako na obrázku 5.



Obr. 5: Príklad filtrovania obrázkov pomocou jednotlivých pojmov v softvéri Kontent.ai

Celý tento proces je aktuálne v systéme možné vykonávať iba ručne. Je teda najskôr potrebné vytvoriť skupinu taxonómií, ručne tam vložiť množstvo pojmov a jednotlivé obrázky týmito pojmami označiť. Až potom je možné podľa nich v knižnici vyhľadávať. Tento proces môže byť časovo náročný (veľké projekty môžu obsahovať aj tisícky súborov) a najmä nezáživný. V dnešnej dobe ale existujú možnosti, vďaka ktorým nemusí človek manuálne označovať čo sa nachádza na tisícoch obrázkov. A práve tu prichádza do hry inteligentná klasifikácia a rozpoznávanie obrazu pomocou umelej inteligencie.

Pomocou technológií rozpoznávania obrazu by mohol byť vyššie popísaný proces zautomatizovaný. To by mohla byť obrovská výpomoc pre projekty, ktoré majú veľkú a nezorganizovanú knižnicu súborov. Bolo by žiaduce, aby jednoduchým a intuitívnym spôsobom mohli užívatelia celú knižnicu rýchlo a spoľahlivo označiť. Pre rozpoznávanie obsahu na rôznych druhoch súborov (primárne zameranie je na obrázky, ale implementované bude aj riešenie pre videá) bude využitý jeden z verejne dostupných poskytovateľov, ktorí mali možnosť svoje služby natrénovať na obrovských sadách dát a preto sú ich výsledky spoľahlivé a vo väčšine prípadov presné. Pomocou toho bude navrhnutá skupina taxonómií s množstvom vygenerovaných pojmov podľa rozpoznaného obsahu na súboroch. Ďalej budú jednotlivé pojmy automaticky k obrázkom priradené. Pomocou týchto služieb by sa tiež dali generovať textové popisy k obrázkom, ktoré by mohli byť uložené do príslušnej kolonky v detaile obrázku. Na to všetko je potrebné v prvom rade vybrať vhodného poskytovateľa technológií rozpoznávania obsahu.

3 Dostupní poskytovatelia technológií rozpoznávania obsahu

Na trhu existuje mnoho spoločností špecializujúcich sa na softvér poskytujúci umelú inteligenciu. Tieto spoločnosti sa v dnešnej dobe dostali do centra pozornosti a vytvárajú významné inovácie v tejto oblasti. V nasledujúcej časti budú predstavení najväčší technologickí giganti ako Google či Amazon, či lídri v tomto odvetví, napríklad Microsoft, IBM, NVIDIA. Tieto spoločnosti investovali rozsiahle zdroje a expertízu na vývoj mocných riešení založených na umelej inteligencii, ktoré umožňujú iným podnikom porozumieť vizuálnym dátam. Predstavené budú však aj menej známe a open-source riešenia, ktoré takisto poskytujú zaujímavé riešenia zdarma alebo v nižších cenových kategóriách než spomínaní giganti. [3]

Kvôli hlavnému cieľu tejto práce, ktorým je implementácia aplikácie na klasifikáciu obsahu, a vykonaného prieskumu typov obsahu najviac využívaných zákazníkmi Kontent.ai, ktorý je popísaný v kapitole 3.2, bol prieskum poskytovateľov zameraný najmä na riešenia, ktoré obsahujú:

- Možnosť klasifikácie či označovania obsahu (v terminológii umelej inteligencie bežne používaný anglický pojem tagging pre označovanie obsahu, v tomto prípade označovanie obrázkov a videí kľúčovými slovami podľa toho, čo sa na nich nachádza), prípadne tiež generovania popisu.
- Podpora obrázkov, videí a PDF dokumentov.

Pri každom poskytovateľovi boli skúmané poskytované služby, cena, spôsob použitia (API či iné), výkon a prípadné API limitácie a výstup analýzy (kľúčové slová, Confidence – percentuálna istota vo výsledku, či popis celou vetou). Každý z poskytovateľov má k dispozícií väčšie množstvo služieb, s rozdielnymi cenami a možnosťami. Rešerš a výpočty sa týkajú vždy najmä služieb, ktoré by boli pre účely tejto práce najviac prínosné. Ceny za služby budú porovnávané za jeden milión obrázkov.

Google Cloud Vision AI

Táto technológia od Google stojí na popredí systémov rozpoznávania obrázkov a poskytuje rýchle a presné výsledky. Využíva pokročilé algoritmy strojového učenia a umelej inteligencie na spracovanie a interpretáciu obrazových dát. Google svoje neurónové siete trénoval na masívnych rôznorodých dátových súboroch. Toto učenie na rozsiahlych trénovacích dátach zaisťuje, že algoritmy sú schopné porozumieť a presne analyzovať širokú škálu typov obrázkov, pričom poskytuje veľkú úroveň presnosti a spoľahlivosti. [4]

Poskytované služby: [4]

1. **Vertex AI Vision** Plne spravované prostredie na vývoj end-to-end aplikácií, ktoré umožňuje jednoducho vytvárať, nasadzovať a spravovať aplikácie počítačového videnia pre konkrétne potreby podnikov.

2. **Custom ML models** Umožňuje automatizovať trénovanie vlastných modelov strojového učenia na klasifikáciu obrázkov či detekciu objektov.

3. **Vision API** Využitie natrénovaných modelov cez REST/RTC API na označovanie obrázkov, klasifikáciu do miliónov preddefinovaných kategórií, detekciu objektov, tvárí, OCR vytlačeného aj písaného textu (Optical Character Recognition – Optické rozpoznávanie znakov, metóda digitalizácie tlačeného textu), zisťovanie metadát ako vlastnosti obrázku, farby, hľadanie nevhodného obsahu.

Pre potreby tejto diplomovej práce by najviac pripadala do úvahy služba Vision API. Tá poskytuje technológiu na označovanie obrázkov, nie však tvorbu popisu obrázku celou vetou. Podporuje obrázky, videá aj texty, teda všetky typy obsahu, na ktoré má byť práca zameraná.

Čo sa týka ceny za službu, platby sú účtované podľa využitia. Poplatky sú účtované za obrázok, pri PDF dokumente je každá strana braná ako jeden obrázok. Každá vlastnosť využitá pre daný obrázok je zvlášť zúčtovateľnou jednotkou (anglicky *billable unit*). Pre rôzne vlastnosti sú ceny len mierne odlišné, približne by však cena vychádzala následovne: [5][2]

- Prvých 1000 jednotiek je každý mesiac zdarma,
- 1001–5 miliónov jednotiek/mesiac = približne $1,5 za 1000 jednotiek,
- 5 miliónov a viac jednotiek/mesiac = približne $0,6 za 1000 jednotiek.

= za milión obrázkov (konkrétne LABEL_DETECTION):

$$1000000 - 1000 = 999000$$

$$999000 \div 1000 = 999$$

(zaokrúhlenie na celú začatú 1000ku)

$$1000 \cdot \$1,5 = \mathbf{\$1500}$$

API limitácie Google Cloud Vision AI sú maximálne 1800 žiadostí za minútu a maximálna veľkosť obrázku 20 MB.

[2] Cenník aktuálny k 22. 02. 2024

Amazon Web Services – Amazon Rekognition

Služba Amazon Rekognition poskytuje automatizáciu a zníženie nákladov na rozpoznávanie obrázkov a analýzu videa pomocou strojového učenia. Sľubuje rýchle pridanie pred-trénovaných či prispôsobiteľných API počítačového videnia do aplikácií bez nutnosti vytvárať vlastné modely strojového učenia a infraštruktúru od nuly, analýzu miliónov obrázkov a videí v priebehu sekúnd a škálovanie podľa konkrétnej potreby zákazníka s platbou iba za obrázky a videá, ktoré budú zanalyzované.

Medzi základné služby patrí porovnávanie tvárí, rozpoznávanie tvárí v obrázkoch či videách, atribútov ako otvorené oči, okuliare a ďalšie. Ďalej detekciu nevhodného obsahu, generovanie textu, označovanie (objekty, scény, aktivity), segmenty vo videách či rozpoznávanie celebrít.

Podporované sú obrázky a videá, PDF dokumenty dokáže spracovávať zvlášť služba Amazon Textract. Špecialitou Amazonu, ktorá pre prípad tejto práce nie je úplne výhodná, je nutnosť posielať dáta do Amazonu ako bitové pole, prípadne uložiť ich do Amazon S3 bucket. Samotná komunikácia už potom prebieha pomocou REST API. [6]

Cena za túto službu sa odvíja podľa toho, koľkokrát bolo prevolané API. Zavolanie viacerých API na jeden obrázok sa ráta ako spracovanie viacerých obrázkov. Na 12 mesiacov je služba pre 5000 obrázkov mesačne ponúkaná zdarma na vyskúšanie. Inak je cenník rozdelený na 2 skupiny podľa typu služby. Konkrétne pre prípad označovania obrázkov (API DetectLabels) je **cena za milión obrázkov:** [7][3]

$$1000000 \div \$0,001 = \mathbf{\$1000}.$$

Pri väčšom počte obrázkov sa cena za obrázok znižuje. Čo sa týka videí, vrámci spomínanej 12 mesačnej doby je zdarma 1000 minút video analýzy mesačne. Inak je to $0,10/min.

Azure Cognitive Services – Computer Vision

Azure AI Vision je zjednotená služba od spoločnosti Microsoft, ktorá ponúka inovatívne možnosti počítačového videnia. Poskytuje analýzu obrázkov, ktorá využíva viac ako 10000 konceptov a objektov na detekciu, klasifikáciu a vytváranie popisov. Pomocou priestorovej analýzy je možné pochopiť prítomnosť a pohyb ľudí vo fyzickom priestore v reálnom čase. Ponúka tiež extrahovanie textu z obrázkov a dokumentov (PDF, TIFF), zhrnutie hlavných objektov vo videu či odhalenie citlivého obsahu na obrázku. Prehľad všetkých služieb je dostupný v [8].

Žiadosti sú posielané pomocou REST API s akýmkoľvek typom obsahu: image/*, application/octet-stream alebo JSON s URL adresou s obrázkami.

[3] Cenník aktuálny k 22. 02. 2024

Najzaujímavejšou službou pre túto prácu by bolo extrahovanie bežných tagov z obrázkov. Pri počítaní ceny za službu je každá vybraná vlastnosť počítaná ako jedna transakcia. Jeden mesiac je možné služby využívať v skúšobnej inštancii zdarma. Maximum je pritom 20 transakcií za minútu a 5000 transakcií na daný mesiac. Zvyšné služby sú rozdelené do rôznych cenových skupín, viď [9].[4] Klasifikácia službou Tag by pre **milión obrázkov** mala výslednú cenu:

$$1000000 \div 1000 \cdot \$1 = \mathbf{\$1000}$$

Táto služba funguje na obrázkoch, ktoré splňujú nasledujúce požiadavky:

- Obrázok musí byť v jednom z týchto formátov: JPEG, PNG, GIF, BMP, WEBP, ICO, TIFF alebo MPO.
- Obrázok musí byť menší ako 20 MB.
- Rozmery obrázka musia byť väčšie ako 50 × 50 pixelov a menšie ako 16 000 × 16 000 pixelov. [10]

Imagga API

Imagga API sľubuje posilnenie inteligentných aplikácií pomocou prispôsobiteľnej technológie strojového učenia. Je to súbor technológií na porozumenie a analýzu obrazu dostupných ako webová služba, ktorá poskytuje automatizáciu procesu analýzy, organizácie a vyhľadávania vo veľkej kolekcii neštruktúrovaných obrázkov. Služba je dostupná na báze softvér ako služba (SaaS – anglicky Software as a Service), čo ju činí veľmi flexibilnou, škálovateľnou (aplikovateľná na akúkoľvek veľkosť zbierky obrázkov) a cenovo dostupnou. Podporuje označovanie obrázkov, vizuálne vyhľadávanie, rozpoznávanie tvárí, kategorizáciu a ďalšie. Žiadosti na analýzu sú posielané pomocou REST API s autentifikáciou, pričom vracané sú napríklad tagy so stupňom dôveryhodnosti. Je možné tiež natrénovať vlastné modely.

Čo sa týka ceny, najskôr je potrebné vybrať si plán, ktorý je ale možné kedykoľvek zvýšiť či znížiť. V inštancii, ktorá je poskytovaná zdarma, je možných 1000 API žiadostí mesačne. V triede Indie je to $79 mesačne za 70000 žiadostí, Pro je za $349 za mesiac s 300000 žiadosťami. Na zistenie ceny za prípadný Enterprise plán s vlastným trénovaním modelov a 1 miliónom žiadostí je potrebné firmu kontaktovať. [11]

Clarifai

Clarifai je jednou z popredných technologických firiem, ktorá prináša riešenia v oblasti spracovania obrazu. Ponúka široké spektrum služieb a nástrojov, ktoré umožňujú rýchle a presné analýzy a interpretácie vizuálnych

[4] Cenník aktuálny k 22. 02. 2024

dát. Možné sú využiť natrénované technológie rozpoznávania obrazu, analýzu videa, OCR či generovanie obrázkov. Používa pokročilé klasifikačné modely na podrobnú klasifikáciu a analýzu údajov, čo umožňuje rýchle a presné rozhodovanie.

Clarifai podporuje rôzne formáty videí, obrázkov, audio i textu. Pre jednoduchú integráciu všetkých služieb je v jazyku C# dostupný Clarifai Nuget balíček. Prevolávanie technológií rozpoznávania obrazu potom prebieha pomocou REST API. Čo sa týka stanovenia ceny, existujú rôzne plány a vrámci nich rôzne možnosti využívaných služieb. Konkrétne pre klasifikáciu obsahu cena vychádza na $1,2 za 1000 žiadostí, pre detekciu obsahu $2 za 1000 žiadostí. Celková cena by teda mohla byť až **$2000 za milión obrázkov**. [12]

Ďalšie riešenia

IBM Watson Visual Recognition bola aplikácia strojového učenia pre označovanie a klasifikáciu obrazových dát a nasaditeľná na širokú škálu účelov. V roku 2021 však bol jej vývoj a podpora prerušená a viac nie je dostupná.[5]

Čo sa týka spoločnosti **NVIDIA**, tá poskytuje viac špecifické riešenia rozpoznávania obsahu, ako rozpoznávanie osôb či áut vo videu, či NVIDIA Clara Imaging pre zdravotnícky priemysel.[6]

Ďalšou možnosťou je **OpenCV**, v dobe písania tohto textu však neposkytovala službu vhodnú pre účely tejto diplomovej práce, zameranie je najmä na rozpoznávanie tvárí.[7]

3.1 Porovnanie poskytovateľov

Pre porovnanie jednotlivých poskytovateľov služieb a výber vhodného pre účely tejto práce bola vytvorená tabuľka vlastností poskytovateľov 1, ktorá ich porovnáva na základe jednotlivých, pre tento prípad použitia dôležitých, vlastností.

Aktuálne sa najvhodnejšími kandidátmi zdajú byť Azure Cognitive Services, Amazon Rekognition a Google Vision AI. Sú to služby najznámejšie a najvyvinutejšie na trhu, čo znamená, že rozpoznávanie bolo pravdepodobne natrénované na väčších súboroch dát, a tým pádom bude dávať presnejšie výsledky ako ostatné dostupné služby. Tiež sú to poskytovatelia, ktorí pri väčších objemoch dát, ktoré zákazníci Kontent.ai majú, žiadajú najnižšie ceny za poskytované služby.

[5] https://www.ibm.com/us-en
[6] https://www.nvidia.com/cs-cz/
[7] https://opencv.org/

	Vlastnosti		Typy súborov			Cena
	Tagy	**Popis**	**Obrázky**	**Videá**	**PDF**	**(1M žiadostí)**
Azure	✓	✓	✓	✓	✓	$999
Amazon	✓	✗	✓	✓	✓	$1000
Google	✓	✗	✓	✓	✓	$1500
Imagga	✓	✗	✓	✗	✗	neznáma
Clarifai	✓	✓	✓	✓	✓	$2000

Tabuľka 1: Tabuľka vlastností poskytovateľov technológií rozpoznávania obrazu

3.2 Testovanie vybraných poskytovateľov

Pre zistenie pomeru rôznych typov súborov, ktoré majú zákazníci uložené v softvéri Kontent.ai, bolo potrebné nahliadnuť do produkčnej databázy a správne položiť dotazy na typy súborov v knižnici súborov. Na základe týchto dotazov bolo zistené, že 85 % súborov sú obrázky, 12.5 % PDF dokumenty a 5 % videá (zaokrúhlene). Zvyšné percentá tvoria iné typy súborov, ktoré môžem pre tvorbu prototypu aplikácie pre túto prácu zanedbať. Podľa tohto rozloženia, odpovedajúceho distribúcii na produkcii, bola pripravená množina súborov, na ktorých prebehlo testovacie porovnanie 3 spomenutých vybraných poskytovateľov. Testovanie prebiehalo na online skúšobných verziách API jednotlivých poskytovateľov:

- Azure Vision Studio,[8]
- Google Cloud Vision,[9]
- Amazon Rekognition (je potrebná registrácia).[10]

Testovanie prebiehalo na základe subjektívnych názorov 4 osôb, ktoré si manuálne prešli vygenerované tagy od každého sprostredkovateľa ku každému obrázku a vytvorili si vždy poradie, ktoré tagy najlepšie a najpresnejšie vystihovali či najviac sa priblížili skutočnosti, teda tomu čo rozozná na obrázku človek. Na základe takéhoto ohodnotenia pre dostatočne veľkú sadu obrázkov potom určili celkové poradie týchto sprostredkovateľov.

Z tohto testovania okrem celkového výsledku podľa nezávislého názoru 4 ľudí (viď tabuľka 2) tiež vznikli postrehy a poznámky k jednotlivým službám, pretože po otestovaní na pár obrázkoch sa dali odsledovať špecifické javy či črty jednotlivých sprostredkovateľov k spôsobu označovania. Hoci sa jednalo o nezávislé a subjektívne názory, veľa z nich sa zhodovalo.

[8] https://portal.vision.cognitive.azure.com/demo/generic-image-tagging
[9] https://cloud.google.com/vision?hl=en#section-6
[10] https://eu-west-2.console.aws.amazon.com/rekognition/home?region=eu-west-2#/label-detection

	Subjekt 1	Subjekt 2	Subjekt 3	Subjekt 4	Výsledok
1.	Azure	Azure	AWS	AWS	**Azure**
2.	AWS	Google	Azure	Azure	**AWS**
3.	Google	AWS	Google	Google	**Google**

Tabuľka 2: Poradie poskytovateľov určené testovaním štyrmi osobami

Azure

- generuje najmenší počet tagov (stručné), ale veľmi presné a výstižné, presnosťou asi najlepšie
- generuje tagy ktoré sú skôr obecné, nezameriava sa moc na detaily
- neporadí si príliš dobre s netradičným obrázkom
- chyby sa vyskytovali až pri veľmi nízkej dôveryhodnosti (zaužívaný anglický pojem Confidence), vysoké čísla Confidence dávali správne výsledky

Záver: menej chybový, menej tagov, zameraný viac na celok, čo je pre účely tejto práce ideálna kombinácia.

Google

- generuje najväčší počet tagov, ale občas nesprávne
- veľa tagov má malú Confidence
- tagy s najväčšou Confidence neboli úplne najvýstižnejšími pojmami pre daný obrázok
- niektoré tagy sú až zbytočné (príliš zložité, nepoužívané a cudzojazyčné pojmy, ktoré by nevystihovalo viac obrázkov, len daný konkrétny)

Záver: Pre správne využitie by bolo potrebné určiť nejaký prah Confidence, od ktorej by boli tagy pre obrázok prijaté, aby tým boli odfiltrované tie nesprávne. Inak je Google zrovnateľný s Azure, generuje toho trochu viac.

AWS

- generuje najdetailnejšie tagy, všíma si však niekedy zbytočné a nepodstatné drobnosti
- generuje veľké množstvo tagov
- u správnych tagov dáva vysoké Confidence, čo je veľmi dobré
- poradil si pomerne dobre aj na netradičnom obrázku

- bolo tu však spozorovaných najviac chýb v porovnaní s ostatnými službami, a to s pomerne vysokou Confidence

Záver: Pre použitie na túto prácu generoval príliš veľké množstvo nepresností s vysokými Confidence, čo nie je úplne vhodné. Tento nástroj by bolo dobré použiť pre hľadanie väčších detailov, kde by nevadili časté falošne pozitívne výsledky.

Dôkaz niektorých z predošlých výrokov je možné vidieť na nasledujúcich príkladoch konkrétnych obrázkov a aké tagy na nich našli jednotliví poskytovatelia. Je potrebné zdôrazniť, že nájdené črty jednotlivých poskytovateľov bolo možné odpozorovať na väčšom množstve rôznych obrázkov a nasleduje len demonštrácia nájdených vlastností. Tagy sú generované v anglickom jazyku.

Ako prvý príklad odsledujeme rozdiely na obrázku 6.

Obr. 6: Obrázok veľkomesta s budovami, obchodmi, ulicou s autami a ľudmi[11]

Azure Computer Vision rozpoznalo v tomto obrázku 20 tagov. Všetky boli pravdivé a pomenovávali väčšie a výraznejšie objekty v obrázku. Google ich zas rozpoznalo 50, pričom bolo využitých veľa synoným (vehicle, car, motor vehicle, personal luxury car). Všetky tiež boli správne, tiež s pomerne veľkou Confidence. AWS ich tu rozpoznalo 28, pričom objavilo aj detaily ako handbag, accessories, footwear, ktoré nie sú úplne najvýraznejšou súčasťou tohto obrázku.

[11] Zdroj:https://www.pexels.com/photo/crowd-of-people-on-street-during-night-time-1737957/

Veľký zmysel pre detail u AWS je možné odsledovať aj na obrázku 7. Ten s veľkou Confidence (89,9) našiel na obrázku dospelého človeka, čo si človek na prvý pohľad ani nevšimne, a takisto sa nedá naisto tvrdiť že v pozadí je muž (takisto Confidence 89,9). Tiež tu nesprávne našiel ananás či rastlinu. Azure na tomto obrázku našiel 16 tagov, opäť všetky správne. Čo je tiež zaujímavé na tomto obrázku odsledovať sú tagy od Google, ktorých je opäť pomerne veľa (45) a obsahujú aj slová ako Cicchetti, À la carte food či Hors d'oeuvre, čo nie sú úplne bežne používané výrazy a pravdepodobne by sme podľa nich nešli vyhľadávať obrázok v knižnici obrázkov.

Obr. 7: Obrázok stola s donutmi a inými zákuskami[12]

Ďalší obrázok na ktorom boli služby otestované, je nasledujúci netradičný obrázok 8 (netradičným je myslené že to nie je úplne bežne sa vyskytujúca scéna, a teda pravdepodobne trénovanie služieb na veľa podobných obrázkoch neprebehlo, prípadne sa objekty podobajú na niečo iné).

Všetky 3 služby si poradili s týmto obrázkom podobne. Každá rozpoznala niečo spojené so svadbou, svadobné šaty, svadobný odev. Jedine služba AWS našla aj slovo cloak (plášť), čo je viac trefné. Zaujímavé je, že pričom AWS aj Google mali opäť tagov pomerne veľa, služba Azure ich našla iba 6 (wedding dress, outdoor, skyscraper, building, city, head covering), čo len potvrdzuje že Azure generuje len tagy ktorými je si skutočne istá a zbytočne si „nevymýšľa".

[12]Zdroj: https://www.pexels.com/photo/donuts-and-bagel-display-205961/

[13]Zdroj:https://www.pexels.com/photo/unrecognizable-couple-in-ghost-costumes-in-city-5911012/

Obr. 8: Obrázok dvoch osôb zakrytých plachtami s okuliarmi v meste[13]

3.3 Konečný výber poskytovateľa a zdôvodnenie

Na základe poskytovaných vlastností, testovania popísaného v predchádzajúcej kapitole a spozorovaných faktov spôsobu označovania bolo rozhodnuté, že najvhodnejšou službou pre túto diplomovú prácu je Azure Cognitive Services. Táto služba je najmenej chybová, stručná a zameraná viac na celok, čo je pre naše účely ideálne. K rozhodnutiu pozitívne prispel aj fakt, že firma Kontent.ai využíva od spoločnosti Microsoft Azure celé spektrum cloudových služieb. Vďaka tomu môžu byť na testovacie účely a vytvorenie proof-of-concept aplikácie využité služby Vision AI už z existujúcej infraštruktúry Azure a cena za vývoj a testovanie bude vďaka tomu zanedbateľná.

4 Návrh riešenia, implementácia a použité technológie

Pre vyriešenie problému klasifikácie obsahu v systéme Kontent.ai bolo firmou vyžiadané navrhnúť a implementovať aplikáciu, ktorá by otestovala a demonštrovala schopnosť využitia vybraných služieb. Navrhnutá mala byť jednoduchá webová aplikácia, do ktorej zákazník zadá identifikátor a kľúč k svojmu projektu a bude môcť spustiť klasifikáciu nad existujúcimi súbormi (obrázkami a videami) informujúcu o progrese. Vygenerované informácie si bude môcť zákazník prehľadne prezrieť. Pomocou aplikácie má byť tiež možné vygenerovanú taxonómiu exportovať do projektu a zaradiť do nej jednotlivé súbory, tiež s možnosťou generovania popisov obrázkov.

Celú logiku aplikácie je možné rozdeliť na 3 základné časti. Prvou časťou bude prezentačná vrstva, teda samotné **používateľské rozhranie** webovej aplikácie. Druhou časťou je jej komunikácia so softvérom Kontent.ai. Na tieto účely slúži firmou udržiavané verejné API, nazývané **Management API**, ku ktorému je dostupné SDK ako Nuget balíček použiteľný priamo v .NET aplikácii. Poslednou, treťou časťou, je analyzátor obrázkov a videí od poskytovateľa tretej strany a komunikácia s týmito službami, na čo boli vybraté služby od **Microsoft Azure**. Všetky tieto časti budú podrobne rozobrané a popísané v tejto kapitole spolu s implementačnými detailami a ukážkami zdrojového kódu vzniknutej aplikácie.

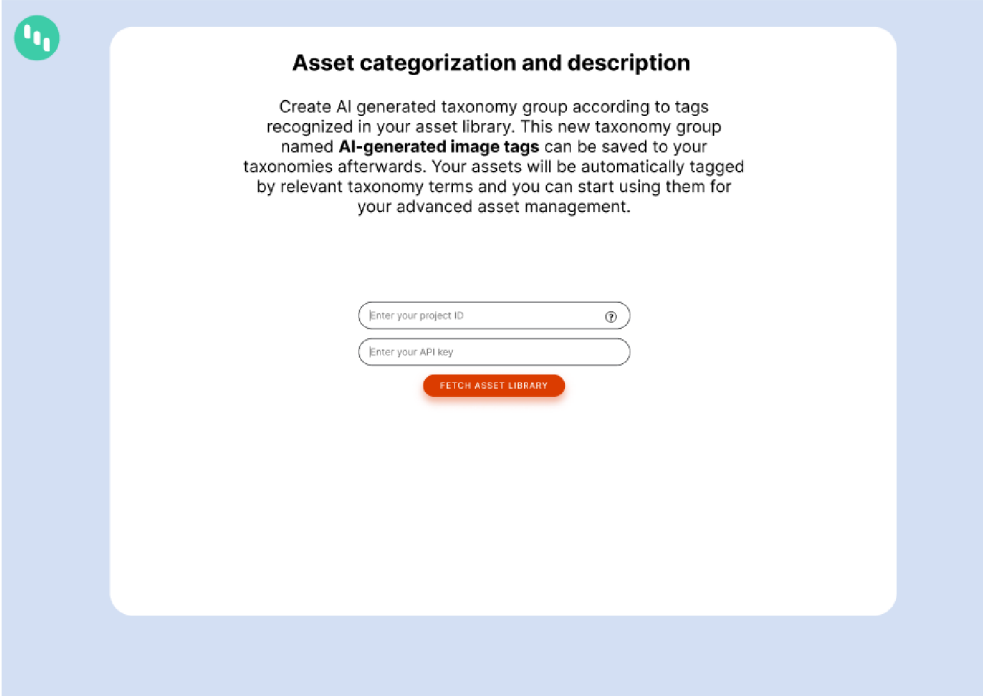
4.1 Návrh používateľského rozhrania

V tejto časti je možné vidieť snímky obrazovky z návrhu používateľského rozhrania vytvoreného v grafickom editore Figma.[14]

Jediným prípadom použitia aplikácie je použitie zákazníkom, ktorý má založený projekt v systéme Kontent.ai a má v ňom minimálne jeden súbor – obrázok alebo video. Užívateľ musí zo systému zistiť identifikačné údaje projektu – ID daného projektu (Environment ID) a vygenerovať Management API kľúč. Pomocou týchto údajov sa na úvodnej stránke aplikácie prihlási k danému projektu, viď obrázok 9. Kľúč je pritom citlivou informáciou, pretože pomocou neho sa dajú do projektu pridávať aj odstraňovať dáta. Tiež je potrebný informačný text o tom, na čo aplikácia slúži.

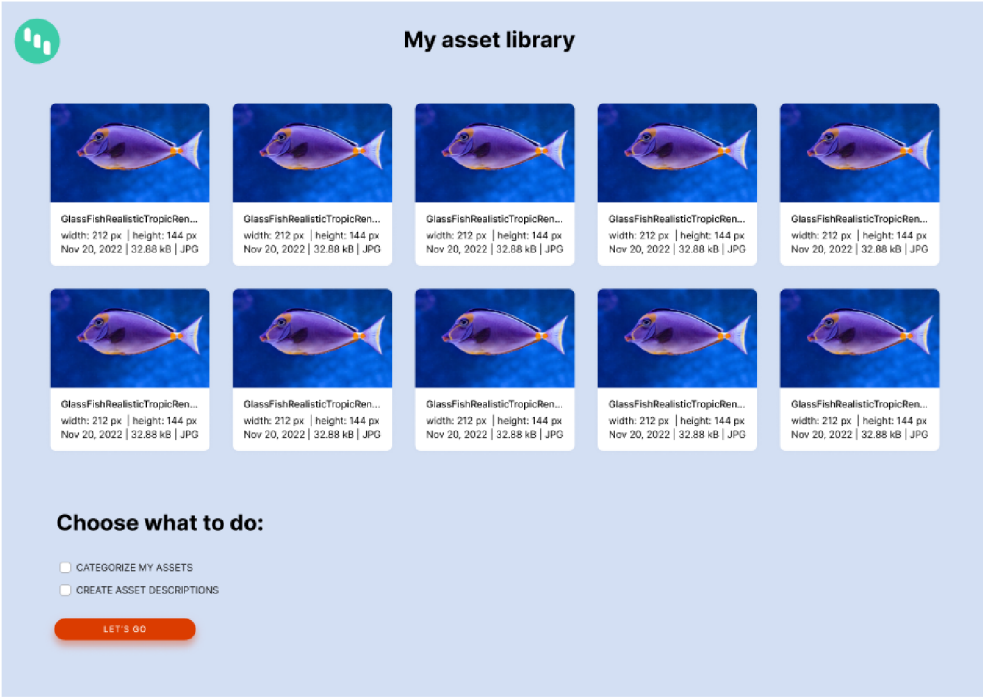
Po prihlásení sa zobrazí celá knižnica súborov daného projektu. Zákazník si môže prezrieť svoje súbory a následne si vybrať čo chce s nimi robiť – kategorizovať či vytvoriť popisy súborov. Návrh je možné vidieť na obrázku 10. Pôvodne navrhnuté užívateľské rozhranie bolo časom rozšírené o možnosť spracovania videí, takže vo finálnom produkte sú na výber 3 zaškrtávacie políčka – označiť obrázky, vytvoriť popisy obrázkov alebo označiť videá (to je možné vidieť obrázku 16). Je možné si vybrať aj viacero možností, aplikácia ich

[14] https://www.figma.com/design/ZCsEGBKsDpqcIRvirL3h8a/Diplomka?node-id=0-1&m=dev&t=NYt321mGfGmL2PgC-1



Obr. 9: Návrh úvodnej stránky aplikácie

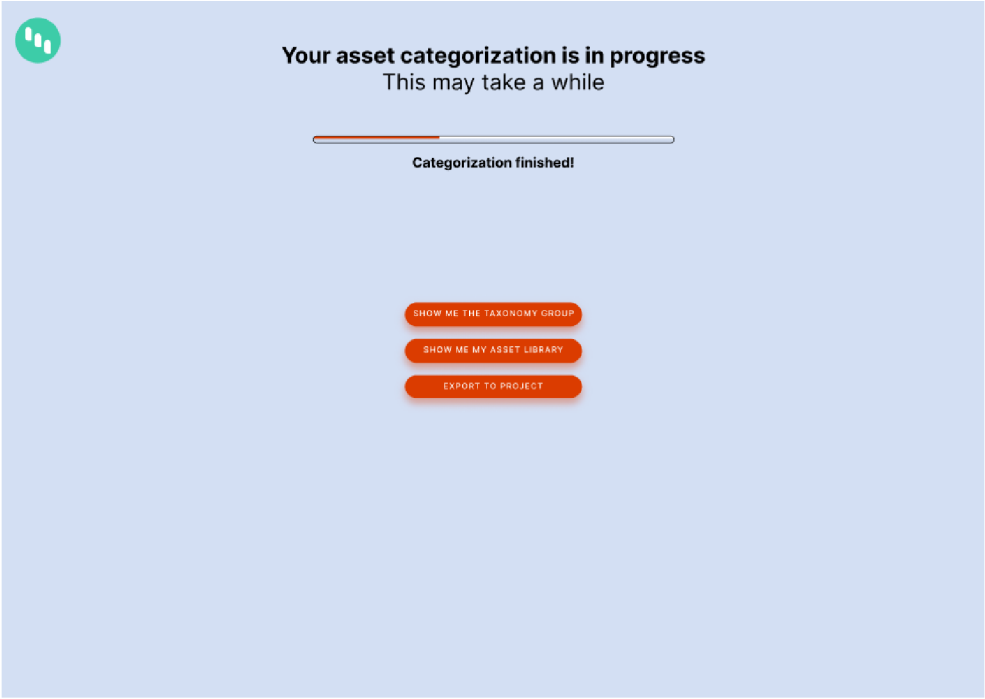
spracuje z užívateľského pohľadu súčasne.



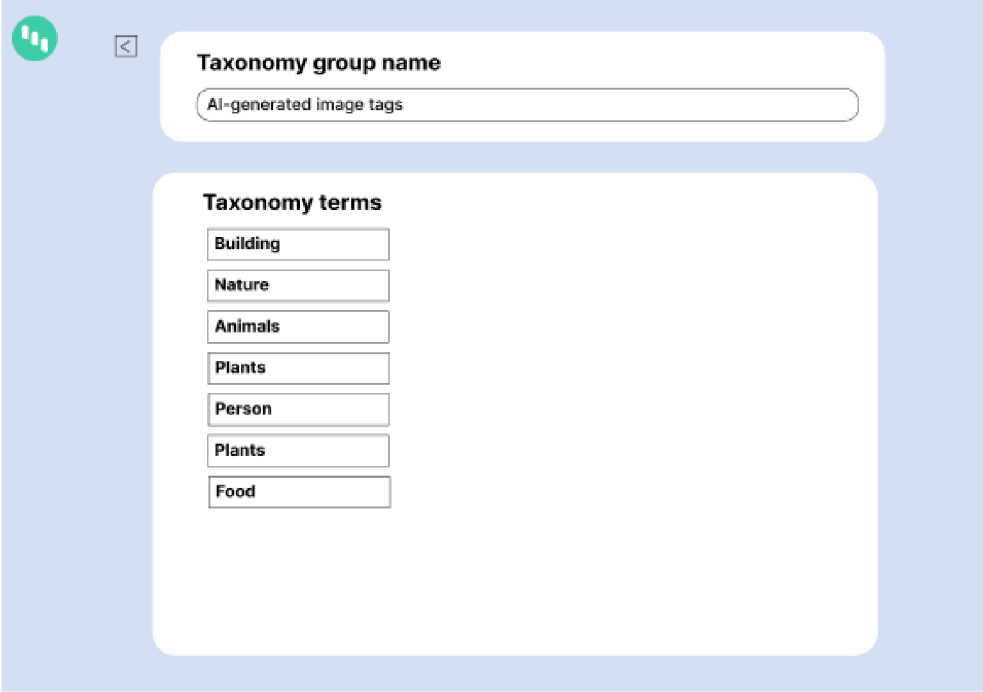
Obr. 10: Návrh druhej stránky aplikácie po načítaní knižnice súborov

Následne mal byť v aplikácii prítomný ukazovateľ progresu prebiehajúcej analýzy. Analýza obrázkov i vytváranie popisov je pomerne rýchlou činnosťou, analýza videí trvá dlhšie. Po skončení analýzy má užívateľ vrámci aplikácie ďalšie možnosti (viď obrázok 11):

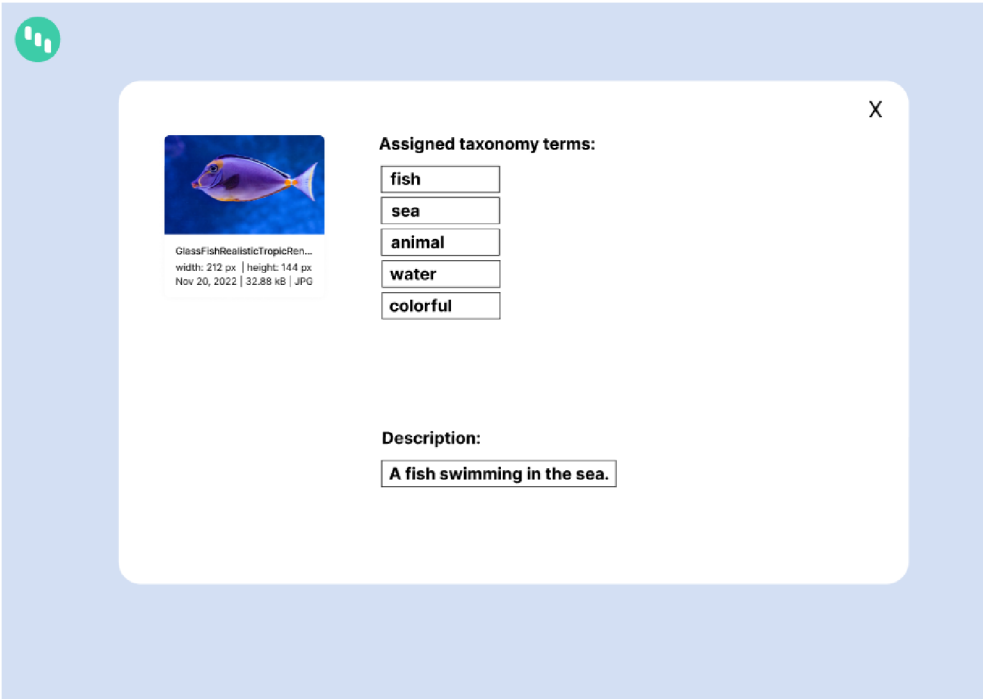
- Zobraziť si vygenerovanú skupinu taxonómií (viď obrázok 12) – počas programovania aplikácie prišlo k rozhodnutiu vygenerovať 2 skupiny taxonómií – zvlášť pre obrázky a zvlášť pre videá. Súčasťou týchto taxonómií bude aj pojem vyjadrujúci typ súboru – obrázok alebo video. Užitočné to bude pri filtrácii knižnice súborov, teda bude možné zobraziť si len obrázky alebo len videá.
- Zobraziť si opäť celú knižnicu súborov, avšak po kliknutí na jednotlivé obrázky alebo videá je zobrazený detail súboru aj s príslušnými vygenerovanými dátami. Návrh takéhoto detailu je možné vidieť na obrázku 13.
- Exportovať dáta do projektu. Po tomto kroku beh tejto relácie aplikácie skončí.



Obr. 11: Návrh tretej stránky aplikácie, zobrazujúci tlačítka s možnosťami ďalších akcií v aplikácii



Obr. 12: Návrh stránky aplikácie so zobrazením vygenerovanej skupiny taxonómií



Obr. 13: Návrh stránky aplikácie so zobrazením detailu zanalyzovaného obrázku

4.2 Kontent Management API

Management API je zabezpečené REST API (aplikačné programovacie rozhranie, ktoré umožňuje komunikáciu medzi systémami cez internet pomocou štandardných HTTP metód ako GET, POST, PUT a DELETE), ktoré poskytuje prístup na čítanie a zápis do Kontent.ai projektov. Využíva sa na správu obsahu, súborov, dátového modelu či nastavení projektu. [13]

Pre posielanie požiadaviek je potrebné mať v projekte rozhranie Management API aktivované. Musí byť vygenerovaný súkromný API kľúč a ním autorizované bezpečné požiadavky posielané cez HTTPS. Na zjednodušenie práce je verejne dostupné .NET SDK, čo je v podstate klientská knižnica, ktorá poskytuje funkcie na volanie API požiadaviek. Toto SDK sa do projektu pridá ako Nuget balíček.[15] Základné dostupné funkcie využité v tejto aplikácii sú: získanie všetkých súborov z knižnice súborov (`ListAssetsAsync`), ďalej vytvorenie skupiny taxonómií (`CreateTaxonomyGroupAsync`) a následné priradenie jednotlivých pojmov zo skupiny príslušným súborom (`UpsertAssetAsync`). Celá práca s Management API je naprogramovaná v súbore `ManagementApi.cs`.

4.3 Azure AI Vision Image Analysis

Služba Image Analysis od Microsoft Azure dokáže z obrázkov extrahovať množstvo rôznych vizuálnych informácií. Je možné ju používať cez klientskú SDK knižnicu alebo priamo volaním REST API. Pre knižnicu je oficiálne vydaný Nuget balíček. Ten poskytuje algoritmy pre spracovanie obrázkov a navrátenie informácií o ich obsahu. Umožňuje extrahovať jeden alebo viac vizuálnych informácií z obrázku zároveň, ako napríklad titulok k obrázku, extrahovanie textu z obrázku či detekcia objektov.

Pre vytvorenie akejkoľvek služby v Azure je najprv potrebné založiť predplatné. Pre všetky služby využité na účely tejto práce bolo vytvorené testovacie predplatné od firmy Kontent.ai s externým prístupom. Sú uložené pod jednou skupinou zdrojov, viď obrázok 14. Väčšina zdrojov na obrázku bude podrobnejšie popísaná vrámci tejto kapitoly.

Name	Type	Location
ai-content-classification	App Service plan	West Europe
ai-content-classification	App Service	West Europe
ai-content-classification	Application Insights	West Europe
Application Insights Smart Detection	Action group	Global
diplomkaAI	Computer vision	West Europe
diplomkaAI-video	Azure AI Video Indexer	West Europe
diplomkaKeyVault	Key vault	West Europe
diplomkastorageaccount	Storage account	West Europe
Failure Anomalies - ai-content-classification	Smart detector alert rule	Global

Obr. 14: Skupina zdrojov so všetkými vytvorenými službami pre účely tejto práce

[15] https://www.nuget.org/packages/Kontent.Ai.Management

Pod týmto predplatným je vytvorený zdroj **Computer Vision**. Pre autorizáciu na túto službu je potrebné vytvoriť inštanciu triedy `ImageAnalysisClient` a predať do nej 2 dôležité údaje – autorizačný kľúč a adresu koncového uzlu tejto služby (anglicky zaužívaný pojem *endpoint*). Implementácia v aplikácii je zobrazená v zdrojovom kóde 1.

```
private readonly ImageAnalysisClient _client;
private readonly IHubContext<ProgressHub> _progressHubContext;

public ImageAnalysis(IConfiguration configuration,
  IHubContext<ProgressHub> progressHubContext)
{
  var secretName = "ImageAnalysisKey";
  var key = configuration[secretName];

  if (key == null)
  {
   Console.WriteLine("No secret found.");
  }
  else
  {
   var endpoint = "https://diplomkaai.cognitiveservices.azure.com/";
   _client = new ImageAnalysisClient(
        new Uri(endpoint),
        new AzureKeyCredential(key));
  }
  _progressHubContext = progressHubContext;
}
```

Zdrojový kód 1: Vytvorenie triedy `ImageAnalysisClient` s potrebnou konfiguráciou

Ďalším potrebným krokom je výber vizuálnych vlastností, ktoré chceme na obrázku analyzovať. Všetky možnosti sú špecifikované triedou `VisualFeatures`. Nás zaujímajú dve z týchto možností – `VisualFeatures.Tags`, ktorá dokáže rozoznať tisíce objektov, živých bytostí, scenérií a akcií, ktoré sa na obrázku nachádzajú, a `VisualFeatures.Caption`, ktorá vygeneruje ľudsky čitateľnú vetu popisujúcu obsah obrázka. Užívateľ aplikácie má možnosť si vybrať, ktoré z týchto možností chce na obrázku vykonať, ako bolo možné vidieť v časti 4.1.

Posledným krokom v tomto procese je samotné odoslanie obrázka na analýzu. Balíček poskytuje funkciu `AnalyzeAsync`, ktorou je obrázok odoslaný cez verejne dostupnú URL do služby spolu s vybranými vizuálnymi vlastnosťami. Služba si obrázok stiahne, vykoná požadovanú analýzu a výsledok vráti v podobe triedy `ImageAnalysisResult`. Z nej sú výsledky spracované a uložené ku konkrétnym obrázkom pre ďalšie spracovanie aplikáciou. [14]

Celý tento proces s výberom vizuálnych vlastností, kontrolou požiadaviek

na obrázky vypísaných v kapitole 3, odoslaním obrázkov a spracovaním výsledkov je naprogramovaný v súbore `ImageAnalysis.cs`.

4.4 Azure Video Indexer

Azure AI Video Indexer je jednou zo služieb Microsoft Azure využívajúcich umelú inteligenciu k analýze obsahu. Video Indexer analyzuje video a audio obsah s pomocou 30+ modelov umelej inteligencie a má možnosť generovať pomerne rozsiahly obsah: prepisy, rozpoznávanie znakov, tváre, témy, označenia, emócie a ďalšie. Analyzovanie obsahu nazýva indexovaním a po nahratí videa buď cez online portál alebo pomocou REST API vracia JSON odpoveď s rozpoznanými detailami o videu. Každý typ analýzy navyše obsahuje aj konkrétne časové rozsahy kedy sa čo vo videu nachádza. Jednou z možností analýzy je aj identifikácia vizuálnych objektov či akcií, ktoré sa na videu nachádzajú. Práve táto služba je vhodná pre generovanie taxonómií do produktu Kontent.ai. [15]

Táto služba nemá žiadny verejný Nuget balíček, existuje iba projekt na GitHube s oficiálnym vzorovým zdrojovým kódom[16] a API dokumentácia[17]. Na posielanie jednotlivých žiadostí je preto potrebné vždy zostaviť správnu URL žiadosti s konfiguračnými údajmi a prípadnými parametrami, vytvoriť HTTP klienta, a pomocou vstavanej .NET metódy `PostAsync()` alebo `GetAsync()` asynchrónne žiadosť odoslať. Príklad odoslania GET žiadosti na index konkrétneho videa je možné vidieť v zdrojovom kóde 2.

```
1 var requestUrl = $"{VideoIndexerConsts.ApiEndpoint}/
2                    {VideoIndexerConsts.Location}/Accounts/
3                    {_configuration["ViAccountId"]}/Videos/
4                    {indexerVideoId}/Index";
5 var response = await _httpClient.GetAsync(requestUrl);
6 response.VerifyStatus(System.Net.HttpStatusCode.OK);
```

Zdrojový kód 2: Vytvorenie URL a odoslanie GET žiadosti metódou `GetAsync()`. Žiadosť je poslaná službe Video Indexer na získanie indexu konkrétneho videa. Riadok 3 overuje úspešnosť žiadosti. Trieda `VideoIndexerConsts` obsahuje konštanty s názvami potrebných služieb.

Pre prístup k službe je potrebné v prvom rade vytvoriť zdroj *Azure AI Video Indexer* v Microsoft Azure portále a tiež k nemu pripojený *Storage account*, ktorý bude slúžiť ako úložisko analyzovaných videí. Pre prístup k zdroju je najprv potrebná autorizácia. Najskôr je potrebné vygenerovať ARM (Azure Resource Manager – služba správy pre Azure) prístupový token. Ten potom slúži na autorizáciu k vygenerovaniu prístupového tokenu

[16] https://github.com/Azure-Samples/azure-video-indexer-samples/tree/master
[17] https://api-portal.videoindexer.ai/api-details#api=Operations

ku konkrétnemu účtu Video Indexera. Existuje niekoľko úrovní prístupu a token sa generuje s vybranou úrovňou typu a rozsahu povolenia. Typom povolenia môže byť *Reader* (práva len na čítanie) alebo *Contributor* (práva na úpravy). Rozsah môže byť daný na konkrétne video, projekt alebo celý účet.

Pomocou tohto druhého tokena je potom možné overenie pri posielaní API žiadostí (token označený ako *Bearer*, slovensky nositeľ, čo vyjadruje poskytnutie prístupu nositeľovi daného tokenu a je to bežná schéma overenia pomocou HTTP). Celá táto autorizačná logika je naprogramovaná v súbore `AccountTokenProvider.cs`. [15]

Keď už máme token k dispozícii, je možné prejsť na samotné indexovanie.[18] Aplikácia najskôr nahrá pomocou API žiadosti postupne všetky videá z knižnice súborov, a to kvôli paralelnému spracovaniu videí. Do žiadosti je potrebné zadať názov videa a URL s jeho zdrojom. Indexovanie začína automaticky nahratím videa do Video Indexera. V tejto chvíli začína čakanie na vykonanie analýzy. Aplikácia sa v pravidelných časových intervaloch (30 sekundové čakania) dotazuje na indexy jednotlivých videí. Video je zanalyzované, keď je stav nastavený na *Processed*. Keď je analýza hotová, žiadané výsledky sú prečítané z JSON výstupu a uložené do dátových štruktúr na ďalšie spracovanie.

4.5 ASP .NET Core MVC a Razor Pages

Webová aplikácia je napísaná v jazyku C# vo frameworku **ASP.NET Core MVC**. Ten zostavuje grafické rozhranie dynamicky na serveri s pomocou Razor Pages a generuje HTML a CSS každej stránky ako odpoveď na žiadosť prehliadača. Stránka tak prichádza klientovi pripravená na zobrazenie. Využitý je architektonický vzor **MVC (Model-View-Controller)**. Tento vzor separuje aplikáciu na tri hlavné skupiny komponent – modely, pohľady a controllery. Užívateľské žiadosti zadané do prehliadača sú smerované do controllera. Toho úlohou je spracovať žiadosť pomocou modelu aplikácie, vykonať žiadané akcie (výpočty, spracovanie výsledkov) a nakoniec vybrať stránku ktorá bude užívateľovi zobrazená a poskytuje mu dáta, ktoré si vyžiadal. Hlavnou výhodou tohto vzoru je oddelenie jednotlivých častí pre maximálnu flexibilitu v biznis modeli, ktorý sa môže časom jednoducho vyvíjať bez veľkých zmien. [16]

Razor je značkovacia syntax pre zabudovanie .NET kódu do webových stránok. Základným jazykom na písanie webových stránok je stále HTML. Pomocou symbolu @ je ale možný prechod z jazyka HTML do jazyka C#. Súbory s Razor syntaxou majú súborovú príponu `.cshtml`. Táto syntax je veľmi výhodná pre použitie v tejto aplikácii, napríklad na vykresľovanie knižnice súborov *for-cyklom* či podmieňovanie toho, čo má byť zobrazené, podmieneným príkazom *if*. Príklad ľubovoľného striedania syntaxe HTML a C# je možné vidieť v zdrojovom kóde 3.

[18] Služba má limit 10 API žiadostí za sekundu a najviac 120 žiadostí za minútu.

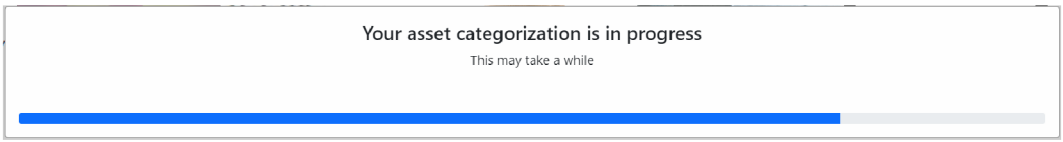
```
@if(taggedAssetLibrary.VideoTaxonomyGroup != null)
{
  <p><b>AI-Generated video tags (with percentage of videos with this
      tag)</b></p>
  <div class="paragraphs">
  @foreach (var taxonomy in taggedAssetLibrary.VideoTaxonomyGroup)
  {
   <p>@taxonomy.Name (@(Math.Round(
                      taxonomy.PercentageOfAssets * 100, 1))%)</p>
  }
  </div>
}
```

Zdrojový kód 3: Kód demonštrujúci ľubovoľný prechod medzi HTML a C# syntaxou. Zobrazuje na stránku vygenerovanú taxonómiu pre videá v knižnici.

Jediným **controllerom** v aplikácii je `HomeController` uložený v priečinku `Controllers`. Každú verejnú (`public`) metódu v ňom je možné zavolať ako koncový bod HTTP. Tieto metódy sú nazývané aj akčné metódy a obecne vracajú `IActionResult` alebo zdedenú triedu. To umožňuje metóde vrátiť `View()`, a teda že metóda využíva šablónu na vrátenie odpovedi prehliadaču. Ak nie je názov šablóny zadaný, predvolenou hodnotou je stránka s rovnakým názvom ako metóda. Prvou zobrazenou stránkou je `Index`, ktorá zobrazí úvodnú stránku aplikácie. Ostatné metódy sú postupne odpoveďami na dotazy vyvolané stlačením tlačidiel aplikácie.

Jedným z argumentov konštruktora `HomeController` je `IConfiguration`. Konfigurácia aplikácie v ASP.NET Core je vykonávaná pomocou jedného alebo viacerých poskytovateľov. Tí čítajú konfiguračné údaje z párov kľúč-hodnota pomocou rôznych zdrojov, jedným z nich je aj Azure Key Vault. V ňom sú pre túto aplikáciu uložené tajné hodnoty kľúčov ku všetkým využívaným službám, ako je napríklad Azure Computer Vision alebo Video Indexer. Azure Key Vault je podrobnejšie popísaný v časti 4.6.

Druhým argumentom konštruktora je `IHubContext<ProgressHub>`. Potrebný je pre pridanie real-time funkcionality pomocou knižnice SignalR, ktorá je použitá na informovanie používateľa o stave analýzy súborov. Stav je zobrazený ukazovateľom progresu pomocou percentuálneho posúvania čiary, viď obrázok 15. Využitý je predvolený `ProgressBar` z knižnice Bootstrap.[19]



Obr. 15: Ukazateľ progresu analýzy súborov v aplikácii

[19]https://getbootstrap.com/

Argument `IHubContext<ProgressHub>` využíva v .NET vstavaný dizajnový vzor v programovaní známy ako **Dependency Injection** (DI, voľne preložené ako vkladanie závislostí). Táto technika spôsobuje inverziu kontroly medzi triedami a ich závislosťami. Závislosť je v podstate objekt, na ktorom iný objekt závisí. Táto závislosť existuje vtedy, keď je v objekte potrebné vytvoriť inštanciu triedy závislosti. Táto inštanciácia spôsobuje niekoľko problémov a je vhodné sa jej vyhýbať – pri zmene implementácie závislosti je nutné zmeniť závislý objekt; ak má aj závislosť ďalšie závislosti, tie musia byť tiež konfigurované v závislom objekte; a ďalšie problémy. Dependency Injection práve tieto problémy adresuje: [17]

- Implementácia závislosti je abstrahovaná pomocou rozhrania. Nepoužíva sa teda priamo konkrétna trieda, ale iba rozhranie, ktoré ju implementuje. Implementácia je vďaka tomu jednoducho modifikovateľná.
- Závislosť je registrovaná v servisnom kontajneri. Servisy sa registrujú pri spustení aplikácie a môžu byť registrované s rôznou životnosťou.
- Injekcia alebo vloženie servisy do konštruktora triedy, kde bude použitá. To je práve spomenutý argument `IHubContext<ProgressHub>` triedy `HomeController`. O vytvorenie inštancie závislosti aj jej odstránenie, keď nie je potrebná, sa stará .NET.

SignalR je open-source knižnica vstavaná do ASP.NET Core, ktorá umožňuje vytváranie real-time funkcionality v aplikáciách. Pracuje veľmi dobre v prípadoch, kedy chceme prezentovať zmeny v užívateľskom rozhraní v reálnom čase . V aplikácii je využitá na priebežné posúvanie spomenutého progresu z obrázku 15.

Knižnica je rozdelená na dve časti – serverovú a klientskú. Serverová časť je priamo vstavaná do ASP.NET Core balíčkov a využíva koncept centrálneho spracovania (Hub) pre posielanie správ. Na implementáciu je potrebných len pár riadkov konfigurácie v súbore `Program.cs`, a to `AddSignalR` pre jeho pridanie do servís potrebných pri spustení aplikácie, a metóda `MapHub`, ktorá sprístupní koncový bod na prevolávanie medzi klientom a serverom. Na to je ale najskôr potrebné vytvorenie triedy `ProgressHub`, ktorá bude dediť z triedy `Hub`.

Klientská knižnica je jednoduchý stiahnuteľný JavaScriptový súbor, ktorý sa pridá do akéhokoľvek projektu. V súbore `admin.js` je potom potrebné pridať JavaScriptový kód na pripojenie sa ku koncovému uzlu a samotný kód, ktorý bude pri prijatí správy meniť percento progresu na ukazovateli priebehu (správa je nazvaná `ReceiveProgress`). Implementáciu je možné vidieť v zdrojovom kóde 4.

Na samotné zavolanie a odoslanie správy pripojenému klientovi slúži vlastnosť `Clients` triedy `Hub`. Toto volanie v tejto aplikácii prebieha počas analýzy obrázkov alebo videí v cykle, kedy je najskôr vypočítaný percentuálny progres a potom jeho odoslanie pomocou správy, ako je možné vidieť v zdrojovom kóde 5. [18]

```
connection.on("ReceiveProgress", function (progress) {
   $('#progressBar').css('width', progress + '%')
                     .attr('aria-valuenow', progress);
});
```

Zdrojový kód 4: Zmena ukazateľa progresu v užívateľskom rozhraní pri prijatí správy `ReceiveProgress` od servera

```
counter++;
float progress = ((float)(counter)/(float)(totalNumberOfAssets))
   *100;
await _progressHubContext.Clients.All.SendAsync("ReceiveProgress",
   progress);
```

Zdrojový kód 5: Vypočítanie percentuálneho progresu analýzy a odoslanie správy s týmto číslom pripojenému klientovi

Všetky súbory zdrojového kódu logicky patriace k **modelu** aplikácie sú uložené v priečinku `Models`. Patria sem najmä implementácie jednotlivých servis opísaných v predchádzajúcich častiach – súbor `ImageAnalysis.cs` pre analýzu obrázkov, `VideoAnalysis.cs` a celá zložka `VideoIndexer` s kompletnou implementáciou na analýzu videí a `ManagementApi.cs` na prevolávanie API softvéru Kontent.ai. Ďalej sú tu uložené polia reťazcov `ImageFormats` a `VideoFormats` s povolenými typmi súborov pre obrázky a videá. Poslednou časťou je zložka `DTOs`, ktorá obsahuje tzv. Data Transfer Objects (objekty na prenos údajov), čo sú v podstate štruktúry na uloženie a pre-posielanie všetkých získaných výsledkov využívané v celej aplikácii.

Medzi jednotlivými metódami akcie controllera je nutné nejakým spôsobom prenášať získané dáta. Pri každom zavolaní metódy sa totiž vytvára nový kontext (HTTP protokol je bez-stavovým protokolom), a teda uloženie dát na globálnej úrovni aplikácie (globálne premenné) neprichádza do úvahy. Informácie o stave aplikácie nie sú medzi žiadosťami zachované, každá žiadosť je nezávislá od ostatných. Hlavnými dátami, ktoré musíme prenášať, je knižnica súborov; najskôr len po načítaní z projektu, potom zanalyzovaná umelou inteligenciou pre zobrazenie na jednotlivých stránkach a export späť do projektu.

Pre MVC aplikácie sú k dispozícii rôzne techniky riadenia stavu, teda niekoľko rôznych spôsobov ako ukladať a získavať dáta medzi jednotlivými žiadosťami. Vďaka správe stavu je tiež používateľský zážitok plynulejší tým, že užívatelia môžu pokračovať kde skončili a nemusia opätovne zadávať svoje informácie. Bežné prípady použitia sú napríklad uloženie užívateľských preferencií v aplikácii, obsah nákupného košíku či prihlásenie a overenie užívateľa. Najčastejšie spôsoby na uchovávanie stavu v ASP.NET Core MVC

sú: [19]

- **cookies** – Cookie je kus dát uložený v užívateľovom počítači a pomáha ho identifikovať. Je posielaný v každej žiadosti, preto by jeho veľkosť mala byť udržiavaná na čo najmenšiu. Sú reprezentované ako dvojice kľúč-hodnota a môžu byť užívateľom vymazané. Vo všeobecnosti sú ale najtrvalejšou formou uchovávania údajov na klientovi. Používané sú najmä na personalizáciu na prispôsobenie obsahu danému užívateľovi či prístup k jeho uloženým nastaveniam.

- **session state** – Mechanizmus pre uloženie užívateľských dát na strane servera. Užívateľov prehliadač pošle serveru žiadosť s informáciami o užívateľskej relácii vždy keď navštívi webovú stránku. Server potom vytvorí novú reláciu a uloží užívateľské dáta v nej. Celá táto relácia aj s dátami je po opustení stránky odstránená. Vhodné využitie je na uloženie malého množstva dát, ktoré majú byť zachované medzi rôznymi žiadosťami od jedného používateľa – napríklad uloženie nákupného košíka či užívateľských preferencií. Má takisto podobu kľúč-hodnota. Pre ukladanie komplexnejších typov informácií pomocou session state je potrebné dáta vhodne serializovať a deserializovať.

- **skryté polia** – Uloženie dát na klientskej strane bez zobrazenia v užívateľskom rozhraní. Typické riešenie pri viac-stranových formulároch, keď potrebujeme dáta poslať na server keď užívateľ vykoná nejakú akciu. Informácie sa jednoducho uložia v skrytých poliach formulára a odoslané sú v ďalšej žiadosti.

- **vlastnosť TempData** – Uloženie dát pokiaľ ich aplikácia prečíta, je možné k ním pristúpiť v controlleroch a pohľadoch. Pomocou funkcií `Keep()` a `Peek()` je možné dáta prečítať a ponechať, a teda nebudú prečítaním vymazané. Sú výborným riešením pre uloženie dát, ktoré potrebujeme vo viacerých žiadostiach. Sú dostupné počas jednej relácie používateľa.

- **cachovanie** – Ďalší efektívny spôsob uloženia stavových dát medzi žiadosťami. Aplikácia môže kontrolovať životnosť takto uložených položiek. Vhodné použitie je na dáta, ktoré sa príliš často v aplikácii nemenia.

- **query strings** – Uloženie a prenos malých dát priamo v reťazci dotazu. Keďže sú ale tieto reťazce verejne viditeľné, sú absolútne nevhodným riešením na presun citlivých dát. Takisto môžu spôsobiť, že aplikácia bude zraniteľná na útok typu CSRF (cross-site request forgery).

V aplikácii boli využité dve zo spomenutých metód. Na uloženie ID projektu užívateľa a API kľúča, ktoré zadáva na úvodnej stránke aplikácie, bola

použitá metóda session state. Tieto údaje potrebujeme okrem počiatočného využitia na načítanie knižnice súborov uložiť na neskôr pre vytvorenie skupiny taxonómií a export zanalyzovaných dát do projektu užívateľa. Keďže nechceme tieto dáta pýtať od užívateľa znova, uložíme ich vrámci relácie na serveri a použijeme, keď budú potrebné. Pre povolenie tohto spôsobu je potrebné v súbore `Program.cs` zavolať tri konfiguračné metódy – `AddDistributedMemoryCache()` ktorá umožňuje ukladať dáta v pamäti, `AddSession()` ktorá pridá služby potrebné na vyžitie session state, a `UseSession()` ktorá povolí session state v aplikácii. Potom už je možné ukladať dáta metódou `HttpContext.Session.SetString()`, ako je možné vidieť v zdrojovom kóde 6. Prečítanie dát je zas možné metódou `HttpContext.Session.GetString()`.

```
1 HttpContext.Session.SetString("ProjectId", cl.ProjectId);
2 HttpContext.Session.SetString("ApiKey", cl.ApiKey);
```

Zdrojový kód 6: Uloženie ID projektu a API kľúča pomocou techniky riadenia stavu session state

Na uloženie a presúvanie už spomínanej knižnice dát medzi žiadosťami je vyžitá vlastnosť TempData. Ukladaná je pomocou triedy `AnalyzedAssetListModel`, čo je objekt na prenos dát, ktorý sa bude postupom v aplikácii meniť – pribudnú tam analyzované výsledky. Táto trieda obsahuje list analyzovaných výsledkov, kde sú uložené jednotlivé súbory, k nim list príslušných vygenerovaných taxonómií, typ súboru a prípadný umelou inteligenciou vygenerovaný popis (platí len pre obrázky). Ďalej je tu uložená zvlášť množina taxonómií pre obrázky a pre videá. K jednotlivým pojmom v taxonómii je vždy uložená aj miera dôvery (Confidence) a takisto je v aplikácii počítané po zanalyzovaní všetkých súborov percento súborov, ktoré sú daným pojmom označené. Celá táto štruktúra je serializovaná do JSON reťazca a ukladaná do TempData, ako je možné vidieť v zdrojovom kóde 7.

```
1 TempData["TaggedAssetLibrary"] = JsonConvert.SerializeObject(
    taggedAssetLibrary);
```

Zdrojový kód 7: Uloženie dát knižnice súborov do TempData pomocou serializácie

Pre prečítanie dát je ich vždy nutné najskôr deserializovať, viď zdrojový kód 8. Rovnakým spôsobom je možné ich čítať v controlleri i v pohľadoch. Keďže ich ale chceme zachovať, po každom prečítaní je nutné zavolať metódu `TempData.Keep()`, inak by boli po prečítaní vymazané.

```
var taggedAssetLibrary = JsonConvert.DeserializeObject
   <AnalyzedAssetListModel>((string)TempData["TaggedAssetLibrary"]);
TempData.Keep("TaggedAssetLibrary");
```

Zdrojový kód 8: Čítanie dát knižnice súborov z TempData pomocou deserializácie a ich zachovanie pomocou metódy `Keep()`

Pri scenároch v aplikáciách so vstup-výstupnými operáciami, ako je napríklad vyžadovanie dát zo siete, prístup k databáze či čítanie a zapisovanie do súborového systému, je vhodné využiť **asynchrónne programovanie**. .NET ponúka výbornú správu asynchrónnych operácií pre dobrý výkon aplikácií pomocou vzoru **async/await**. Tento vzor je používaný z viacerých dôvodov: [20]

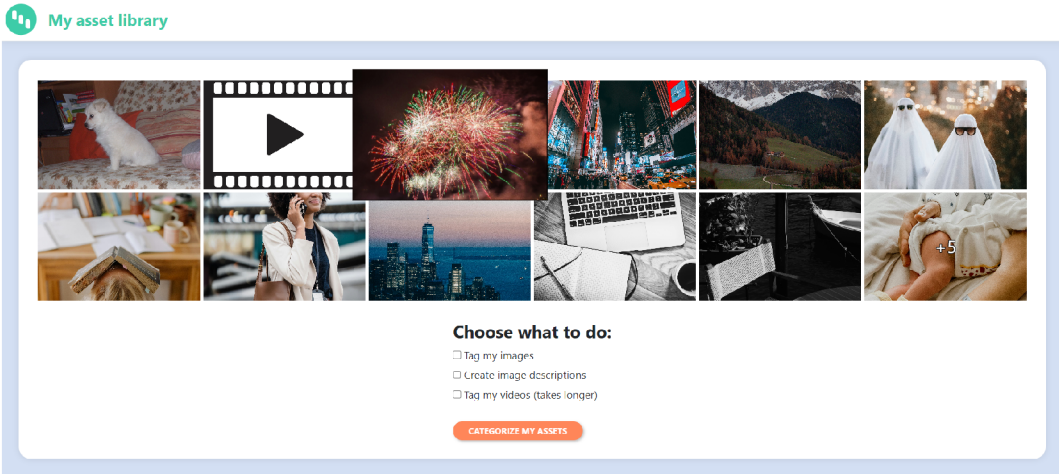
- umožňuje aby používateľské rozhranie ostalo responzívne pri vykonávaní časovo náročných operácií,
- aplikácia je škálovateľnejšia vďaka možnosti efektívneho spracovávania súbežných úloh,
- vďaka rýchlejšiemu vykonávaniu dlhotrvajúcich úloh je celková používateľská skúsenosť lepšia,
- vlákna aplikácie sú využívané efektívnejšie, čím sa redukuje spotreba zdrojov.

Základom asynchrónneho kódu sú objekty `Task` a `Task<T>` spolu s kľúčovými slovami `async` a `await`. Takýto kód môže vykonávať viacero úloh súčasne bez blokovania hlavného vlákna aplikácie. Je potrebné očakávať (`await`) operáciu, ktorá vracia objekt `Task` alebo `Task<T>`, vnútri `async` metódy. Pri použití kľúčového slova `await` v metóde je kontrola navrátená volajúcemu metódy, až dokým je asynchrónna operácia vykonaná a vďaka tomu umožňuje používateľskému rozhraniu ďalej reagovať. Keď asynchrónna operácia svoju prácu dokončí, `await` zachytí výsledok a pokračuje vo vykonávaní zvyšku metódy. Kód je veľmi podobný synchrónnemu kódu, čo zjednodušuje jeho pochopenie a údržbu. Asynchrónne metódy majú podľa konvencie v názve slovo „Async“. [21]

V tejto aplikácii je využité asynchrónne programovanie na viacerých miestach. Prvým je prevolávanie Management API Kontent.ai, kde všetky poskytnuté metódy sú asynchrónnymi, takže aj ich volanie musí prebiehať asynchrónne. Ďalším je analýza obrázkov, kde sú takisto volania API asynchrónne. Posielanie žiadostí na URL služby Video Indexer je vykonané pomocou asynchrónnych operácií HTTP klienta `PostAsync` a `GetAsync`. Každá vstup-výstupná komunikácia s externými zdrojmi je teda podľa odporúčaní vykonaná asynchrónne.

Väčšina **stránok** aplikácie využíva základné HTML komponenty na vykreslenie či vypísanie potrebných informácií na stránku a CSS na formátovanie. Využívaný je tiež nástroj **Bootstrap**, ktorý poskytuje sadu nástrojov pre tvorbu responzívneho frontendu. Základné rozloženie komponent ktoré dedia všetky stránky je v súbore `_Layout.cshtml`. K dispozícii je tiež stránka `Error.cshtml`, na ktorú je aplikácia presmerovaná v prípade chyby a chybová hláška je užívateľsky zrozumiteľne vypísaná na obrazovku. Všetky pohľady aplikácie sú uložené v priečinku `Views`. Mená sa zhodujú názvami akčných metód v controlleri.

Ďalším nástrojom, využitým pre vykresľovanie peknej galérie súborov, je **nanogallery2**.[20] Je to nástroj pre prehľadné vykreslenie galérie na stránku s možnosťou rôznych efektov, nastavení a zobrazením obrázkov na celú obrazovku v plnej kvalite. Využitá je na stránke `FetchAssetLibrary` po načítaní knižnice z užívateľovho projektu. Snímka obrazovky z aplikácie s príkladom tejto knižnice je na obrázku 16.

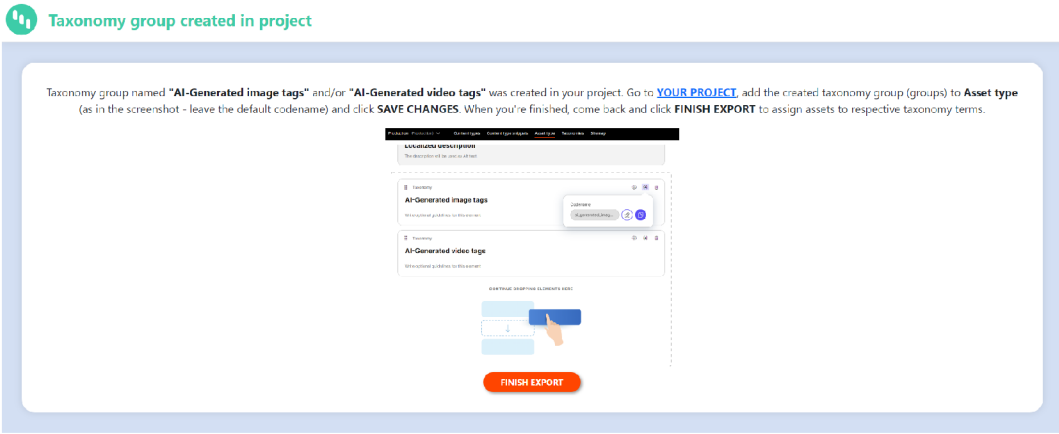


Obr. 16: Snímka obrazovky užívateľského rozhrania aplikácie zobrazujúca vykreslenie galérie súborov pomocou nástroja *nanogallery2*

Čo sa týka konečného vzhľadu užívateľského rozhrania, zväčša sa držal návrhu popísaného v časti 4.1. Jedným problémom ktorý sa ukázal až počas implementácie bol export. Vytvorenie taxonómie do projektu je možné pomocou Management API, avšak ďalší krok potrebný v exporte je nutné vykonať v projekte manuálne, keďže API neposkytuje metódu na priradenie elementu vytvorenej taxonómie do *AssetType*. Do aplikácie bola teda pridaná stránka informujúca o tom, ako tento medzikrok vykonať, aj s odkazom priamo na správne miesto v softvéri Kontent.ai. Snímku z obrazovky aplikácie je možné vidieť na obrázku 17. Po správnom vykonaní tejto akcie je možné export dokončiť a pojmy budú priradené príslušným súborom.

Celkový vzhľad aplikácie bol inšpirovaný vzhľadom softvéru Kontent.ai.

[20] https://nanogallery2.nanostudio.org/



Obr. 17: Snímka obrazovky užívateľského rozhrania aplikácie zobrazujúca medzikrok v exporte taxonómií do projektu. Vysvetľuje ako ručne priradiť vytvorenú taxonómiu do *AssetType* a nabáda na dokončenie exportu tlačidlom *FINISH EXPORT*.

Počas písania tejto práce bol však vzhľad softvéru zmenený, preto aplikácia s ním naďalej nekorešponduje.

4.6 Verziovanie a nasadenie aplikácie

Verziovanie aplikácie prebiehalo pomocou nástroja **git** a projekt je uložený v aplikácii **GitHub**.[21] Nasadenie aplikácie prebiehalo pomocou **GitHub Actions**, čo je nástroj na automatizáciu pracovného postupu pomocou CI/CD pipeline. Pri vložení (git push) novej verzie do hlavnej vetvy projektu (main) bola aplikácia zostavená a nasadená do služby Azure Web App.

Azure App Service je služba od Microsoft Azure založená na HTTP na hosťovanie webových aplikácií a API. Podporuje rôzne programovacie jazyky vrátane .NET. Aplikácie môžu bežať na windows alebo linux prostredí a poskytuje automatické škálovanie a zjednodušenie správy infraštruktúry. Ponúka tiež integráciu s ďalšími Azure službami ako databázy, monitorovanie a bezpečnosť, čo uľahčuje vytváranie komplexných a bezpečných aplikácií. Tiež je možné využiť jej rôzne DevOps schopnosti, ako je kontinuálne nasadzovanie s Azure DevOps, GitHub či Docker Hub. Platba za službu je iba za využité zdroje. [22].

Aplikácia je nasadená pod zdrojmi firmy Kontent.ai, ktorá využíva širokú škálu služieb od Azure. URL nasadenej aplikácie je

https://ai-content-classification.azurewebsites.net/.

Azure Key Vault je cloudová služba od firmy Microsoft určená na správu a ochranu kryptografických kľúčov a tajných údajov používaných aplikáciami

[21] https://github.com/lenkaklimcikova/ai-content-classification

a službami. Umožňuje bezpečné ukladanie a prístup k citlivým informáciám, ako sú rôzne heslá, certifikáty a kľúče, čím zvyšuje úroveň zabezpečenia aplikácií, znižuje riziko úniku tajných informácií a pomáha organizáciám spĺňať bezpečnostné požiadavky. [23]

Medzi ostatnými zdrojmi od firmy je tiež pre túto aplikáciu vytvorený Azure Key Vault. Uložené v ňom a čítané aplikáciou sú kľúče na prístup k Azure Image Analysis a Video Indexer a ID daného predplatného.

5 Testovanie a škálovanie aplikácie

V tejto kapitole je popísané testovanie vytvorenej aplikácie aj s popisom automatizovaného testovania, ktoré by bolo vytvorené v prípade skutočného nasadenia. Ďalej sú spomenuté možné vylepšenia v prípade ďalšieho vývoja. Tiež je tu zamyslenie, ako by bolo potrebné aplikáciu a využité služby škálovať a ako by mohla byť jej funkcionalita integrovaná priamo do softvéru Kontent.ai miesto samostatnej aplikácie.

5.1 Testovanie aplikácie

Hlavným spôsobom testovania vytvorenej aplikácie bolo manuálne testovanie. Keďže aplikácia poskytuje jeden prípad použitia, testovanie bolo priamočiare a znamenalo najmä zaistenie zamýšľanej funkcionality, teda že všetky pripojené služby odpovedia na dotazy, vygenerované pojmy je možné prehľadne prezrieť a export do projektu prebehne úspešne. Ošetriť a otestovať bolo nutné krajné prípady ako zadanie nesprávneho kľúča k projektu, neexistencia žiadneho súboru v knižnici k analýze, vytvorenie už existujúcej taxonómie či nepriradený element taxonómie do *AssetType*. Tiež sú v aplikácii ošetrované typy súborov, ktoré sú služby Microsoft Azure schopné spracovať, a tiež veľkosti súborov (napríklad povolené rozmery obrázkov). Takisto bolo nutné otestovať správne rozdiely toho, čo sa objaví na stránkach pri výbere možností analýzy od užívateľa – napríklad pri výbere len generovania tagov pre obrázky sa nevytvorí a neexportuje aj taxonómia pre videá. Ďalej bolo nutné skontrolovať, či sa exportované údaje uložili do správnych polí v projekte.

Samozrejme by v prípade skutočného nasadenia aplikácie bolo potrebné vytvoriť aj testy automatizované. To by mohlo byť v prípade pripojenia k produktom Kontent.ai vykonávané tak ako v samotnom softvéri Kontent.ai. Na testovanie používateľského rozhrania by bolo možné použiť testovacie nástroje pre webové aplikácie ako **Cypress**[22] alebo **Selenium**[23]. Tie je možné potom pripojiť do celej automatizovanej CI/CD pipeline.

Integračné testy by mohli napodobňovať scenáre a interakciu medzi jednotlivými komponentami aplikácie, ako napríklad API koncové body a prevolávanie Management API či služieb umelej inteligencie. Testovaná by nimi bola tiež celá infraštruktúra potrebná pre aplikáciu. Na automatizovanie týchto testov by mohla byť využitá **Azure Devops** pipeline, čo je nástroj pre zostavovanie, testovanie a nasadzovanie aplikácií na akúkoľvek platformu či cloud. Do pipeline sa dá priamo vybrať GitHub projekt. Konfigurácie takýchto pipeline sa píšu vo formáte YAML. Pre infraštruktúru by sa nasadzovali potrebné zdroje. Toto celé by tvorilo potrebné testovacie prostredie. Pomocou pipeline by sa tiež aplikácia po otestovaní automaticky nasadila do Azure App Service. Testy by boli udržiavané a priebežne prispôsobované

[22] https://www.cypress.io/
[23] https://www.selenium.dev/

prípadným zmenám aplikácie.

Ďalším typom možných testov by boli Unit testy, čo sú testy overujúce správnosť jednotiek zdrojového kódu, teda jednotlivých metód. Na písanie týchto druhov testov je zvyčajne použitý framework ako napríklad **xUnit**[24] spolu s knižnicou **NSubstitute**[25]. Táto knižnica umožňuje programátorom vytvárať tzv. mockové objekty (objekty slúžiace ako náhrady za triedy a rozhrania). Vďaka tomu je písanie testov zjednodušené a testovacie scenáre flexibilné.

5.2 Možné vylepšenia

Cieľom tejto práce bolo vytvoriť proof-of-concept aplikáciu, teda aplikáciu demonštrujúcu základné funkcionality vybranej služby na rozpoznávanie obrazu a jej možnosti. Ponúkajú sa však možné rozšírenia a vylepšenia, ktoré by mohli byť pre ďalší vývoj aplikácie užitočné.

Jednou z možností vylepšenia aplikácie by bolo dať užívateľovi možnosť vybrať si označenia k jednotlivým obrázkom, ktoré tam nechce, pretože sú nevhodné alebo nepravdivé. Túto možnosť už má ale priamo v systéme Kontent.ai, kde môže manuálne odstrániť buď pojem z celej skupiny taxonómií alebo ho od-značiť od konkrétneho obrázka. Čo sa týka správnosti označení, práve na tieto účely slúži dôvera (Confidence), ktorá je službou generovaná ku každému predpovedanému výstupu a v aplikácii je zobrazená. Toto skóre predstavuje percentuálnu presnosť predikcie. Bolo by možné napríklad udať minimálny prah presnosti a v prípade že ho daný pojem nepresiahne, nebude k danému obrázku priradený.

Analýza obrázkov od Microsoft Azure je veľmi odolnou službou, existujú však faktory akými sú rozlíšenie, kvalita obrázka, svetelná expozícia či kontrast, ktoré môžu presnosť výsledkov výrazne ovplyvniť. V tomto prípade sa teda môže jednoducho stať, že je prah presnosti potrebný, lebo nekvalitné obrázky môžu generovať pojmy s nízkym skóre dôvery a ktoré sú nesprávne (tzv. False positive – systém nesprávne generuje výsledok ktorý sa na obrázku nenachádza). [24]

Ďalším praktickým rozšírením by bolo, keby pri exporte vygenerovaných pojmov do aplikácie nebol nutný spomenutý medzikrok (viď obrázok 17). Na to by bolo potrebné pridať do Kontent.ai Management API metódu na priradenie elementu taxonómie do časti *AssetType*. Celý export by bolo možné vďaka tomu vykonať z užívateľského hľadiska pomocou kliknutia jedného tlačidla.

Ďalším možným rozšírením, ktorý aktuálne v aplikácii chýba, je generovanie textových popisov k videám (služba *textual video summarization*). Pre obrázky je toto implementované, avšak pre videá je potrebné pripojiť Azure Video Indexer k Azure openAI službe. Tá je však dostupná len na vyžiadanie a je potrebné prejsť procesom schválenia od Microsoft Azure.

[24] https://xunit.net/
[25] https://nsubstitute.github.io/

5.3 Škálovanie a integrácia do produktu

V prípade skutočného nasadenia aplikácie do produkcie by bolo potrebné vyriešiť škálovateľnosť služieb umelej inteligencie, ako aj samotnej aplikácie. Škálovateľnosť je v podstate schopnosť služby reagovať na vyššiu prevádzku, teda na viacero žiadostí súčasne. Každá služba Azure AI má vopred nakonfigurovanú rýchlosť (počet transakcií za sekundu), ktorá limituje, koľko súčasných volaní do tejto služby smú zákazníci vykonať v danom časovom rámci. Platí, že zákazníci zaplatia iba za úspešné prevolanie API služby. [25]

Služba Azure Cognitive Services škáluje automaticky. Existujú dve typické možnosti škálovania – vertikálne, kde sa zvyšuje napríklad veľkosť virtuálneho stroja, alebo horizontálne, kde sa zvyšuje počet súbežných strojov pracujúcich na požiadavkách. Funkcia automatického škálovania dokáže automaticky znižovať alebo zvyšovať limity rýchlosti zdrojov zákazníka, a to na základe metrík využívania zdrojov v reálnom čase a metrík kapacity koncových služieb. Táto funkcia je pre každý nový zdroj predvolene vypnutá. Pomocou Azure portálu alebo Azure klienta je možné túto funkciu pre žiadaný zdroj zapnúť. [26]

Škálovanie samotnej aplikácie nasadenej v Azure App Service je možné dvomi spôsobmi. Jedným zo spôsobov je *Scale up* – získanie viac CPU, pamäti, diskového priestoru či virtuálnych strojov. Toto je možné zabezpečiť zmenou cenového plánu danej App Service. Zmena trvá v jednotkách sekúnd a nie je potrebné meniť kód či odznova aplikáciu nasadiť. Druhým spôsobom je *Scale out* – zvýšenie počtu inštancií virtuálnych strojov, na ktorých aplikácia beží. Je tiež k dispozícii automatické škálovanie počtu inštancií pomocou vopred definovaných pravidiel. [22]

Zároveň je potrebné zamyslieť sa nad časom trvania analýzy. Pri veľkom množstve súborov môže totiž celá analýza trvať pomerne dlhý čas. Tento čas môže byť ovplyvnený rôznymi faktormi, ako je veľkosť a rozlíšenie obrázka, typ analýzy, dĺžka a kvalita videa, oneskorenie siete či zaťaženie služby. Na základe skúseností užívateľov zdieľaných vo fórach (ako napríklad Microsoft Q&A) i vlastným testovaním je zvyčajne proces označenia obrázkov rýchly, v jednotkách milisekúnd až maximálne pár sekúnd. Čo sa týka videa, analýza trvá oveľa dlhšie, môže trvať približne 1-2 násobok dĺžky videa.

Práve pri spomínaných videách by bolo možné spraviť v aplikácii jedno vylepšenie, ktoré by ale potrebovalo viac premyslenia pri skutočnom nasadení. Týka sa duplikátov videí, teda keď rovnaké video chceme poslať na analýzu viackrát. Aktuálna implementácia duplikáty povoľuje (parameter `preventDuplicates=false` v žiadosti pri nahratí videa), čo znamená, že rovnaké video viackrát poslané na analýzu bude indexované vždy odznova. To je však zbytočne časovo neefektívne. Riešením by bolo vždy najskôr overiť, či už dané video nebolo indexované, a ak áno, iba zažiadať o zanalyzované informácie. Služba vie síce rozpoznať, či video už bolo indexované a ak áno, vráti konflikt. Problémom je API, ktorá pri písaní tejto práce neobsahuje metódu na získanie videa podľa identifikátora a ani spôsob, ako vlastné

identifikátory videám prideľovať, a tiež by sme si ich museli nejakým spôsobom pamätať. Je možné získať všetky videá v danom účte služby Video Indexer a porovnávať napríklad názvy, tie sa však môžu meniť a tiež by tu ostávala otázka bezpečnosti.

Ďalším problémom pri videách (v nasadenej verzii aplikácie v Azure App Service) je vypršanie časového limitu pri príliš dlhých žiadostiach, čo sa aktuálne deje pri nahraní dlhších videí do služby Video Indexer (alebo ak je služba príliš zaťažená). Azure Load Balancer má predvolený časový limit 4 minúty, po ktorých ak služba neodpovie na žiadosť, aplikácia vráti klientovi chybu.[26] Tento problém by bolo možné vyriešiť spracovaním na pozadí pomocou Azure Webjobs, to je však funkcionalita nad rámec tejto práce.

Táto aplikácia mala slúžiť najmä ako prototyp demonštrujúci využitie služieb umelej inteligencie. Reálnym využitím pre firmu Kontent.ai pravdepodobne nebude táto aplikácia ako celok, ale integrácia daných služieb priamo do softvéru Kontent.ai. V tom prípade by bolo potrebné premyslieť, akým spôsobom by sa analýza volala a aké prípady použitia by boli možné. Prvou otázkou by bolo, ako zaistiť možnosť analýzy všetkých súborov aktuálne prítomných v knižnici a ako ju zaistiť pre každý nový pridaný súbor.

V prípade analýzy celej knižnice by sa mohlo nachádzať priamo v knižnici tlačidlo, po ktorého kliknutí by bola prejdená celá knižnica a všetky súbory analyzované. Mohlo by ale tiež byť zobrazené vyskakovacie okno, v ktorom by si užívateľ najskôr vybral, tak ako v tomto prototype, čo všetko chce s knižnicou vykonať. Výsledky by následne neboli nikde vypísané tak, ako v prototype, ale priamo exportované do taxonómií a automaticky priradené jednotlivým súborom. Táto možnosť by nemusela byť priamo v knižnici, ale napríklad v nastaveniach projektu, pretože nie pre každého by bola zaujímavá a tiež nie je potrebné ju mať k dispozícií pri každodennej práci používateľa. Spustenie analýzy by bolo asynchrónne, aby užívateľ nemusel čakať v prípade veľkých knižníc na dokončenie aj desiatky minút, ale mohol pokračovať v práci a po dokončení analýzy by bol informovaný. Tiež by bolo možné predom informovať zákazníka o časovom odhade trvania analýzy (podľa počtu a typu súborov) a tiež cenovom odhade, pretože za túto službu navyše by bol samozrejme účtovaný.

Funkcionalitu označovania súborov by bolo možné pridať aj vždy pri vložení nového súboru do knižnice. Možnosť ju zapnúť by mohla byť v nastaveniach projektu. Potom by sa automaticky pri každom novom súbore zavolala správna služba (podľa typu súboru buď Computer Vision alebo Video Indexer) a spolu s uložením súboru do knižnice by boli pridané pojmy do taxonómie (s kontrolou duplikátov) a daný súbor nimi označený. V prípade obrázkov by to bolo rýchle, v prípade videí by asi tiež bolo potrebné asynchrónne spracovanie, aby používateľ zbytočne dlho nečakal.

[26] https://learn.microsoft.com/en-us/troubleshoot/azure/app-service/web-apps-performance-faqs#why-does-my-request-time-out-after-230-seconds

Zaujímavou otázkou by bolo, ako pracovať v taxonómiách so synonymami slov. V niektorých prípadoch môžu byť viaceré slová rovnakého významu zbytočné (napríklad ak by sme vygenerované pojmy chceli použiť niekde ďalej a nie je ich potrebných zbytočne veľa), v inom prípade by sa zas viac synoným mohlo hodiť (napríklad vyhľadávanie v knižnici ručným zadaním slova by malo väčšiu šancu úspechu). V oboch prípadoch by ako riešenie mohlo prísť vhod využitie API synonymických slovníkov, ako napríklad *Thesaurus API* od *API Ninjas*.[27] Každé slovo taxonómie by mohlo byť pomocou API vyhľadané a navrátené by boli jeho synonymá. Tie by sme mohli buď z taxonómie odstrániť (ak sa tam nachádzajú) a ponechať len pôvodné slovo, alebo naopak nejaký počet z nich pridať. Záležalo by na žiadanom použití.

V Kontent.ai existuje aj hierarchická štruktúra taxonómií. V praxi to znamená, že je možné vytvoriť napríklad pojem zviera a pod neho uložiť pod-taxonómie konkrétnych zvierat (mačka, pes, zajac). V oblasti umelej inteligencie je to v podstate problém klasifikácie slov do kategórií, ktoré by bolo možné pred-definovať (učenie s učiteľom) alebo automaticky rozoznať (učenie bez učiteľa). Kategórie v rozpoznávaní obrazu má k dispozícií priamo aj poskytovateľ Amazon Web Services. Ten ku každému pojmu vo výsledku vracia aj rodičovské pojmy a možné kategórie, ktoré by mohli slúžiť práve na účel hierarchickej štruktúry taxonómií. Služba Azure Cognitive Services výsledky v takomto formáte negeneruje.

[27] https://api-ninjas.com/api/thesaurus

Záver

Jedným z cieľov tejto diplomovej práce bolo vykonať prieskum existujúcich technológií rozpoznávania obsahu pomocou umelej inteligencie. Nájdení poskytovatelia boli porovnaní na základe vlastností, ktoré vedia s obsahom vykonávať (označovanie či tvorba popisu), podľa typov súborov (obrázky, videá, text) a ceny za služby (počítaná na jeden milión žiadostí). Vybraní poskytovatelia boli otestovaní na ich skúšobných verziách a nájdené boli obecné vlastnosti, akými pristupujú k označovaniu obrázkov. Podľa toho bol následne vybraný najvhodnejší poskytovateľ pre použitie v systéme Kontent.ai, kde bude služba slúžiť na označovanie veľkej knižnice súborov. Na základe kritérií bola vybraná služba Azure Cognitive Services, pričom výber podporil aj fakt, že firma využíva veľké množstvo služieb od Microsoft Azure.

Druhým a hlavným cieľom bol návrh a implementácia prototypu aplikácie, ktorá by načítala používateľskú knižnicu súborov a na ktorých vykonala požadovanú analýzu pomocou vybraného poskytovateľa. Primárne zameranie bolo na označovanie a generovanie textového popisu obrázkov pomocou Azure Computer Vision. Implementácia bola úspešne zrealizovaná pre základné formáty a prípustné rozmery obrázkov. Aplikácia bola následne rozšírená aj o podporu videí a označovanie ich obsahu pomocou Azure Video Indexer. Aplikácia bola implementovaná vo frameworku .NET Core podľa architektúry MVC s využitím Razor Pages. Komunikácia so softvérom Kontent.ai je realizovaná pomocou Management API a po vykonanej analýze je možné všetky výsledky exportovať do projektu používateľa. Tam budú využívané pri každodennej práci zákazníkov na filtrovanie či vyhľadávanie v knižnici.

Aplikácia je úspešne nasadená v Azure App Service s využitím Azure Key Vault pre správu tajných údajov. Bola precízne manuálne otestovaná a schválená konzultantom z firmy Kontent.ai. V poslednej časti práce sú taktiež popísané ďalšie možné vylepšenia aplikácie, možnosti škálovania v prípade skutočného nasadenia do produkcie a spôsoby, akými by mohla byť funkcionalita integrovaná priamo do produktu.

Conclusions

One of the objectives of this diploma thesis was to conduct a research of existing image recognition technologies using artificial intelligence. The providers found were compared to each other based on the properties they can perform with the content (tagging and caption generation), according to file types (images, videos or text files) and price for the service (counted on one million requests). Chosen providers were tested on their trial versions and general characteristics for each service of how they approach image tagging were found. Afterwards, the most suitable provider was selected for use in the Kontent.ai system, where the service will be used to tag a large asset library. Based on the criteria, the selected service was Azure Cognitive Services, while the selection was also supported by the fact that the company uses big amount of Microsoft Azure services.

The second and main objective was to design and implement a prototype application that would load the user's asset library and perform the required analysis on it using the selected provider. The primary focus was on image tagging and generating text descriptions using Azure Computer Vision. The implementation was succesfully carried out for main image formats and acceptable image dimensions. The application was subsequently extended to support videos and label their content using Azure Video Indexer. The application was implemented in .NET Core framework following the MVC architecture with Razor Pages. Communication with the Kontent.ai system was carried out by using Management API, and after the analysis, all results can be exported to the user's project. There it will be used in customer's everyday work for asset library filtering or search.

Application is succesfully deployed in Azure App Service with the use of Azure Key Vault for managing secret data. It was precisely manually tested and approved by a Kontent.ai consultant. The final section of the thesis also describes further possible application improvements, options for scaling in the case of actual production deployment, and ways in which the functionality could be integrated directly into the product.

A Obsah elektronických dat

`text/`

Adresár s textom práce vo formáte PDF, vytvorený s použitím záväzného štýlu KI PřF UP v Olomouci pre záverečné práce, včetne všetkých (textových) príloh, a všetky súbory potrebné pre bezproblémové vytvorenie PDF dokumentu textu (v ZIP archíve).

`README.txt`

Textový súbor s informáciami o spôsobe použitia ostatných dát práce.

`source/`

Adresár s ostatnými autorskými dátami práce (v ZIP archíve) – kompletné zdrojové kódy software.

Literatúra

[1] *Kontent.ai*. [online]. [cit. 2024-4-3]. Dostupné z www: ⟨https://kontent.ai⟩.

[2] *Kontent.ai Learn - Kontent.ai Documentation*. [online]. [cit. 2024-4-5]. Dostupné z www: ⟨https://kontent.ai/learn/docs/overview⟩.

[3] Writer, AIT Staff. *Top AI Software Companies for Image Recognition: From Pixels to Insights: Unveiling the Titans of AI Image Recognition* [online]. [cit. 2024-3-25]. Dostupné z www: ⟨https://aithority.com/vision/top-ai-software-companies-for-image-recognition/⟩.

[4] *Google Cloud Vision AI*. [online]. [cit. 2024-3-25]. Dostupné z www: ⟨https://cloud.google.com/vision?hl=en#section-6⟩.

[5] *Google Cloud Vision AI Prices*. [online]. [cit. 2024-3-25]. Dostupné z www: ⟨https://cloud.google.com/vision/pricing/#prices⟩.

[6] *Amazon Rekognition*. [online]. [cit. 2024-3-25]. Dostupné z www: ⟨https://aws.amazon.com/rekognition/⟩.

[7] *Amazon Rekognition pricing*. [online]. [cit. 2024-3-25]. Dostupné z www: ⟨https://aws.amazon.com/rekognition/pricing/⟩.

[8] *Azure AI Vision Studio*. [online]. [cit. 2024-3-25]. Dostupné z www: ⟨https://portal.vision.cognitive.azure.com/gallery/featured⟩.

[9] *Azure AI services pricing*. [online]. [cit. 2024-3-25]. Dostupné z www: ⟨https://azure.microsoft.com/en-us/pricing/details/cognitive-services/#pricing⟩.

[10] *Azure AI Vision*. [online]. [cit. 2024-3-25]. Dostupné z www: ⟨https://azure.microsoft.com/en-us/products/ai-services/ai-vision⟩.

[11] *Imagga API*. [online]. [cit. 2024-4-1]. Dostupné z www: ⟨https://docs.imagga.com/#introduction⟩.

[12] *Clarifai*. [online]. [cit. 2024-4-1]. Dostupné z www: ⟨https://www.clarifai.com/computer-vision⟩.

[13] Kontent.ai. *Management API v2* [online]. [cit. 2024-6-29]. Dostupné z www: ⟨https://kontent.ai/learn/docs/apis/openapi/management-api-v2/⟩.

[14] Microsoft. *Azure Image Analysis client library for .NET* [online]. [cit. 2024-6-30]. Dostupné z www: ⟨https://www.nuget.org/packages/Azure.AI.Vision.ImageAnalysis/1.0.0-beta.2⟩.

[15] Microsoft. *Learn about Azure AI Video Indexer* [online]. [cit. 2024-7-3]. Dostupné z www: ⟨https://learn.microsoft.com/en-us/azure/azure-video-indexer/⟩.

[16] Microsoft. *ASP.NET documentation* [online]. [cit. 2024-6-30]. Dostupné z www: ⟨https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0⟩.

[17] Microsoft. *.NET dependency injection* [online]. [cit. 2024-7-2]. Dostupné z www: ⟨https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection⟩.

[18] Appel, Rachel. *Create Real-time Applications with ASP.NET Core SignalR – Using Rider* [online]. 2020 [cit. 2024-7-3]. Dostupné z www: ⟨https://blog.jetbrains.com/dotnet/2020/05/19/create-real-time-applications-asp-net-core-signalr-using-rider/⟩.

[19] Kanjilal, Joydip. *6 state management techniques for ASP.NET Core MVC* [online]. [cit. 2024-7-3]. Dostupné z www: ⟨https://www.infoworld.com/article/3680070/6-state-management-techniques-for-aspnet-core-mvc.html⟩.

[20] Sahay, Rahul. *Best Practices for Using Async/Await in C# with .NET Core* [online]. [cit. 2024-7-13]. Dostupné z www: ⟨https://blog.stackademic.com/best-practices-for-using-async-await-in-c-with-net-core-b067ea3fa9d3⟩.

[21] Microsoft. *Asynchronous programming scenarios* [online]. [cit. 2024-7-13]. Dostupné z www: ⟨https://learn.microsoft.com/en-us/dotnet/csharp/asynchronous-programming/async-scenarios⟩.

[22] Microsoft. *App Service documentation* [online]. [cit. 2024-7-7]. Dostupné z www: ⟨https://learn.microsoft.com/sk-sk/azure/app-service/⟩.

[23] Microsoft. *Azure Key Vault basic concepts* [online]. [cit. 2024-7-13]. Dostupné z www: ⟨https://learn.microsoft.com/en-us/azure/key-vault/general/basic-concepts⟩.

[24] Microsoft. *Characteristics and limitations for using Image Analysis* [online]. [cit. 2024-6-30]. Dostupné z www: ⟨https://learn.microsoft.com/en-us/legal/cognitive-services/computer-vision/imageanalysis-characteristics-and-limitations⟩.

[25] Microsoft. *Autoscale AI services limits* [online]. [cit. 2024-7-16]. Dostupné z www: ⟨https://learn.microsoft.com/en-us/azure/ai-services/autoscale?tabs=portal⟩.

[26] Boyd, Amy. *5 Ways Azure Cognitive Services Scale* [online]. [cit. 2024-7-16]. Dostupné z www: ⟨https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/5-ways-azure-cognitive-services-scale/ba-p/2664021⟩.