FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	CzechCaptcha – modul importu dat
Student:	Jakub Bůlfinek
Vedoucí:	Ing. Marek Sušický
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Pro nástroj CzechCaptcha implementujte modul, který umožní uživatelsky příjemné vkládání nových obrazových datasetů do aplikace. Uživatel zvolí zdroj dat, aplikace provede detekci objektů, obrazy rozřeže a vybere ty se zajímavým obsahem. Pokud bude dostupná částečná anotace dat ve vhodném formátu, bude jí možné využít. Následně proběhne konfigurace pro nástroj CzechCaptcha a nasazení datasetu k anotaci.
Práce naváže na existující diplomovou práci [1]. Proveďte analýzu stávajících object detection algoritmů a vyberte nejvhodnější pro dané užití. Vyberte vhodný anotační nástroj, který bude podporován a implementujte pipeline dle popisu výše.

[1] - Bc. Otakar Vinklář - Czech Captcha, 2022; https://github.com/opendatalabcz/czech-captcha

Elektronicky schválil/a Ing. Magda Friedjungová, Ph.D. dne 21. května 2022 v Praze.

ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F8

Fakulta informačních technologií
Katedra aplikované matematiky

Bakalářská práce

CzechCaptcha – modul importu dat

OpenDataLab

Jakub Bůlfinek
Obor Znalostní inženýrství

Leden 2023
Vedoucí práce: Ing. Marek Sušický

Poděkování / Prohlášení

V prvé řadě bych chtěl poděkovat Ing. Marku Sušickému za vedení této práce a motivaci při jejím zpracování. Svým nejbližším pak děkuji za podporu, kterou mi věnovali po celou dobu studia.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 5. ledna 2022

.....

Abstrakt / Abstract

Open-source aplikace CzechCaptcha je CAPTCHA systém, který lze použít jednak k obraně proti nežádoucí činnosti automatizovaných robotů na internetu, a jednak k anotování dat lidmi pomocí řešení CAPTCHA úloh.

V rámci této práce byl do aplikace CzechCaptcha přidán modul pro import dat obsahující model pro detekci objektů v obrázku EfficientDet-V4 z frameworku KotlinDL natrénovaný na datasetu COCO. Při nahrávání obrázku do aplikace lze určit, které objekty v něm mají být detekovány. V případě, že model jejich detekci podporuje, jsou na základě jeho výstupů do lokálního úložiště uloženy výřezy obrázku určené pro použití ve výběrové obrázkové CAPTCHA úloze. V opačném případě je původní obrázek použit v nové CAPTCHA úloze pro detekci objektů, v níž testovaný uživatel pomocí obdélníků označuje požadované objekty. Po získání několika odpovědí, jsou vypočítány pravděpodobné pozice objektů. Anotace s již známými pozicemi lze nahrát společně s obrázkem pomocí nového webového rozhraní.

Klíčová slova: CAPTCHA, crowdsourcing, označování dat, detekce objektů, frameworky pro detekci objektů, formáty anotací pro detekci objektů, webová aplikace, open-source

Open-source application CzechCaptcha is a CAPTCHA system that can be used both to defend against unwanted activities of automated robots on the Internet and to annotate data by humans by solving CAPTCHA tasks.

In this work, a data import module was added to the CzechCaptcha application. It contains the EfficientDet-V4 object detection model from the KotlinDL framework trained on the COCO dataset. When uploading an image to the application, it is possible to specify which objects in the image should be detected. If the model supports their detection, image cut-outs based on its outputs are saved to the local storage for later use in the image selection CAPTCHA task. Otherwise, the original image is used in a new CAPTCHA task for object detection, in which a tested user uses rectangles to localize the desired objects. After obtaining several answers, the probable positions of the objects are calculated. Annotations with already known positions can be uploaded together with the image using a new web user interface.

Keywords: CAPTCHA, crowdsourcing, data labeling, object detection, object detection frameworks, annotation formats for object detection, web application, open-source

Title translation: CzechCaptcha – data import module (OpenDataLab)

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Dle některých odhadů je v současnosti více než třetina provozu na internetu tvořena roboty [1]. Většina této automatizované činnosti se neděje s dobrým úmyslem. Jedná se o útoky hrubou silou (např. DoS – Denial of Service), pokusy o nelegální sběr dat, generování spamu a další.

Jednou z možností, jak tomuto problému čelit, je využití ověřovacího systému CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart). Ten povolí přístup ke zdrojům či operacím až poté, co uživatel splní nějaký úkol. Koncepce úkolu by měla být taková, aby její vyřešení bylo relativně jednoduché pro člověka, ale velmi obtížné pro současné počítačové programy. Nalezení takové úlohy je s rychlým vývojem v oblasti strojového a hlubokého učení stále obtížnější.

CAPTCHA systémy je dále možné využívat jako platformu pro dělbu práce mezi velké množství lidí, tzv. crowdsourcing. Toho je docíleno pomocí koncipování úloh tak, aby jejich výsledky bylo možné aplikovat při řešení dalších problémů. V praxi se výstupy úkolů využívají většinou k anotování nestrukturovaných dat, neboli přidávání značek například k obrázkům. Tento proces hraje zásadní roli při přípravě trénovacích dat pro modely hlubokého učení.

Příkladem tohoto využití může být aplikace reCAPTCHA od společnosti Google. Ta v minulosti požadovala od uživatelů zadání slov ze strojově nečitelném skenu textu, čímž pomohla nejen digitalizovat archiv deníku The New York Times [2], ale také natrénovat algoritmy rozpoznávající text z obrázků. Později, po zdokonalení této technologie, se místo přepisování textů začala po uživatelích vyžadovat kategorizace obrázků. Ta opět pomáhá při tvorbě trénovacích dat pro modely strojového vidění.

Stejný princip se využívá ve službě CzechCaptcha, open-source CAPTCHA systému, který dostal název podle země původu hlavních vývojářů. Aplikace je dostupná jak pro majitele webových stránek a aplikací, kteří chtějí zamezit automatizovanému přístupu, tak i pro vlastníky dat, jež je potřeba anotovat lidmi, například právě pro účely strojového učení.

Aplikace CzechCaptcha je předmětem této bakalářské práce navazující na diplomovou práci Ing. Otakara Vinkláře [3], v rámci níž byla vytvořena první verze. Její serverová část naprogramovaná v jazyce Kotlin umožňuje konfiguraci, vytváření a vyhodnocování CAPTCHA úloh. Uživatelé mohou do aplikace nahrát svá data (např. obrázky) a využívat je v úlohách. K dispozici je také vestavěný mechanismus pro anotování dat řešiteli CAPTCHA úloh. Webová klientská část naprogramovaná v jazyce JavaScript komunikuje se serverovou pomocí REST rozhraní a poskytuje velmi základní uživatelské rozhraní.

Tato práce se zaměřuje na vylepšení modulu pro nahrávání dat do aplikace a předzpracovaní obrázků pomocí algoritmu pro detekci objektů. Modul umožňuje nový způsob ukládání datových souborů, včetně čistšího uživatelského rozhraní.

Text je rozdělen do celkem sedmi kapitol. Ve třetí kapitole jsou shrnuty informace o CAPTCHA systémech, detekci objektů, anotačních formátech a hierarchickém shlukování. Čtvrtá kapitola se zabývá analýzou aplikace CzechCaptcha v době počátku této

práce. Předmětem páté kapitoly je design a implementace přidávaného modulu importu dat. Šestá kapitola shrnuje testování nového modulu.

Kapitola 2
Cíle práce

Cílem teoretické části práce je seznámit čtenáře se systémem CAPTCHA, jeho vlastnostmi, různými variantami a vztahem k umělé inteligenci. Dalším dílčím cílem je prozkoumat tematiku detekce objektů, včetně používaných metrik, aktuálních modelů a konkrétních dostupných frameworků. Posledním dílčím cílem je představit současné anotační formáty používané v kontextu strojového vidění.

V praktické části práce je cílem na základě výstupů z teoretické části navrhnout a implementovat modul pro nahrávání dat do aplikace CzechCaptcha, open-source verze CAPTCHA systému. Modul bude umožňovat nahrát data ze zadaného zdroje a v případě, že se jedná o obrázky, je i následně předzpracovat pomocí algoritmu pro detekci objektů. Do systému pak budou uloženy části obrázků obsahující objekty vhodné k anotaci lidmi. V případě poskytnutí již existujících anotací k obrázkům během nahrávání, modul je také uloží a zohlední. Následně budou nahraná data nasazena do aplikace k označení uživateli. Druhým dílčím cílem praktické části je vytvořit uživatelsky přívětivé webové rozhraní pro nahrávání dat do systému.

Výsledky bakalářské práce budou přínosné pro budoucí uživatele veřejně přístupné aplikace CzechCaptcha, kteří budou nahrávat nová data k anotování.

Kapitola 3
Úvod do problematiky

Tato kapitola obsahuje informace potřebné jednak pro obecné seznámení se s tematikou a jednak pro učinění rozhodnutí během samotného návrhu a implementace modulu importu dat pro aplikaci CzechCaptcha. Je rozdělena do dvou podkapitol, přičemž první se týká historie, typů a využití CAPTCHA úloh a druhá se zaobírá problematikou detekce objektů v obrázcích.

3.1 CAPTCHA

Slovo CAPTCHA poprvé použili Luis von Ahn, Manuel Blum, Nicholas Hopper a John Langford z Univerzity Carnegieho–Mellonových v Pensylvánii v roce 2000. [4] Je tvořeno počátečními písmeny termínu *Completely Automated Public Turing test to tell Computers and Humans Apart*, neboli kompletně automatizovaný veřejný Turingův test k odlišení počítačů a lidí. Vzhledem k popularitě textových CAPTCHA v minulosti, některé zdroje uvádí ještě jednu interpretaci tohoto slova – *CAPTure CHAracters*. [5]

Předmětem originálního Turingova testu představeného v roce 1950 matematikem a informatikem Alanem Mathisonem Turingem je schopnost člověka odlišit, zda pomocí textového kanálu (např. klávesnice a obrazovky) komunikuje s jiným člověkem, či počítačem. [6] Naproti tomu v případě CAPTCHA testu, jak naznačují první dvě písmena akronymu, je do role toho, kdo rozlišuje lidské či strojové chování, postaven počítač. Třetí slovo, Public, pak vyjadřuje, že by potřebná data, postup tvorby i vyhodnocení CAPTCHA úloh měly být veřejně dostupné, až na prvek náhody aplikovaný během procesu generování. Je to záruka bezpečnosti, neboť ani když CAPTCHA systém přesně známe, nejsme schopni testy strojově řešit s velkou úspěšností. [7] Tuto vlastnost ovšem většina moderních CAPTCHA systémů nemá.

Formální definice CAPTCHA testu je uvedena v článku *CAPTCHA: Using Hard AI Problems For Security*. [8] Pro potřeby této práce však není nutné ji uvádět. Stačí říci, že CAPTCHA je program, který umí generovat a vyhodnocovat testy, jež většina lidí umí jednoduše vyřešit, ale současný software to umí pouze s nízkou úspěšností nebo to vůbec neumí. [7] Samozřejmě tyto vlastnosti testů není možné formálně dokázat.

3.1.1 Vznik a využití CAPTCHA úloh

Důvod vzniku CAPTCHA systémů je prostý. S postupně se rozšiřujícím používáním internetu širokou veřejností, vznikaly webové aplikace a platformy, kde uživatelé mohli sami tvořit obsah a komunikovat s ostatními. Ovšem toto lidské chování je možné napodobit pomocí programů s řádově vyšší rychlostí a většinou se tak děje z nekalých důvodů. Takto automaticky generovaný obsah může snižovat užitečnost zveřejněných informací, či přímo ohrozit fungování aplikací. Příkladem možných problémů je zahlcení služby velkým množstvím požadavků (DoS útoky), slovníkový útok na uživatelská hesla a následný neoprávněný přístup k datům, či šíření nevyžádaných informací, např. formou emailů nebo příspěvků na internetových fórech a blozích.

Řešením je umožnění přístupu k některých datům či funkcím až po prokázání, že uživatel je člověk a ne počítač právě pomocí splnění nějaké CAPTCHA úlohy. Dalšími možnostmi je kupříkladu omezení počtu požadavků za nějaký časový úsek nebo kontrola a případná restrikce IP adres požadavků, ovšem tyto alternativy nemusí být vždy vhodné či dostačující. [7]

3.1.2 Bezpečnost CAPTCHA úloh

CAPTCHA testy jsou většinou postaveny na základě aktuálních problémů umělé inteligence (AI – Artificial Intelligence), které nejsme v současné době schopni efektivně řešit pomocí programu. Teoreticky tento fakt má výhodu v tom, že motivuje nejen vědce, ale i další programátory k pokroku v oblasti AI. Prolomení CAPTCHA úlohy nutně znamená posunutí hranice schopností softwaru (a samozřejmě také nutnost vymyslet nový typ úlohy). Naopak neschopnost test strojově vyřešit znamená, že existuje způsob, jak odlišit lidi a počítačů. Obě situace jsou tedy pozitivní. [8] V minulosti již situace, kdy vývoj v oblasti umělé inteligence umožnil CAPTCHA testy řešit automaticky, nastaly. Je ovšem prakticky nemožné určit, jakou zásluhu na tom měly právě samotné CAPTCHA úlohy.

Příkladem pokořené problematiky je rozpoznávání textu z obrázku (OCR – Optical Character Recognition), které bylo a stále je velmi využívané k odlišení lidí od počítačů. Technologie a algoritmy se zlepšovali, na což se reagovalo ztěžováním CAPTCHA úloh větším zkreslením textu. V roce 2014 vědci ze společnosti Google zjistili, že i přes vyšší obtížnost, má jejich konvoluční neuronová síť srovnatelnou, a v některých případech i vyšší, úspěšnost vyplňování textových CAPTCHA testů než lidé. Díky tomuto algoritmu byl teoreticky vyřešen problém rozpoznávání krátkých textů z obrázku. [9]

Ovšem nejen textové CAPTCHA úlohy byly překonány. Mnoho implementací i jiných typů testů se podařilo prolomit s vysokou mírou úspěšnosti během několika let, což je vidět na obrázku 3.1.

CAPTCHA implementace by měly být bezpečné na dvou hlavních úrovních. První z nich je samotná podstata úlohy. Ta by měla být založena na nevyřešené problematice umělé inteligence. Generování by mělo probíhat s použitím co nejvíce náhodně volených parametrů, aby útočník musel přijít s obecným řešením. Šance na prolomení úlohy by také neměla výrazně stoupat s rostoucím počtem uživatelů a ani při případném masovém rozšíření. Druhou úrovní jsou pak implementační detaily. Je třeba zajistit, aby úloha nešla programově obejít a její předání i vyhodnocení bylo bezpečné. V případě existence neveřejných informací a dat nepostradatelných pro fungování CAPTCHA úlohy, je nutné zachovat jejich důvěrnost. Pokud se používá obsah negenerovaný automaticky (např. fotografie, audio nahrávky, anotace), je nezbytné mít velké množství těchto dat, aby byla zaručena dostatečná různorodost zadání. [4]

Ovšem ani při dodržení všech zmíněných požadavků, není zaručené, že uživatel, který prošel CAPTCHA testem není program. Existují totiž placené služby přeposílající CAPTCHA úlohy k vyřešení jinému člověku. Často se jedná o lidi z rozvojových zemí takto pracující za velmi nízký plat (v porovnání s vyspělými státy). Cena za vyřešení 1000 CAPTCHA úloh se odvíjí od jejich typu a pohybuje se zhruba od 0,5 USD do 5 USD. Příkladem těchto služeb je 2Captcha, Anti Captcha nebo EndCaptcha. [10–12]

Může tedy vyvstat otázka, zda CAPTCHA systémy mají vůbec smysl, když je lze prolomit, či obejít. Odpověď zní, že mají. CAPTCHA je stále efektivní ne proto, že ji nelze automaticky vyřešit, ale proto, že útok na ni je složitý natolik, že se v některých případech ekonomicky nevyplatí.

Obrázek 3.1. Graf s typy CAPTCHA úloh a názvy jejich konkrétních implementací. Na časové ose je vyznačeno, jak dlouho trvalo, než byla daná implementace prolomena. Procenta v červených pruzích určují nejlepší úspěšnost útoků. [13]

3.1.3 Problémy s CAPTCHA úlohami

Největší nevýhodou CAPTCHA systémů je zhoršení uživatelského zážitku nuceným plněním úloh. S rostoucí komplexností testu se zvyšuje bezpečnost, ale také frustrace, neboť schopnost lidí projít testem klesá. Je vždy tedy nutné tyto dvě protichůdné vlastnosti vyvažovat.

Obtížnost vyplnění CAPTCHA testu pro některé, obzvláště starší, uživatele dokazuje i vznik české služby CAPTCHA Help od společnosti CZ.NIC, která pomocí rozšíření ve webovém prohlížeči Google Chrome umožňuje zachytit textovou CAPTCHA úlohu na obrazovce a poslat ji operátorovi technické podpory k vyřešení. [14]

Dalším problémem je přístupnost úloh pro všechny uživatele včetně těch fyzicky (nevidomí, hluší) a psychicky (dyslektici) postižených, na které se často zapomíná. Pro

některé z nich mohou být CAPTCHA úlohy prakticky neřešitelné. Překážkou může být také neznalost anglického jazyka používaného v některých CAPTCHA systémech. Částečným řešením je možnost volby z více různých typů testů využívajících odlišné smysly a schopnosti. [15]

3.1.4 Typy CAPTCHA úloh

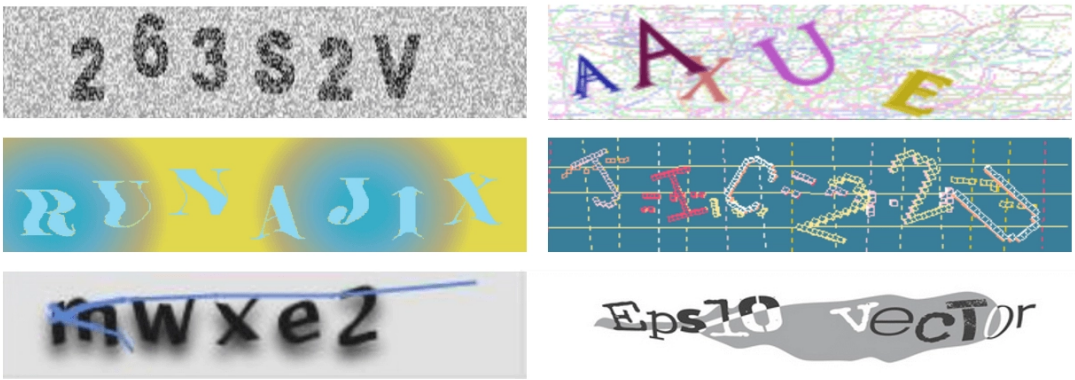
Vzhledem k velmi obecné definici CAPTCHA úlohy, existuje několik různých typů lišících se v samotné podstatě testu. Každý typ pak má mnoho rozdílných implementací s různými výhodami a nevýhodami. Nekompletní seznam implementací a jejich typů je vidět na obrázku 3.1.

3.1.4.1 Textové

Tento typ se stal synonymem pro slovo CAPTCHA díky jeho vysoké rozšířenosti. Uživatel má za úkol přepsat posloupnost písmen a číslic z obrázku. Obtížnost se může velmi lišit na základě následujících parametrů vygenerovaného obrázku:

- posloupnost znaků – reálné slovo, či náhodná sekvence, použití malých a velkých písmen a číslic,
- znaky – velikost, font, barvy, pootočení, deformace (zkroucení, rozmazání, ...), překryv s ostatními znaky,
- pozadí – barvy, textura, další čáry a linie, částečná inverze s barvou textu.

Příklady takovýchto různých úprav jsou vidět na obrázku 3.2. Jejich cílem je ztížit některou z fází automatického rozpoznávání textu, například odstranění pozadí, rozdělení sekvence na jednotlivé znaky, či samotné rozpoznávání symbolů. Ovšem v současné době už ani tato opatření nejsou dostatečná a textové CAPTCHA úlohy nejsou považovány za bezpečné. [16]



Obrázek 3.2. Příklady různých úprav textu pro ztížení jeho rozpoznání pomocí technologie OCR [17]

3.1.4.2 Obrázkové

Obrázkové CAPTCHA testy jsou modernější alternativou k textovým. Jejich podstatou je problém klasifikace či detekce objektů (viz kapitolu 3.2.1). Existuje mnoho různých implementací. Liší se zadáním úlohy a také způsobem interakce s ní. V některých člověk musí něco obtáhnout, spojit, či nakreslit, v jiných zase něco někam přesunout, nebo přetáhnout. Nejčastější podkategorií je ovšem tzv. výběrová, jejímž úkolem je na základě textového zadání zvolit některé z několika prezentovaných obrázků. [13]

Obecně, nevýhodou těchto úloh je potřeba velkého množství obrázků s anotacemi. Pro tvorbu zadání je tedy nutná nějaká počáteční lidská práce. Oproti tomu výhodou je srozumitelnost úloh a příjemné ovládání i na mobilních zařízeních.

3.1.4.3 Matematické

Jedná se o lehce prolomitelný typ testů, kdy je uživateli na obrázku, či v textu prezentován primitivní matematický příklad (např. 7 + dva =), jehož výsledek je odpovědí. V případě zadání přímo v podobě textu lze úlohu snadno algoritmicky vyřešit bez použití strojového učení.

3.1.4.4 Zvukové

Hlavním účelem tohoto typu CAPTCHA úloh je umožnění splnění i pro postižené lidi – dyslektiky a nevidomé. Většinou jsou tedy audio úlohy nabízeny jako alternativa k vizuálním, které jsou pro normálního uživatele přívětivější. Cílem je rozpoznat a přepsat několik číslic, písmen, či slov řečených v audio nahrávce obsahující rušivé zvuky v pozadí.

Zvukové úlohy jsou ovšem problematické. První překážkou pro některé lidi může být neznalost anglického jazyka, který se často používá pro předčítání znaků. Dalším problémem je fakt, že rozpoznání slov v audio stopě je pro, nejen zrakově postižené, lidi obtížné kvůli velkému umělému zkreslení zvuku. [15] Naopak moderní algoritmy pro rozpoznávání řeči si s tímto typem CAPTCHA testů umí dobře poradit, čehož využívá například program Buster automaticky řešící alternativní zvukovou úlohu u známých reCAPTCHA úloh. [18]

3.1.4.5 Behaviorální

Tento typ CAPTCHA úloh verifikuje uživatele na základě jeho chování – pohybů myší, dotyků obrazovky, úderů do klávesnice, či cookies a dříve navštívených stránek. Některé implementace stále vyžadují od člověka vyřešení nějakého úkol a sledují jeho chování pouze během plnění. Jiné se řídí heslem, že nejlepší CAPTCHA je žádná CAPTCHA, a sledují uživatele během celé interakce s webovou stránkou.

Velkou výhodou tohoto typu je minimalizace nutné interakce ze strany uživatele. Na druhou stranu pracuje nějakým způsobem s osobními daty, což může být problematické z hlediska soukromí.

3.1.4.6 Ostatní

Vzhledem k již dlouhé historii CAPTCHA testů, není překvapivé, že se za tu dobu objevila spousta nápadů na další typy úloh. Většina z nich se však z různých důvodů masově nerozšířila, či úplně zanikla.

Prvním příkladem jsou testy využívající video či jiné animace. Jejich podoba se může velmi lišit – od přepsání postupně se odkrývajícího textu až po popsání pohybu člověka prezentovaného na videoklipu. Zajímavými zástupci jsou Solve Media a někdejší CAPTCHAAD prezentující uživateli reklamní spot s následným dotazem na jeho obsah. [19–20]

Dalším typem jsou CAPTCHA úlohy bojující proti odporu uživatelů k nim tím, že se snaží být více zábavné. Jedná se o různé hry založené na pravidlech či sémantických významech, které jsou těžko pochopitelné pro stroj. Představitelem je např. PlayThru CAPTCHA. [13]

Posledním zajímavou ukázkou je historický projekt social CAPTCHA společnosti Facebook, jenž ověřoval, zda uživatel je člověk, na základě schopnosti rozpoznání svých přátel ze sociální sítě Facebook. [21]

3.1.5 CAPTCHA crowdsourcing

Dle některých odhadů každý den lidé dohromady stráví okolo 500 let vyplňováním CAPTCHA úloh. [22] Ať už je toto číslo ve skutečnosti jakékoli, není pochyb, že je vysoké, obzvláště vezmeme-li v potaz, že tento čas slouží pouze k dokázání, že lidé jsou opravdu lidé. Míru tohoto plýtvání lidským potenciálem si uvědomili už v minulosti a přišli se způsobem, jak CAPTCHA úkoly využít i k dalším prospěšným věcem.

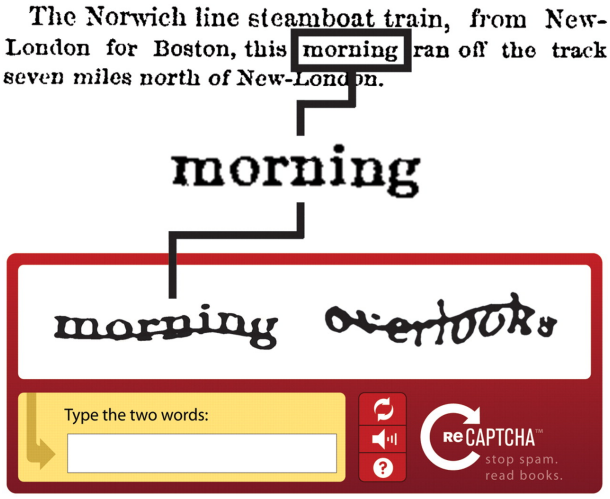
Hlavní oblastí, která benefituje z vyplňování CAPTCHA úloh, je anotování dat (data annotation, data labeling). Jedná se o proces přidávání charakterizujících značek (label) k nestrukturovaným datům jako jsou například obrázky. Existence takovýchto anotací hraje zásadní roli při trénování modelů hlubokého učení, které pro správné fungování (předpovídání) potřebují velké množství ukázkových vstupů se správnou odpovědí. Příklady zakomponování anotování dat do CAPTCHA úlohy jsou uvedeny v následujících podkapitolách.

3.1.5.1 reCAPTCHA

Prvním úspěšným projektem využívajícím odpovědi na CAPTCHA úlohy byla reCAPTCHA. S nápadem přišel v roce 2007 kolektiv okolo Luise von Ahna, jenž vymyslel i slovo CAPTCHA. Tou dobou probíhaly digitalizace velkého množství tištěných textů, aby byly zachovány a dalo se s nimi jednodušeji pracovat. Knihy, noviny a další tiskoviny se nejprve naskenovaly a následně se na ně použily algoritmy pro rozpoznávání textu z obrázků. Ty ovšem nebyly dostatečně přesné, takže mnoho slov bylo určeno špatně, případně vůbec. Provést jejich korekturu mohl pouze člověk, ale profesionální přepisovatelé jsou drazí. A právě tento problém řešila reCAPTCHA a to formou crowdsourcingu – rozdělením práce mezi velké množství lidí.

Celý proces ověření uživatele reCAPTCHA a určení slova nerozpoznatelného pomocí OCR probíhal následovně. Na sken tiskoviny byly použity dva různé v té době moderní programy pro rozpoznávání textu. Jejich výstupy byly porovnány navzájem a také s anglickým slovníkem. V případě, že se určená slova neshodovala nebo pokud se nejednalo o anglické slovo, bylo označeno za neznámé. Aby uživatel reCAPTCHA prokázal, že je člověk, musel z prezentovaného obrázku opsat dvě slova. První z nich bylo neznámé slovo a druhé tzv. kontrolní – jeho správný přepis byl známý. Pořadí na obrázku bylo náhodné. Příklad zadání reCAPTCHA úlohy je vidět na obrázku 3.3. V případě, že bylo kontrolní slovo uživatelem správně vyplněno, považoval se za člověka a předpokládalo se, že správně vyplnil i neznámé slovo. Samozřejmě i lidé dělají chyby, takže aby přepis neznámého slova byl označený za správný, musel obdržet alespoň 2,5 hlasu, přičemž odpověď uživatele měla váhu 1 a výstup z algoritmu pro rozpoznávání textu měl váhu 0,5.

Aby byla zaručena vysoká míra bezpečnosti, jako kontrolní slova byla vybírána pouze původně neznámá slova, která oba OCR programy nedokázaly správně rozpoznat a první 3 uživatelé, jimž byla zobrazena se shodli na jednom přepisu. Automatizované řešení této CAPTCHA úlohy by tedy nutně znamenalo zlepšení v moderních OCR technikách.



Obrázek 3.3. Příklad zadání původní reCAPTCHA úlohy. Slovo „morning“ je neznámé a slovo „overlooks“ je kontrolní. Pro ztížení úlohy byla obě ještě lehce zdeformována. [23]

Tímto způsobem bylo i bez znalosti kontextu z okolních slov dosaženo více než 99% přesnosti, což odpovídá standardu profesionálních přepisovatelů. Zároveň rychlost přepisů mnohonásobně vzrostla a náklady se snížili na minimum. [23] Projekt reCAPTCHA byl natolik úspěšný, že ho v roce 2009 odkoupila společnost Google a využívala ho při tvorbě obsahu služby Google Books. [24] Později se také v zadání úkolu začaly objevovat obrázky s čísly domů a názvy ulic pocházející z Google Street View (viz obrázek 3.4). Jejich rozpoznávání pomohlo vylepšit aplikaci Google Maps. [25] Provoz této verze (v1) reCAPTCHA systému byl oficiálně ukončen v roce 2018. [26]

Obrázek 3.4. Příklad zadání původní reCAPTCHA úlohy s použitím obrázku čísla domu ze služby Street View [27]

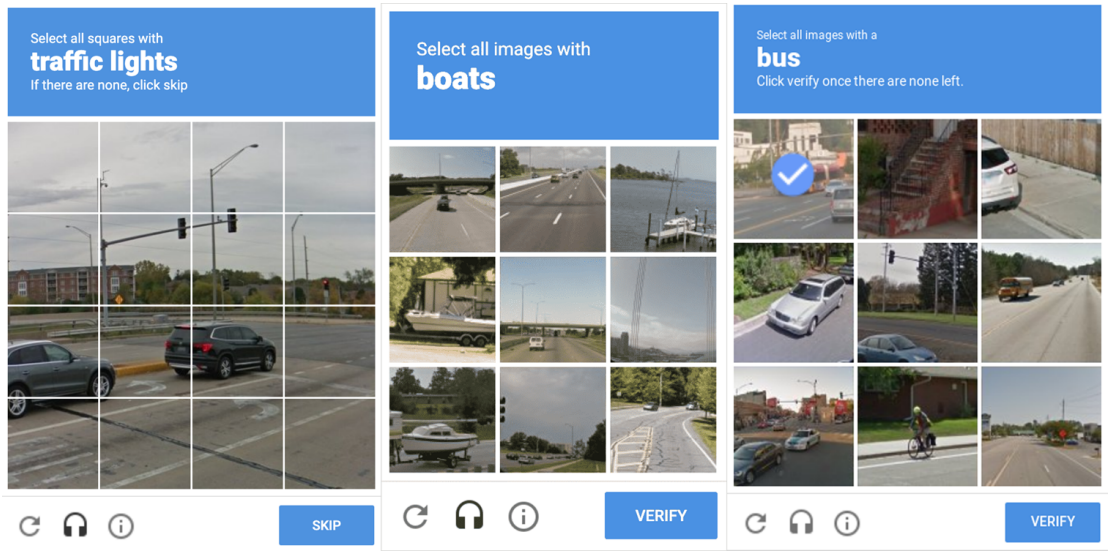
3.1.5.2 reCAPTCHA v2

V reakci na úspěšnost projektu reCAPTCHA a interní výzkum ohledně bezpečnosti textových CAPTCHA úloh (viz kapitolu 3.1.2) společnost Google představila v roce 2014 systém reCAPTCHA v2, též prezentovaný jako No CAPTCHA reCAPTCHA, který ačkoli byl prolomen, je funkční dodnes.

Jeho hlavním prvkem je tzv. systém pokročilé analýzy rizik (advanced risk analysis system) vyhodnocující, zda je uživatel člověk či robot, pomocí například pohybů myši a zmáčknutých kláves, historie prohlížení či cookies souborů. [28] Úkolem uživatele je pouze označit ikonické zaškrtávací pole „Nejsem robot“. Přesná podoba algoritmu pro ověření není známá a lze ji zkoumat pouze pomocí testování za různých podmínek. Společnost Google využívala sledování chování uživatele již v pokročilejším stádiu první verze reCAPTCHA, kdy výstupy analýzy byly použity ke zvolení obtížnosti prezentovaného obrázku s textem. Uživatelé chovající se více jako lidé dostávali jednodušší zadání. [29] V tomto behaviorálním trendu pokračuje Google dodnes v podobě reCAPTCHA v3, jež od uživatele nevyžaduje žádnou interakci.

V případě, že se uživatel chová podezřele, např. velmi přesně a rychle kliká, je mu zobrazena jedna kontrolní obrázková úloha ze tří možných typů, které lze vidět i na obrázku 3.5:

- mřížka 4×4 pro detekci objektů – uživatel musí na jednom rozřezaném obrázku zvolit pole, která obsahují zadaný objekt,
- mřížka 3 × 3 pro klasifikaci objektů – cílem je z 9 různých obrázků označit ty, co obsahují zadaný objekt,
- mřížka 3 × 3 pro klasifikaci objektů s měnícími se obrázky – obdoba předchozí úlohy, ovšem obrázek po zvolení zmizí a je nahrazen novým. [30]



Obrázek 3.5. Různé typy reCAPTCHA v2 úloh – zleva mřížka 4 × 4 pro detekci objektů, mřížka 3 × 3 pro klasifikaci objektů, mřížka 3 × 3 pro klasifikaci objektů s měnícími se obrázky [31–33]

Opět přesný způsob vyhodnocení těchto úloh není veřejně známý, ale lze předpokládat, že obdobně jako v první verzi reCAPTCHA jsou některé obrázky kontrolní a některé neznámé a že na základě několika shodných hlasů se neznámým obrázkům přiřazují anotace.

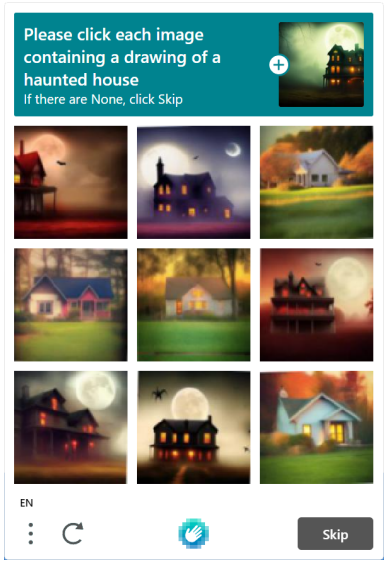
Tematika obrázků se v průběhu času měnila, avšak v poslední době se ustálila na objektech ze silničního provozu jako jsou auta, kola, semafory, či hydranty. To by mohlo napovídat o tom, že označená data jsou použita k trénování modelů hlubokého učení sloužících k řízení autonomních aut, ovšem dceřinná společnost Waymo, dříve známá jako Google self-driving car project, tvrdí, že výstupy z reCAPTCHA v2 systému nepoužívá. [34]

3.1.5.3 hCaptcha

Aplikace hCaptcha vznikla ve společnosti Intuition Machines jako reakce na potřebu anotování dat pro učení neuronových sítí. V roce 2019 však byla zveřejněna jako placená služba i pro další firmy. Anotování dat internetovými uživateli je tak cenné, že hCaptcha platí webům, které ji používají, za správné vyplňování úloh. Veškeré tyto transakce jsou zveřejňovány prostřednictvím protokolu Human Protocol, decentralizované účetní knihy, která je postavená nad blockchainem Ethereum. [35] Dle oficiálního webu hCaptcha momentálně běží na zhruba 15 % internetových stránek. [36]

Z uživatelského hlediska úloha funguje obdobně jako reCAPTCHA s mřížkou 3 × 3 pro klasifikaci objektů, která je ovšem zobrazena dvakrát, takže člověk má na výběr

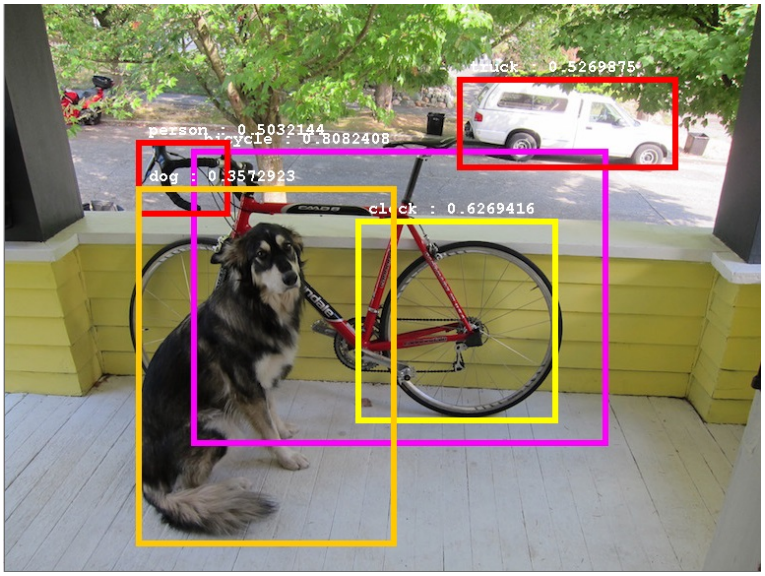
z celkem 18 obrázků. Díky datasetům od různých společností jsou úlohy různorodější, takže namísto objektů ze silničního provozu musí člověk označit například všechny opasky, dorty či strašidelné domy (viz obrázek 3.6).



Obrázek 3.6. Příklad zadání hCaptcha úlohy

3.2 Detekce objektů

Detekce objektů spadá pod oblast strojového vidění. Cílem je na obrázku detekovat a pomocí *ohraničujícího obdélníku* (bounding box) označit nějaké objekty. Její pole uplatnění je široké – od dopravy (autonomní vozidla, detekce SPZ) přes průmysl (detekce anomálií) až k zabezpečení (rozpoznávání obličejů). Příklad detekce objektů je vidět na obrázku 3.7.



Obrázek 3.7. Příklad výstupu detekce objektů na obrázku [37]

K detekci objektů se využívají techniky hlubokých neuronových sítí (deep neural networks). Neuronové sítě jsou výpočetní systémy určené k predikci. Svůj název dostaly na základě konceptuální podobnosti s mozkem. Jejich základními stavebními kameny jsou

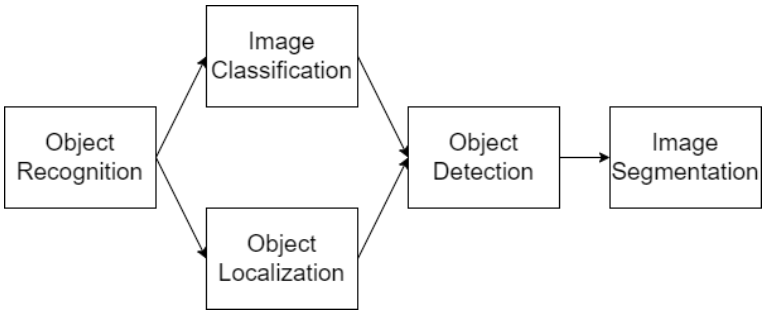
neurony, které jsou seřazeny do navzájem propojených vrstev. Slovo hluboké v termínu pak odkazuje na vysoký počet těchto vrstev.

Určité uspořádání a propojení vrstev se nazývá *model*. Samotný model k dobrým predikcím ovšem nestačí. Je nutné ho natrénovat, neboli upravit spojení mezi neurony. To probíhá tak, že na vstup modelu se dávají tzv. *trénovací data*, model provede na jejich základě nějakou predikci a ta je porovnána s požadovaným, správným výstupem. Podle výsledku tohoto porovnání se pak upraví jednotlivá spojení mezi neurony, čímž se v ideálním případě zlepší další predikce. Zpravidla čím větší je množství trénovacích dat, tím lépe lze model natrénovat, ale také tím větší jsou náklady na trénink a to jak výpočetní, tak časové. [38] Detailnější popis fungování a trénování neuronových sítí není předmětem této práce.

Sada (trénovacích) dat se nazývá *dataset*. Struktura modelu spolu s trénovacím datasetem určují, co a jak bude model schopný predikovat. Pro detekci objektů jsou k dispozici velké datasety obsahující statisíce obrázků s označenými objekty. Jedná se o několik desítek až stovek objektů z každodenního života, například auta, lidi, psi. Nejznámější modely pro detekci objektů jsou trénovány právě na těchto datasetech, takže jejich schopnost detekce se omezuje právě na tyto kategorie objektů. [39]

3.2.1 Příbuzné pojmy

Pojem detekce objektů je zaměňován s pojmy rozpoznávání objektů (Object Recognition), klasifikace obrázků (Image Classification), lokalizace objektů (Object Localization) a segmentace objektů (Object Segmentation). Liší se ovšem svým výstupem. Vztah jednotlivých pojmů je zachycen na obrázku 3.8.



Obrázek 3.8. Vztah mezi pojmy příbuznými detekci objektů

Rozpoznávání objektů je oblast strojového a hlubokého učení, která si klade za cíl naučit počítače rozpoznávat objekty na obrázcích a ve videích podobně, jako to umí lidé. Je to pojem nadřazený všem následujícím.

Klasifikace obrázků přiřadí každé kategorii objektů pravděpodobnost s jakou se nachází na vstupním obrázku. Suma pravděpodobností všech kategorií je rovna 1. Často se následně vybere pouze jedna kategorie s nejvyšší pravděpodobností a ta se přiřadí k obrázku. I pro členitý obsah dostáváme tedy pouze jednu kategorii nejlépe charakterizující obrázek jako celek.

Výstupem *lokalizace objektů* jsou čtyři hodnoty určující obdélník rovnoběžný s osami x a y (možnosti jeho zakódování jsou uvedeny v kapitole 3.2.5) ohraničující lokalizovaný objekt. Objektu je také přiřazena pravděpodobnost na jakou byl správně identifikován. Na rozdíl od klasifikace obrázků, která je klasifikačním problémem (řešení se nachází v diskrétním prostoru), se jedná o problém regresní (řešení se nachází ve spojitém prostoru).

Detekce objektů se skládá z lokalizace a klasifikace. Oproti lokalizaci lze detekovat více objektů ze stejných i různých kategorií najednou.

Posledním příbuzným pojmem je *segmentace objektů*, kterou lze rozdělit na dva základní typy:

- sémantická segmentace (Semantic Segmentation) – každému pixelu obrázku je přiřazena kategorie, více instancí objektu ze stejné kategorie je označeno jako jeden objekt,
- segmentace instancí (Instance Segmentation) – nalezení přesného ohraničení každé instance objektu z dané kategorie.

V obou případech tedy máme přesnou informaci o tvaru jednotlivých kategorií namísto ohraničujících obdélníků. Příklady obou typů segmentace lze vidět na obrázku 3.9. [40]

Obrázek 3.9. Rozdíl mezi sémantickou segmentací a segmentací instancí [41]

3.2.2 Metriky

Pro porovnávání výsledků modelů detekujících objekty existuje několik metrik určujících míru správnosti předpovědí. Mezi ty nejčastěji používané patří průměrná přesnost a střední průměrná přesnost. K jejich výpočtu je potřeba definovat několik dalších pomocných pojmů.

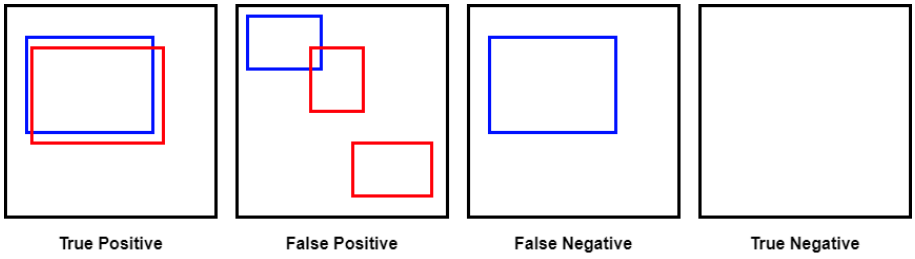
Průnik nad sjednocením (Intersection over Union – IoU) je metrika určující, jak moc se překrývají dva libovolné 2D tvary. Může nabývat hodnot 0 až 1, přičemž je roven 1 v případě, že jsou tvary shodné, a roven 0 v případě, že se vůbec nepřekrývají. Výpočet je následující:

$$IoU = \frac{tvar_1 \cap tvar_2}{tvar_1 \cup tvar_2}$$

V kontextu detekce objektů jsou oba tvary obdélníky, přičemž jeden z nich ohraničuje skutečný objekt a druhý je předpověď modelu. Aby bylo možné určit, zda je daná předpověď správná či nikoli, je potřeba určit hraniční hodnotu α. V případě, že IoU je větší nebo rovno α, pak se předpověď považuje za správnou, v opačném případě za špatnou.

Následně je třeba definovat, co v detekci objektů představují následující pojmy, jejichž význam je zachycen na obrázku 3.10:

- *True Positive (TP)* je správně detekovaný existující objekt.
- *False Positive (FP)* je špatně detekovaný objekt (neexistující nebo existující, ale nedostatečně přesně ohraničený).
- *False Negative (FN)* je špatně nedetekovaný existující objekt.
- *True Negative (TN)* je správně nedetekovaný neexistující objekt. Nepoužívá se, neboť je nekonečně mnoho neexistujících objektů.



Obrázek 3.10. True Positive, False Positive, False Negative, True Negative v kontextu detekce objektů. Modré obdélníky představují skutečná ohraničení objektů, červené jsou predikce modelu. Druhý příklad ukazuje oba možné případy False Positive – detekování neexistujícího objektu a nedostatečně přesné detekování objektu.

Přesnost (precision – P) je metrika určující, jak moc je model schopný predikovat pouze relevantní objekty.

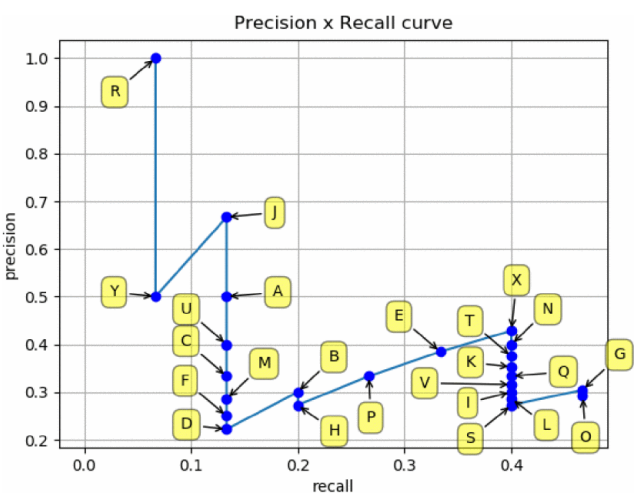
$$P = \frac{TP}{TP + FP}$$

Recall (R) je metrika určující, jak moc je model schopný predikovat všechny správné skutečné hodnoty.

$$R = \frac{TP}{TP + FN}$$

Precision a recall metriky jsou úzce spjaté – když jedna klesá, druhá roste. Graf jejich závislosti se nazývá *precision-recall křivka*. Příklad jejího tvaru na základě predikcí modelu je vidět na obrázku 3.11.

confidence	TP	FP	acc TP	acc FP	precision	recall
95%	1	0	1	0	1	0.0666
95%	0	1	1	1	0.5	0.0666
91%	1	0	2	1	0.6666	0.1333
88%	0	1	2	2	0.5	0.1333
84%	0	1	2	3	0.4	0.1333
80%	0	1	2	4	0.3333	0.1333
78%	0	1	2	5	0.2857	0.1333
74%	0	1	2	6	0.25	0.1333
71%	0	1	2	7	0.2222	0.1333
70%	1	0	3	7	0.3	0.2
67%	0	1	3	8	0.2727	0.2
62%	1	0	4	8	0.3333	0.2666
54%	1	0	5	8	0.3846	0.3333
48%	1	0	6	8	0.4285	0.4
45%	0	1	6	9	0.4	0.4
45%	0	1	6	10	0.375	0.4
44%	0	1	6	11	0.3529	0.4
44%	0	1	6	12	0.3333	0.4
43%	0	1	6	13	0.3157	0.4
38%	0	1	6	14	0.3	0.4
35%	0	1	6	15	0.2857	0.4
23%	0	1	6	16	0.2727	0.4
18%	1	0	7	16	0.3043	0.4666
14%	0	1	7	17	0.2916	0.4666



Obrázek 3.11. Precision-recall křivka sestrojená na základě predikcí modelu zachycených v tabulce vlevo [42]

Konečně *průměrná přesnost (average precision – AP)* je definovaná jako obsah plochy pod precision-recall křivkou pro dané α. Značí se AP@α či APα. Formálně:

$$AP@\alpha = \int_0^1 P(R)\mathrm{d}R$$

Čím vyšší je tato hodnota, tím vyšší jsou precision a recall metriky, což znamená lepší predikce modelu.

Jelikož výsledky jsou velmi závislé na hodnotě parametru α, většinou se modely porovnávají pomocí průměru AP při různých hodnotách α. Asi nejčastější je AP@[.50:.05:.95], které počítá s 10 hodnotami α od 0,5 po 0,95 lišící se vždy o 0,05.

Střední průměrná přesnost (mean average precision – mAP) se vypočítá jako průměr z průměrných přesností všech tříd objektů. Pro n různých tříd lze vztah zapsat takto:

$$mAP@\alpha = \frac{1}{n}\sum_{i=1}^{n} AP_i@\alpha$$

Obdobně jako pro průměrnou přesnost existuje i pro střední průměrnou přesnost hodnota mAP@[.50:.05:.95] počítaná při různých hodnotách parametru α. [42]

3.2.3 Modely pro detekci objektů

Modely pro detekci objektů se dělí do dvou hlavních skupin – jednofázové (single-stage) a dvoufázové (two-stage). V tabulce 3.1 lze vidět srovnání jednotlivých modelů vytvořené v roce 2021.

3.2.3.1 Dvoufázové modely

Dvoufázové modely historicky předcházely jednofázovým. V první fázi je navrženo libovolné množství částí obrázku, kde by se mohly nacházet nějaké objekty. V druhé fázi se v těchto částech objekty přesně lokalizují a klasifikují. Obecně jsou dvoufázové modely pomalejší v predikci, neboť mají komplikovanější architekturu a musí provést dva oddělené kroky, čímž také částečně postrádají globální kontext.

Hlavními zástupci tohoto typu jsou modely z rodiny R-CNN (Region-Based Convolutional Neural Network), konkrétně R-CNN, Fast R-CNN, Faster R-CNN a Mask R-CNN.

3.2.3.2 Jednofázové modely

Jednofázové modely celý proces detekce provedou v jednom kroku pomocí hustého vzorkování, ke kterému používají předdefinované obdélníky různých velikostí. Jsou proto rychlejší a více hodí pro detekci v reálném čase. V minulosti platilo, že měly menší přesnost, ale v poslední době se ukazuje, že tento typ architektury je lepší i v tomto ohledu.

Zástupci jednofázových modelů jsou YOLO (You Only Look Once) verze v1-v5, SSD (Single Shot MultiBox Detector), RetinaNet, EfficientDet, CenterNet či Swin Transformer. [39]

3.2.3.3 Páteřní architektura

Většina modelů pro detekci objektů je postavená na základě modelů hlubokého učení určených pro klasifikaci obrázků. Těmto modelům se proto říká páteřní architektura (backbone architecture). Využívá se faktu, že tyto klasifikační modely mají velké množství skrytých vrstev, které z obrázku extrahují nejprve základní vlastnosti (úsečky, křivky) a následně i komplexnější tvary. Většina jejich vrstev je zachována nezměněná a upravuje se až několik posledních vrstev, které mají v originálních modelech na starost samotnou klasifikaci.

V případě, že chceme detekovat objekty z kategorií podobných těm, na kterých byla trénována i páteřní architektura, je dobré využít již natrénovaných vah z původního modelu. V případě velmi odlišných dat, která chceme detekovat, je lepší trénovat model od začátku. [43]

Příkladem páteřní architektury je AlexNet, VGG, ResNet, GoogLeNet, DarkNet či EfficientNet. [39] Páteřní architektury využité v jednotlivých modelech pro detekci objektů lze vidět v tabulce 3.1.

Model	Year	Backbone	Size	$AP_{[0.5:0.95]}$	$AP_{0.5}$	FPS
R-CNN*	2014	AlexNet	224	-	58.50%	0.02
SPP-Net*	2015	ZF-5	Variable	-	59.20%	0.23
Fast R-CNN*	2015	VGG-16	Variable	-	65.70%	0.43
Faster R-CNN*	2016	VGG-16	600	-	67.00%	5.00
R-FCN	2016	ResNet-101	600	31.50%	53.20%	3.00
FPN	2017	ResNet-101	800	36.20%	59.10%	5.00
Mask R-CNN	2018	ResNeXt-101-FPN	800	39.80%	62.30%	5.00
DetectoRS	2020	ResNeXt-101	1333	53.30%	71.60%	4.00
YOLO*	2015	(Modified) GoogLeNet	448	-	57.90%	45.00
SSD	2016	VGG-16	300	23.20%	41.20%	46.00
YOLOv2	2016	DarkNet-19	352	21.60%	44.00%	81.00
RetinaNet	2018	ResNet-101-FPN	400	31.90%	49.50%	12.00
YOLOv3	2018	DarkNet-53	320	28.20%	51.50%	45.00
CenterNet	2019	Hourglass-104	512	42.10%	61.10%	7.80
EfficientDet-D2	2020	Efficient-B2	768	43.00%	62.30%	41.70
YOLOv4	2020	CSPDarkNet-53	512	43.00%	64.90%	31.00
Swin-L	2021	HTC++	-	57.70%	-	-

Tabulka 3.1. Modely označené * jsou porovnávány na datasetu PASCAL VOC 2012, zatímco ostatní na MS COCO. [39]

3.2.4 Frameworky pro detekci objektů

Tato kapitola obsahuje přehled dostupných frameworků pro hluboké učení použitelných v programovacím jazyce Kotlin, potažmo spustitelných na Java Virtual Machine (JVM), které by bylo možné použít k detekci objektů v aplikaci CzechCaptcha.

3.2.4.1 KotlinDL

KotlinDL je vysokoúrovňové programovací rozhraní (Application Programming Interface – API) pro hluboké učení napsané v programovacím jazyce Kotlin. Využívá TensorFlow Java API a ONNX Runtime API pro Javu. Podporuje kompletní trénování modelů i využívání předtrénovaných Keras a ONNX modelů. [44]

ONNX (Open Neural Network Exchange) je open-source formát reprezentující modely strojového učení. Byl vytvořen, aby bylo možné přenášet modely mezi různými nástroji a frameworky. Definuje základní stavební bloky strojového a hlubokého učení, pomocí kterých reprezentuje graf modelu. [45] V rámci projektu také vznikla kolekce běžných modelů, tzv. model zoo, právě ve formátu `.onnx`. Z oblasti detekce objektů se v ní nachází například Faster-RCNN, RetinaNet a YOLOv4. [46]

3.2.4.2 ML Kit

ML Kit je framework pro hluboké učení od společnosti Google napsaný v jazyce Kotlin a Java. Je určený primárně pro mobilní zařízení s operačním systémem Android, či iOS, takže podporuje například použití kamery jako zdroje obrázků. Obsahuje implementaci blíže nespecifikovaného modelu pro detekci objektů, který podporuje pouze 5 předdefinovaných kategorií. Lze do něj nahrát a používat modely ve formátu `.tflite` a `.lite`. [47]

3.2.4.3 Deep Java Library

Dalším open-source vysokoúrovňovým frameworkem pro hluboké učení napsaným v Javě je Deep Java Library (DJL). Je navržen tak, aby bylo možné využít libovolný

engine pro hluboké učení. [48] V současnosti jsou podporovány následující enginy spolu s jejich formáty modelů (v závorkách):

- Apache MXNet (MXNet symbolic model)
- PyTorch (TorchScript model)
- TensorFlow (`.pb`, Keras model)
- fastText (`.bin`, `.ftz`, SageMaker BlazingText)

Dále je možné načíst modely v ONNX formátu [49], či využít integrované model zoo obsahující modely v několika různých formátech. [50]

3.2.4.4 Deeplearning4j

Eclipse Deeplearning4j je soubor nástrojů pro hluboké učení na JVM. Sestává z několika modulů – například DL4J (vysokoúrovňové API pro hluboké učení), ND4J (knihovna pro lineární algebru, ekvivalent knihovny numpy v jazyce Python), či SameDiff (nízkoúrovňové API pro hluboké učení). [51] Součástí Deeplearning4j je podpora nahrání Keras a TensorFlow (`.pb`) modelů a také model zoo obsahující několik modelů včetně YOLOv2. [52]

3.2.4.5 TensorFlow for Java

Jak již název napovídá, TensorFlow for Java zprostředkovává přístup k TensorFlow, oblíbenému frameworku pro hluboké učení napsanému v jazyce Python. V minulosti byl tento projekt zveřejňován spolu s originálním TensorFlow. V dnešní době už je ovšem oddělený a aktuální verze 0.4.1 využívá TensorFlow 2.7.1. [53]

Samozřejmostí je možnost nahrání modelů v TensorFlow formátu (`.pb`). Ty je možné stáhnout například ze stránky TensorFlow Hub, který obsahuje stovky předtrénovaných modelů včetně těch určených pro detekci objektů. [54]

3.2.4.6 Další frameworky

Existují i další frameworky pro hluboké učení napsané v jazyce Java. Ty ovšem už neposkytují žádné natrénované modely ani možnost jejich importu, takže pro detekci objektů by bylo nutné si modely vytvořit a natrénovat samostatně. Mezi tyto frameworky patří například OpenCV a Neuroph. [55–56]

3.2.5 Anotační formáty pro detekci objektů

Pro supervizované strojové učení, mezi něž spadá i detekce objektů, hraje správné označení dat klíčovou roli, neboť kvalita výsledného modelu je velmi závislá na kvalitě dat. Tento fakt vyjadřuje i fráze používaná v oblasti neuronových sítí „Garbage In Garbage Out“. [57]

Jak již bylo zmíněno v kapitole 3.2.1, data pro detekci objektů se skládají primárně z pěti hlavních informací – čtyř souřadnic určujících polohu obdélníku ohraničujícího daný objekt a názvu daného objektu. V případě výstupu z modelu se ještě setkáváme s pravděpodobností správného přiřazení, ovšem pro vstupní trénovací data se tento údaj nevyužívá. Dále se ovšem o objektů dá uchovávat množství dalších informací jako například jeho nadřazená kategorie, zda se jedná o jednu, či více instancí, nebo zda je objekt částečně zakrytý.

To, jaké údaje se o objektech ukládají a v jaké podobě, určuje použitý anotační formát. Pro detekci objektů neexistuje žádný jednotný standard, dle kterého by se anotace uchovávaly. Existuje ovšem pár většinově používaných formátů a následně spousta proprietárních formátů pocházejících primárně od tvůrců nástrojů pro ruční anotaci obrázků. Vzhledem k tomu, že každý formát uchovává různé informace, jsou mezi sebou převeditelné pouze částečně. [58]

Dokonce i jednotlivé obdélníky ohraničující objekty jsou zakódovány různými způsoby. Ty lze však jednoduše převést. První rozdíl může být v tom, zda jsou rozměry uvedeny v absolutních či relativních hodnotách. U formátů, kde jsou použity absolutní rozměry, je vždy uvedeno i rozlišení obrazového souboru, aby je bylo možné přepočítat v případě zvětšení či zmenšení obrázku. Druhým rozdílem může být kombinace hodnot určujících obdélník. Možnosti jsou následující:

- x a y souřadnice levého horního vrcholu a x a y souřadnice pravého dolního vrcholu
- x a y souřadnice levého horního vrcholu a šířka a výška
- x a y souřadnice středu a šířka a výška

3.2.5.1 YOLO

Anotační formát YOLO se stal známým díky Darknet frameworku pro neuronové sítě použitému při implementaci YOLO modelů. Pro všechny anotované obrázky existují textové soubory (`.txt`) se stejnými jmény. Na každém řádku těchto souborů se pak nachází jeden objekt v následujícím formátu.

```
<číslo-třídy-objektu> <x-střed> <y-střed> <šířka> <výška>
```

Složka s obrázky dále obsahuje ještě jeden textový soubor, v němž jsou uvedeny názvy tříd objektů. Číslo třídy objektu odpovídá řádku v tomto souboru. [59]

Výhodou tohoto formátu je jeho stručnost a jednoduchost. Dále také fakt, že veškeré rozměry jsou uváděny v relativních hodnotách (od 0 do 1). Naopak nevýhodou je dvojnásobné množství souborů ve složce s daty, což může zpomalovat práci s nimi.

3.2.5.2 COCO

Dalším anotačním formátem je COCO, jenž byl využit k anotaci často používaného Microsoft COCO datasetu. Veškeré informace jsou zachyceny v jednom JSON (JavaScript Object Notation) souboru. Ten obsahuje objekty s následujícími daty:

- obecné informace – autor, verze, datum vytvoření, popis, ...
- licence
- obrázky – id, jméno souboru, rozměry v pixelech, ...
- kategorie – id, jméno, nadřazená kategorie
- anotace – id obrázku, id kategorie, absolutní souřadnice ohraničujícího obdélníku (levý horní vrchol, šířka a výška), plocha obdélníku, příznak, zda se jedná o jednu, či více instancí, ... [60]

Předností COCO formátu je přítomnost všech informací v jednom souboru nezávisle od místa uložení samotných obrázků. To zlehčuje například vyhledávání všech obrázků obsahujících objekt dané kategorie. Užitečné mohou být i informace o nadřazené kategorii, či zda obdélník ohraničuje více instancí objektu. V neposlední řadě lze formát s drobnými změnami použít k uložení informací potřebných nejen pro detekci objektů, ale i pro další úlohy strojového vidění. Negativem může být zbytečně velké množství uchovávaných informací a tím pádem větší velikost souboru.

3.2.5.3 Pascal VOC

Formát Pascal VOC byl vytvořen pro použití v soutěži v oblasti strojového vidění PASCAL Visual Object Classes Challenge. Pro každý anotovaný obrázek existuje soubor ve formátu XML (Extensible Markup Language) se stejným jménem. Ten obsahuje elementy s informacemi, jako je jméno souboru, jeho umístění a velikost, a pak samotné objekty – název jejich kategorie, ohraničující obdélník (levý horní a pravý dolní vrchol),

typ pohledu na objekt a příznaky, zda je objekt oříznutý, částečně překrytý, či těžko rozpoznatelný. [61–62]

Tyto detailní informace o objektu jsou hlavním pozitivem formátu Pascal VOC. Nevýhodou je zdlouhavá XML syntaxe a duplicita ukládaných informací, tedy výsledná velikost anotačních dat.

3.2.5.4 TFRecord

Tensorflow TFRecord je binární formát obsahující nejen anotace, ale i samotná data. Díky tomu je velmi efektivní jak z hlediska paměťového, tak z hlediska rychlosti čtení a zapisování, které lze dobře paralelizovat. Na druhou stranu je však nečitelný pro lidi, což může být problém, když během práce s ním něco nefunguje. [63–64]

3.3 Hierarchické shlukování

Aby bylo možné správně porozumět, jak funguje CAPTCHA úloha pro detekci objektů popsaná v kapitole 5.4, je nutné se seznámit s hierarchickým shlukováním.

Hierarchické shlukování je jeden ze dvou základních typů shlukování (anglicky clustering). Cílem shlukování je vytvořit ze vstupních dat shluky (skupiny) tak, aby byla data ve stejném shluku co nejpodobnější a naopak data z různých shluků co nejodlišnější. Shlukování je metoda nesupervizovaného strojového učení a slouží k rozřazování dat do kategorií nebo k vyhledávání podobnosti mezi daty.

Hierarchické shlukování vytváří skupiny následujícím způsobem. Na začátku je každý datový bod shluk. Následně se v každém kroku algoritmu spojí dva shluky, které jsou k sobě nejblíže, čímž se celkový počet shluků sníží o jedna. Nakonec vznikne pouze jeden shluk obsahující veškerá data. Výše popsaný algoritmus je tzv. aglomerativní hierarchické shlukování. Druhým typem hierarchického shlukování je divizivní, které přistupuje k problému obráceně – rozděluje jeden počáteční shluk tak dlouho, dokud každý datový bod není samostatný shluk.

Pro určení vzdálenosti dvou shluků je možné využít několika metod. Nejjednodušššími jsou metoda nejbližšího, resp. nejvzdálenějšího souseda, ve kterých je vzdálenost dvou skupin rovna vzdálenosti mezi dvěma nejbližšími, resp. nejvzdálenějšími datovými body ze shluků. Dalšími příklady metod jsou centroidní a mediánová.

Pro určení vzdálenosti dvou datových bodů se nejčastěji využívá různých metrik, jako je například Manhattanská a Euklidovská. Metrika na p-rozměrném euklidovském prostoru E_p je funkce $\delta\colon \mathrm{E}_p \times \mathrm{E}_p \to [0, \infty)$ taková, že pro každé $r, s, t \in \mathrm{E}_p$ platí:

- $\delta(r, s) \geq 0$
- $\delta(r, s) = 0 \Leftrightarrow r = s$
- $\delta(r, s) = \delta(s, r)$
- $\delta(r, t) \leq \delta(r, s) + \delta(s, t)$

Hierarchické shlukování lze dobře vizualizovat pomocí dendrogramu zachycujícího proces spojování, resp. dělení shluků. Další výhodou tohoto algoritmu je to, že není nutné na vstupu uvádět požadovaný počet vytvořených shluků a lze ho určit až na základě výstupů algoritmu. [65]

Kapitola 4
Analýza aplikace CzechCaptcha

Tato kapitola představuje stav aplikace CzechCaptcha, českého open-source CAPTCHA systému, v době počátku této bakalářské práce.

První verze aplikace CzechCaptcha vznikla v rámci diplomové práce Ing. Otakara Vinkláře [3]. Zdrojový kód je volné přístupný v Git repozitáři zveřejněném pomocí služby GitHub[1].

Hlavním cílem aplikace je zpřístupnit provozovatelům webových stránek jednoduché a vysoce konfigurovatelné ověření uživatelů CAPTCHA úlohou. Další důležitou funkcí je využití cenného času uživatelů stráveného řešením CAPTCHA úloh k vytvoření nějaké hodnoty, konkrétně pomocí anotování dat (obrázků, videí, audio souborů, atd.).

4.1 Serverová aplikace

Serverová aplikace CzechCaptcha umožňuje komunikovat (konfigurovat, generovat, vyhodnocovat úlohy) s více klienty zároveň přes REST API. Je napsaná v programovacím jazyce Kotlin, který ke svému běhu využívá JVM, čímž dovoluje využít i knihovny psané pro jazyk Java. K sestavení a testování aplikace se používá nástroj Gradle, jenž je dobře integrovaný s jazykem Kotlin. Další zásadní použitou technologií je Spring, konkrétně jeho části Spring Framework (pro umožnění Inversion of Control – IoC), Spring MVC (vytvoření HTTP endpointů), Spring Security (zabezpečení HTTP endpointů) a Spring Data (integrace s databází). Jako perzistentní úložiště je použita NoSQL dokumentová databáze MongoDB v kombinaci s migračním nástrojem Mongock. Tato databáze je charakteristická svou škálovatelností a flexibilním databázovým schématem.

CzechCaptcha umožňuje provozovatelům webu vybrat si z několika různých CAPTCHA úloh a ty nakonfigurovat včetně potřebného skóre pro projití testem či samotných dat použitých v zadání. Momentálně je k dispozici matematická (vypočítání jednoduchého příkladu), textová (přepsání zkresleného textu) a obrázková (vybrání obrázků odpovídajících popisu) CAPTCHA úloha. Další je možné jednoduše přidat vytvořením konfiguračního JSON schématu a implementováním rozhraní `TaskTemplate` a jeho metod `generateTask` a `evaluateTask`.

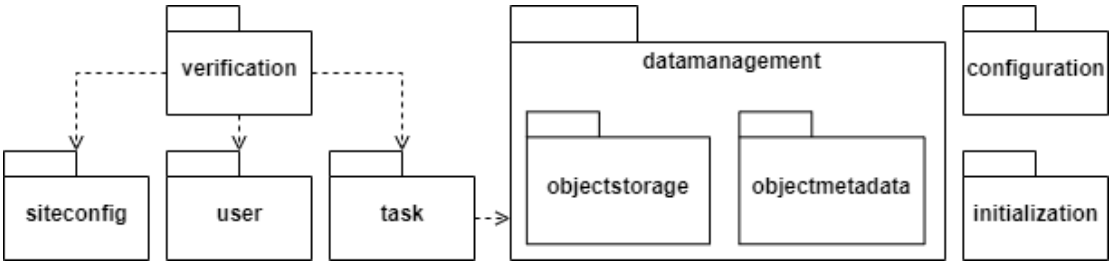
Aplikace je logicky rozdělena do následujících hlavních modulů a submodulů dle jejich funkce:

- *Verification* zprostředkovává proces verifikace uživatele (hlavní funkcionalitu CzechCaptcha).
- *Task* obsahuje implementace jednotlivých CAPTCHA úloh.
- *User* autentikuje uživatele aplikace CzechCaptcha a uchovává informace o nich.
- *SiteConfiguration* umožňuje konfiguraci CAPTCHA úloh pro použití v dalších aplikacích.
- *DataManagement* uchovává veškerá data potřebná pro tvorbu jednotlivých CAPTCHA úloh.

[1] `https://github.com/opendatalabcz/czech-captcha`

- *ObjectStorage* uchovává datové soubory (obrázky, audio, atd.) a základní informace o nich.
- *ObjectMetadata* uchovává informace o datech (stav anotování, uživatelské štítky, atp.), realizuje anotování dat.

Tyto moduly odpovídají i balíčkům použitých v implementaci aplikace (viz obrázek 4.1), což do budoucna umožňuje snadnější rozdělení na mikroslužby v případě nových funkcí nebo růstu počtu uživatelů.



Obrázek 4.1. Struktura balíčků aplikace CzechCaptcha

Pro potřeby této práce je důležité pochopit, jak funguje ukládání datových souborů a jejich metadat a dále proces anotace dat, neboli submoduly ObjectStorage a TaskMetadata.

4.1.1 ObjectStorage submodul

Hlavní třídou tohoto submodulu je `ObjectService` zprostředkovávající operace s tzv. datovými objekty, které lze do aplikace nahrát. Může se jednat o obrázky, zvukové, či textové soubory. V momentální chvíli je podporováno ukládání objektů pouze pomocí jejich URL adresy (Uniform Resource Locator) – samotný objekt se tedy neukládá, uloží se pouze odkaz na něj. V rámci této práce je implementováno ukládání také do lokálního souborového systému (viz kapitolu 5.2). V budoucnu lze aplikaci jednoduše rozšířit o další typy úložišť pomocí implementace rozhraní `FileRepository`.

K jednoznačné identifikaci datového objektu slouží vygenerované UUID (universally unique identifier), které je součástí třídy `ObjectStorageInfo` spolu s informacemi o uživateli, jenž objekt nahrál, cestě k objektu (URL) a typu použitého úložiště (URL typ). Veškeré tyto informace jsou perzistentně uloženy v Mongo databázi.

4.1.2 TaskMetadata submodul

Struktura TaskMetadata submodulu je podobná jako u ObjectStorage submodulu. Ústřední třídou je `ObjectMetadataService` manipulující s metadaty datových objektů. Ta jsou zapouzdřena ve třídě `ObjectMetadata` obsahující identifikátor, uživatele, který objekt nahrál, typ objektu (obrázek / zvukový soubor / textový soubor), uživatelem definované štítky, datovou strukturu zachycující průběh anotace a mapu rozhraní `OtherMetadataType` umožňující ukládání dalších libovolných informací. Opět je vše uloženo v Mongo databázi, přičemž se zde využívá jejího volného databázového schématu.

Důležitým konceptem při anotaci dat v aplikaci CzechCaptcha je *label group* – skupina značek. Tyto skupiny definují uživatelé používající aplikaci k anotaci svých dat či zabezpečení webových stránek. Mají sloužit k upřesnění procesu anotace a odlišení značek s různým významem. Každá značka je tedy definována unikátním názvem skupiny, do které patří, a svým názvem.

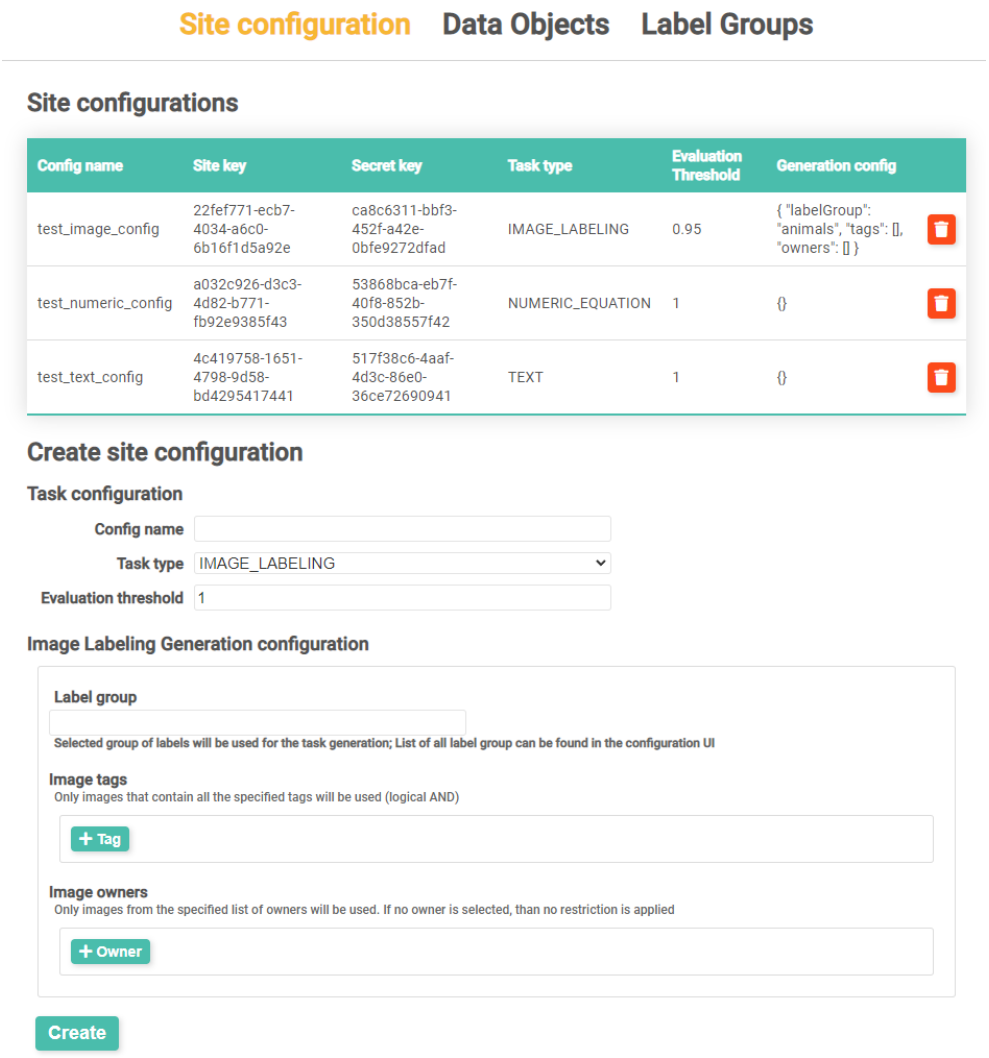
Samotná anotace probíhá stejně pro obrázky i jiné typy datových objektů. Pro daný objekt a danou label group je zachycena pomocí třídy `Labeling`. Ta obsahuje binární

indikátor, zda už je anotace hotová, seznam kladných značek (např. předmětů nacházejících se na obrázku), seznam záporných značek (např. předmětů nepřítomných na obrázku) a statistiky pro zatím nerozhodnuté značky.

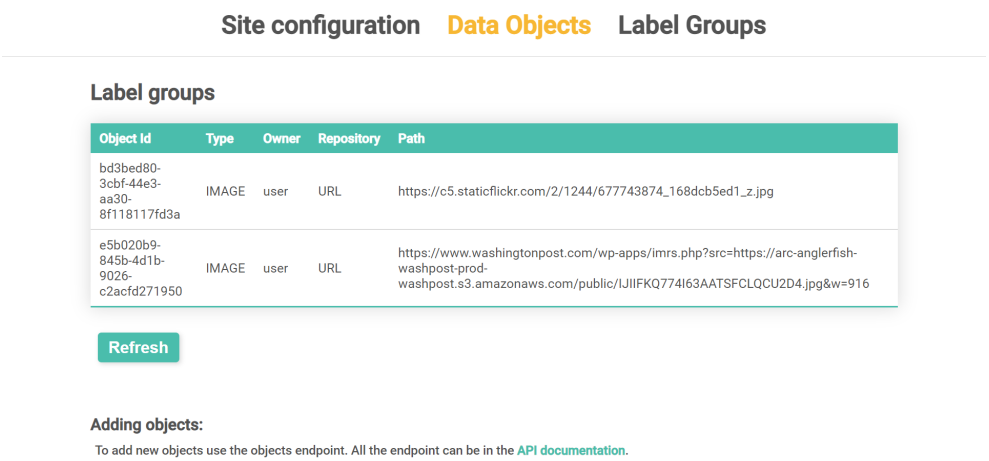
Pro přiřazení značky mezi kladné, resp. záporné, je potřeba dosáhnout „skóre“ 3, resp. -3. Skóre začíná na 0. Pokud uživatel v CAPTCHA úloze označí, že značka charakterizuje daný objekt, přičte se ke skóre 1, v opačném případě -1. Pro případ, že by nebylo jasné, zda značka objekt charakterizuje či nikoli, existuje ještě podmínka říkající, že lze skóre změnit maximálně devětkrát. Pokud ani po devíti změnách není skóre 3, nebo -3, je daná značka označena jako nerozhodnutelná.

4.2 Webová aplikace

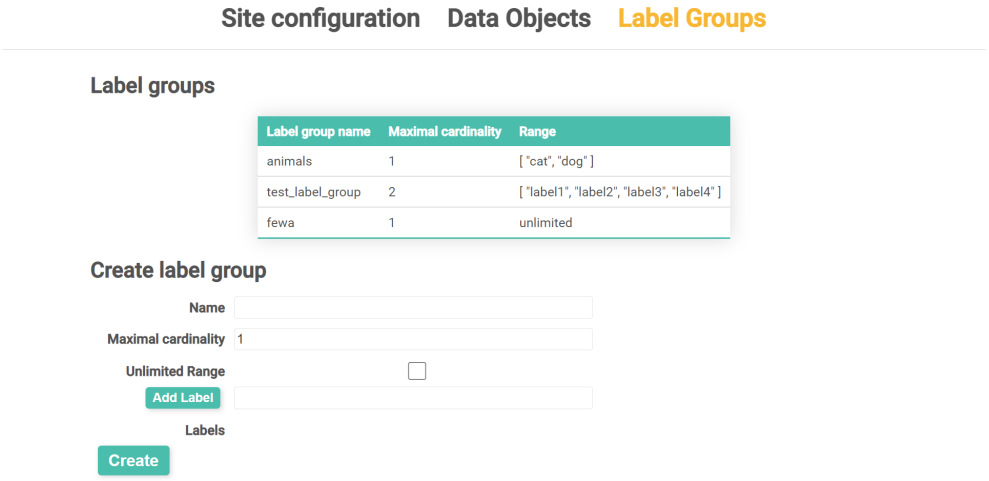
Klientská aplikace CzechCaptcha po přihlášení umožňuje skrze strohé grafické uživatelské rozhraní (User Interface – UI) základní práci s entitami. Konkrétně lze zobrazit, vytvářet a mazat konfigurace CAPTCHA úloh pro jednotlivé webové stránky (obrázek 4.2), zobrazit datové objekty (obrázek 4.3) a zobrazit a vytvářet skupiny značek (obrázek 4.4).



Obrázek 4.2. Screenshot webové aplikace CzechCaptcha zachycující uživatelské rozhraní sloužící ke konfiguraci CAPTCHA úloh pro další webové stránky



Obrázek 4.3. Screenshot webové aplikace CzechCaptcha zachycující uživatelské rozhraní sloužící k zobrazení datových objektů

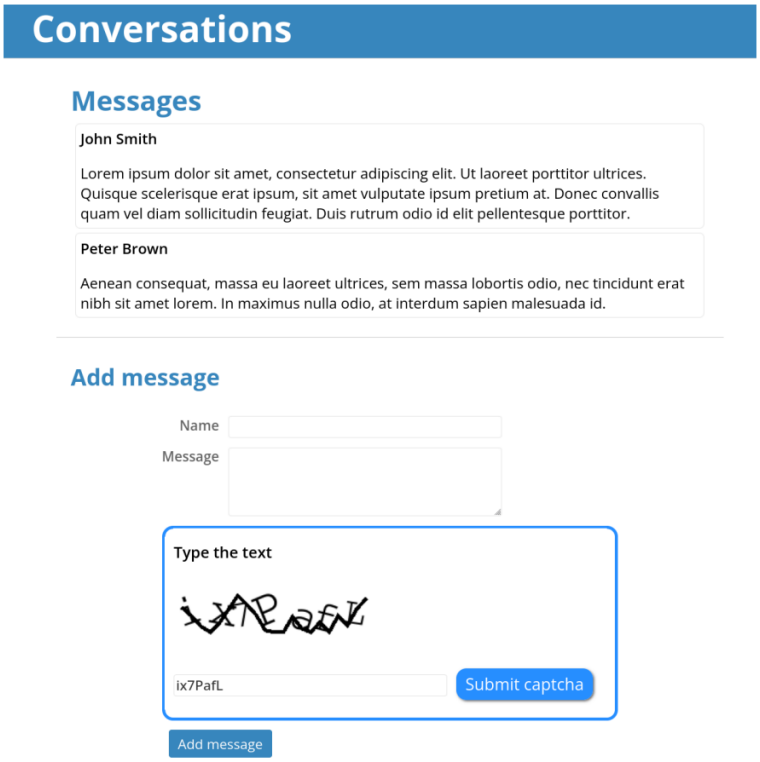


Obrázek 4.4. Screenshot webové aplikace CzechCaptcha zachycující uživatelské rozhraní sloužící k zobrazení a vytváření skupin značek

Klientská aplikace je zprostředkovaná pomocí Spring Boot a Spring Security jako statický obsah ve složce `/resources/static`. Po spuštění serverové aplikace je přístupná na adrese localhost:8080. Napsaná je v programovacím jazyce JavaScript. Využívá framework Vue pro vytváření uživatelského rozhraní a Axios klienta k realizaci HTTP požadavků na server.

V rámci diplomové práce vznikla také jednoduchá webová aplikace pro posílání zpráv prezentující zabezpečení pomocí CzechCaptcha úlohy (viz obrázek 4.5). Nachází se ve vlastním Git repozitáři[2] a obsahuje ukázkovou knihovnu v jazyce JavaScript pro integraci ověření do HTML formuláře. Knihovna obstarává získání a zobrazení CAPTCHA úlohy, odeslání řešení a předání tokenu s vyhodnocením.

[2] `https://github.com/OtakarVinklar/captcha-exampleApp`



Obrázek 4.5. Screenshot ukázkové aplikace využívající textovou CzechCaptcha úlohu k ověření uživatele před odesláním zprávy [3]

Kapitola 5
Design a implementace modulu importu dat

Tato kapitola se zabývá přínosy této bakalářské práce v podobě vylepšení modulu importu dat do aplikace CzechCaptcha. Představuje také novou CAPTCHA úlohu pro detekci objektů v obrázku.

Již v první verzi bylo možné aplikaci použít jako nástroj ke klasifikaci obrázků (přiřazení značek). Kombinace dvou zmíněných novinek přidává možnost použití také k detekci objektů v obrázcích. Nová funkcionalita je určena pro uživatele, jenž mají obrázky s objekty, o kterých chtějí kromě jejich názvů znát také jejich pozici v obrázku, a zároveň chtějí mít tyto informace potvrzené od skutečných lidí. Hlavním příkladem využití těchto informací jsou trénovací data pro modely detekující objekty.

5.1 Zakomponování detekce objektů do aplikace

Tato kapitola popisuje obecný pohled na proces nahrání, zpracování a uložení obrázků, ve kterých uživatel chce detekovat nějaké objekty. Následující kapitoly pak popisují jednotlivé komponenty a jejich fungování více do detailu.

5.1.1 Nahrání obrázků pro detekci objektů

Pro nahrávání obrázků určených k detekci objektů byly vytvořeny dva nové endpointy, každý pro jeden typ nahrávaného souboru:

- `POST api/datamanagement/objects/image/url` pro obrázky dostupné na nějaké URL adrese,
- `POST api/datamanagement/objects/image/file` pro obrázky nahrávané z lokálního úložiště.

V těle HTTP požadavku je nutné uvést obecné informace o souboru stejně jako při nahrávání jiného datovém objektu. Dále lze volitelně přidat parametry detekce objektů. Ty obsahují množinu všech názvů objektů (značek), které chce uživatel na obrázku detekovat. Je nutné je specifikovat, neboť detekování (případně označení nepřítomnosti) všech možných typů objektů by bylo velmi časově náročné a ve většině případů také zbytečné. Parametry obsahují ještě hodnoty `thresholdOneVote` a `thresholdTwoVotes`, jejichž význam je vysvětlen dále. V těle požadavku je též možné volitelně poskytnout informace o již detekovaných objektech – značky, jejich skupiny a ohraničující obdélníky.

K nahrání obrázku a veškerých dalších potřebných dat lze využít nové uživatelské rozhraní aplikace CzechCaptcha popsané v kapitole 5.5. Požadavky lze také vytvořit „ručně“ a odeslat je například pomocí nástroje curl nebo Postman.

5.1.2 Zpracování obrázků pro detekci objektů

Po nahrání jsou obrázky zpracovány následujícím způsobem. Do aplikace byl přidán modul pro detekci objektů, více popsaný v kapitole 5.3, který obsahuje model detekující objekty. Pro každý typ objektu (značku), který chce uživatel detekovat, může nastat jedna ze dvou situací.

1. Vestavěný model umí daný typ objektu detekovat (skupina značek je *object_detection*, viz 5.3.1), takže tak učiní. Na základě jeho výstupů jsou z obrázku vytvořeny výřezy detekovaných objektů, jež jsou uloženy způsobem popsaným v kapitole 5.2 jako standardní datové objekty určené k použití ve výběrové obrázkové CAPTCHA úloze. Obsahují pouze pár informací navíc. První z nich je identifikátor původního obrázku. Obdobně původní obrázek obsahuje data o jeho výřezech, aby bylo možné je navzájem jednoduše dohledat.

   Další informací navíc jsou hlasy přidané typu objektu, jež obsahují, na základě automatické detekce, podobně jako to dělala aplikace reCAPTCHA. K určení počtu přidaných hlasů slouží parametry `thresholdOneVote` a `thresholdTwoVotes` nahrané společně s obrázkem a pravděpodobnost detekovaných objektů určená modelem. Pokud je pravděpodobnost vyšší než hodnota `thresholdOneVote`, resp. `thresholdTwoVotes`, je přidán jeden, resp. dva hlasy.

   Vždy tedy minimálně jeden reálný uživatel musí v úloze potvrdit, že se na výřezu skutečně daný objekt nachází, neboť model je náchylný k chybám. Například u kozy je vysoká pravděpodobnost, že bude detekována jako pes.
2. Vestavěný model neumí daný typ objektu detekovat. K obrázku je uložena informace, že daný typ objektu má být detekovaný pomocí CAPTCHA úlohy pro detekci objektů popsané v kapitole 5.4.

Po zpracování detekovaných typů objektů jsou k obrázku přidány poskytnuté anotace (již detekované objekty), které jsou považovány za pravdivé a finální, takže jejich detekce pomocí CAPTCHA úlohy neprobíhá. Tato možnost slouží primárně při nahrávání obrázků sloužících k ověření, zda uživatel je člověk, pomocí CAPTCHA úlohy pro detekci objektů (viz kapitolu 5.4.1). Aby mohla být vygenerována úloha pro detekci určitého typu objektu, musí totiž existovat alespoň jeden obrázek s dokončenou detekcí pro tento typ. Načtení informací o detekovaném objektu ze souboru s anotacemi se prování již ve webové aplikaci CzechCaptcha a je popsáno v kapitole 5.5.1.

Nakonec jsou nahrané a zpracované obrázky uloženy a připraveny k použití ve výběrové obrázkové CAPTCHA úloze a případně také v úloze pro detekci objektů.

5.1.3 Uchovávání informací o detekci objektů

K uchování veškerých informací týkajících se detekce objektů je využit atribut `otherMetadata`, originálně pojmenovaný jako `templateData`, třídy `ObjectMetadata` zmíněný v kapitole 4.1.2. Ten slouží k ukládání libovolných doplňujících informací o objektu. Pod klíčem *object-detecting* se v něm nachází instance třídy `ObjectsDetectionData`. Pro každou značku, kterou nahrávající uživatel specifikoval, že chce detekovat, obsahuje stav její detekce v podobě třídy `ObjectDetectionData`, jež je značně podobná třídě `Labeling`, zmíněné opět v kapitole 4.1.2. Obsahuje seznam reálných ohraničujících obdélníků a seznam ohraničujících obdélníků zadaných uživateli vyplňujícími CAPTCHA úlohu pro detekci objektů.

5.2 Ukládání datových objektů

Ukládání nových datových objektů je v původní diplomové práci [3] uvedeno jako jeden z budoucích cílů vývoje aplikace. Je užitečné pro ukládání jednak nových datových objektů, ale také výřezů obrázků, které jsou výstupem algoritmu pro detekci objektů. Při implementaci ukládání by bylo možné využít tyto možnosti:

- Ukládat datové objekty někde tak, aby byly přístupné pomocí URL, a následně je nahrát do aplikace pomocí třídy `UrlObjectRepository`, původně pojmenované `UrlFileRepository`, spravující URL datové objekty. Úložiště by mohlo být na nějakém vlastním serveru, či cloudovém objektovém úložišti jako je Amazon S3, Azure Blob Storage nebo Google Cloud Storage. To by ovšem znamenalo úložiště zřídit, správně nakonfigurovat a případně také platit. Dalším problémem by byla správa těchto souborů, neboť třída `UrlObjectRepository` si pouze ukládá, a případně maže, záznamy s URL adresami, ale práci se samotnými soubory nepodporuje.
- Ukládat datové objekty přímo do Mongo databáze jako datový typ `BinData`. Mongo dokumenty (záznamy) jsou ovšem limitovány maximální velikostí 16 MB. Pro větší objem dat je možné využit rozšíření GridFS, které velké soubory rozdělí a uloží do menších dokumentů. Následně umožňuje přístup jak k celému původnímu souboru, tak i k jednotlivým rozděleným částem. [66]
- Ukládat datové objekty na lokální souborový systém. V případě provozu několika instancí aplikace CzechCaptcha, by bylo nutné vyřešit správný přístup všech instancí ke všem datům formou nějakého sdíleného souborového systému.

Nakonec byla zvolena poslední možnost ukládání datových souborů, lokální souborový systém, díky své přímočaré implementaci.

5.2.1 Ukládání do lokálního úložiště

Rozhraní `ObjectRepository`, původně pojmenované jako `FileRepository` s metodami `getFile`, `saveFile` a `removeFile` umožňuje jednoduché rozšíření o další typy úložišť a jejich vzájemnou kombinaci. Komunikaci s lokálním souborovým systémem zajišťuje nová třída `FilesystemObjectRepository` implementující právě toto rozhraní. Soubory jsou ukládány do adresáře specifikovaného v konfiguračním souboru. Výchozí hodnota je nastavena na `./programs/czech-captcha/saved-files`.

V případě, že nahrávaný soubor je obrázek, je zkontrolována jeho velikost. Pokud některý jeho rozměr v pixelech přesahuje maximální hodnotu nastavenou v konfiguračním souboru (výchozí hodnota je 1024), je obrázek proporcionálně zmenšen pomocí knihovny imgscalr tak, aby jeho větší strana měla právě tuto maximální velikost. Žádná z CAPTCHA úloh momentálně nepotřebuje obrázky ve velkém rozlišení a tímto způsobem se šetří úložištěm.

Nahrávání a ukládání nových datových objektů je klientům zprostředkované skrze nově přidaný HTTP endpoint `POST api/datamanagement/objects/file`, který v těle požadavku vyžaduje data typu `multipart/form-data` obsahující nahrávaný soubor typu `MultipartFile` a třídu `FileObjectCreateDTO` s informacemi o datovém objektu. Pro objekt je vygenerováno UUID, které ho jednoznačně identifikuje v rámci celé aplikace. Soubor je uložen s názvem stejným jako toto UUID a původní příponou, což řeší problematiku různě dlouhých názvů souborů při ukládání s vyvažováním adresářů popsaném níže. Originální název souboru je ukládán jako nový atribut třídy `ObjectStorageInfo`.

5.2.1.1 Ukládání s vyvažováním adresářů

Lze předpokládat, že množství uložených souborů bude v produkčním prostředí dosahovat řádů deseti tisíců i více. V závislosti na použitém souborovém systému by v případě ukládání všech souborů do jednoho adresáře pak mohl nastat problém s pomalým přístupem k datům. Třída `FilesystemObjectRepository` mu předchází pomocí vyvažování počtu souborů v adresářích.

Pokud se ukládá nový soubor a počet souborů uložených v adresáři by přesáhl maximální hranici určenou v konfiguraci aplikace (výchozí hodnota 10 000), vytvoří se

podadresář s jednoznakovým názvem shodným s prvním písmenem ukládaného souboru. Soubor se uloží do tohoto nového podadresáře a následně jsou všechny soubory z původního adresáře začínající na stejný znak přesunuty také do podadresáře. Tento algoritmus je následně rekurzivně uplatněn také na podadresáře.

Přístup k souboru funguje analogicky, nejprve se zkontroluje, zda již neexistuje podadresář se stejným jménem jako první písmeno hledaného souboru, pokud ne, hledá se v daném adresáři, pokud ano, hledá se rekurzivně v podadresáři.

5.3 Modul pro detekci objektů

Pro detekci objektů na obrázcích byl do modulu DataManagement přidán nový submodul ObjectDetection (balíček `objectdetection`).

5.3.1 Model pro detekci objektů

Aby bylo možné k detekci objektů používat různé modely, bylo vytvořeno rozhraní `ObjectDetector` s následujícími metodami:

- `getSupportedLabels` vrací množinu objektů, které umí model rozpoznat.
- `detect` vrací množinu objektů detekovaných na obrázku poskytnutém jako parametr zachycených pomocí instancí třídy `DetectedObject` obsahující jejich názvy (label), pravděpodobnosti a ohraničující obdélníky.

Jedinou třídou implementující toto rozhraní je `ObjectDetectorKotlinDL`. Z názvu je patrné, že využívá framework KotlinDL pro detekci objektů popsaný v kapitole 3.2.4, konkrétně jeho verzi 0.4.0. Ten byl vybrán na základě jednoduché integrace a možnosti využití již natrénovaných modelů. Toho aplikace využívá a po spuštění zkontroluje přítomnost předtrénovaného modelu v adresáři definovaném v konfiguračním souboru a popřípadě ho do něj automaticky stáhne. Jako výchozí adresář je nastaven `./programs/czech-captcha/od-cache`.

Jako samotný model byl zvolen EfficientDet-D4. Jeho rychlost i přesnost je srovnatelná s ostatními aktuálními modely. Verze D4 byla vybrána, neboť velikost vstupních obrázků je 1024×1024 pixelů, což je i maximální velikost ukládaná v aplikaci (viz kapitolu 5.2.1). V případě, že jsou obrázky menší, či nemají poměr stran 1:1, jsou proporcionálně zvětšeny tak, aby delší strana byla přesně 1024 pixelů a kratší strana je doplněna o černý obdélník s příslušnou velikostí. Model je předtrénovaný na COCO datasetu, takže rozpoznává 90 různých objektů vyskytujících se v tomto datasetu. Všechny značky podporovaných objektů se nachází ve skupině značek *object_detection*.

5.3.2 Object Detection Service

Hlavní třídou komunikující s dalšími moduly je `ObjectDetectionService`, která kromě zprostředkování výstupů modelu pro detekci objektů umí také odhalit překryvy objektů detekovaných na jednom obrázku.

Třída `DetectedObjectWithOverlappingLabels` obsahuje ohraničující obdélník, název a pravděpodobnost původního detekovaného objektu, ale také názvy a pravděpodobnosti objektů, jejichž ohraničující obdélník má nenulový průnik s obdélníkem původního objektu. Originální pravděpodobnost těchto překrývajících objektů je vynásobena podílem průniku obdélníků a obsahu původního obdélníku, což kompenzuje to, že objekt nemusí být z jeho části zcela zřetelný. V případě, že se překrývají objekty stejného druhu, je brán v potaz pouze ten s vyšší pravděpodobností.

Tato funkcionalita je vhodná pro urychlení následného anotování jednotlivých výřezů detekovaných objektů.

5.4 CAPTCHA úloha pro detekci objektů

Uživateli jsou v této úloze postupně prezentovány dva obrázky a jeho úkolem je pomocí obdélníků na nich označit všechny objekty odpovídající danému popisu (názvu), nebo případně říct, že na daném obrázku se žádné takové objekty nenachází. Pozice objektů na jednom z obrázků je již známá z anotací nahraných společně s obrázkem nebo z dřívějších výsledků této úlohy. Porovnání známých obdélníků s odpovědí uživatele slouží k ověření, zda se jedná o člověka. Pozice objektů na druhém obrázku (uživateli může být prezentován i jako první) naopak známé nejsou. Pokud uživatel u známého obrázku odpověděl dostatečně přesně, je jeho odpověď zaznamenána u neznámého obrázku pro budoucí vytvoření anotace.

5.4.1 Ověření uživatele

Pro ověření, zda uživatel je člověk, je potřeba z odpovědi a správných hodnot nějakým způsobem vypočíst skóre mezi 0 a 1. V konfiguraci CAPTCHA úlohy je pak možné určit hranici tohoto skóre, které je nutné dosáhnout pro projití testem.

Jak již bylo zmíněno, ověřování probíhá pomocí obrázku, u kterého jsou obdélníky ohraničující objekty známé. Pro jejich porovnání s těmi z odpovědi uživatele je nejprve nutné vytvořit dvojice, které k sobě patří, neboť pořadí obdélníků v odpovědi může být libovolné. Skutečné objekty se mohou různě překrývat a nemusí tak být zcela jednoznačné, které obdélníky ohraničují stejný objekt. Lze předpokládat, že obdélník z odpovědi bude odpovídat tomu reálnému obdélníku, se kterým se nejvíce překrývá. K vytvoření dvojic tedy bylo použito upravené aglomerativní hierarchické shlukování (základní varianta je popsaná v kapitole 3.3). Použitá vzdálenostní funkce je popsaná v následující podkapitole.

Na začátku vyhodnocení správnosti odpovědi je spočítána vzdálenost každého reálného obdélníku od všech obdélníků v odpovědi, celkem tedy počet reálných obdélníků krát počet obdélníků v odpovědi vzdáleností. Tyto vzdálenosti jsou seřazeny od nejmenší k největší a postupně se prochází. Pokud alespoň jeden z obdélníků, mezi nimiž byla vypočítána aktuální vzdálenost, už je součástí nějaké dvojice, vzdálenost se přeskočí. V opačném případě se z obdélníku vytvoří dvojice. Pro vzdálenosti větší než 0,8 (IoU menší nebo rovno 0,2), už se dvojice nevytváří, neboť je nepravděpodobné, že by oba obdélníky měly ohraničovat stejný objekt. Díky tomuto opatření a „vyřazování“ obdélníků vytvářením dvojic nevznikne na konci algoritmu jedna skupina obsahující všechny obdélníky, jak tomu je u obecného aglomerativního hierarchického shlukování, nýbrž vznikne až n skupin (dvojic, či samostatných obdélníků), kde n je součet správných obdélníků a obdélníků v odpovědi (žádný obdélník z odpovědi by se neprotínal s těmi skutečnými).

Po vytvoření dvojic se pro každou z nich spočítá IoU. Pokud obdélník nemá dvojici (uživatel neoznačil nějaký objekt nebo naopak nějaký označil navíc), jeho průnik s neexistujícím obdélníkem je nulový a sjednocení je rovno obsahu daného obdélníku – IoU je tedy také nulové. Pro výpočet finálního skóre by bylo možné vzít všechna tato IoU a spočítat jejich průměr, který triviálně bude mezi 0 a 1. Jelikož ovšem IoU charakterizuje relativní překryv, bude například neoznačení malého objektu v pozadí penalizováno stejným způsobem jako neoznačení velkého objektu v popředí. Proto bylo pro výpočet finálního skóre využito „akumulované“ IoU, které je rovno podílu součtu všech průniků a součtu všech sjednocení jednotlivých dvojic obdélníků, jehož hodnota se také nachází mezi 0 a 1.

V praxi je pro robota, ale i člověka, téměř nemožné získat skóre blízké nebo rovné 1. Je nutné o tomto faktu informovat uživatele při vytváření konfigurace pro tuto úlohu, aby adekvátně nastavili hranici pro projití testem.

5.4.1.1 Vzdálenostní funkce pro tvorbu dvojic obdélníků

Použitá vzdálenostní funkce je založená na IoU (průnik nad sjednocením) popsaném v kapitole 3.2.2. Jedná se o metriku dle její definice, neboť splňuje všechny axiomy zmíněné v kapitole 3.3. [67] Její předpis je následující:

$$d(A, B) = 1 - \frac{A \cap B}{A \cup B}$$

kde A, B jsou porovnávané ohraničující obdélníky.

Obsah průniku obdélníků lze spočítat jako součin překryvu jejich šířky a výšky. Formálně:

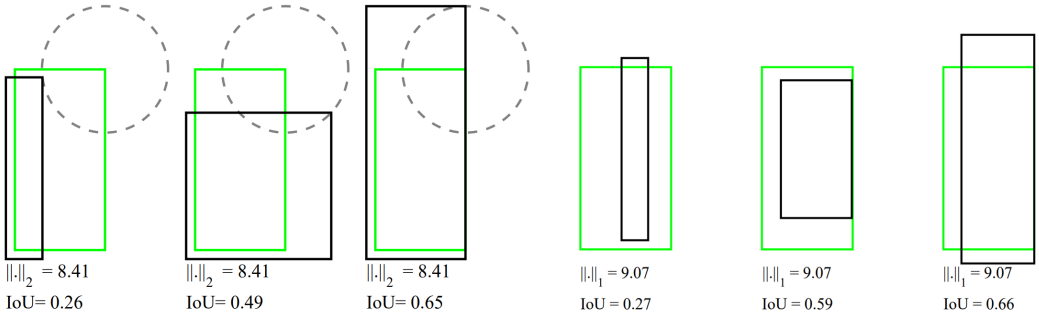
$$A \cap B = \max(0, \min(x_A + w_A, x_B + w_B) - \max(x_A, x_B)) \cdot \\ \cdot \max(0, \min(y_A + h_A, y_B + h_B) - \max(y_A, y_B))$$

kde x_A, y_A značí x a y souřadnici levého horního vrcholu obdélníku A a w_A, h_A značí jeho šířku a výšku.

Obsah sjednocení obdélníků lze následně spočítat jednoduše jako součet jejich obsahů mínus jejich průnik. Formálně:

$$A \cup B = w_A \cdot h_A + w_B \cdot h_B - A \cap B$$

Dalšími kandidáty na vzdálenostní funkci byly Manhattanská a Euklidovská vzdálenost mezi levými horními a pravými dolními vrcholy obdélníků, či mezi jejich středy a šířkami a výškami. Na obrázku 5.1 lze vidět, že stejná hodnota těchto funkcí může charakterizovat několik velmi odlišných předpovědí s různými IoU. Dále také berou v potaz pouze absolutní velikost obdélníků, takže pro dvojice obdélníků se stejným překryvem (IoU), ale jinou velikostí, mají různé hodnoty. Z těchto důvodů nejsou vhodné jako vzdálenostní funkce pro vytváření dvojic obdélníků.



Obrázek 5.1. Porovnání hodnot IoU a Manhattanské, resp. Euklidovské vzdálenosti na dvou trojicích obdélníků v levé, resp. pravé části obrázku [68]

5.4.2 Výpočet finálních pozic objektů z odpovědí

V případě, že odpověď při ověření uživatele dostane skóre vyšší, než je hranice definovaná v konfiguračním souboru (výchozí hodnota je 0,6), uloží se jeho odpověď s obdélníky ohraničujícími objekty na neznámém obrázku. Po dosažení určitého počtu uložených odpovědí, opět definovaného v konfiguračním souboru (výchozí počet je 10), se vypočítají finální ohraničující obdélníky, které se následně budou považovat za reálné.

Při výpočtu finálních obdélníků je, podobně jako při výpočtu skóre uživatele, nejprve nutné vytvořit skupiny obdélníků ohraničující stejné objekty. Jednotlivé odpovědi mohou obsahovat různý počet obdélníků v různém pořadí. Pro vytvoření skupin se proto opět používá upravené aglomerativní hierarchické shlukování popsané v kapitole 5.4.1. Jediným rozdílem je, že místo dvojic se vytváří skupiny o velikosti až v, kde v je celkový počet uložených uživatelských odpovědí. Je zaručeno, že v jedné skupině se nebude nacházet více obdélníků z jedné odpovědi, protože se předpokládá, že uživatel neoznačí jeden objekt vícekrát. Z popisu použitého algoritmu plyne, že pro shlukování je použita metoda nejbližšího souseda. Podobně jako v algoritmu pro slučování dvojic nevznikne na konci shlukování jedna velká skupina, nýbrž vznikne až n skupin, kde n je celkový počet obdélníků ve všech zaznamenaných odpovědích.

Pokud takto vytvořená skupina obsahuje alespoň $\lceil \frac{n}{2} \rceil$ obdélníků, jsou jejich pozice a velikosti zprůměrovány do jednoho obdélníku, který je označen za finální.

5.4.3 Bezpečnost úlohy

Tato CAPTCHA úloha je bezpečná, neboť hledané objekty jsou z mnoha různých oblastí, které definují uživatelé nahrávající data. Naopak se v úloze nevyskytují objekty z nejčastějších kategorií jako jsou například auta, lidé či psi, protože tyto objekty jsou při nahrání dat detekovány modulem pro detekci objektů. Bylo by tedy obtížné vytvořit model pro detekci objektů rozeznávající velké množství netriviálních a různorodých kategorií nacházejících se v této CzechCaptcha úloze.

5.4.4 Technické provedení

O veškerou funkcionalitu CAPTCHA úlohy pro detekci objektů se stará třída `ObjectDetectingTemplate` implementující rozhraní `TaskTemplate` popsané v kapitole 4.1. Pro práci s metadaty datových objektů využívá třídu `ObjectMetadataService` a pro získání obrázků, které jsou posílány klientovi v base64 kódování, používá třídu `ObjectService`.

5.5 Webové uživatelské rozhraní

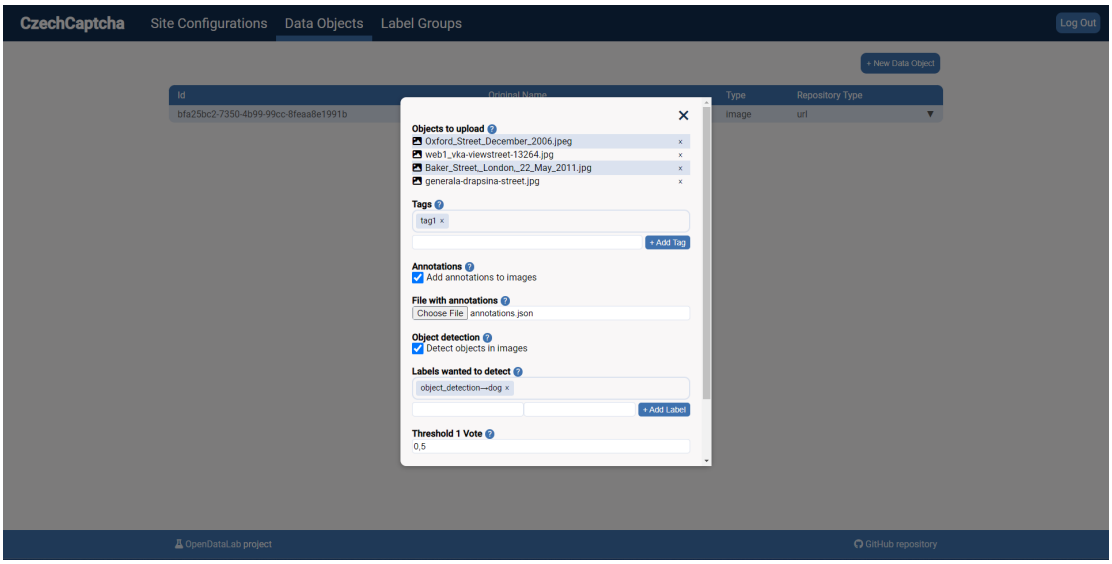
K nahrávání datových objektů do aplikace CzechCaptcha lze nově využít webového uživatelského rozhraní. Uživatel najede myší na tlačítko *+ New Data Object* a zvolí zdroj objektu – URL, soubor, či složku s více soubory. Následně je mu zobrazen dialog pro výběr souboru, či složky z jeho lokálního úložiště, nebo okno s možností zadání URL adresy. V modálním okně (obrázek 5.2) jsou poté vypsány všechny soubory připravené k nahraní. Lze zde také ke všem souborům přidat štítky. V případě, že alespoň jeden nahrávaný soubor je obrázek, je možné nahrát soubor s anotacemi (viz kapitolu 5.5.1) a označit obrázky k detekci objektů společně s nastavením příslušných parametrů. Kategorie objektů určené k detekci se zadávají pomocí kombinace skupiny značek (label group) a samotné značky (label). Veškerá vstupní pole jsou opatřena ikonou s nápovědou, která po najetí myší ukáže detailnější popis. Nahrání objektů se potvrdí pomocí tlačítka *Add Data Objects*, čímž se zavře i modální okno. Jelikož detekce objektů v serverové části aplikace může trvat několik sekund, jsou v této době nahrávané objekty zobrazeny v tabulce všech objektů s označením *uploading*. Jednotlivé řádky této tabulky reprezentují datové objekty v aplikaci a obsahují základní informace o nich. Všechna podrobnější data se zobrazí po rozbalení řádku. Tabulku lze vidět na obrázku 5.3.

Spolu s nahráváním objektů byl upraven design i ostatních stránek v uživatelském rozhraní pro správu konfigurací CAPTCHA úloh pro jednotlivé webové stránky (Site

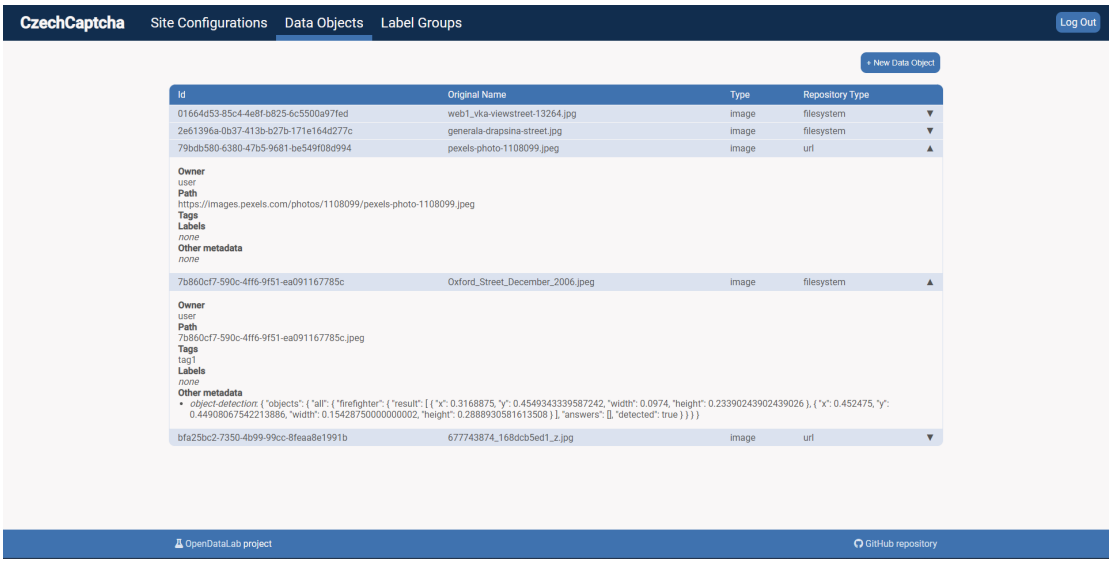
Configurations) a pro správu skupin značek (Label Groups) – viz obrázky 5.4 a 5.5. Způsob zobrazení jednotlivých položek v tabulce a jejich vytváření pomocí modálního okna je obdobný jako u nahrávání datových objektů.

Dále byla přidána nová úvodní stránka přístupná i bez přihlášení obsahující krátké představení aplikace CzechCaptcha a jejích možností využití, kterou lze vidět na obrázku 5.6.

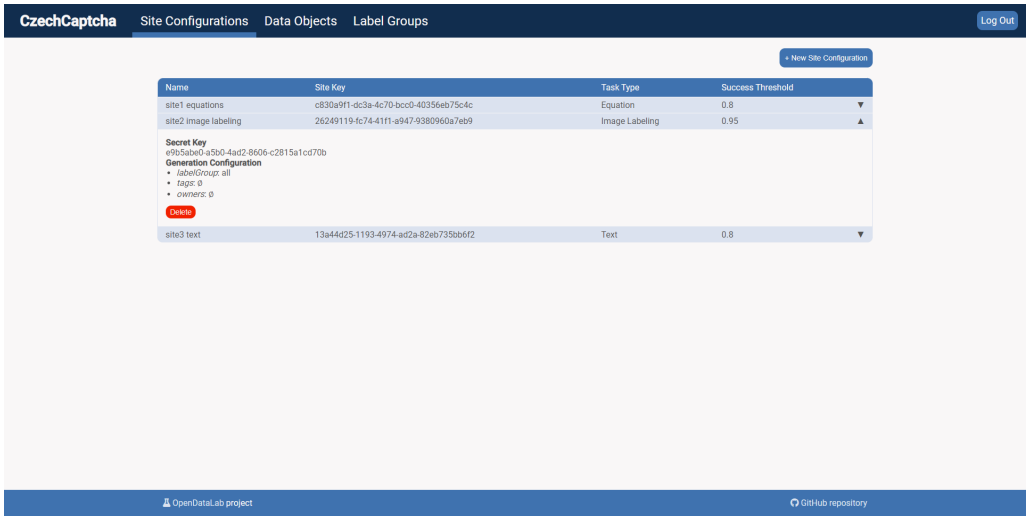
Design záhlaví stránek a modálního okna pro vytváření datových položek byl inspirován aplikací pro správu úkolů Habitica (viz obrázek 5.7). Hlavní barevné schéma se skládá tří odstínů modré a světle béžové barvy a působí klidným a seriózním dojmem.



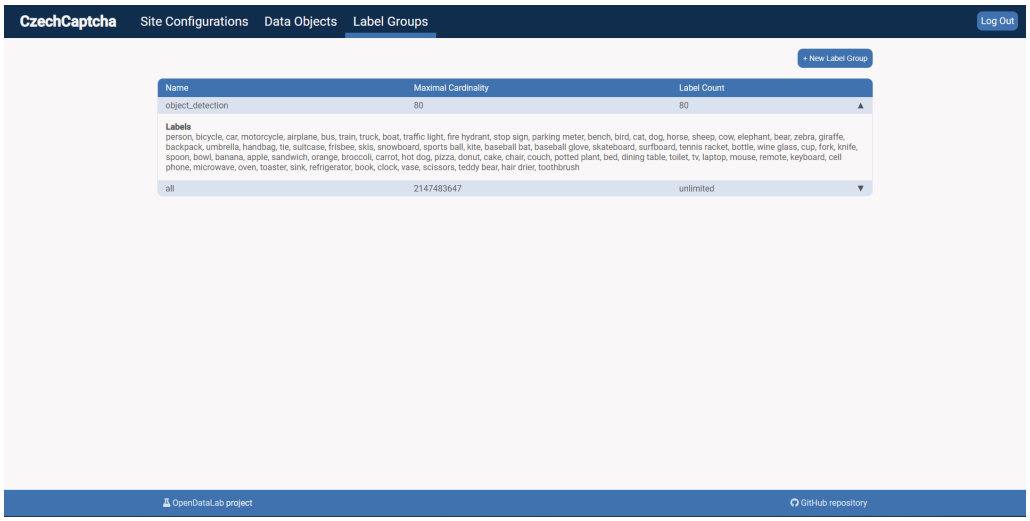
Obrázek 5.2. Screenshot nového modálního okna pro přidávání nových datových objektů



Obrázek 5.3. Screenshot nového uživatelského rozhraní pro správu datových objektů

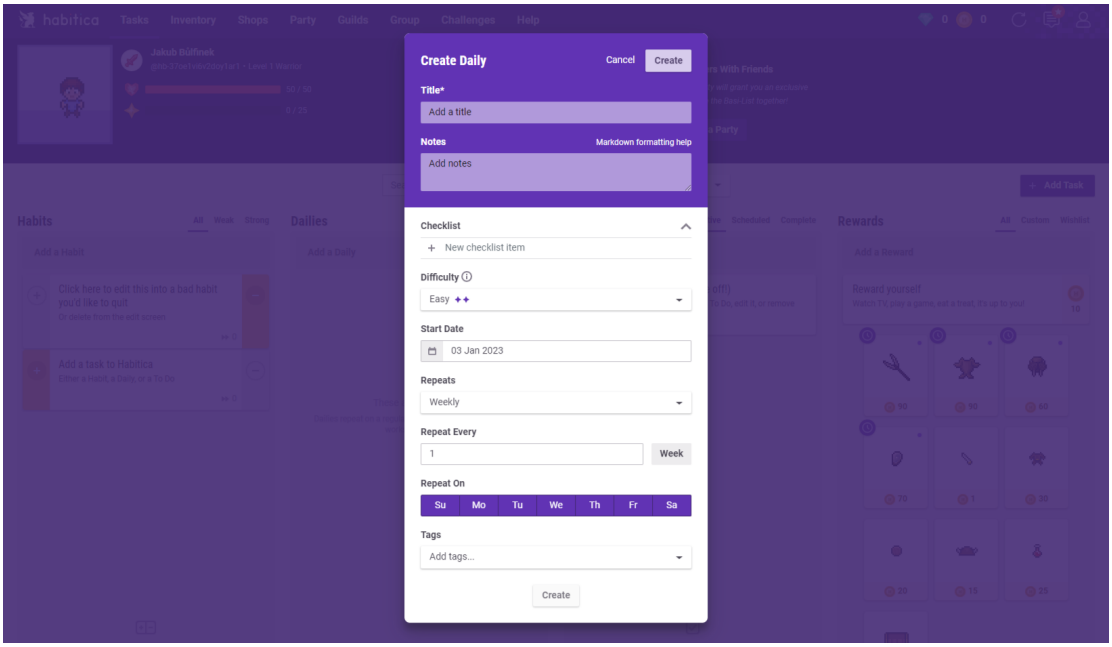


Obrázek 5.4. Screenshot nového uživatelského rozhraní pro správu konfigurací CAPTCHA úloh pro další webové stránky



Obrázek 5.5. Screenshot nového uživatelského rozhraní pro správu skupin značek

Obrázek 5.6. Screenshot nové úvodní stránky webového rozhraní CzechCaptcha



Obrázek 5.7. Screenshot webové aplikace Habitica, která sloužila jako inspirace při designování uživatelského rozhraní CzechCaptcha

5.5.1 Načtení anotací ze souboru

Obrázky do aplikace lze přidat spolu s anotacemi. Uživatel při nahrávání zvolí soubor s anotacemi ve formátu COCO popsaném v kapitole 3.2.5. Tento formát byl zvolen díky možnosti uvedení nadřazené kategorie (položka supercategory), která je v aplikaci CzechCaptcha interpretována jako skupina značek. V případě, že nadřazená kategorie není uvedena, nastaví se skupina značek na *all*, což je nově přidaná základní skupina, která může obsahovat libovolné množství libovolných značek.

Další výhodou COCO formátu je to, že se jedná o JSON soubor, takže je jednoduše převeden na JavaScript objekt a následně je pomocí JSON Schema a knihovny Ajv potvrzen správný formát anotací. Z tohoto objektu jsou získány veškeré potřebné informace. Anotace jsou přiřazeny k nahrávaným souborům na základě shody jmen souborů. Lze tedy nahrát anotace i k URL obrázkům, ovšem je nutné, aby název souboru uvedený v COCO formátu byl shodný s URL adresou obrázku. Soubory, ke kterým jsou přiřazeny nějaké anotace, jsou v seznamu nahrávaných souborů modře zvýrazněny. Během přiřazování jsou také absolutní souřadnice z COCO anotací převedeny na relativní pomocí rozměrů obrázku, též uvedených v COCO souboru.

5.6 Další úpravy aplikace

Během práce byly provedeny další malé úpravy aplikace CzechCaptcha. Mezi ně patří automatická detekce typu (obrázek, audio, atd.) a formátu souboru během jeho nahrávání, takže tyto údaje již není nutné zasílat v těle požadavku. Bylo odstraněno pole `user` ve třídě `ObjectStorageInfo`, neboť tato informace je duplicitně uložena také ve třídě `ObjectMetadata`. Bylo provedeno drobné refaktorování kódu včetně změny některých názvů, smazání nepoužívaného kódu a nahrazení seznamů (`List`) za množiny (`Set`) v opodstatněných případech.

Kapitola 6
Testování modulu

Tato kapitola představuje způsob, jakým byly testovány nové součásti aplikace CzechCaptcha popsané v kapitole 5.

Veškeré nově přidané třídy a metody jsou testovány pomocí automatických jednotkových (unit) testů. Pro jejich přípravu a porovnávání hodnot se využívá testovací framework JUnit 5. K vytváření mock objektů je použita knihovna Mockk speciálně napsaná pro programovací jazyk Kotlin.

Model pro detekci objektů je testován tak, že je mu prezentován kvalitní obrázek člověka na bílém pozadí a očekává se, že ho detekuje. Pravděpodobnost ani přesná pozice detekovaného člověka nejsou kontrolovány.

Funkcionalita nahrávání nových objektů je též testována pomocí automatických integračních testů. Pro jejich běh je kromě výše zmíněných knihoven použita také anotace `@SpringBootTest` z knihovny spring-boot-starter-test. Testovací Mongo databáze je vytvořena a spravována pomocí knihovny Testcontainers, která ji spustí v Docker prostředí.

Pokrytí řádků kódu testy (lines code coverage) určené pomocí nástroje IntelliJ IDEA code coverage runner (IntelliJ IDEA verze 2021.3.3 Ultimate Edition) je pro balíček `objectdetection` 93 %, pro balíček `objectmetadata` 92 % a pro balíček `objectstorage` 94 %. Balíček `objectdetectiontemplate` má pokrytí 96 %. Detailní rozpis pokrytí je zobrazen v tabulce 6.1.

Bylo dopsáno několik testů i ke třídám nesouvisejícím s novými funkcionalitami. Dále byla přidána testovací konfigurace `RepositoriesTestConfig` a anotace `SpringBootTestWithoutMongoDB`, takže testovací třídy vytvářející celý Spring kontext již nemusejí obsahovat všechny mock objekty, které přistupují k databázi.

Webové uživatelské rozhraní není momentálně automaticky testováno. Byly provedeny pouze manuální UI testy. K jejich provedení byla spuštěna instance Mongo databáze a aplikace CzechCaptcha. Následně byly provedeny všechny operace dostupné ve webové aplikaci.

Package	Class	Method	Line
/	66% (8/12)	80% (16/20)	75% (36/48)
configuration	100% (12/12)	100% (28/28)	100% (96/96)
datamanagement	100% (12/12)	72% (64/88)	77% (204/264)
dto	94% (64/68)	89% (68/76)	95% (168/176)
objectdetection	88% (32/36)	93% (112/120)	93% (388/416)
objectmetadata	94% (68/72)	89% (356/400)	92% (1088/1172)
objectstorage	100% (28/28)	96% (112/116)	94% (396/420)
initialization.mongock	100% (20/20)	66% (40/60)	89% (168/188)
siteconfig	100% (20/20)	81% (52/64)	89% (140/156)
dto	0% (0/12)	0% (0/12)	0% (0/12)
task			
taskconfig	100% (8/8)	75% (24/32)	81% (36/44)
templates	100% (12/12)	100% (32/32)	100% (44/44)
imagelabelingtemplate	66% (8/12)	20% (8/40)	15% (36/236)
objectdetectiontemplate	100% (20/20)	100% (88/88)	96% (632/656)
simplequationtemplate	100% (4/4)	100% (12/12)	100% (40/40)
texttemplate	100% (4/4)	100% (48/48)	97% (344/352)
user	66% (16/24)	69% (64/92)	69% (100/144)
verification	100% (32/32)	85% (72/84)	90% (188/208)
dto	71% (20/28)	71% (20/28)	83% (40/48)
entities	80% (64/80)	73% (68/92)	84% (128/152)

Tabulka 6.1. Pokrytí kódu testy v jednotlivých balíčcích určené pomocí nástroje IntelliJ IDEA code coverage runner (hlavní balíček `cz.opendatalab.captcha` je kvůli zkrácení ve všech řádcích vynechán, odsazení znamená podbalíček)

Kapitola 7
Závěr

V dnešní době se provozovatelé webových stránek a aplikací musí chránit před činností automatizovaných robotů, kteří se často snaží nějakým způsobem škodit. Jedním možným způsobem obrany jsou CAPTCHA systémy vyžadující před zpřístupněním zdrojů či operací splnění úkolu, jehož vyřešení je relativně jednoduché pro člověka, ale obtížné pro program. Nevýhodou tohoto řešení je velké množství času stráveného řešením těchto úloh. V současnosti se ovšem výstupy z úkolů kromě ověření uživatelů využívají také k přidávání anotací k datům, primárně obrázkům.

Aplikace CzechCaptcha je český open-source CAPTCHA systém, jehož první verze vznikla v rámci diplomové práce Ing. Otakara Vinkláře [3]. Je možné ji zdarma použít jak pro zabezpečení webových stránek, tak pro anotování dat určených například pro trénování hlubokých neuronových sítí.

Cílem této bakalářské práce bylo do aplikace CzechCaptcha přidat modul pro import dat, který bude umožňovat nahrát data ze zadaného zdroje a v případě, že se jedná o obrázky, je i následně předzpracuje pomocí algoritmu pro detekci objektů. Vhodně ořezané obrázky budou uloženy do aplikace a označeny k anotovaní lidmi skrze CAPTCHA úlohy. Spolu s obrázkem bude možné nahrát také již existující anotace a modul je během nahrávání zohlední. Druhým cílem bylo vytvoření uživatelsky přívětivého webového rozhraní pro nahrávání dat do systému.

Všechny cíle této bakalářské práce byly splněny. Bylo implementováno ukládání datových objektů (např. obrázků) do lokálního úložiště. Byl přidán modul pro detekci objektů obsahující model hlubokého učení detekující objekty. Uživatel při nahrávání obrázků zvolí, které objekty chce detekovat. Pokud je model podporuje, zkusí je detekovat. Na základě jeho výstupů jsou uloženy výřezy obrázku určené pro použití ve výběrové obrázkové CAPTCHA úloze. V případě, že model daný objekt detekovat neumí, je původní obrázek uložen a označen k použití v nové CAPTCHA úloze pro detekci objektů. V ní uživatel pomocí obdélníků označuje v kontrolním a neznámém obrázku požadované objekty. Po zaznamenání dostatečného počtu odpovědí, jsou všechny agregovány a jsou z nich vytvořeny finální pozice objektů. K nahrání objektů do aplikace je možné využít nové webové rozhraní vysvětlující veškeré parametry celého procesu. Zde je též možné nahrát soubor s anotacemi k obrázku a příslušné hodnoty jsou následně využity při vytváření kontrolních obrázků v nové CAPTCHA úloze.

Na tuto práci lze v budoucnu navázat v několika ohledech. Jak je již zmíněno v diplomové práci [3], je možné implementovat perzistentní ukládání tokenů a zadání úloh, či frontend knihovnu pro jednoduchou integraci CzechCaptcha ověření do webových stránek. Návrhy na rozšíření jsou též umožnění exportu anotací vytvořených prostřednictvím CAPTCHA úloh a zlepšení manipulace s daty a jejich vizualizace pomocí filtrování, vyhledávání a statistik. Další eventuální možností navázání je automatická úprava a trénink integrovaného modelu hlubokého učení s využitím transfer learning technik a dat nově anotovaných v CAPTCHA úloze pro detekci objektů.

Literatura

[1] Clement, J. *Global share of human and bot web traffic 2019–2020* [online]. 2021. [vid. 2022/06/22]. Dostupné na https://www.statista.com/statistics/1264226/human-and-bot-web-traffic-share/.

[2] Gugliotta, Guy. Deciphering Old Texts, One Woozy, Curvy Word at a Time. *The New York Times* [online]. 03, 2011. [vid. 2022/12/11]. ISSN 1553-8095. Dostupné na https://www.nytimes.com/2011/03/29/science/29recaptcha.html.

[3] Vinklář, Otakar. *Server-side application for CAPTCHA system* [online]. České vysoké učení technické v Praze, Fakulta informačních technologií, 2022 [vid. 2022/12/11]. Diplomová práce. Dostupné na http://hdl.handle.net/10467/101151.

[4] *The Official CAPTCHA Site* [online]. © 2000–2010. [vid. 2022/09/28]. Dostupné na http://www.captcha.net/.

[5] Reimer, Jeremy. *The most adorable spambot killer ever* [online]. 2006. [vid. 2022/10/02]. Dostupné na https://arstechnica.com/uncategorized/2006/04/6554-2/.

[6] TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind* [online]. 10, 1950, ročník LIX, č. 236, s. 433–460 [vid. 2022/09/28]. ISSN 0026-4423. Dostupné na DOI 10.1093/mind/LIX.236.433. Dostupné na https://doi.org/10.1093/mind/LIX.236.433.

[7] Ahn, Luis von, Manuel Blum a John Langford. Telling Humans and Computers Apart Automatically. *Communications of the ACM* [online]. New York, NY, USA: Association for Computing Machinery, 02, 2004, ročník 47, č. 2, s. 56–60 [vid. 2022/09/28]. ISSN 0001-0782. Dostupné na DOI 10.1145/966389.966390.

[8] Ahn, Luis von, Manuel Blum, Nicholas J. Hopper a John Langford. CAPTCHA: Using Hard AI Problems for Security. In: Eli Biham, editor. *Advances in Cryptology — EUROCRYPT 2003* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003 [vid. 2022/09/28]. s. 294–311. ISBN 978-3-540-39200-2.

[9] Goodfellow, Ian J., Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud a Vinay Shet. *Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks* [online]. [vid. 2022/09/28]. Dostupné na DOI 10.48550/ARXIV.1312.6082. Dostupné na https://arxiv.org/abs/1312.6082.

[10] *2Captcha: Captcha Solving Service, reCAPTCHA Recognition and Bypass, Fast Auto Anti Captcha* [online]. [vid. 2022/10/02]. Dostupné na https://2captcha.com/.

[11] *Anti Captcha: Captcha Solving Service. Bypass Recaptcha, FunCaptcha Arkose Labs, image captcha, GeeTest, HCaptcha.* [online]. [vid. 2022/10/02]. Dostupné na https://anti-captcha.com/.

[12] *EndCaptcha - Premium Captcha Service* [online]. [vid. 2022/10/02]. Dostupné na https://www.endcaptcha.com/.

[13] Guerar, Meriem, Luca Verderame, Mauro Migliardi, Francesco Palmieri a Alessio Merlo. Gotta CAPTCHA 'Em All: A Survey of 20 Years of the Human-or-Computer Dilemma. *ACM Computing Surveys* [online]. New York, NY, USA: Association for Computing Machinery, 2021, ročník 54, č. 9, s. 1–33 [vid. 2022/10/02]. ISSN 0360-0300. Dostupné na DOI 10.1145/3477142.

[14] *S CAPTCHA Help doplňkem o krok dál* [online]. 2013. [vid. 2022/10/04]. Dostupné na `https://blog.nic.cz/wp-content/uploads/2013/10/CAPTCHA_Help_rozsireni_dokument1.pdf`.

[15] Brewer, Judy, Michael Cooper, John Foliot, Scott Hollier, Stephen Noble, Janina Sajka, David Sloan a Jason White. *Inaccessibility of CAPTCHA* [online]. 2021. [vid. 2022/10/04]. Dostupné na `https://www.w3.org/TR/2021/DNOTE-turingtest-20211216/`. W3C Group Draft Note.

[16] Bursztein, Elie, Matthieu Martin a John Mitchell. Text-Based CAPTCHA Strengths and Weaknesses. In: *Proceedings of the 18th ACM Conference on Computer and Communications Security* [online]. New York, NY, USA: Association for Computing Machinery, 2011 [vid. 2022/10/02]. s. 125–138. ISBN 9781450309486. Dostupné na DOI 10.1145/2046707.2046724.

[17] *What Does CAPTCHA Mean? | CAPTCHA Types & Examples | Imperva* [online]. © 2022. [vid. 2022/10/30]. Dostupné na `https://www.imperva.com/learn/application-security/what-is-captcha/`.

[18] *Buster: Captcha Solver for Humans* [online]. [vid. 2022/10/02]. Dostupné na `https://github.com/dessant/buster`.

[19] *Engagement Advertising Technology | CAPTCHA, Pre-roll, Brand Tags - Solve Media* [online]. © 2010–2023. [vid. 2022/10/30]. Dostupné na `https://www.solvemedia.com/`.

[20] *US20090210937A1 - Captcha advertising - Google Patents* [online]. 2020. [vid. 2022/10/16]. Dostupné na `https://patents.google.com/patent/US20090210937A1/en`.

[21] Zeltser, Lenny. *Exploring Facebook's New Social CAPTCHA Authentication* [online]. 2010. [vid. 2022/10/16]. Dostupné na `https://zeltser.com/facebook-social-captcha-authentication/`.

[22] Meunier, Thibault. *Humanity wastes about 500 years per day on CAPTCHAs. It's time to end this madness* [online]. 2021. [vid. 2022/10/23]. Dostupné na `https://blog.cloudflare.com/introducing-cryptographic-attestation-of-personhood/`.

[23] Ahn, Luis von, Benjamin Maurer, Colin McMillen, David Abraham a Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science* [online]. 2008, ročník 321, č. 5895, s. 1465–1468 [vid. 2022/10/23]. ISSN 1095-9203. Dostupné na DOI 10.1126/science.1160379.

[24] *Official Google Blog: Teaching computers to read: Google acquires reCAPTCHA* [online]. 2009. [vid. 2022/12/11]. Dostupné na `https://googleblog.blogspot.com/2009/09/teaching-computers-to-read-google.html`.

[25] Perez, Sarah. *Google Now Using ReCAPTCHA To Decode Street View Addresses* [online]. 2012. [vid. 2022/10/23]. Dostupné na `https://techcrunch.com/2012/03/29/google-now-using-recaptcha-to-decode-street-view-addresses/`.

[26] *Choosing the type of reCAPTCHA* [online]. [vid. 2022/12/11]. Dostupné na `https://developers.google.com/recaptcha/docs/versions`.

[27] *Google používá reCAPTCHA ke čtení čísel domů* [online]. 2012. [vid. 2022/10/23]. Dostupné na https://www.root.cz/zpravicky/google-pouziva-recaptcha-ke-cteni-cisel-domu/.

[28] Shet, Vinay. *Google Online Security Blog: Are you a robot? Introducing "No CAPTCHA reCAPTCHA"* [online]. 2014. [vid. 2022/10/24]. Dostupné na https://security.googleblog.com/2014/12/are-you-robot-introducing-no-captcha.html.

[29] Shet, Vinay. *Google Online Security Blog: reCAPTCHA just got easier (but only if you're human)* [online]. 2013. [vid. 2022/10/24]. Dostupné na https://security.googleblog.com/2013/10/recaptcha-just-got-easier-but-only-if.html.

[30] Bergström, O. a J. Larsson. *Image Recognition for Solving Google's reCAPTCHA: An Investigation of how Different Aspects Affects the Security of Google's reCAPTCHA* [online]. School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 2022 [vid. 2022/10/24]. Diplomová práce. Dostupné na http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-320012.

[31] *How to Choose a CAPTCHA in WPForms* [online]. © 2016–2023. [vid. 2022/10/30]. Dostupné na https://wpforms.com/docs/setup-captcha-wpforms/.

[32] *reCaptcha v2 vs v3: Bieten sie wirksamen Botschutz?| DataDome* [online]. 2021. [vid. 2022/10/30]. Dostupné na https://datadome.co/de/bot-management-und-schutz/recaptcha-v2-vs-v3-bieten-sie-wirklich-wirksamen-botschutz/.

[33] *Bypass recaptcha v2 and image captcha using our bestcaptchasolver* [online]. © 2022. [vid. 2022/10/30]. Dostupné na https://bestcaptchasolver.com/bypass-captcha-solver-automation.

[34] Vega, Edward. *Why captchas are getting harder* [online]. 2021. [vid. 2022/10/24]. Dostupné na https://www.vox.com/22436832/captchas-getting-harder-ai-artificial-intelligence.

[35] Schwab, Katharine. *hCaptcha is a popular alternative to Google's reCaptcha* [online]. 2019. [vid. 2022/10/30]. Dostupné na https://www.fastcompany.com/90377406/suspicious-of-googles-recaptcha-heres-a-popular-alternative.

[36] *hCaptcha Is Now The Largest Independent CAPTCHA Service, Runs on 15% Of The Internet* [online]. 2022. [vid. 2022/10/30]. Dostupné na https://www.hcaptcha.com/post/hcaptcha-now-the-largest-independent-captcha-service.

[37] *Object Detection with KotlinDL and Ktor | The Kotlin Blog* [online]. 2022. [vid. 2022/12/11]. Dostupné na https://blog.jetbrains.com/kotlin/2022/01/object-detection-with-kotlindl-and-ktor/.

[38] *What is a Neural Network? AI and ML Guide – AWS* [online]. © 2022. [vid. 2022/12/11]. Dostupné na https://aws.amazon.com/what-is/neural-network/.

[39] Zaidi, Syed Sahil Abbas, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar a Brian Lee. A survey of modern deep learning based object detection models. *Digital Signal Processing* [online]. 2022, vydání 126, článek 103514 [vid. 2022/06/20]. ISSN 1051-2004. Dostupné na DOI https://doi.org/10.1016/j.dsp.2022.103514.

[40] Nathan, Aravind Ram. *Object Recognition vs Object Detection vs Image Segmentation* [online]. 2021. [vid. 2022/06/20]. Dostupné na `https://www.kaggle.com/getting-started/169984`.

[41] Arnab, Anurag, Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Mans Larsson, Alexander Kirillov, Bogdan Savchynskyy, Carsten Rother, Fredrik Kahl a Philip Torr. Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation: Combining Probabilistic Graphical Models with Deep Learning for Structured Prediction. *IEEE Signal Processing Magazine* [online]. 01, 2018, ročník 35, s. 37–52 [vid. 2022/06/20]. Dostupné na DOI 10.1109/MSP.2017.2762355.

[42] Padilla, Rafael, Sergio Netto a Eduardo da Silva. A Survey on Performance Metrics for Object-Detection Algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)* [online]. 2020 [vid. 2022/06/20]. s. 237–242. Dostupné na DOI 10.1109/IWSSIP48289.2020.

[43] *Backbone network in Object detection* [online]. 2021. [vid. 2022/06/21]. Dostupné na `https://stackoverflow.com/questions/66915996/backbone-network-in-object-detection`.

[44] *KotlinDL: High-level Deep Learning API in Kotlin* [online]. [vid. 2022/06/21]. Dostupné na `https://github.com/Kotlin/kotlindl`.

[45] *Open Neural Network Exchange* [online]. © 2019. [vid. 2022/06/21]. Dostupné na `https://onnx.ai/`.

[46] *ONNX Model Zoo* [online]. [vid. 2022/06/21]. Dostupné na `https://github.com/onnx/models`.

[47] *Machine learning for mobile developers* [online]. [vid. 2022/06/21]. Dostupné na `https://developers.google.com/ml-kit`.

[48] *Deep Java Library (DJL)* [online]. [vid. 2022/06/30]. Dostupné na `https://docs.djl.ai/index.html`.

[49] *FAQ - Deep Java Library* [online]. [vid. 2022/06/30]. Dostupné na `https://docs.djl.ai/docs/faq.html`.

[50] *djl/model-zoo.md at master · deepjavalibrary/djl* [online]. [vid. 2022/06/30]. Dostupné na `https://github.com/deepjavalibrary/djl/blob/master/docs/model-zoo.md`.

[51] *Deeplearning4j Suite Overview* [online]. © 2022. [vid. 2022/06/30]. Dostupné na `https://deeplearning4j.konduit.ai/`.

[52] *eclipse/deeplearning4j* [online]. [vid. 2022/06/30]. Dostupné na `https://github.com/eclipse/deeplearning4j`.

[53] *tensorflow/java: Java bindings for TensorFlow* [online]. [vid. 2022/06/30]. Dostupné na `https://github.com/tensorflow/java`.

[54] *TensorFlow Hub* [online]. [vid. 2022/06/30]. Dostupné na `https://tfhub.dev/`.

[55] *OpenCV* [online]. © 2022. [vid. 2022/06/30]. Dostupné na `https://opencv.org/`.

[56] *Java Neural Network Framework Neuroph* [online]. [vid. 2022/06/30]. Dostupné na `http://neuroph.sourceforge.net/`.

[57] Pokhrel, Sabina. *Image Data Labelling and Annotation – Everything you need to know* [online]. 2020. [vid. 2022/09/24]. Dostupné na `https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1`.

[58] *Computer Vision Annotation Formats* [online]. © 2022. [vid. 2022/09/24]. Dostupné na https://roboflow.com/formats.

[59] *YOLO Darknet TXT Annotation Format* [online]. © 2022. [vid. 2022/09/24]. Dostupné na https://roboflow.com/formats/yolo-darknet-txt.

[60] *COCO - Data format* [online]. 2020. [vid. 2022/09/24]. Dostupné na https://cocodataset.org/#format-data.

[61] EVERINGHAM, Mark a John WINN. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Development Kit* [online]. 2012. [vid. 2022/09/24]. Dostupné na http://host.robots.ox.ac.uk/pascal/VOC/voc2012/devkit_doc.pdf.

[62] *Pascal VOC XML Annotation Format* [online]. © 2022. [vid. 2022/09/24]. Dostupné na https://roboflow.com/formats/pascal-voc-xml.

[63] *TFRecord and tf.train.Example* [online]. 2022. [vid. 2022/09/24]. Dostupné na https://www.tensorflow.org/tutorials/load_data/tfrecord#tfrecords_format_details.

[64] POULOPOULOS, Dimitris. *What a TFRecord Is and How to Create It* [online]. 2021. [vid. 2022/09/24]. Dostupné na https://towardsdatascience.com/what-a-tfrecord-is-and-how-to-create-it-6be3c92c13cc.

[65] KELBEL, Jan a David ŠILHÁN. *Shluková analýza* [online]. [vid. 2022/12/22]. Dostupné na http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis_prednasky/zapis_02/13/shlukovani.pdf.

[66] *GridFS – MongoDB Manual* [online]. © 2022. [vid. 2022/11/13]. Dostupné na https://www.mongodb.com/docs/manual/core/gridfs/.

[67] ZHANG, Yi-Fan, Weiqiang REN, Zhang ZHANG, Zhen JIA, Liang WANG a Tieniu TAN. Focal and efficient IOU loss for accurate bounding box regression. *Neurocomputing* [online]. 2022, ročník 506, s. 146–157 [vid. 2022/11/13]. ISSN 0925-2312. Dostupné na DOI 10.1016/j.neucom.2022.07.042.

[68] REZATOFIGHI, Hamid, Nathan TSOI, JunYoung GWAK, Amir SADEGHIAN, Ian REID a Silvio SAVARESE. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. 2019 [vid. 2022/11/13]. s. 658–666. Dostupné na DOI 10.1109/CVPR.2019.00075.

Příloha A
Instalační příručka

Zdrojový kód je dostupný jednak v digitální příloze práce, jednak v Git repozitáři zveřejněném pomocí služby GitHub[1].

Postup sestavení, otestování a spuštění aplikace se neliší od postupu popsaného v diplomové práci [3] v kapitole 3.9.1. Využívá se Docker kontejnerů v kombinaci s nástrojem Docker compose, takže je nutné tyto programy mít nainstalované.

Před spuštěním aplikace je nutné nastavit tyto proměnné prostředí:

- `CAPTCHADB_USERNAME` obsahuje jméno výchozího uživatele MongoDB.
- `CAPTCHADB_PASS` obsahuje heslo výchozího uživatele MongoDB.
- `MONGO_STORAGE_PATH` obsahuje složku, do které se budou ukládat data z databáze.
- `DATA_OBJECTS_STORAGE_PATH` obsahuje složku, do které se budou ukládat datové objekty.

K nastavení proměnných lze využít připravený soubor `.env-example` a přepínač `--env-file .env-example` příkazu `docker-compose`. Případně lze tento soubor přejmenovat na `.env` a již není potřeba zadávat přepínač.

Pro základní práci s aplikací lze použít následující příkazy:

```
# spustí testy, vytvoří jar soubor s aplikací,
# sestaví Docker obraz s aplikací
docker-compose build
# spustí aplikaci (Docker kontejnery)
docker-compose -p captcha up -d
# zastaví aplikaci (Docker kontejnery)
docker-compose -p captcha stop
```

Služba `captcha` se skládá ze dvou kontejnerů. V jednom kontejneru se nachází databáze MongoDB, jejíž obraz je stažen ze služby Docker Hub. V druhém kontejneru je spuštěna aplikace CzechCaptcha, jejíž obraz je vytvořen na základě připraveného Dockerfile. Všechny funkcionality jsou přístupné na adrese `127.0.0.1:8080`. Pro otevření webového rozhraní zadejte ve webovém prohlížeči tuto adresu.

V případě, že databáze je při spuštění prázdná, aplikace CzechCaptcha do ní automaticky nahraje základní data. Mezi nimi je i uživatelský účet „admin“ s heslem „admin“ a administrátorskými právy a účet „user“ s heslem „user“ a běžnými uživatelskými právy. Tyto údaje lze využít pro přihlášení do webové aplikace.

[1] `https://github.com/opendatalabcz/czech-captcha`

Příloha B
Obsah přiloženého média

- `readme.md` ..... stručný popis obsahu média
- `czech-captcha` ..... zdrojový kód aplikace CzechCaptcha
- `text` ..... text práce
  - `src` ..... zdrojový kód práce ve formátu OpTEX
  - `thesis.pdf` ..... text práce ve formátu PDF