FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Webová aplikace pro monitoring ochranných známek
Student:	Eduard Stehlík
Vedoucí:	Ing. Jiří Novák, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat webovou aplikaci, která bude monitorovat konflikty existujících českých ochranných známek a názvů nově vzniklých právnických osob.

1. Nastudujte problematiku ochranných známek, proces jejich registrace a námitkového řízení.
2. Nastudujte možné přístupy pro vyhodnocení kolizí ochranných známek.
3. Analyzujte vhodné zdroje otevřených dat.
4. Proveďte rešerši obdobných aplikací a služeb monitoringu poskytovaných známkovými kancelářemi.
5. Navrhněte a implementujte metodu, která bude vyhodnocovat možné kolize českých ochranných známek a názvů nově vzniklých subjektů.
6. Navrhněte a implementujte webovou aplikaci, ve které bude možné přidávat sledované ochranné známky, zobrazovat kolize a nastavovat upozornění na kolize.
7. Výsledné řešení otestujte.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 27. října 2023 v Praze.

Bakalářská práce

WEBOVÁ APLIKACE PRO MONITORING OCHRANNÝCH ZNÁMEK

Eduard Stehlík

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jiří Novák, Ph.D.
16. května 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především Ing. Jiřímu Novákovi, Ph.D., který mou práci vedl a poskytoval mi cenné rady v rámci konzultací. Zároveň bych chtěl poděkovat své rodině a přítelkyni za vytvoření příjemného prostředí a podporu po celou dobu, kdy jsem se bakalářské práci věnoval. Zejména bych chtěl vyzdvihnout pomoc mé matky, která pomohla s gramatickou a stylistickou úpravou práce.

Prohlášení

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 16. května 2024

Abstrakt

Tato bakalářská práce řeší monitoring ochranných známek podle změn, které proběhly v české legislativě k 1. 1. 2019. Při monitoringu dochází k vyhodnocování možných kolizí sledovaných označení v systému s nově přihlašovanými ochrannými známkami. Pro vyhodnocování možných slovních kolizí je používána Damerauova vzdálenost. V případě grafických prvků je využívána metoda Template matching.

Výsledkem práce je systém, který provádí periodické kontroly přihlášek ochranných známek a obchodních firem nově vzniklých společností. K tomu využívá otevřená data poskytovaná úřady v České republice i Evropské unii. V případě odhalené kolize notifikuje uživatele pomocí několika komunikačních kanálů. Součástí systému je webová aplikace, která registrovaným uživatelům umožňuje spravovat jejich sledovaná označení a vzniklé kolize. Program, který vyhodnocuje kolize, komunikuje s webovou aplikací pomocí interního REST API a společně s webovou aplikací zpracovává uživatelské přílohy nahrané v objektovém úložišti.

Klíčová slova webová aplikace, ochranné známky, Damerauova vzdálenost, otevřená data, Go, PHP, objektové úložiště, monitoring, OpenCV

Abstract

This bachelor thesis deals with the monitoring of trademarks according to the changes that took place in the Czech legislation as of 1 January 2019. During the monitoring, possible conflicts between the monitored signs in the system and newly applied trademarks are evaluated. The Damerau distance is used to evaluate possible word conflicts. In the case of graphic elements, the Template matching method is used.

The result of the work is a system that performs periodic checks of trademark applications and business names of newly established companies. For this purpose, it uses open data provided by offices in the Czech Republic and

the European Union. In case of a detected conflict, it notifies the user through several communication channels. The system includes a web application that allows registered users to manage their monitored signs and the resulting conflicts. The program that evaluates the collisions communicates with the web application using an internal REST API and together with the web application it processes user attachments uploaded in the object store.

Keywords web application, trademarks, Damerau distance, open data, Go, PHP, object storage, monitoring, OpenCV

Seznam zkratek

API	Aplikační rozhraní
ARES	Administrativní registr ekonomických subjektů
B2C	Business to Client (obchodník k zákazníkovi)
CRUD	Create, Read, Update, Delete
CSS	Kaskádové styly
ČSÚ	Český statistický úřad
DIA	Datová a informační agentura
DTO	Data transfer object
EU	Evropská unie
EUIPO	Úřad Evropské unie pro duševní vlastnictví
GDPR	Obecné nařízení o ochraně osobních údajů
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IČO	Identifikační číslo osoby
ID	identifikátor
IDN	Internationalized Domain Name
IP	Internet Protocol
IRI	Internationalized Resource Identifier
JSON	JavaScriptový Object Notation
NKOD	Národní katalog otevřených dat
ORDBMS	Objektově-relační systém pro správu databáze
ORM	Objektově relační mapování
OSA	Optimal string alignment
P2P	Peer to Peer (rovný s rovným nebo klient klientovi)
RAM	Random Access Memory
RES	Registr ekonomických subjektů
REST	Representational state transfer
SIFT	Scale-invariant feature transform
SMTP	Simple Mail Transfer Protocol
TLD	Top Level domain
URL	Uniform Resource Locator
USPTO	U.S. Patent and Trademark Office
ÚPV	Úřad průmyslového vlastnictví
WIPO	Světová organizace duševního vlastnictví
XML	Extensible Markup Language
XSD	XML Schema Definition
YAML	YAML Ain't Markup Language

Úvod

Značky jsou každodenní součástí našeho života a společnosti. Setkáváme se s nimi po celý den a není prakticky možné se jim vyhnout. Můžeme díky nim identifikovat konkrétní produkt, službu nebo výrobce. Je s nimi spojována jejich kvalita a mohou být pro spotřebitele zárukou. Kromě známého jména mohou mít tyto značky i vybudovanou komunitu, která jejich produkty nebo služby přímo adoruje. Co ale brání v prodeji vlastní Coca-Coly nebo iPhone?

Státy po celém světě tak pro ochranu tohoto duševního vlastnictví využívají ochranných známek. Ty si můžeme za poplatek registrovat u příslušných úřadů, a získat tak právní ochranu názvů, sloganů, vizuální identity nebo i audia. Hlavním důvodem existence ochranných známek je tak zamezení užívání zaregistrovaného označení třetí osobě, která k tomu nemá svolení a mohla by ať už vědomě, či nevědomě parazitovat na značce nebo se podílet na jejím poškozování.

V České republice spravuje tuzemské ochranné známky Úřad průmyslového vlastnictví (ÚPV). Ten až do 1. 1. 2019 prováděl intenzivnější kontrolu nových přihlášek, kdy kontroloval, zda nová přihláška ochranné známky nekoliduje s již existující, a případně ji zamítl. Novelou zákona č. 441/2003 Sb. však došlo ke změně, a od 1. 1. 2019 je povinnost kontrolovat a chránit ochrannou známku přenesena na vlastníka ochranné známky. Prakticky to znamená, že sám vlastník musí periodicky kontrolovat věštníky a databáze úřadů, aby odhalil kolizní přihlášky, a ochránil tak svá práva.

Úřady však poskytují tato data v různorodých formátech ve formě otevřených dat. My tak můžeme proces periodického hlídání ochranných známek automatizovat v přehledné aplikaci a ušetřit čas vlastníkům ochranných známek.

Když jsem si registroval několik ochranných známek a zjišťoval si o nich bližší podrobnosti, velmi mě překvapilo, že si vlastník ochranné známky musí danou známku chránit proti přihláškám stejného či obdobného znění nebo vyobrazení. Byl jsem si dobře vědom, že mi plyne povinnost známku užívat, protože by pak mohla být v souladu s § 31 zákona č. 441/2003 Sb. zrušena, ale tato povinnost ochrany v případě pokusu třetí strany o zápis shodné nebo

podobné ochranné známky mě až zaskočila. Měl jsem za to, že úřad, který disponuje registrem již existujících ochranných známek a za časově omezený zápis ochranné známky si účtuje poplatek, dokáže takové kolize odhalit. Není zřejmé, zda to nedokáže nebo nechce, každopádně je jisté, že teď už ani nesmí. Vlastníci ochranných známek mají vcelku krátkou lhůtu, ve které musejí přihlášku kolizní ochranné známky odhalit a napadnout u ÚPV pomocí námitek.

Cílem bakalařské práce je tak vytvoření webové aplikace, do níž si vlastníci ochranných známek mohou přidat své ochranné známky za účelem jejich hlídání. Aplikace pak bude denně vyhodnocovat možné kolize s nově podanými přihláškami ochranných známek, aby co nejrychleji identifikovala možné kolize, a získala tak vlasníkům čas pro podání námitek u ÚPV.

Cíle

Cílem bakalářské práce je analyzovat vhodné zdroje otevřených dat a přístupy pro vyhodnocování možných kolizních ochranných známek, porovnat již existující řešení, která se danou problematikou zabývají. Na základě předchozích výstupů navrhnout a implementovat webovou aplikaci pro správu monitorovaných ochranných známek s programem pro periodické kontrolování nových českých ochranných známek a obchodních firem nově vzniklých společností, který se bude soustředit na vyhodnocování možných kolizí. Výsledné řešení následně otestuji.

Kapitola 1
Teoretická část

Součástí této kapitoly je přiblížení problematiky ochranných známek, procesu jejich registrace, námitkového řízení a jejich upevnění v české legislativě. Zároveň kapitola popisuje, co to jsou otevřená data, jak se dělí a jak je na ně nahlíženo v České republice. V neposlední řadě se pak věnuje fungování možných přístupů pro porovnávání obrázků a textů a možnostem následného využití některých z nich jako pilířů výsledné bakalářské práce.

1.1 Ochranné známky

Ochrannou známkou je označení, typicky název nebo logo značky, pro které je zajištěno jeho výhradní užívání vlastníkem ochranné známky. Velice zjednodušeně by se ochranná známka dala vysvětlit i jako takový „patent“ značky. Její registrace nás chrání před neoprávněným užíváním našeho vlastnictví, zejména názvu značky, loga atd.

Zákon o ochranných známkách ji hned v § 1a definuje následovně: „*Ochrannou známkou může být za podmínek stanovených tímto zákonem jakékoliv označení, zejména slova, včetně osobních jmen, barvy, kresby, písmena, číslice, tvar výrobku nebo jeho obal nebo zvuky*“, takové označení musí být navíc „*způsobilé odlišit výrobky nebo služby jedné osoby od výrobků nebo služeb jiné osoby a být vyjádřeno v rejstříku ochranných známek způsobem, který příslušným orgánům a veřejnosti umožňuje jasně a přesně určit předmět ochrany poskytnuté vlastníkovi ochranné známky.*“[1]. Splnění definice podle § 1a však nezaručuje, že dané označení opravdu bude zapsáno jako ochranná známka. Existuje totiž ještě množství jak absolutních, tak relativních důvodů pro zamítnutí přihlášky.

Ochrana označení netrvá navždy. Ochranná známka je dle § 29 zákona o ochranných známkách platná 10 let ode dne podání přihlášky. Na rozdíl od patentu však můžeme u ochranné známky opakovaně prodlužovat platnost a maximální délka není stanovena. Žádost o prodloužení platnosti však můžeme podat nejdříve 12 měsíců a zároveň nejpozději 6 měsíců před uplynutím

platnosti.[1]

1.1.1 Rozdělení

Ochranná známka nemusí chránit jen název nebo logo, ale má mnoho dalších typů. Dalšími, vcelku neobvyklými, ale rovněž užívanými typy jsou ochranné známky zvukové, které chrání zvuk nebo kombinaci zvuků, a pohybové, které jsou tvořeny pohybem nebo změnou pozice.

Kompletní seznam typů je definován v příloze č. 1 zákona o ochranných známkách a tento seznam vychází ze seznamu typů evropských ochranných známek. Ochranná známka je ze zákona následující:

- slovní ochranná známka,
- obrazová ochranná známka,
- prostorová ochranná známka,
- poziční ochranná známka,
- ochranná známka se vzorem,
- barevná ochranná známka,
- zvuková ochranná známka,
- pohybová ochranná známka,
- multimediální ochranná známka,
- holografická ochranná známka,
- jiný druh ochranné známky[1].

1.1.2 Přihláška

Proces registrace tuzemské ochranné známky začíná podáním přihlášky u ÚPV. Pro každou ochrannou známku musí být podána přihláška samostatně a každá přihláška podléhá také správnímu poplatku.

V první řadě je nutné si zvolit, jaký typ ochranné známky registrovat. V rámci sekce 1.1.1 jsou vypsány všechny typy, které lze v České republice registrovat. Typicky se jedná buď o ochrannou známku slovní, nebo obrazovou. Následně je nutné vybrat i třídy, pro které chceme získat ochranu.

Třídy ochranných známek představují další dělení. V České republice se podle § 19a odst. 1 zákona o ochranných známkách postupuje podle Niceské dohody o mezinárodním třídění výrobků a služeb pro účely zápisu známek, která definuje jednotlivé třídy[1]. Celkem v ní lze nalézt 45 tříd, kdy třídy 1–34 platí pro výrobky a třídy 35–45 pro služby[2].

Po připojení přílohy, pokud je vyžadována, je potřeba doplnit údaje o přihlašovateli a případně údaje pro uplatnění práva přednosti dle mezinárodních smluv. Poté je možné přihlášku odeslat na ÚPV a zaplatit příslušný správní poplatek, který se odvíjí i od počtu tříd, pro které požadujeme ochranu.

Ve chvíli, kdy je přihláška dodána ÚPV, může úřad zahájit řízení o přihlášce podle hlavy VI zákona o ochranných známkách. Před zveřejněním přihlášky ve věstníku, ze kterého bude systém vyhodnocovat data, provede úřad formální a věcný průzkum přihlášky. Úřad tedy projde možné absolutní důvody zamítnutí přihlášky a podle výsledku ji zamítne, nebo uveřejní ve věstníku.[1]

1.1.3 Námitkové řízení

Zveřejněním přihlášky ve věštníku začíná podle § 24 a § 25 zákona o ochranných známkách běžet tříměsíční lhůta, během níž mohou vlastníci ochranných známek nebo široká veřejnost vyjádřit k této přihlášce připomínky nebo podat námitky. Zmíněná lhůta je velice důležitá, protože nejen úřad, ale hlavně zákon její překročení nepřipouští. Zároveň je nutné v rámci této lhůty dodat i důkazy pro své připomínky a námitky.[1]

Podání námitky zahájí námitkové řízení, jehož postup je popsán v rámci § 26 zákona o ochranných známkách. Úřad pro každou námitku zejména zkontroluje, zda byla podána v již zmíněné tříměsíční lhůtě, jestli ji podala oprávněná osoba a jestli obsahuje námitka důvody a důkazy. V případě, že úřad z předchozích důvodů námitku nezamítne, vyrozumí o dané námitce přihlašovatele a stanoví mu lhůtu pro vyjádření.[1]

Na základě vyjádření, údajů ze spisu a námitky úřad rozhodne, zda skutečně došlo k zásahu do práv majitele starší ochranné známky podle § 7 zákona o ochranných známkách. V případě, že úřad shledá zásah do práv majitele starší ochranné známky, zamítne přihlášku úplně nebo jen pro třídy, pro které existuje důvod. V opačném případě námitku zamítne.[1]

1.1.4 Zákon 286/2018 Sb.

Zákon 286/2018 Sb. změnil s účinností od 1. 2. 2019 zákon o ochranných známkách. Zapracoval do něj směrnici Evropského parlamentu a Rady (EU) 2015/2436, kterou se sbližují právní předpisy členských států o ochranných známkách. Nejvíce diskutovanou změnou je zrušení § 6 zákona o ochranných známkách, které mělo zrychlit a pro úřad zjednodušit proces registrace ochranných známek.

Uvedený § 6 zákona o ochranných známkách, který platil do 31. 12. 2018, zněl následovně: „*Do rejstříku se nezapíše označení, pokud je shodné se starší ochrannou známkou, která je přihlášena nebo zapsána pro jiného vlastníka či přihlašovatele pro shodné výrobky či služby; to neplatí, pokud vlastník či přihlašovatel starší ochranné známky udělí písemný souhlas k zápisu pozdější ochranné známky do rejstříku.*“[1]. Zrušením tohoto paragrafu pozbyli vlastníci

ochranných známek jednu z ochran svého označení a musí se již spoléhat pouze na svou iniciativu, případně iniciativu své známkoprávní kanceláře, a uplatňovat namítání přihlašovaných označení. Zároveň platilo, že přihlášku, která by podle § 6 zákona o ochranných známkách byla úřadem zamítnuta, mohl připomínkovat podle § 24 téhož zákona kdokoliv. Úřad tedy mohl o možné kolizi vyrozumět i jiný subjekt. Po novelizaci tak mohou učinit pouze vlastníci dotčeného označení.

Další značnou změnou bylo zrušení písmene m § 4 zákona o ochranných známkách, které stanovovalo, že označení se do rejstříku nezapíše „*jestliže je zjevné, že přihláška ochranné známky nebyla podána v dobré víře*“[1]. Vzhledem k tomu, že podle § 4 se posuzují absolutní důvody pro zamítnutí přihlášky v rámci formálního průzkumu úřadu, přišel tak úřad o další zákonný důvod zamítnutí přihlášky, jímž však mohl vlastníky chránit.

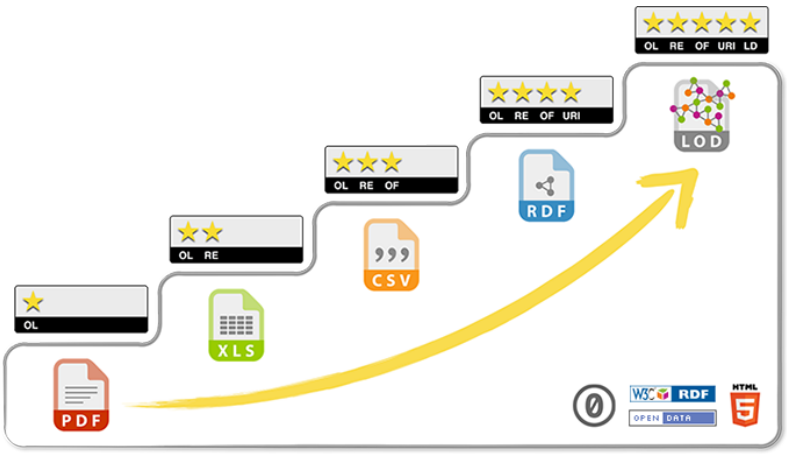
1.2 Otevřená data

Definice otevřených dat shrnutá do jedné věty je podle Open Knowledge následující: „*Otevřená data a obsah může kdokoli volně používat, upravovat a sdílet za jakýmkoliv účelem.*“[3]. Vzhledem k tomu, že otevřená data jsou dílo, pak můžeme uplatnit i definici otevřeného díla, které, podle stejné organizace, musí být poskytováno pod otevřenou licencí, mělo by být dostupné celé s možností bezplatného stažení prostřednictvím internetu. Zároveň by data měla být strojově čitelná a poskytnuta v otevřeném formátu, jehož specifikace je veřejně dostupná.[4]

1.2.1 Hodnocení otevřených dat

Vynálezce internetu sir Tim Berners-Lee přišel s návrhem na pětistupňové hodnocení otevřenosti dat, které je popsáno na obrázku č. 1.1. Jeho cílem bylo podpořit správné otevírání dat nejen u jednotlivců a společností, ale zejména u státní správy. Stupnice klade důraz na to, aby byla data skutečně otevřená, tedy poskytovaná pod otevřenou licencí.

- 1. hvězda – data jsou dostupná na webu, poskytována pod otevřenou licencí.
- 2. hvězda – data jsou strukturovaná a strojově čitelná a splňují vše předchozí.
- 3. hvězda – data jsou poskytována v otevřeném formátu a splňují vše předchozí.
- 4. hvězda – pro identifikaci je v datech užíváno IRI a splňují vše předchozí.
- 5. hvězda – data jsou propojena s jinými datovými sadami a splňují vše předchozí.[6]



Obrázek 1.1 5hvězdičková stupnice otevřených dat, převzato z [5]

1.2.2 Tuzemský přístup

Česká republika a Evropská unie pracují s termínem otevřená data ve svých zákonech, resp. směrnicích již delší dobu. V případě Česka je termín užíván v rámci zákona č. 106/1999 Sb. o svobodném přístupu k informacím. V § 3a odst. 5 je uvedeno, že: „*Otevřenými daty se pro účely tohoto zákona rozumí informace zveřejňované způsobem umožňujícím dálkový přístup v otevřeném a strojově čitelném formátu, jejichž způsob ani účel následného využití není povinným subjektem, který je zveřejňuje, omezen a které jsou evidovány v národním katalogu otevřených dat.*“[7].

Tento zákon zároveň v § 2, § 2a a § 2b definuje tzv. povinné subjekty, které musí poskytovat informace. Dle JUDr. MgA. Jakuba Míška, Ph.D., jenž vedl pro DIA kurz o právní úpravě otevřených dat, je pak pro otevřená data důležitý § 5a odst. 2 téhož zákona, který ukládá povinným subjektům, spravujícím určitý registr dat, povinnost tato data publikovat ve formě otevřených dat. Z aktuální právní úpravy tedy vyplývá, že musí tato data publikovat způsobem umožňujícím dálkový přístup v otevřeném a strojově čitelném formátu a účel následného využití nemůže být omezen. Prakticky to znamená, že povinné subjekty musí ze zákona tato data zveřejňovat přinejmenším jako tříhvězdičková otevřená data podle Berners-Leeovy 5hvězdičkové škály otevřených dat.[8]

Zároveň mají tyto subjekty povinnost tato data publikovat v NKOD, který je popsán v § 4c zákona 106/1999 Sb. a spravuje jej DIA. Lze v něm nalézt jednotlivé datové sady, poskytovatele těchto sad a aplikace, které se veřejně přihlašují k užívání těchto sad.

Uvedený český přístup, který zapracovává směrnici Evropského parlamentu a Rady (EU) 2019/1024 o otevřených datech a opakovaném použití informací veřejného sektoru, má za důsledek svobodnější přístup a nakládání s veřejnými informacemi, a tím i možnost vzniku desítek užitečných aplikací, které pomáhají např. ke kontrole veřejných financí, skóringu firem, potírání korupce.

Pomáhají i běžným občanům ve výběru střední školy, nalezení vhodné nádoby či zařízení na odpad nebo informují o mimořádnostech v městské hromadné dopravě.

1.3 Porovnání textu

V praxi je často potřeba nějakým způsobem vyjádřit podobnost dvou řetězců, v ideálním případě číslem, pro následné vyhodnocení této podobnosti. Pro tyto potřeby tak vznikly různé vzdálenosti, které podobnost řetězců číselně reprezentují. Využití pak nacházejí nejen u vyhodnocování možných kolizí ochranných známek, ale zejména např. při detekci chyb[9].

1.3.1 Hammingova vzdálenost

Tato vzdálenost je pojmenována po americkém matematikovi a držiteli Turingovy ceny Richardu Hammingovi. Definuje číselnou vzdálenost mezi dvěma řetězci nebo vektory stejné délky jako počet pozic, na kterých se liší[9]. Tedy pro řetězce FIT a FEL bude platit $D(FIT, FEL) = 2$, protože musí nastat záměna I za E a T za L, tedy řetězce se liší na dvou pozicích.

1.3.2 Levenshteinova vzdálenost

Levenshteinova vzdálenost vychází ze tří jednoduchých editačních operací: přidání znaku, odebrání znaku a záměny znaku[10]. Publikoval ji v roce 1965 ruský matematik Vladimir Levenshtein. Je využívána např. při detekci překlepů.

Postup při jejím výpočtu je vcelku jednoduchý. Pokud se znaky na stejných pozicích rovnají, neprobíhá žádná operace a výpočet pokračuje. V případě, že se znaky na stejných pozicích nerovnají, rekurzivně proběhne provedení operací vložení znaku, smazání znaku a záměny znaku a z těchto tří operací se vybere ta s minimální vzáleností, ke které se přičte 1. Rekurze se zastaví ve chvíli, kdy probíhá porovnání dvou prázdných řetězců a vrátí se tedy 0, nebo když jeden řetězec je prázdný, pak se vrátí délka zbývajícího řetězce. Soustava rovnic č. 1.1 popisuje postup výpočtu standardní Levenshteinovy vzdálenosti, převzato z [11].

$$
\begin{aligned}
L(\varepsilon, W) &= |W| \\
L(V, \varepsilon) &= |V| \\
L(aV, bW) &= \begin{cases} L(V, W) & \text{pokud } a = b \\ 1 + min \begin{cases} L(V, W) \\ L(aV, W) \\ L(V, bW) \end{cases} & \text{pokud } a \neq b \end{cases}
\end{aligned} \tag{1.1}
$$

pro $V, W \in \Sigma^*$ a $a, b \in \Sigma$

Pro příklad lze uvést, že řetězce Kočka a Kokča (řeka v Afghánistánu) mají Levenshteinovu vzdálenost 2, protože proběhne odstranění znaku č, Kočka → Koka a následně vložení znaku č, Koka → Kokča. Alternativně mohla proběhnout dvakrát záměna znaků.

Pro výpočet Levenshteinovy vzdálenosti se užívá Wagner-Fischer[1] algoritmus, který je založen na dynamickém programování.

1.3.3 Damerauova vzdálenost

Tato vzdálenost, označována i jako Damerau-Levenshteinova vzdálenost, je rozšířením Levenshteinovy vzdálenosti popsané v sekci č. 1.3.2. Kromě tří editačních operací, které přebírá z Levenshteinovy vzdálenosti, přidává ještě čtvrtou operaci, prohození znaků. Tento přístup měl vylepšit odhalování překlepů[12].

Levenshteinova vzdálenost řetězců Kočka a Kokča v sekci č. 1.3.2 byla 2. V případě, že bychom chtěli pro příklad získat Damerauovu vzdálenost těchto dvou řetězců, výsledkem bude 1. Dojde totiž pouze k jedné editační operaci, a to prohození písmen č a k.

1.3.3.1 OSA vzdálenost

Podle L. Boytsova lze vypočítat dva typy Damerauovy vzdálenosti, omezenou vzdálenost a neomezenou vzdálenost. Jako omezená je označována OSA vzdálenost, která, jak již název napovídá, spoléhá na optimální seřazení řetězce. V závislosti na umístění znaků v řetězci tak nemusí vrátit opravdovou Damerauovu vzdálenost, ale Levenshteinovu vzdálenost. Její nevýhodou je limitace provedení pouze jedné operace v danou chvíli. Tedy není možné provést prohození dvou znaků a následně mezi ně znak vložit.[13]

Výpočet OSA vzdálenosti vychází z Wagner-Fischer dynamického algoritmu pro výpočet Levenshteinovy vzdálenosti a jeho úprava pro OSA vzdálenost je vyobrazena v pseudokódu č. 1.1, kterou navrhli Oommen a Loke[14].

[1] WAGNER, R. A.; FISCHER, M. J. The String-to-String Correction Problem https://doi.org/10.1145/321796.321811

Pro výpočet opravdové Damerauovy vzdálenosti lze použít algoritmus Lawrence a Wagnera[2].

Výpis kódu 1.1 Výpočet OSA vzdálenosti [14, 15, 16]

```
OSAVzdalenost(a_1,...,a_n; b_1,...,b_m)
    T[0,0] := 0
    Pro i := 1,...,n:
        T[i,0] := i
    Pro i := 1,...,m:
        T[0,i] := i
    Pro i := 1,...,n:
        Pro j := 1,...,m:
            c := 1
            Pokud a_i = b_j:
                c := 0
            T[i,j] := min(T[i,j-1] + 1,      // vložení
                          T[i-1,j] + 1,      // smazání
                          T[i-1,j-1] + c)    // záměna
            Pokud i > 1 a j > 1 a a_i = b_{j-1} a a_{i-1} = b_j:
                T[i,j] := min(T[i,j],
                              T[i-2,j-2] + 1)  // prohození
    Vrať T[n,m]
```

1.4 Porovnání obrázků

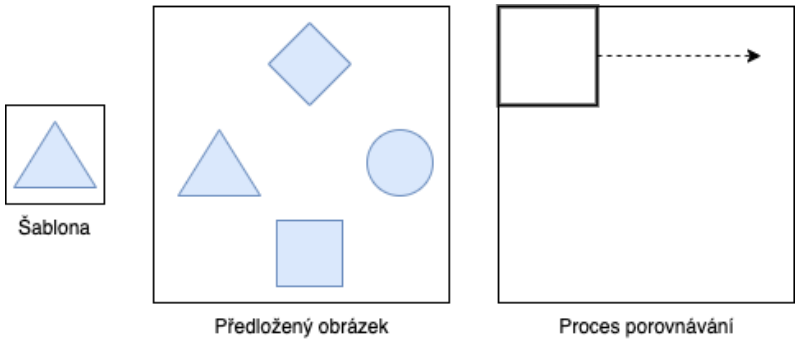
Při práci s obrázky dochází často k situaci, kdy je potřeba je algoritmicky zpracovávat. Typicky jde o případy, kdy je nutné identifikovat stejné obrázky, které pouze zaplňují místo[17], v případě kontroly kvality[18], v medicíně[19], případně při seskupování obrázků podle objektů a vyhodnocování objektů nacházejících se na obrázku[20]. Níže jsou popsány vybrané dvě metody, které se porovnáváním obrázku zabývají.

1.4.1 Template matching

Jedná se o jednoduchý, avšak účinný přístup počítačového vidění k vyhledávání prvku z šablony v rámci předložených obrázků. Šablona se pak typicky sestává pouze ze signifikantního prvku, který chceme identifikovat nebo nám jeho přítomnost pomůže identifikovat větší celek. Některé jeho techniky dokáží s dobrými výsledky rozpoznávat i obličeje[21].

[2]LOWRANCE R.; WAGNER R. A. An Extension of the String-to-String Correction Problem https://doi.org/10.1145/321879.321880

Přístup vychází z částí předloženého obrázku oproti šabloně, kterou chceme v předloženém obrázku nalézt. V rámci tohoto procesu je pak pro každou pozici šablony v předloženém obrázku počítána korelace, která vyjadřuje míru vhodnosti umístění šablony na dané pozici. Z této míry následně můžeme usoudit, zda je opravdu šablona obsažena v předloženém obrázku, případně jestli se mu podobá. Tento přístup má dobré výsledky v případě, že se šablona na obrázku vyskytuje nebo jsou si obrázky velice podobné.[19]



Obrázek 1.2 Proces porovnávání v rámci metody Template matching, přeloženo a převzato z [19]

1.4.2 SIFT

SIFT je algoritmus počítačového vidění, který publikoval v roce 1999 David Lowe. Tento algoritmus popisuje a identifikuje lokální prvky v obrázku na základě bodů zájmu, které jsou lokalizovány jako lokální maxima, resp. minima. Tento algoritmus tak lze použít pro rozpoznávání objektů v obrázcích. Typicky tak lze identifikovat stejnou stavbu na dvou různých fotografiích, kdy každá je focena z jiného úhlu a za jiného denního světla.

V první řadě SIFT identifikuje zájmové body, z nichž vyřadí ty, které by mohly být způsobeny nějakým šumem a mají malý kontrast, nebo se vyskytují na hranách. Následně pro tyto body zájmu spočítá orientaci v jejich okolí. Každý výsledný zájmový bod pak v rámci $4 \cdot 4 \cdot 8$ vektoru svého okolí popisuje prvek s pomocí histogramů spočítaných orientací. Tyto prvky je možné u dvou obrázků porovnat a zjistit, zda došlo ke shodě.[20, 22]

Kapitola 2
Analýza

Tato kapitola se věnuje analýze již existujících systémů a zdrojů dat. Popisuje tedy zavedené postupy a řešení, vyzdvihuje jejich funkce a upozorňuje na možné komplikace. V případě zdrojů dat se zaměřuje na otevřená data jak v souvislosti s ochrannými známkami, tak s údaji o nově vzniklých společnostech, kde přichází s možnými přístupy jejich detekce.

2.1 Existující řešení

Služba monitoringu ochranných známek je běžnou praxí a známkoprávní kanceláře ji poskytovaly již před novelizací zákona o ochranných známkách. Obzvlášť v zahraničí je propojení software s právním světem označováno již delší dobu jako Legal Tech a těší se vzestupu. Novela zákona o ochranných známkách měla za následek prohloubení tohoto propojení i v České republice, kde nejen jednotlivci, ale právě i samotné známkoprávní kanceláře svěřují kontroly a ochranu práv duševního vlastnictví specializovaným programům.

2.1.1 Programy

Tato sekce stručně popisuje některá z existujících řešení, provádějících automatickou kontrolu kolizí ochranných známek, případně se zaměřují i na zajišťování další automatické detekce a ochrany. Je nutné zmínit, že tyto programy nejsou poskytovány zdarma a jejich využití je zpoplatněno pro každou monitorovanou ochrannou známku zvlášť. Obvykle přístup nacenění monitoringu ochranné známky závisí na oblastech, které mají být sledovány.

2.1.1.1 RightHub

Populární a mezinárodní řešení RighHub poskytuje komplexní ochranu duševního vlastnictví online. Monitoring ochranných známek poskytuje pod službou Watch, která kontroluje celosvětové databáze a snaží se identifikovat možné

kolize. Kromě služby Watch nabízí Right Hub ještě službu Brand Protection, která pomáhá identifikovat padělky na online tržištích, případně i portál pro správu všech informací týkajících se duševního vlastnictví.[23]

2.1.1.2 tramatm

Celosvětová služba, která kromě monitoringu ochranných známek poskytuje i právní služby v oblasti duševního vlastnictví. Dle jejich webových stránek provádí kontroly registrů jednou za týden a systém průběžně kalibrují, aby snížili počet falešně pozitivních výsledků. Služba podle svých statistik monitoruje přes dva tisíce ochranných známek a měsíčně údajně odhalí přes 10 tisíc možných kolizí.[24]

2.1.1.3 Legal Zoom

Další službou, která provádí automatický online monitoring ochranných známek je Legal Zoom. Dle jejich webových stránek v případě nalezení možné kolize dokáží propojit vlastníka ochranné známky s právníkem se zaměřením na duševní vlastnictví. Za nevýhodu můžeme považovat jejich zúženou působnost, která je omezena na data z USPTO.[25]

2.1.1.4 JUMP Trademarks

Zahraniční řešení JUMP Trademarks nabízí kromě monitoringu i služby registrace, rešerše a prodloužení ochranné známky. Na svých webových stránkách uvádějí, že sbírají data od úřadů po celém světě a mají značné množství velkých klientů. Jejich služby se však omezují na ochranné známky a jejich monitoring v registrech. Nelze tak v rámci této služby monitorovat např. konflikty s nově vzniklými společnostmi nebo vyhledávání padělků.[26]

Ačkoliv by za touto službou měla stát právní kancelář z Anglie, absentují na jejich webu identifikační informace a právní dokumenty, na které se však odkazují. V porovnání s ostatními službami se tak jedná o méně důvěryhodné řešení.

2.1.1.5 Hlídač OZ

České řešení pro monitoring ochranných známek Hlídač OZ vyvíjí skupina ATLAS GROUP. Naneštěstí je webová stránka prezentující tento produkt značně strohá a poskytuje s vyjímkou kontaktních informací velice obecný popis. Dle webu by se tak mělo jednat o komplexní řešení zahrnující jak nástroj pro rešerši, tak monitoring ochranných známek s možností nastavit hranice podobnosti pro vyhodnocování kolizí a notifikace. Upozornění zasílá na e-mail a je také dostupné v aplikaci. Zároveň by z dostupných informací mělo být využíváno velkým množstvím českých společností.[27]

2.1.2 Známkoprávní kanceláře

Agendu související s ochrannými známkami zpracovávají převážně advokátní kanceláře, které dedikují část svého působení duševnímu vlastnictví. Řeší tedy registrace ochranných známek, jejich prodlužování, rešerše a i jejich monitoring (watching).

Právě v případě monitoringu a jeho provádění se jejich přístupy liší. Typicky kanceláře, které mají pobočky v několika světových zemích, inklinují k využití Legal Tech, která se stává velice populární, protože dokáže efektivně pracovat s dokumenty a provádět rešerši. Dle statistik je využití Legal Tech v advokátních kancelářích na vzestupu[28].

V České republice v případě kanceláří, které nemají takový zahraniční přesah, je běžnějším přístupem manuální monitoring a vyhodnocování možných kolizí ve věstnících, které probíhají jednou za jeden až dva měsíce, aby bylo možné v případě potřeby stihnout podat námitky v rámci lhůty. Právě z důvodu těchto krátkých lhůt je téměř nutné využívat automatizovaná řešení pro odhalování kolizí. Do budoucna tak bude nejspíše podíl jejich využití v jednotlivých kancelářích jen sílit.

V případě čistě manuální kontroly je velkou výhodou minimalizace falešně negativních výsledků, které by systém neodhalil, ale jednalo by se o kolizní přihlášky. V ideálním případě by tak známkoprávní kanceláře měly provádět kontrolu s pomocí specializovaného software v co nejkratších časových úsecích a následně provádět analýzu jak výstupu programu, tak v určitém časovém intervalu provádět i manuální kontrolu, aby došlo k minimalizování výskytu jak falešně pozitivních, tak falešně negativních výsledků.

2.2 Zdroje dat

O datech se říká, že jsou ropou 21. století. Společnosti s nimi obchodují a bedlivě si je střeží. V případě, že jejich součástí jsou i údaje o fyzických osobách, podléhají tato data regulaci ze strany EU (GDPR) a při jejich nevhodném zpracování hrozí správcům pokuty.

Celá naše aplikace je postavena na datech a práci s nimi. Stěžejní je tedy způsob získání dat. V našem případě budou data, která chceme vyhodnocovat, většinou pocházet od státu. Lze využít legislativu EU týkající se otevřených dat a její následnou implementaci napříč EU včetně České republiky.

V ostatních případech je nutné spoléhat na organizace a společnosti, že se otevřená data nebo alespoň nějakou formu dat, rozhodnou publikovat samy. Kde i to nestačí, můžeme se uchýlit k scrapingu veřejně dostupných zdrojů.

Následující část práce tak popisuje možné zdroje dat pro aplikaci a přístup, s jakým jsou publikována, a co je zapotřebí k jejich získání.

2.2.1 Ochranné známky

Pro zdroje dat o ochranných známkách je nutné volit ověřené zdroje, které jsou aktuální a pokud možno poskytují data bezplatně. Tento požadavek však výběr značně zužuje na státní úřady zabývající se duševním vlastnictvím.

2.2.1.1 ÚPV

Jako základní zdroj připadá v úvahu český ÚPV. Ten publikuje svá data o ochranných známkách jako otevřená data. Úřad jednou publikoval celý export databáze a následně tento export pravidelně aktualizuje pomocí balíčků se změnami databáze. Pro zprovoznění aktuální verze databáze je tedy zapotřebí zpracovat původní, celý export databáze a následně postupně aktualizovat databázi s pomocí změnových souborů.

Pro naše využití je však tento přístup vhodný. Zajímají nás pouze nové přihlášky a ty v těchto změnových souborech jednoduše nalezneme. Úřad pro publikaci dat zvolil formát XML a publikuje vždy dvě verze, tj. data ve vlastní XSD specifikaci a data podle WIPO standardu ST.96. Hlavní rozdíl představuje struktura ZIP souboru, který je publikován. V případě vlastní specifikace úřadu je přímo v kořenu ZIP souboru XML soubor, který obsahuje všechny změny obalené elementem transakce, veškeré přílohy se také nachází v kořeni. V případě změnového souboru podle WIPO standardu jsou však v kořenu ZIP souboru složky, pro každou známku jedna, a v nich jsou jednotlivé přílohy a XML soubor s informací o konkrétní ochranné známce.

Ukázka změnového souboru podle vlastní specifikace ÚPV je v příloze A.1. Obsahuje transakci s jednou ochrannou známkou. Z ukázky je vynechán popis tříd, pro které je požadována ochrana.

2.2.1.2 EUIPO

České společnosti obvykle nejdříve expandují do ostatních zemí EU. Proto jako další možný zdroj dat o přihláškách ochranných známek připadá v úvahu EUIPO. Jedná se o úřad, který sídlí ve Španělsku a zpracovává registrace evropských ochranných známek platných v celé EU. Pro přístup k datům o evropských ochranných známkách poskytuje REST API.

Aby však bylo možné se tohoto REST API na data dotázat, je nutné si u EUIPO vytvořit uživatelský účet, přidat aplikaci, která bude data využívat a poslat žádost, v níž odsouhlasíme podmínky užití. Žádost musí být následně schválena. EUIPO také poskytuje sandboxové prostředí tohoto API, které lze využít pro vývoj, a schválení aplikace zde netrvá dlouho. Nevýhodou je, že i pro přístup k sandboxovému API je nutné požádat, a tato žádost podléhá manuální kontrole.

Po povolení přístupu k REST API, si můžeme pomocí OAuth2 vystavit token, se kterým se budeme u API autorizovat. Pro vyhledávání konkrétních

známek lze využít vcelku komplexního dotazovacího jazyka, kterým stačí formulovat požadovaný dotaz. Následná odpověď je pak ve formátu JSON a podporuje stránkování. Je nutné zmínit, že toto API implementuje rate-limiting, a je proto nutné využívat jeho zdroje s rozmyslem. Ukázkový výstup doložený v příloze A.2 ukazuje odpověď na volání seznamu ochranných známek.

Pokud projdeme ukázkový výstup z přílohy A.2, zjistíme, že každá známka obsahuje „markBasis“ atribut. Ten podle dokumentace může nabývat následujících hodnot:

EU_TRADEMARK značí evropskou ochrannou známku.

INTERNATIONAL_TRADEMARK má označovat mezinárodní ochrannou známku.[29]

Ochranné známky, které jsou dle tohoto atributu označeny jako mezinárodní, pocházejí od organizace WIPO, jež s EUIPO spolupracuje. Jedná se o ochranné známky s designací v EU a využívá je i nástroj EUIPO pro vyhledávání ochranných známek TMView.[30, 31]

2.2.1.3 WIPO

Pro co nejširší ochranu registrují společnosti ochranné známky u organizace WIPO, která jim dokáže na základě mezinárodních smluv poskytnout téměř celosvětovou ochranu jednou přihláškou. Zároveň se tato organizace snaží vytvořit komplexní nástroj pro vyhledávání ve své databázi a také databázi spolupracujících úřadů. V rámci její Global Brand Database tak lze manuálně vyhledávat možné podobné ochranné známky, získávat statistiky a zobrazit si mapu pokrytí.

Ačkoliv nalezneme v této databázi data z Egypta, Itálie, Německa, Súdánu a dalších států, nejsou dostupná, mimo jiné, data o českých a slovenských ochranných známkách. Další nevýhodou je pak neexistence API, skrze které by mohla aplikace s databází komunikovat. Zároveň je v rámci podmínek užití služby zakázáno programatické zpracování jejího obsahu. Jedinou možností pro práci s ní by pak přicházely v úvahu manuální exporty a jejich následné zpracování programem. Zde však nastává komplikace, protože aplikace povoluje export pouze pro prvních 180 záznamů.

2.2.2 Nově vzniklé společnosti

V případě, že si zaregistrujeme ochrannou známku shodnou, jako je název naší společnosti v obchodním rejstříku, jsme do jisté míry chránění. Podle § 132 občanského zákoníku musí být název společnosti odlišitelný od jiné společnosti[32]. To znamená, že případné pokusy o zapsání nové společnosti se stejným, nebo podobným názvem by měly být zastaveny ještě před zápisem do obchodního rejstříku samotným rejstříkovým soudem.

Jiný případ však může nastat ve chvíli, kdy ochrannou známku vlastníme, ale nemáme v obchodním rejstříku společnost, která by registrované označení užívala ve svém názvu. V tomto případě se již na tuto do určité míry automatickou ochranu při zápisu ostatních společností nelze spolehnout a právo, které nám plyne z § 8 zákona o ochranných známkách, musíme uplatňovat sami[1].

Nevýhodou je, že se o nových společnostech dozvíme až v době, kdy jsou zapsány do obchodního rejstříku, takže porušení našich práv následně musíme řešit buď smírnou cestou jejich kontaktováním, nebo se případně domáhat svých práv soudní cestou.

2.2.2.1 Datové schránky

Každé právnické osobě zapsané v obchodním rejstříku je dle § 5 zákona o elektronických úkonech a autorizované konverzi dokumentů zřízena datová schránka, kterou musí používat při komunikaci se státem[33]. Zároveň je seznam datových schránek i s dalšími identifikátory jako je IČO a název společnosti, dostupný jako otevřená data ve formátu XML, která publikuje DIA. Tento soubor je aktualizován každé tři hodiny a měl by tedy obsahovat aktuální údaje o všech právnických osobách v registru.

Následně by bylo možné nově vzniklé společnosti nacházet způsobem, kdy si k určitému datu vytvoříme databázi klíč–hodnota, kde jako klíč můžeme využít IČO subjektu. Po jejím úvodním naplnění daty ze souboru následně můžeme pro každé další stažení identifikovat nově vzniklé společnosti tak, že pokud klíč, jejich IČO, není v databázi, jedná se o nově vzniklou společnost. To lze následně i ověřit dotazem do ARES, kde zjistíme i datum vzniku společnosti.

Tento přístup, který využívá současnou legislativu, představuje velmi rychlou pomoc při identifikaci nově vzniklé společnosti, o níž si následně na základě jejího IČO můžeme vyžádat další informace z ostatních databází. Jeho nevýhodou je, že musí vyhodnocovat a uchovávat pro své fungování velké množství dat. Jeden soubor s informacemi o datových schránkách právnických osob totiž dosahuje k dnešnímu dni (12. 4. 2024) velikosti 651 MB.

2.2.2.2 ČSÚ RES

Další možností je využít dat, která ve svém RES eviduje ČSÚ podle § 20 zákona o státní statistické službě. Jedná se o seznam všech ekonomických subjektů v České republice, jejichž údaje získal registr na základě různých zákonů[34]. Pro nás je obzvlášť zajímavý údaj o vzniku společnosti a jejím názvu. Celý registr je publikován jako otevřená data ve formátu CSV a je aktualizován dvakrát měsíčně. Data jsou vždy vázána k 15. a poslednímu dni měsíce. Zároveň jsou tato data dostupná v přehledné online aplikaci, která podporuje pokročilé vyhledávání a export následných výsledků.

Samotný soubor s veškerými daty je poměrně rozsáhlý. K dnešnímu dni (12. 4. 2024) má velikost 479 MB. Velkou výhodou oproti souboru se seznamem

datových schránek je pak informace o aktuálnosti dat u každého záznamu a informace o typu poslední změny. Není proto nutné si ukládat jednotlivé verze lokálně a následně oproti nim porovnávat současnou verzi, aby bylo možné identifikovat nové společnosti, ale postačí pouze při sekvenčním zpracování souboru vyhodnocovat tyto příznaky a podle nich poté vybírat nově vzniklé společnosti.

2.2.2.3 Komerční řešení

Na trhu existuje i několik komerčních řešení, která využívají jak otevřených dat, tak jiných spoluprací, na jejichž základě pak staví rozsáhlé databáze firem a podnikatelů. Exporty z těchto databází pak prodávají dalším společnostem, které je dále využívají pro své podnikání.

V případě nově vzniklých společností se tato data nejčastěji využívají pro oslovení těchto společností s nabídkou služeb. Typicky se pak jedná o služby marketingové, právní a nabídku tvorby webových stránek. Tyto nabídky se snaží oslovit začínající podnikatele, u nichž předpokládají velkou šanci na úspěch. Výjimkou nejsou ani podvodné pokusy, které se za enormní poplatky snaží přesvědčit nově vzniklé subjekty, aby se zapsaly do jejich zcela nepotřebných katalogů a registrů.

Vzhledem k tomu, že tato řešení jsou postavena z větší části na otevřených datech, nebude jejich služeb v rámci této práce využito a tato možnost je zde uvedena pouze pro úplnost.

Kapitola 3
Návrh

Kapitola se podrobně věnuje požadované funkcionalitě systému, na jejímž základě definuje scénáře případů užití. Rozebírá navrhovanou architekturu systému a popisuje výběr technologií, ze kterých se bude systém skládat a které budou použity pro jeho vývoj. Závěrem pak prezentuje návrhy funkcionalit jednotlivých obrazovek ve formě wireframes.

3.1 Požadavky

Součástí této sekce je specifikace jak funkčních, tak nefunkčních požadavků na finální systém. Kombinace těchto dvou skupin pak definuje rozsah výsledného systému a identifikuje klíčové funkcionality.

3.1.1 Funkční

Níže je seznam 11 funkčních požadavků, které definují klíčovou funkcionalitu. Kromě požadavků, které se týkají přímo uživatele a jeho interakce se systémem, jsou definovány požadavky i přímo na aplikaci, která bude periodicky s pracovat s daty.

F1 Registrace a přihlášení – Neregistrovaný uživatel bude mít možnost si v aplikaci vytvořit svůj účet. Uživatel se poté může přihlásit a bude mu tím umožněna práce s aplikací.

F2 Úprava účtu – Uživatel bude mít možnost upravit údaje a heslo svého účtu.

F3 Obnova hesla – V případě zapomenutí hesla bude mít uživatel možnost si jej obnovit.

F4 Opakované zaslání aktivačního odkazu – V případě, že by uživatel neobržel po registraci aktivační odkaz nebo mu tento odkaz vypršel, může si nechat vygenerovat a zaslat nový.

F5 Přidání ochranné známky – Uživatel bude mít možnost do systému přidat slovní nebo obrazové označení, které chce monitorovat.

F6 Správa ochranných známek – Uživatel bude mít možnost v systému zobrazit, vyhledávat, editovat a mazat všechny ochranná známky, které do systému sám přidá.

F7 Přizpůsobení monitoringu – Uživatel bude mít možnost pro každou ochrannou známku v systému upravit vyhodnocovací algoritmus.

F8 Kolize ochranných známek – Systém bude vyhodnocovat možné kolize registrovaných ochranných známek v systému s nově podanými přihláškami u úřadů.

F9 Kolize názvů společností – Systém bude vyhodnocovat možné kolize registrovaných ochranných známek v systému s nově vytvořenými právnickými osobami v České republice.

F10 Zobrazení kolize – Uživatel si bude moct v systému zobrazit jednotlivé kolize svých ochranných známek a jejich seznam.

F11 Notifikace v případě kolize – Systém v případě detekce kolize bez zbytečného odkladu notifikuje vlastníka daného označení v systému. Způsob zaslání notifikace si definuje uživatel sám pro každou ochrannou známku.

3.1.2 Nefunkční

Pro obecné fungování systému je definováno 8 nefunkčních požadavků. Z těchto požadavků se pak vychází pro obecný návrh systému, jeho architektury a postupy při vývoji a produkci.

N1 Dostupnost přes web – Aplikace bude dostupná jakožto webová aplikace skrze internetový prohlížeč. Zároveň bude responzivní, a bude tak umožňovat pohodlný přístup z mobilních zařízení.

N2 Integrace služeb pro okamžité zasílání zpráv – Systém bude napojen na služby pro okamžité rozesílání zpráv.

N3 Objektové úložiště – Pro zjednodušení případné škálovatelnosti budou veškeré obrázky nahrané uživateli a získané jako podklady kolizí přihlášky s monitorovanou ochrannou známkou uchovávány v objektovém úložišti.

N4 Kontejnerizace – Části systému budou kontejnerizovány pro jednodušší správu, škálovatelnost a nasazení v cloudu. Zároveň tím bude zaručena vysoká podobnost vývojového a produkčního prostředí.

N5 Zdroje dat – Pro vyhodnocování kolizí bude využíváno více zdrojů otevřených dat, které publikují důvěryhodné instituce.

N6 Multijazyčnost – Webové rozhraní systému bude možné zobrazit v českém a anglickém jazyce.

N7 Asynchronní zpracování – Scraper bude pro vyhodnocování kolizí zpracovávat získaná data o přihláškách ochranných známek asynchronně s pomocí vláken.

N8 Konfigurovatelnost scraperu – Scraper bude možné konfigurovat, a přizpůsobit tak využívání zdrojů.

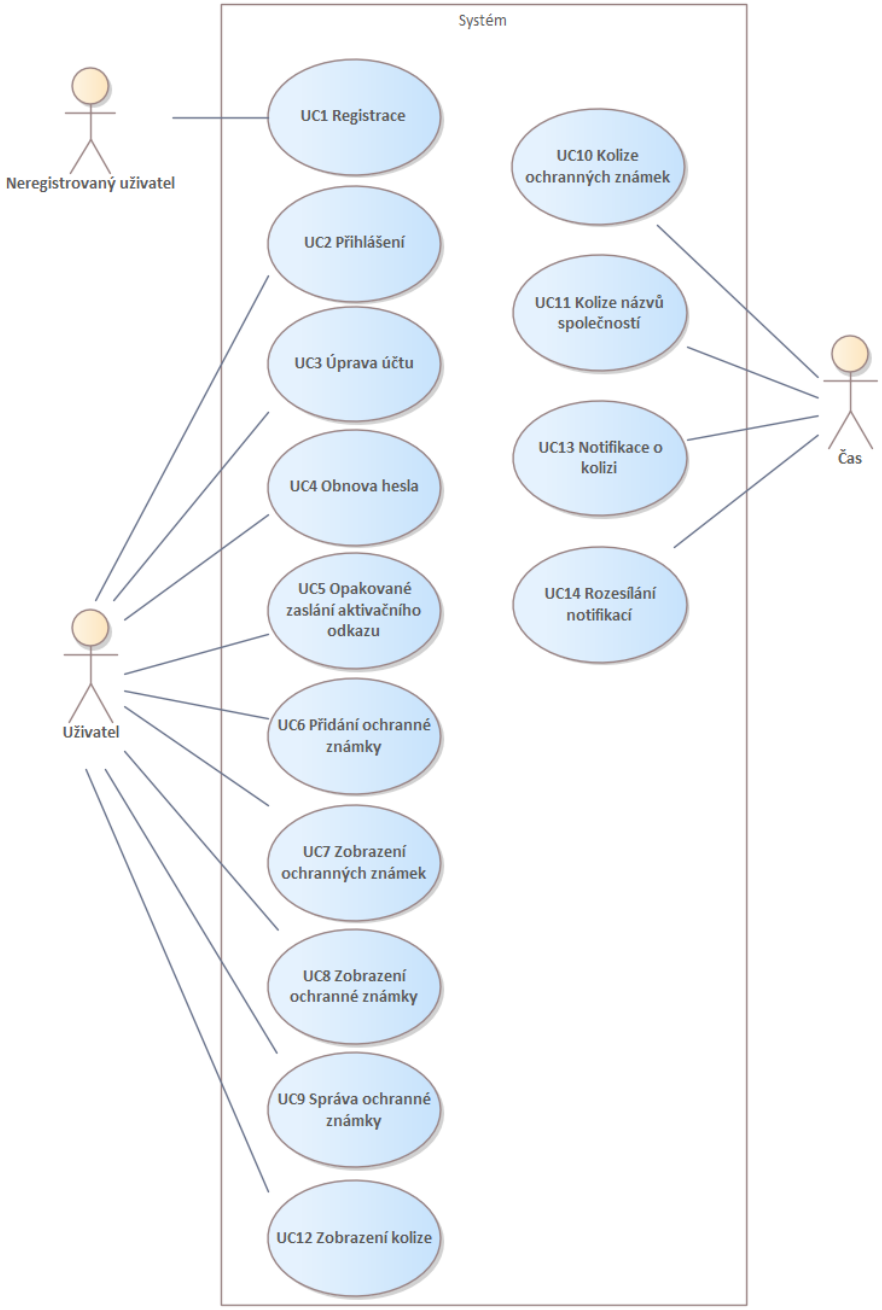
3.2 Případy užití

Pro návrh systému je nutné pochopení požadavků a na jejich základě namodelování případů užití. Níže je seznam funkcionalit, které bude systém poskytovat jednotlivým aktérům. V tomto případě budou se systémem pracovat tři aktéři. Prvním je *neregistrovaný uživatel*. Ten si kromě registrace bude moct procházet základní stránky webové aplikace. Druhým je *uživatel*. Jedná se již o osobu s vytvořeným uživatelským účtem v aplikaci. Ten má dostatečná práva pro práci s celou webovou aplikací. Posledním aktérem je pak *čas*. Na jeho základě dochází ke spouštění velké části klíčových funkcionalit v systému. Bližší znázornění aktérů a případů užití, s nimiž pracují, je k nahlédnutí na obrázku č. 3.1. Pokrytí funkčních požadavků jednotlivými případy užití znázorňuje tabulka č. 3.1.

UC1 Registrace – Umožňuje neregistrovanému uživateli vytvořit si uživatelský účet v aplikaci a získat možnost práce s aplikací.

1. Případ užití začíná příchodem neregistrovaného uživatele na stránku *Registrace*.
2. Neregistrovaný uživatel vyplní informace o své osobě pro vznik uživatelského účtu. Zejména pak jméno, e-mail a heslo.
3. Po odeslání formuláře systém provede kontrolu formuláře a duplicit. Pokud systém nevyhodnotí formulář jako bezchybný, vrátí jej uživateli s požadovanou hláškou k přepracování. V opačném případě scénář pokračuje.
4. Systém vygeneruje a odešle na zadanou e-mailovou adresu časově omezený aktivační odkaz.
5. Uživatel odkaz rozklikne, čímž potvrdí vlastnictví dané e-mailové adresy.
6. Systém účet aktivuje. Tím dojde k dokončení registrace.

UC2 Přihlášení – Umožňuje uživateli s již zaregistrovaným uživatelským účtem se přihlásit.



Obrázek 3.1 Případy užití a aktéři

1. Případ užití začíná příchodem nepřihlášeného uživatele na *Hlavní stránku* webové stránky systému.
2. Uživatel vybere v pravém rohu navigace možnost *Přihlásit*.
3. Uživatel vyplní přihlašovací formulář, který se skládá z e-mailové adresy a hesla.
4. Po odeslání formuláře provede systém kontrolu formuláře. Pokud systém nevyhodnotí formulář jako bezchybný nebo pokud nenalezne uživatelský účet, vrátí formulář s informací uživateli k přepracování. V opačném případě scénář pokračuje.
5. Systém uživatele přihlásí a změní rozložení menu pro přihlášeného uživatele.

UC3 Úprava účtu – Umožňuje přihlášenému uživateli upravit svůj uživatelský profil a nastavit kanály pro komunikaci.

Úprava základních údajů

1. Případ užití začíná příchodem uživatele na stránku *Nastavení*.
2. Uživatel vyplní formulář, kde může upravit své jméno a telefonní číslo.
3. Po odeslání formuláře provede systém kontrolu formuláře. Pokud systém nevyhodnotí formulář jako bezchybný, vrátí formulář s informací uživateli k přepracování. V opačném případě scénář pokračuje.
4. Systém provede požadovanou změnu.

Nastavení komunikačních kanálů

1. Případ užití začíná příchodem uživatele na stránku *Nastavení*.
2. Uživatel si z pravého menu vybere položku *Nastavení notifikací*.
3. Systém uživateli zobrazí informace o propojení konkrétních služeb, případně o možnostech jejich propojení, a formulář pro jejich změnu.
4. Uživatel vyplní formulář, případně v jeho rámci odpojí některou ze služeb.
5. Po odeslání formuláře provede systém kontrolu formuláře. Pokud systém nevyhodnotí formulář jako bezchybný, vrátí formulář s informací uživateli k přepracování. V opačném případě scénář pokračuje.
6. Systém provede požadovanou změnu.

Změna hesla

1. Případ užití začíná příchodem uživatele na stránku *Nastavení*.
2. Uživatel si z pravého menu vybere položku *Nastavení hesla*.
3. Uživatel vyplní formulář, kde zadá původní heslo a nové heslo s jeho potvrzením.

4. Po odeslání formuláře provede systém kontrolu formuláře. Pokud systém nevyhodnotí formulář jako bezchybný, vrátí formulář s informací uživateli k přepracování. V opačném případě scénář pokračuje.
5. Systém provede požadovanou změnu.

UC4 Obnova hesla – Umožňuje uživateli s již zaregistrovaným uživatelským účtem obnovit své heslo v případě jeho ztráty.

1. Případ užití začíná příchodem uživatele na stránku *Přihlášení*.
2. Uživatel vybere pod formulářem možnost *Zapomněli jste heslo?*.
3. Uživatel vyplní do formuláře e-mail spojený s jeho uživatelským účtem.
4. Po odeslání formuláře provede systém kontrolu formuláře. Pokud systém nevyhodnotí formulář jako bezchybný nebo pokud nenalezne uživatelský účet, vrátí formulář s informací uživateli k přepracování. V opačném případě scénář pokračuje.
5. Systém uživateli vygeneruje a odešle časově omezený odkaz pro změnu hesla na zadaný e-mail spojený s uživatelským účtem.
6. Uživatel odkaz rozklikne a zobrazí se mu formulář pro změnu hesla.
7. Po odeslání formuláře provede systém kontrolu formuláře. Pokud systém nevyhodnotí formulář jako bezchybný, vrátí formulář s informací uživateli k přepracování. V opačném případě scénář pokračuje.
8. Systém změní heslo u uživatelského účtu.

UC5 Opakované zaslání aktivačního odkazu – Umožňuje uživateli s již vytvořeným, ale neaktivovaným účtem nechat si zaslat nový aktivační odkaz v případě, že předchozí expiroval nebo nebyl doručen.

1. Případ užití začíná příchodem uživatele na stránku *Přihlášení*.
2. Uživatel vybere pod formulářem možnost *Zaslání aktivačního odkazu*.
3. Uživatel vyplní do formuláře e-mail spojený s jeho uživatelským účtem.
4. Po odeslání formuláře provede systém kontrolu formuláře. Pokud systém nevyhodnotí formulář jako bezchybný nebo pokud nenalezne uživatelský účet, vrátí formulář s informací uživateli k přepracování. V opačném případě scénář pokračuje.
5. Systém uživateli vygeneruje a odešle časově omezený aktivační odkaz na zadaný e-mail spojený s uživatelským účtem.
6. Uživatel odkaz rozklikne, čímž potvrdí vlastnictví dané e-mailové adresy.
7. Systém účet aktivuje. Tím dojde k dokončení registrace.

UC6 Přidání ochranné známky – Umožňuje uživateli přidat do systému ochrannou známku pro její monitorování.

1. Případ užití začíná příchodem uživatele na stránku *Ochranné známky*.
2. Uživatel vybere možnost *Přidat ochrannou známku*.
3. Uživatel vyplní formulář. Zejména název ochranné známky, její typ a označí notifikační kanály, skrze které chce být informován. Dále může v závislosti na typu uživatel přidat slovní vyjádření ochranné známky nebo nejvýraznější grafický prvek.
4. Uživatel může rozkliknout položku *Pokročilé nastavení*, v jejímž rámci může přizpůsobit chování vyhodnocovacího algoritmu.
5. Po odeslání formuláře provede systém kontrolu formuláře. Pokud systém nevyhodnotí formulář jako bezchybný, vrátí formulář s informací uživateli k přepracování. V opačném případě scénář pokračuje.
6. Systém přidá novou ochrannou známku a nahraje případné přílohy do objektového úložiště.

UC7 Zobrazení ochranných známek – Umožňuje uživateli zobrazit všechny jeho ochranné známky v systému.

1. Případ užití začíná příchodem uživatele na stránku *Ochranné známky*.
2. Uživatel může pro specifikaci využít filtr k snížení počtu výsledků.
3. Systém uživateli zobrazí seznam evidovaných ochranných známek rozdělený na stránky.

UC8 Zobrazení ochranné známky – Umožňuje uživateli zobrazit detail ochranné známky.

1. Případ užití začíná příchodem uživatele na stránku *Ochranné známky*.
2. Uživatel si vybere požadovanou ochrannou známku a klikne buď na její název, nebo na tlačítko s ikonkou oka.
3. Systém uživateli zobrazí detail ochranné známky včetně seznamu již nalezených kolizí.

UC9 Správa ochranné známky – Umožňuje uživateli upravit nebo smazat již existující ochrannou známku.

Úprava ochranné známky

1. Případ užití začíná příchodem uživatele na stránku *Ochranné známky*.
2. Uživatel si vybere požadovanou ochrannou známku, se kterou chce provádět akci, a klikne buď na její název, nebo na tlačítko s ikonkou oka.
3. Uživatel klikne na tlačítko *Upravit*.
4. Případ užití pokračuje 3. krokem scénáře UC6.

Smazání ochranné známky

1. Případ užití začíná ve 2. kroku scénáře úpravy ochranné známky.
2. Uživatel klikne na tlačítko *Smazat*.
3. Systém uživateli zobrazí okno s potvrzením.
4. Pokud uživatel akci potvrdí, scénář pokračuje. V opačném případě zde končí a akce byla zrušena.
5. Systém smaže ochrannou známku z databáze a k ní připojené kolize včetně všech uložených obrázků v objektovém úložisti.

UC10 Kolize ochranných známek – Umožňuje systému vyhodnotit nové přihlášky ochranných známek a zjistit možné kolize.

1. Případ užití je periodickou událostí, která začíná spuštěním v určitém čase.
2. Systém stáhne informace o všech nových přihláškách z různých zdrojů.
3. Systém porovná nové přihlášky s ochrannými známkami v systému.
4. V případě podobnosti vytvoří systém pro danou ochrannou známku kolizi.

UC11 Kolize názvů společností – Umožňuje systému vyhodnotit názvy nově vzniklé společnosti a zjistit možné kolize.

1. Případ užití je periodickou událostí, která začíná spuštěním v určitém čase.
2. Systém stáhne informace o všech nově vzniklých společnostech v České republice.
3. Systém porovná obchodní firmy společností s ochrannými známkami v systému.
4. V případě podobnosti vytvoří systém pro danou ochrannou známku kolizi.

UC12 Zobrazení kolize – Umožňuje uživateli zobrazit detailní informace o nalezené kolizi s jeho ochrannou známkou.

1. Případ užití začíná příchodem uživatele na stránku *Kolize*.
2. Uživatel si vybere kolizi, kterou si chce zobrazit. Požadovanou kolizi zobrazí kliknutím na název ochranné známky, ke které se kolize váže, nebo na ikonku oka.
3. Systém uživateli zobrazí na jedné obrazovce vedle sebe informace o kolizní přihlášce a o ochranné známce v systému, pro kterou byla kolize detekována.

UC13 Notifikace o kolizi – Přidává systému možnost vytvořit notifikace o nových kolizích.

1. Případ užití začíná, když je v systému vytvořena nová kolize.
2. Systém zjistí pro ochrannou známku, u které byla identifikována kolize, preferované způsoby komunikace.
3. Systém vytvoří notifikaci o kolizi a pro každý způsob komunikace vytvoří jeden případ rozesílky.
4. Systém počká na signál a provede rozeslání podle UC14.

UC14 Rozeslání notifikací – Umožňuje systému rozeslat notifikace, které čekají na odeslání.

1. Případ užití je periodickou událostí, která začíná spuštěním v určitém čase.
2. Systém každý typ rozesílky notifikace zpracuje zvlášť.
3. Systém vybere všechny notifikace stejného typu rozesílky, které ještě nebyly rozeslány.
4. Systém se pokusí tyto notifikace rozeslat požadovaným způsobem.
5. Systém nastaví u úspěšně rozeslaných notifikací příznak o jejich rozeslání.

Tabulka 3.1 Pokrytí funkčních požadavků

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
UC1	X										
UC2	X										
UC3		X									
UC4			X								
UC5				X							
UC6					X	X	X				
UC7						X					
UC8						X					
UC9						X	X				
UC10								X			
UC11									X		
UC12										X	
UC13											X
UC14											X

3.3 Doménový model

V rámci doménového modelu a této sekce jsou popsány jednotlivé pojmy a entity, které jsou v rámci aplikace a této práce používány. Při implementaci bude správu databáze provádět ORM framework Doctrine, přičemž doménový model na obrázku č. 3.2 je podkladem, z něhož bude vycházet následná implementace tříd entit.

Uživatel je reprezentací konkrétního uživatele a jeho uživatelského účtu v aplikaci. Kromě běžných atributů, které se typicky vážou k uživatelskému účtu, jež má i atributy *discord* a *telegram*, které odkazují na stejnojmenné služby Discord a Telegram a uchovávají buď identifikátory z daných služeb nebo webhooky. Jejich využití je pro následné rozesílání notifikací skrze tyto kanály.

Ochranná známka nemusí být v kontextu aplikace pouze zapsané označení v registru. Je tak možné monitorovat i nezapsaná označení, což může např. identifikovat vstup možného konkurenta na širší trh. Vyplnění čísla přihlášky bude mít za důsledek pouze ignorování změn u přihlášky s daným číslem. V kontextu aplikace se tak jedná o jakékoliv označení, které je v aplikaci zaregistrováno k monitoringu.

Kolize je pak každý případ možné kolize ochranné známky v databázi aplikace s označením v přihlášce nebo v záznamu v registru. V případě, že v aplikaci budou vytvořeny dvě ochranné známky, např. Škoda a Škoda Auto, a někdo se pokusí zaregistrovat u úřadu označení Škoda, dojde v systému k vytvoření dvou kolizí, pro každou ochrannou známku jedna.

API uživatel reprezentuje konkrétní systém nebo aplikaci, která využívá API webové aplikace a je jí vygenerován token a umožněna komunikace s webovou aplikací. Podle nastavených práv se pak může dotazovat na data nebo je i měnit.

Scrape je záznam o již konkrétním stažení a vyhodnocení dat z některého zdroje. Může jej vytvářet API uživatel v případě, že má příslušné oprávnění. Vážou se k němu již konkrétní kolize, které byly v průběhu vyhodnocování získaných dat nalezeny. Při následujícím stahování dat ze stejného zdroje jsou vyhodnocována pouze data, která jsou mladší než poslední scrape.

Notifikace označuje konkrétní notifikaci, se kterou má aplikace pracovat. Může být několika předdefinovaných typů, na jejichž základě se vypisuje pomocí šablon. V atributu data má proměnné prvky reprezentované a serializované jako JSON, které lze v šablonách vypsat a jsou mezi jednotlivými typy tak proměnlivé, že by nebylo vhodné pro ně vytvářet samostatné atributy.

Rozesílka notifikace je každý jednotlivý případ rozeslání konkrétní notifikace. Pokud chceme rozeslat jednu notifikaci na e-mail, Discord i Telegram, budou pro danou notifikaci vytvořeny tři rozesílky, které se následně asynchronně zpracují.

Aktivace je uchovaný každý aktivní požadavek na aktivaci účtu. Aby nedocházela e-mailová upozornění osobě, která se neregistrovala v aplikaci, ale pouze někdo uvedl její e-mail, je nutné každý účet v aplikaci aktivovat skrze odkaz v aktivačním e-mailu.

Reset hesla je každý požadavek, který uživatel vygeneroval pro vyresetování svého zapomenutého hesla ke svému účtu.

Telegram propojení je speciální entita, která uchovává žádosti na propojení účtu v aplikaci s účtem na platformě Telegram. Její existence je nutná, protože Telegram bot API má speciální pravidla pro komunikaci s uživateli, která jsou popsána v implementaci tohoto řešení v sekci 4.1.3.3.

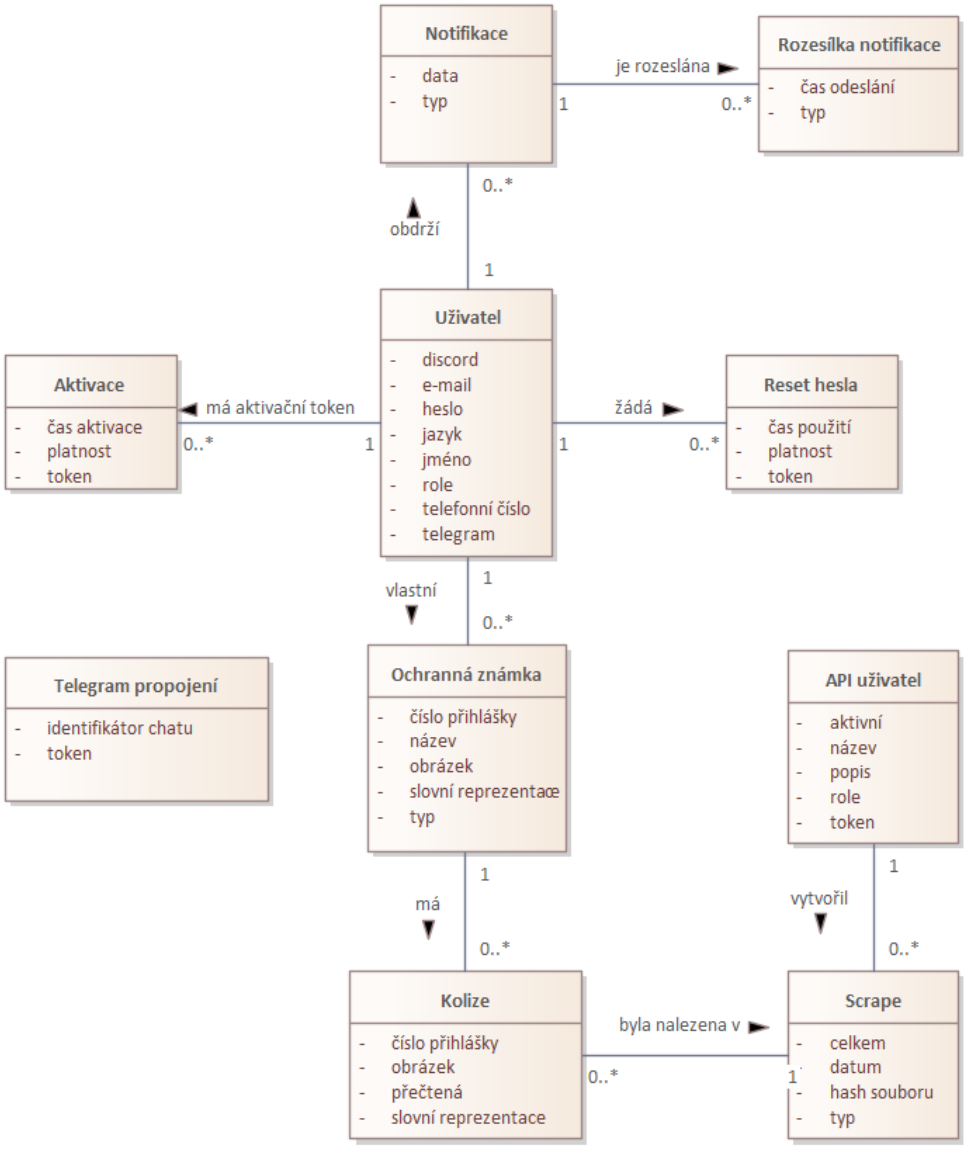
3.4 Architektura

Celý systém bude složen z několika částí, které společně zajistí sběr, uchovávání a vyhodnocování dat, rozesílání notifikací a komunikaci s uživatelem. Pro zjednodušení nejen vývoje, ale i následného provozu jsou jednotlivé části aplikace kontejnerizovány a pro jejich správu je využit Docker. Samotné rozvržení a komunikace jednotlivých částí systému jsou zachyceny na diagramu na obrázku č. 3.3.

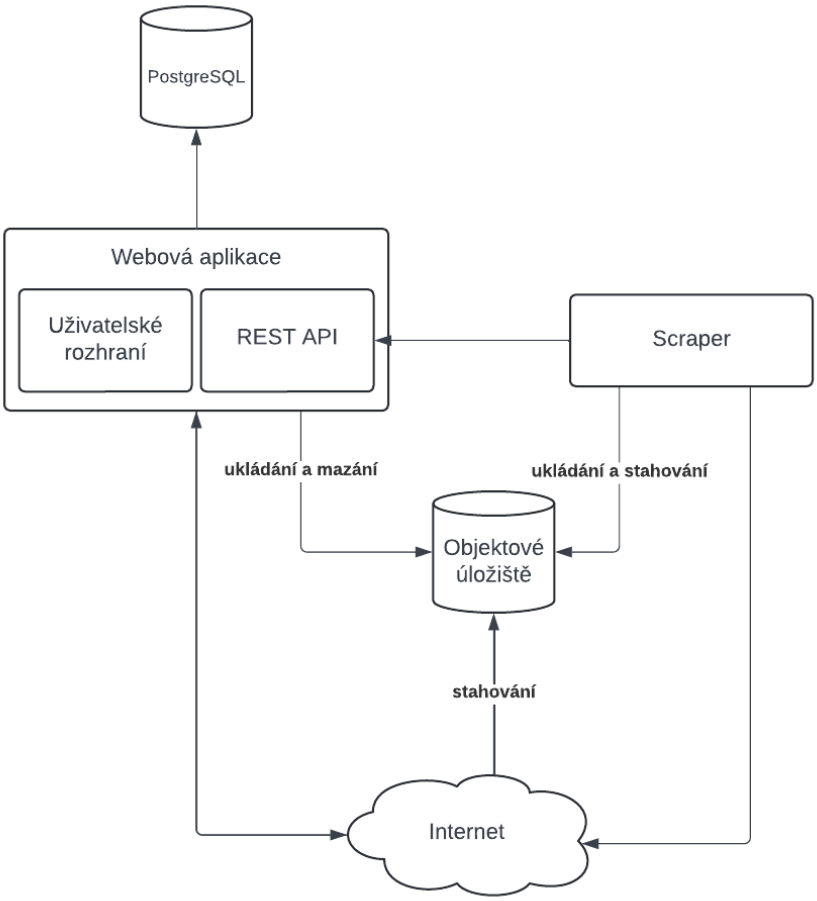
3.4.1 Webová aplikace

Veškerá komunikace jak s uživatelem, tak s programem pro vyhodnocování kolizí bude probíhat přes internet skrze webovou aplikaci. Ta bude pro běžné uživatele nabízet přehledné prostředí s jednoduchou správou. Pouze webová aplikace bude mít přístup k databázi, a tak bude pro ostatní aplikace a systémy vystaveno REST API pro předávání dat.

Aplikace bude naprogramována v interpretovaném jazyce PHP s pomocí populárního frameworku Symfony, který se společně s frameworkem Laravel řadí k těm nejvíce používaným. Byl zvolen pro jeho flexibilitu, množství knihoven a především početnou a aktivní komunitu, která jej udržuje a vyvíjí[35, 36, 37]. Jazyk PHP byl zvolen zejména pro rychlost vývoje, rozšířenost a díky velkému posunu v nových verzích, zejména 7 a 8. Pro zlepšení přívětivosti uživatelského rozhraní aplikace bude využit oblíbený a hojně využívaný CSS framework Bootstrap[38].



Obrázek 3.2 Doménový model



Obrázek 3.3 Architektura systému

3.4.2 Databáze

Veškerá data, se kterými pracuje nejen webová aplikace, ale i na ní závisející systémy, budou uchovávána v centrální databázi běžící na otevřeném řešení PostgreSQL[39]. Jedná se o robustní řešení databázového serveru s rozsáhlou komunitou, která jej udržuje. Zároveň mnozí cloudoví poskytovatelé nabízejí řešení pro jednoduché škálování tohoto velice populárního ORDBMS.

3.4.3 Scraper

Ačkoliv se v pravém slova smyslu nejedná o scraper, protože pro sběr dat využívá již existující API nebo přímo stahuje sady otevřených dat, budu program, který tato data sbírá a vyhodnocuje kolize, nazývat pro jeho funkčnost *scraper*. Jeho hlavním úkolem bude periodicky stahovat změny z datových zdrojů

a tato data následně konkurentně vyhodnocovat oproti registrovaným monitorovaným záznamům z databáze na možná kolizní vyjádření.

Tento program bude napsán v kompilovaném jazyce Go, zejména pro jeho rychlost běhu, rychlost kompilace a jednoduchou práci s vlákny s pomocí tzv. goroutines, které bude využívat[40]. Pro složitější porovnávání obrazových vyjádření využije knihovnu GoCV, která je v jazyce Go implementací populární knihovny pro počítačové zpracování obrazu OpenCV[41, 42].

3.4.4 Objektové úložiště

Pro zajištění jednoduché škálovatelnosti webové aplikace a zamezení zbytečného plýtvání zdrojů serveru s webovou aplikací budou obrázky nahrávané uživateli i obrázky kolizních známek nahrávané scraperem uloženy v objektovém úložišti. Tento přístup, kromě již uvedených výhod, zajistí v případě provozování aplikace u cloudových služeb i finanční úsporu, protože cena za GB objektového úložiště je zpravidla mnohem nižší, než cena za GB úložiště výpočetní instance.

Zvoleno bylo objektové úložiště MinIO, které lze jednoduše provozovat na vlastní infrastruktuře, je rychlé a zejména do budoucna kompatibilní s řešením S3 od společnosti Amazon[43].

3.5 Algoritmy pro vyhodnocování kolizí

Celý systém se ve výsledku zabývá porovnáváním textů a obrázků a měřením jejich podobnosti. Je proto zásadní, aby tyto algoritmy dokázaly co nejpřesněji identifikovat podobné prvky u různých označení, které by poté bylo možné označit za kolizní. Systém obsahuje jeden algoritmus pro vyhodnocování textových podobností a jeden pro vyhodnocování obrazových podobností.

Velice důležitý je algoritmus pro vyhodnocování slovních kolizí, protože pokud se na obrazové příloze známky objevuje text v latince, je přiložen v rámci přihlášky a je nám následně poskytnut v některém ze zdrojů dat. Algoritmus pro vyhodnocování kolizí obrázků je potřebný v případech, kdy obrázek text neobsahuje nebo by kolizní byl pouze grafický prvek.

3.5.1 Slovní reprezentace

Hlavní částí algoritmu pro vyhodnocování textové podobnosti slovních reprezentací je algoritmus pro výpočet OSA vzdálenosti popsaný v sekci 1.3.3.1. Ten je doplněn o úpravy označení a o vyhledávání podřetězců. Protože by prahové hodnoty a některé úpravy nebyly vhodné pro všechna slovní označení, pamatuje algoritmus i na úpravy chování podle přání uživatele. Návrh algoritmu tak, jak bude implementován, je popsán pseukodódem č. 3.1.

Výpis kódu 3.1 Algoritmus porovnání textové podobnosti

```
PodobnostSlovnihoOznaceni(a_1,...,a_n; b_1,...,b_m)
    a_1,...,a_n := verzalky(a_1,...,a_n)
    b_1,...,b_m := verzalky(b_1,...,b_m)

    Pokud uživatel chce převést diakritiku na ASCII znaky:
        a_1,...,a_n := diakritikaNaASCII(a_1,...,a_n)
        b_1,...,b_m := diakritikaNaASCII(b_1,...,b_m)

    Pokud práh podobnosti > 0
    a práh podobnosti ≥ OSAVzdalenost(a_1,...,a_n; b_1,...,b_m):
        Vrať jsou si podobné

    a_1,...,a_n := odstraňMezery(a_1,...,a_n)
    b_1,...,b_m := odstraňMezery(b_1,...,b_m)

    Pokud uživatel chce kontrolovat a podřetězec b
    a n > 2 a a je podřetězec b:
        Vrať jsou si podobné
    Pokud uživatel chce kontrolovat b podřetězec a
    a m > 2 a b je podřetězec a:
        Vrať jsou si podobné

    Vrať nejsou si podobné
```

3.5.2 Grafická reprezentace

Porovnávání grafické podobnosti je oproti porovnání textové podobnosti mnohem náročnější. Pro porovnávání bude systém využívat metodu Template matching, která je implementována v knihovně OpenCV. Za šablonu bude vždy považován obrázek nahraný v systému. Předlohou, ve které bude algoritmus šablonu vyhledávat, budou přílohy přihlášek. Za pomoci této metody získáme maximální korelaci, která se v obrázku vyskytla. Tu vynásobíme 100 a budeme považovat za podobnost v procentech. Zároveň tato metoda předpokládá, že předloha bude větší než šablona. Pokud by tomu bylo naopak, bude nutné jejich velikosti upravit. Výsledný algoritmus, který bude rozhodovat o podobnosti podle uživatelem zvoleného prahu, je přiblížen na pseudokódu č. 3.2.

Výpis kódu 3.2 Algoritmus porovnání grafické podobnosti

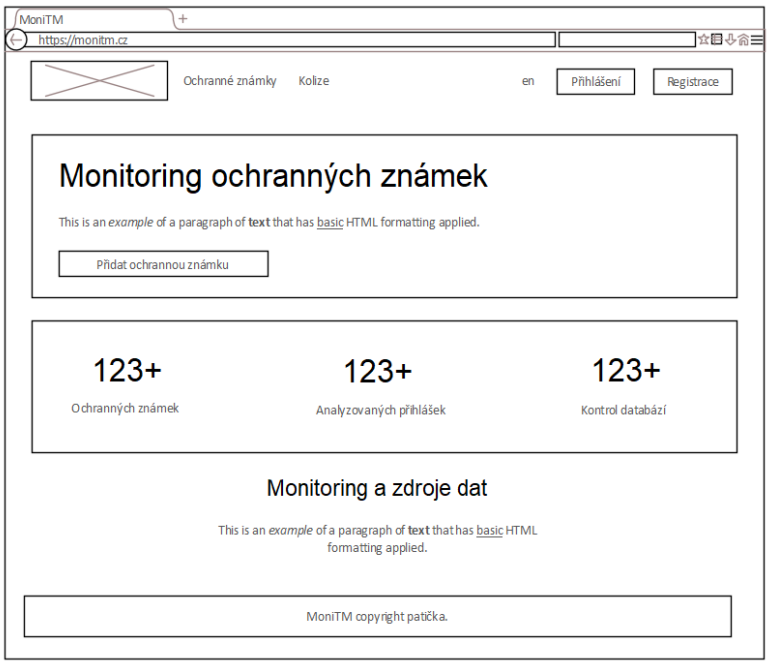
```
PodobnostObrazku(cestaKPředloze, cestaKŠabloně)
    předloha := načti(cestaKPředloze)
    šablona := načti(cestaKŠabloně)

    sPředlohy, vPředlohy := rozměry(předloha)
    sŠablony, vŠablony := rozměry(šablona)

    Pokud sŠablony ≥ sPředlohy nebo vŠablony ≥ vPředlohy:
        sPoměr, vPoměr := poměryStran(předloha, šablona)
        předloha := zvětšit(předloha, max(sPoměr, vPoměr))

    vysledek := TemplateMatching(předloha, šablona)
    korelace := MaxKorelace(vysledek)

    Pokud práh podobnosti ≤ korelace·100:
        Vrať jsou si podobné

    Vrať nejsou si podobné
```

3.6 Wireframes

Wireframe umožňuje popsat rozmístění funkčních prvků při návrhu webu. Jedná se o skicu podoby webu, vytvářející kostru návrhu. V případě webové aplikace byl využit pro definování hlavní stránky, a to i v mobilní verzi, seznamu ochranných známek a detailu ochranné známky. Ostatní stránky na webu pak budou modifikací těchto připravených návrhů, které vycházejí z případů užití.

Na obrázku č. 3.4 je wireframe hlavní stránky webu, která by kromě zobrazení základních informací o systému měla uživatele zaujmou a podprahově jej vyzvat k vytvoření účtu ve webové aplikaci. Zároveň lze na hlavní stránce zobrazit zajímavé statistiky, které by mohly v uživateli vzbuzovat větší pocit důvěry. Součástí tohoto wireframe je i návrh rozložení prvků v menu, které se bude zobrazovat nepřihlášeným uživatelům. Hlavní stránku pak lze nalézt ještě na obrázku č. 3.8, který obsahuje wireframe její mobilní verze.

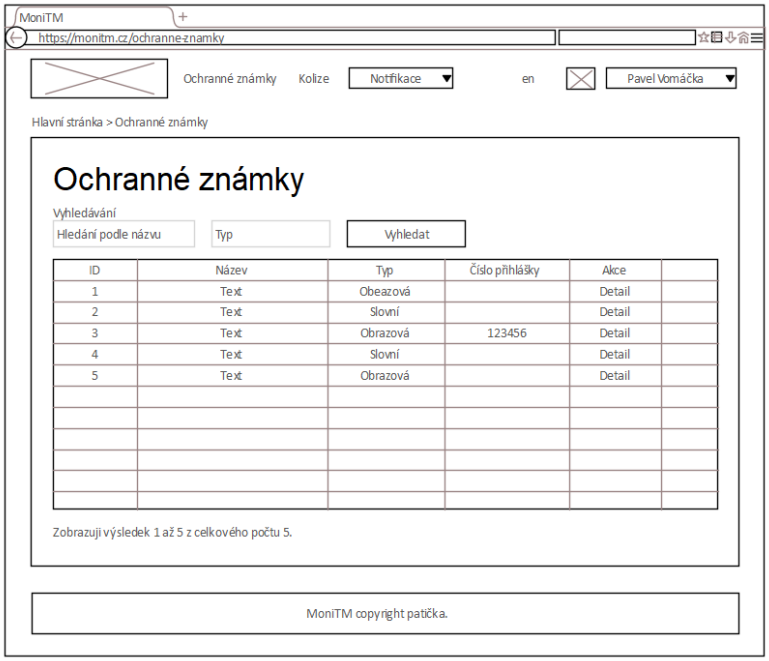
Následující wireframe na obrázku č. 3.5 zobrazuje rozvržení stránky výpisu ochranných známek, které má uživatel v systému zaregistrované, a vychází z UC7. Pro komfort uživatele je přidáno i vyhledávání. Aby se v rámci aplikace uživatel neztratil, je doprovázen drobečkovou navigací. Od tohoto rozvržení následně vychází i další stránky v systému, které využívají tabulky jakožto funkční dominanty. Na tomto wireframe stojí také za povšimnutí roz-



Obrázek 3.4 Wireframe hlavní stránky

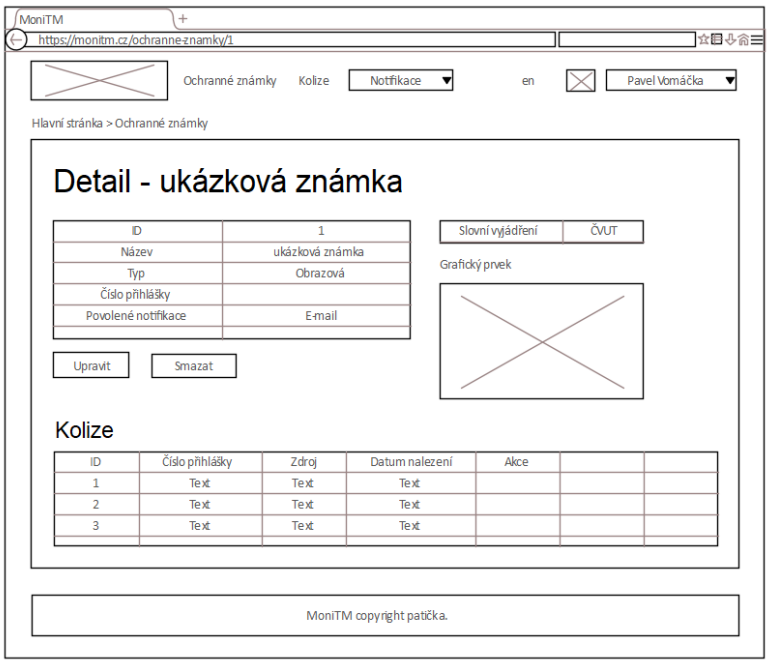
ložení prvků v menu pro již přihlášeného uživatele, jehož součástí jsou rozbalovací prvky uživatelského účtu a notifikací, které lze na obrázku č. 3.6 nalézt rozbalené.

Poslední obrázek č. 3.7 je návrhem rozvržení detailu ochranné známky registrované v systému podle UC8. Obsahuje jak informace o ochranné známce, tak viditelně možnosti další práce s ní v systému. Konkrétně její úpravu, smazání nebo proklik na nalezené kolize.

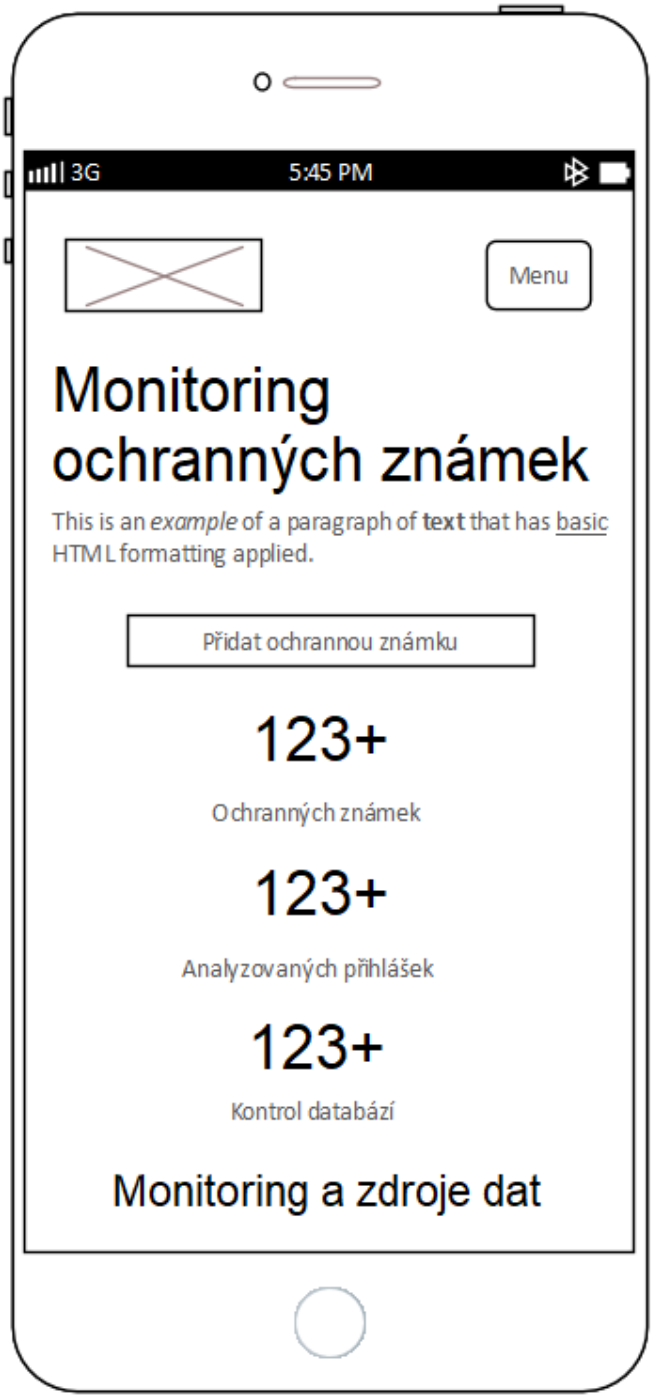


Obrázek 3.5 Wireframe seznamu ochranných známek

Obrázek 3.6 Wireframe seznamu ochranných známek s rozbaleným menu



Obrázek 3.7 Wireframe detailního zobrazení ochranné známky



Obrázek 3.8 Wireframe mobilní verze hlavní stránky

Kapitola 4
Implementace

Tato kapitola se věnuje popisu vývoje systému a nastavení prostředí, ve kterém je vyvíjen. Popisuje konkrétní přístupy a rozebírá i problémy a určité výzvy, s nimiž jsme se při vývoji setkali. Pro jednodušší označování dostal pracovní název *MoniTM*, který vychází ze zkrácení „Monitoring Trademarks“ a je inspirován názvem nástroje EUIPO *TMView*. Je nutné zdůraznit, že se jedná pouze o pracovní název a v produkčním prostředí by nebylo s nejvyšší pravděpodobností možné jej použít. Existují totiž ochranné známky s označením „Moni“ a tento název by pak mohl porušovat práva vlastníků těchto ochranných známek.

4.1 Webová aplikace

Webová aplikace je středobodem systému, neboť seskupuje API a uživatelské rozhraní. Kromě toho zajišťuje i rozesílání notifikací, které se v aplikaci vyskytnou. Jak již bylo specifikováno v kapitole návrhu, bude webová aplikace napsána v jazyce PHP s pomocí frameworku Symfony. To značně zjednodušuje a zrychluje samotný vývoj, protože velké množství problémů je již řešeno existujícími knihovnami.

4.1.1 API

Pro vývoj API nejsou v rámci Symfony použity knihovny, které by generovaly pro specifikované entity CRUD operace. API tedy využívá základu Symfony verze 7. Přístup pro stavbu API byl zvolen REST a odpovědi jsou poskytovány ve formátu JSON. Definujeme tři zdroje, skrze které lze modifikovat nebo získávat informace. Je možné pracovat s ochrannými známkami, kolizemi a jednotlivými scrapes, které označují konkrétní záznamy o stažení a vyhodnocení dat.

Celé API pracuje s DTO, které se dělí na dvě skupiny. Jedná se o DTO pro

práci s odchozími daty, mají příponu *Response*, a o DTO pro práci s příchozími daty, mají příponu *Request*. Response třídy implementují metody pro výběr dat z entit pro výpis, které jsou následně serializovány do formátu JSON. V případě Request tříd dochází k pokusu o deserializaci příchozích dat v těle požadavku a jejich následné jednodušší využití při práci v aplikaci.

V tuto chvíli jsou tyto zdroje používány pouze aplikací. Samotný návrh API ale pamatuje i na možné budoucí rozšíření, kdy by API mohly používat i jiné aplikace, a tak jsou pro jednotlivé aplikace generovány přístupové klíče s oprávněními, které mohou mít až tři role. Lze tak povolit čtení dat, jejich zápis a zároveň i speciální přístupy pouze pro scraper aplikace.

4.1.1.1 Autentizace

Každá aplikace, která bude chtít s API komunikovat, obdrží vygenerovaný klíč, s jehož pomocí se poté může autentizovat. Tento klíč je poté nutné zasílat s každým požadavkem v rámci hlavičky *X-AUTH-TOKEN*. S tou pracuje vlastní autentikátor, který rozšiřuje Symfony balíček Security[44] a je využíván ve webové aplikaci výhradně pro cesty s prefixem */api*.

4.1.1.2 Zdroje

V rámci obecného popisu API byly zmíněny tři zdroje, které se v aplikaci vyskytují. Ty jsou v aplikaci implementovány jako samostatné kontrolery a jejich metody pak řeší dotazování a změnu dat v závislosti na použitých HTTP metodách a informacích v cestě. Jejich bližší popis, včetně podporovaných metod je zmíněn níže.

Ochranné známky lze pouze získávat skrze jejich zdroj př. (4.1). Na druhou stranu je možné filtrovat podle názvu př. (4.2) a typu př. (4.3). Zároveň se očekává, že jedno volání může obsahovat velké množství výsledků. Proto je implementována paginace př. (4.2), v jejímž rámci lze i specifikovat počet požadovaných výsledků př. (4.3).

```
GET /api/trademark                                    (4.1)
GET /api/trademark?query=nazev%20znamky&page=2        (4.2)
GET /api/trademark?type=word&perPage=100              (4.3)
```

Scrape umožňuje se zdrojem více operací a kromě pouhého dotazování př. (4.4) je možné i přidávání nových scrapes př. (4.6). Samotný scraper si totiž nejdříve stáhne informace o posledním běhu pro daný typ zdroje dat pomocí př. (4.5) a následně po konci vyhodnocování kolizí vytvoří nový záznam o stažení dat př. (4.6). Stejně jako u ochranných známek je i zde implementována paginace pro jednodušší získání posledního běhu v souladu s REST přístupem.

GET /api/scrape (4.4)

GET /api/scrape?type=cz_tm_upv&page=1&perPage=1 (4.5)

POST /api/scrape (4.6)

Výpis kódu 4.1 Příklad těla požadavku pro vytvoření nového scrape pomocí př. (4.6)

```
{
    "type": "eu_tm_ipo",
    "date": "2024-01-13 13:30:42",
    "total": 100
}
```

Kolize jsou důležitým zdrojem, který povoluje pouze zápis. Aplikace může zapisovat jednotlivé kolize jednotlivě př. (4.7) nebo dávkově př. (4.8). Při dávkovém zpracování se odpověd skládá z několika odpovědí, pro každý pokus o vytvoření nové kolize je v odpovědi informace o jeho provedení.

POST /api/trademark/id/collision (4.7)

POST /api/collision (4.8)

Dávkový přístup, který využívá zdroj kolizí, sice porušuje REST přístup a návrhově lepším řešením by byl speciální hromadný zdroj, např. */api/bulk*, jenž by obdržel seznam požadavků, které se mají v aplikaci vykonat. Tím by ušetřil jejich jednotlivý přenos přes linku. V rámci Symfony tento postup není obtížné implementovat, protože lze využít subpožadavků, které Symfony nabízí. Nevýhodou tohoto přístupu by však byla rychlost, neboť by pro každý subpožadavek byl znovu inicializován kernel celé aplikace. Za cenu lehkého porušení REST přístupu tak získáváme zrychlení zápisu kolizí pomocí dávek a zároveň šetříme provoz na lince.

4.1.2 Uživatelské rozhraní

Uživatelské rozhraní webové aplikace je, pokud nebudeme počítat komunikační kanály, jejichž prostřednictvím se rozesílají notifikace, jediným bodem pro komunikaci uživatele se systémem. Celý návrh uživatelského rozhraní vychází z wireframes popsaných v sekci 3.6. Ty vycházejí z již definovaných funkčních požadavků a případů užití. Vzhled vychází z komponent frameworku Bootstrap[38], který je velmi rozšířený, a pro uživatele by tak měl prezentovat vzhled a rozložení, s nimiž se již někdy setkal.

Výpis kódu 4.2 Příklad těla požadavku pro dávkové vytvoření nových kolizí pomocí př. (4.8)

```
{
    "scrapeId": 1,
    "collisions": [
        {
            "trademarkId": 1,
            "applicationNumber": "O-1234567",
            "wording": "ČVUT"
        },
        {
            "trademarkId": 999,
            "applicationNumber": "O-7654321",
            "wording": "FIT"
        }
    ]
}
```

Výpis kódu 4.3 Příklad těla odpovědi dávkového vytvoření nových kolizí pomocí př. (4.8)

```
{
    "O-1234567": {
        "data": {
            "id": 1,
            "applicationNumber": "O-1234567",
            "wording": "ČVUT",
            "trademarkId": 1,
            "scrapeId": 1
        },
        "status": {
            "code": 200,
            "message": "OK."
        }
    },
    "O-7654321": {
        "status": {
            "code": 404,
            "message": "Trademark with this ID was not found."
        }
    }
}
```

Jedním z požadavků je také multijazyčnost webové aplikace. Ta má umožnit používání aplikace co nejširšímu okruhu uživatelů. Vzhledem k tomu, že aplikace vyhodnocuje kolize primárně v České republice, je hlavním jazykem čeština. Jako druhý jazyk byla zvolena celosvětově rozšířená angličtina. Přidání více jazyků do aplikace není nijak náročné. Multijazyčnost je implementována s pomocí Symfony Translations[45] a pro nový jazyk tak stačí přidat pouze překlady řetězců v několika souborech.

Velká část uživatelů v dnešní době pracuje s webem zejména ze svého mobilního telefonu. Je proto nutností, aby uživatelské rozhraní bylo responzivní a dobře dostupné z mobilního zařízení. Aplikace tento požadavek splňuje a je díky použitému frameworku Bootstrap responzivní. V několika částech aplikace se vyskytují tabulky. Zobrazení tabulek na mobilním zařízení je náročné a řešení jsou často nepřehledná. Běžně se proto používají dva přístupy. Jednou možností je tyto tabulky zobrazovat stejně jako na počítači, tj. jen v elementu, ve kterém si potom může uživatel tabulku posouvat. Druhou možností je převod tabulky do vertikální orientace. Výhodou tohoto způsobu je, že uživatel to, co potřebuje, má vždy na obrazovce. Dochází ale ke ztrátě přehlednosti, kterou tabulky mají. V aplikaci je využíváno první zmíněné metody.

Pro zlepšení orientace uživatelů v aplikace je přidána drobečková navigace. Ta ukazuje hierarchii dané stránky v aplikaci a umožňuje rychlý návrat na stránky, které jsou hierarchicky výše. Drobečková navigace je v aplikaci implementována jako Twig komponenta využívající balíčku *symfony/ux-twig-component*[46].

Hlavní stránka – První, co každý uživatel uvidí, je hlavní stránka aplikace (obrázek č. C.1). Ta jako první vyzývá uživatele k akci, kterou je přidání ochranné známky do monitoringu. V případě neregistrovaného uživatele bude této akci předcházet ještě registrace. Součástí hlavní stránky jsou i statistiky systému, prezentující počet monitorovaných ochranných známek, počet porovnaných přihlášek a počet všech stažení datových sad. Tyto statistiky jsou jednou za den přepočítány a následně uloženy do cache aplikace, která pro její správu využívá Symfony Cache komponenty[47]. Posledním výrazným elementem je informační sekce, jež uživateli přibližuje zdroje dat aplikace.

Přihlášení – Stránka přihlášení (obrázek č. C.2) by sama o sobě nebyla nijak zajímavá, pokud by valnou většinu autentizace a potažmo i autorizace nezpracovával již zmíněný Symfony Security balíček, který s minimálním vývojem dokáže už „přímo z krabičky“ kompletně obsluhovat přihlášení pro specifikovanou entitu. Nedokáže ale povolit přihlášení pouze aktivovaným uživatelským účtům, proto bylo tedy nutné ho o tuto funkcionalitu rozšířit prostřednictvím *UserChecker*[48].

Seznam ochranných známek – Seznam ochranných známek (obrázek č. C.3) je středobodem aplikace pro přihlášeného uživatele. Lze v něm vyhle-

dávat podle jména a podle typu ochranné známky, což usnadňuje zobrazení detailních informací o požadované známce. Zároveň se předpokládá, že by uživatel mohl mít vyšší desítky ochranných známek, které potřebuje monitorovat. Pro zpřehlednění výpisu je implementováno stránkování, jež je stejně jako drobečková navigace částečně implementováno jako Twig komponenta. Důležitým prvkem seznamu ochranných známek je tlačítko pro přidání ochranné známky. V systému existuje ještě seznam kolizí monitorovaných ochranných známek. Jeho implementace je však až na drobné detaily totožná.

Přidání ochranné známky – Přidat ochrannou známku do systému ke sledování lze pouze prostřednictvím formuláře (obrázek č. C.11) k tomu určeného. Ten se skládá z několika polí, které společně zásadně ovlivňují chování systému. Prvním polem je název, který slouží pouze pro jednodušší identifikaci ochranné známky v systému. Formulář umožňuje přidat až 5 slovních vyjádření, jež chceme sledovat, ke každé ochranné známce. Zároveň v závislosti na vybraném typu lze nechat systém monitorovat i grafickou podobnost. Tato příloha se při vytváření ochranné známky nahraje do objektového úložiště. Uživatel má možnost zadat i číslo přihlášky. Pokud jej zadá, změny u přihlášky s tímto označením budou ignorovány. Poté má uživatel možnost si zvolit, jakým způsobem chce být informován o možné kolizi, pokud vůbec. Poslední částí formuláře je sekce pokročilého nastavení. V jejím rámci lze upravit vyhodnocovací algoritmus pro konkrétní ochrannou známku.

Detail – Pro zobrazení detailnějších informací slouží v systému stránky pro detail ochranné známky (obrázek č. C.4) a detail kolize (obrázek č. C.5). Stránka detailu ochranné známky obsahuje veškeré informace o konkrétní ochranné známce, které jsou systému známé. Součástí je i seznam kolizí, které se k dané ochranné známce vážou. Detail kolize je jednodušší, avšak uživateli přehledně zobrazuje data z kolizní přihlášky a data o ochranné známce vedle sebe, čímž mu zjednodušuje jejich porovnání.

Notifikace – Veškeré notifikace si lze zobrazit na stránce seznamu notifikací (obrázek č. C.6). Zároveň je zde možné si jednotlivé notifikace označit jako přečtené, nebo naopak. Zároveň si poslední tři notifikace může každý přihlášený uživatel zobrazit z menu, kde je obsahuje rozbalovací seznam (obrázek č. C.7).

Nastavení – Stránka nastavení (obrázek č. C.8) se rozděluje na tři podsekce, které se věnují nastavení různých oblastí. Hlavní stránkou u nastavení je sekce pro nastavení informací o uživatelském účtu. Následuje sekce s nastavením komunikačních kanálu, které zde lze připojit, nebo i odpojit. Poslední je sekce pro změnu hesla k uživatelskému účtu.

4.1.3 Notifikace

Očekává se, že uživatel se bude do aplikace přihlašovat pouze za účelem změny údajů a správy ochranných známek. Notifikace jsou tedy jednou z nejdůležitějších částí systému, který je vesměs automatický. Umožňují nám komunikovat změny uživateli okamžitě, aniž by musel sám kontrolovat aplikaci. V aplikaci jsou používány pro komunikaci všech změn od aktivace účtu, přes kolize až po obnovy hesel.

Notifikace jsou v aplikaci implementovány pomocí dvou entit. Tou hlavní je entita notifikace. Ta definuje typ notifikace, podle kterého se volí šablony zpráv, adresáta, příznak o přečtení notifikace ve webové aplikaci a zejména samotná data. Ta jsou ukládána jako JSON objekty, díky tomu můžeme přidávat nové typy notifikací bez zásahu do databáze. V rámci aplikace jsou reprezentována asociativním polem a při uložení do databáze jsou převedena do již zmíněného formátu JSON. Všechny své notifikace si může uživatel zobrazit ve webové aplikaci v záložce *Notifikace*.

Na notifikace jsou pak napojeny jednotlivé rozesílky notifikace. Označujeme tak vyžádané rozeslání skrze konkrétní metodu. Při rozesílce notifikace se vybere podle typu notifikace a metody rozesílky správná šablona, do které se doplní data notifikace. Poté se využije konkrétní metody k rozeslání a nastaví se příznak o odeslání.

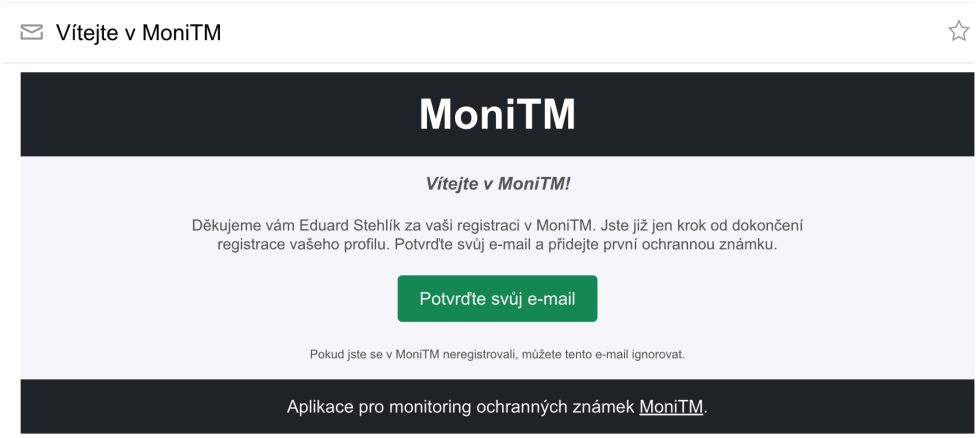
Aby mohly být notifikace multijazyčné, musela nastat u uživatele změna. Protože se notifikace mohou vytvářet i bez přímé akce uživatele, nemohli bychom z jeho požadavku identifikovat aktuální jazyk aplikace a podle toho volit, v jakém jazyce se notifikace odešle. Nyní je tedy součástí jeho účtu i nastavení jazyku komunikace. V tomto jazyce pak následně budou rozesílány veškeré notifikace. Výchozí hodnotou je jazyk, ve kterém byla webová aplikace při odeslání registračního formuláře zobrazena.

Tento přístup nám umožňuje veškeré notifikace centralizovat ve webové aplikaci a zároveň je rozesílat uživateli několika kanály najednou, aniž by docházelo ke zbytečnému zdvojování dat. Další výhodou je, že nezdržujeme uživatele a evidujeme, co aplikace odeslala. Připojení např. k SMTP serveru může být poměrně časově náročné, proto není vhodné rozesílat notifikace ihned při uživatelově akci, protože by došlo ke zpomalení procesu. Takto nám vzniká fronta, která je odbavena v určitém časovém intervalu.

4.1.3.1 E-mail

První metodou, jejímž prostřednictvím lze rozesílat jednotlivé notifikace, jsou e-maily. Jejich implementace v aplikaci není příliš složitá, sestává se z HTML šablony, do které se vkládají informace, a obyčejného připojení k SMTP serveru, jenž následně provede samotné rozeslání zpráv. SMTP server je pro provoz aplikace vyžadován, protože bez něj nelze rozeslat e-maily s aktivačními odkazy. Ukázkový aktivační e-mail je znázorněn na obrázku č. 4.1 (jedná se o snímek obrazovky ze služby Seznam Email, součástí je pro demonstraci i předmět

e-mailu tak, jak se zobrazuje ve zmiňované aplikaci), v případě ostatních typů notifikací dochází pouze ke změně textu, avšak vizuál zůstává stejný.

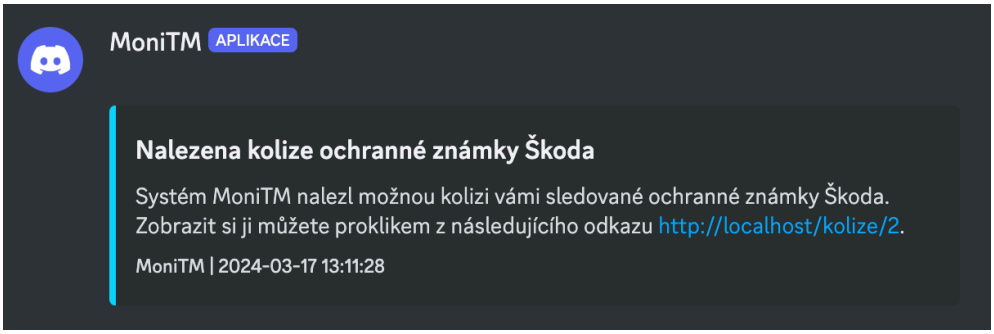


Obrázek 4.1 Ukázkový aktivační e-mail

4.1.3.2 Discord

Velice jednoduchou, ale účinnou metodou je pak využití webhooks v rámci platformy Discord. S pomocí webhooks, které můžeme vygenerovat pro konkrétní kanál serveru na platformě Discord, lze do tohoto kanálu zasílat informace. Ukázková zpráva odesílaná systémem se nachází na obrázku č. 4.2. Zároveň je nutné zmínit, že notifikace, které se týkají aktivace účtu a obnovy hesla, nelze, z pochopitelných důvodů, prostřednictvím této metody zasílat.

Uživatelské nastavení není nijak náročné. V rámci aplikace v případě úpravy kanálu uživatel zvolí možnost Propojení a následně Webhooky. Pak je samotné vytvoření již značně intuitivní. Výhodu představuje, že i rozesílání notifikací je jednoduché. Postačí na uživatelem poskytnutý webhook zaslat POST požadavek, s JSON tělem.



Obrázek 4.2 Ukázková Discord notifikace

Výpis kódu 4.4 Příklad těla požadavku na Discord webhook.

```
{
    "content": "",
    "tts": false,
    "embeds": [
        {
            "type": "rich",
            "title": "Nalezena kolize ochranné známky Škoda",
            "description": "Systém MoniTM nalezl možnou kolizi",
            "color": 54783,
            "footer": {
                "text": "MoniTM | 2024-03-17 13:11:28"
            }
        }
    ]
}
```

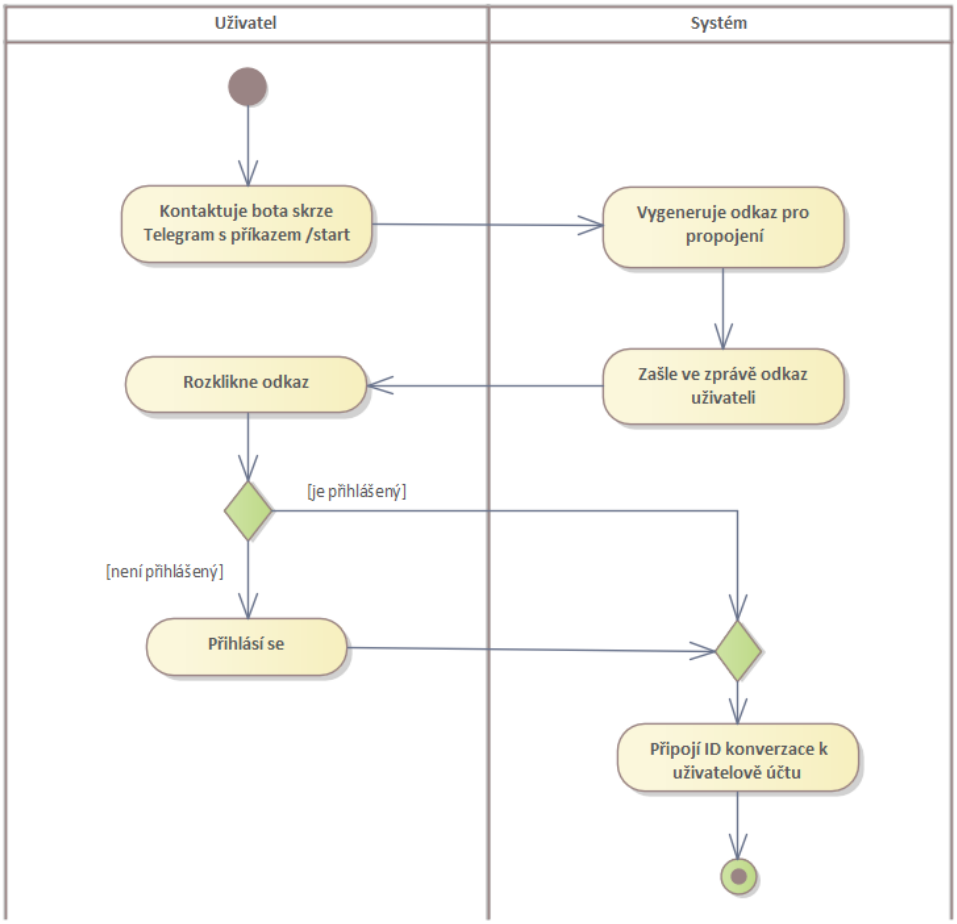
4.1.3.3 Telegram

Populární aplikací pro okamžité doručování zpráv s podporou šifrování je Telegram. Vývojářům vychází velice vstříc tím, že poskytuje API pro komunikaci s uživateli za využití botů. Jedná se o profily, které jsou ovládány čistě prostřednictvím Telegram API.

Uživatelé jsou v rámci platformy Telegram identifikováni buď pomocí jejich telefonních čísel, nebo pomocí jejich uživatelských jmen, která si zvolí. Samotná implementace rozesílání zpráv je ale náročnější, než v případě platformy Discord. Z důvodu ochrany uživatelů neumožňuje Telegram botům kontaktovat uživatele jako první pomocí těchto identifikátorů. Aby mohl bot s uživatelem komunikovat, musí tuto komunikaci iniciovat sám uživatel.[49]

Pro rozesílání notifikací proto musí nejdříve sám uživatel kontaktovat našeho bota prostřednictvím svého účtu. Zde však dochází ke komplikaci, přestože získáme tuto zprávu a můžeme již s pomocí ID této komunikace uživateli zasílat zprávy, nedokážeme si spojit uživatelský účet v systému s tím, který nás právě kontaktoval skrze Telegram. Je proto nutné tyto účty propojit. Za účelem propojení vygenerujeme unikátní odkaz a uchováme si v jeho spojení ID komunikace. Uživateli tento odkaz zašleme přes Telegram. Po jeho rozkliknutí se uživatel přihlásí, takže si již dokážeme spojit ID komunikace s konkrétním účtem v aplikaci.

V první řadě je nutné bota v platformě zaregistrovat. To je poměrně nenáročné, stačí napsat profilu *@BotFather*, který nás provede procesem registrace, v jejímž závěru pak získáme API token pro komunikaci. Běžně se komunikace s botem řeší pomocí příkazů, které začínají lomítkem. Pro zahájení komunikace



Obrázek 4.3 Diagram aktivit popisující propojení účtu v aplikaci s Telegram účtem

se typicky používá příkaz */start*.

Celý proces je závislý na entitě *telegram propojení*, která uchovává unikátní token a ID komunikace. Při každém obdržení příkazu */start* tak systém vygeneruje nový, unikátní, časově omezený token a uživateli zašle odkaz, který jej obsahuje a díky němuž může dokončit propojení. Pro komunikaci s API Telegram je užíváno knihovny *php-telegram-bot/core*, jež je implementací oficiálního API Telegram v PHP[50]. Aplikace, respektive přímo bot reaguje pouze na dva příkazy:

/start – Zahájí proces propojení Telegram účtu s uživatelským účtem v aplikaci.

/disconnect – Odpojí Telegram od všech účtů v aplikaci, které využívají ID konverzace, v níž byla tato zpráva zaslána.

Po propojení je již proces rozesílání zpráv jednoduchý, protože už proběhl

kontakt ze strany uživatele, a my tak můžeme využít ID komunikace pro jejich zasílání. Jedinou komplikaci představuje rate-limiting ze strany Telegramu, takže nelze zaslat za sekundu více než 30 zpráv, a musíme proto v aplikaci tento limit zohlednit[51].

4.1.3.4 Příkazy

Pro zpracování fronty nerozeslaných notifikací a pro mazání expirovaných tokenů pro propojení se službou Telegram jsou naimplementovány konzolové příkazy, které rozšiřují Symfony Console[52]. Tyto akce budou probíhat periodicky, v případě rozesílání notifikací každých 5 minut a v případě mazání expirovaných tokenů každých 30 minut a jejich správu bude zajišťovat prostřednictvím cron samotný systém. Pro rozesílání notifikací tak existuje jeden příkaz, který podle předaných volitelných přepínačů spustí konkrétní metody pro rozeslání. Na příkladu č. (4.9) lze vidět použití příkazu pro rozeslání e-mailových notifikací. Pro ostatní metody existují přepínače *discord* a *telegram*.

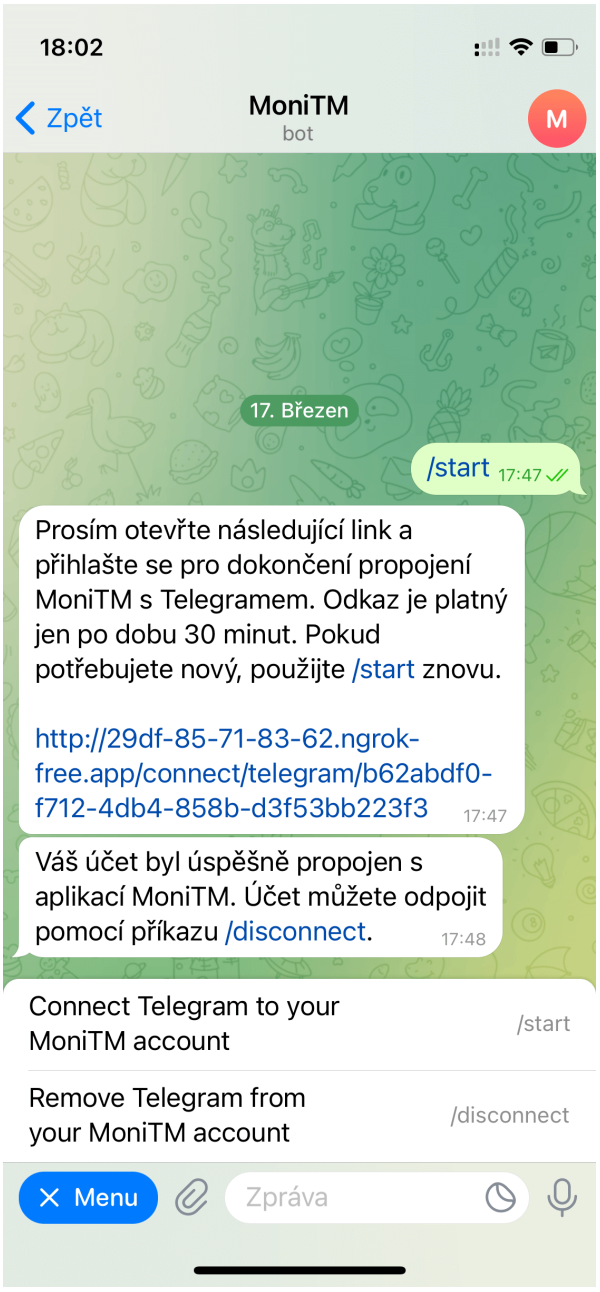
```
php bin/console notifications:send --email          (4.9)
php bin/console database:clear                      (4.10)
```

4.2 Scraper

Klíčovou částí celého systému je scraper, který agreguje data z jednotlivých zdrojů a následně vyhodnocuje možné kolize, jež nahlašuje prostřednictvím API webové aplikace. Je vyvíjen v jazyce Go známém svou rychlostí a prací s vlákny.

Implementace scraperu je rozdělena do několika balíčků, které se snaží být na sobě co nejméně závislé. Cyklus běhu začíná stejně jako u programovacího jazyka C nebo C++ ve funkci *main*. Ta poté vytváří strukturu *App*, která zapouzdřuje běh programu. V ní nejdříve dochází k načtení konfiguračního souboru, registraci loggerů a vytvoření struktury *Execute*, jež nastavuje prostředí pro spuštění jednotlivých příkazů a následně je spouští. Příkazy se v scraperu rozumí struktury implementující již konkrétní práci se zdroji dat. V jejich režii probíhá i následné spuštění vyhodnocovacích funkcí a předání výsledků struktuře *API*, která zajišťuje komunikaci s API webové aplikace.

Po doběhnutí každého příkazu dochází ke smazání dat z disku, jež se k danému příkazu vázala. Zároveň jsou při konci běhu scraperu smazány společné obrázkové cache, které sdílí všechny příkazy. Periodické spouštění scraperu zajistí pomocí cron samotný systém. Po každém úspěšném běhu zasílá scraper požadavek na službu, která monitoruje jeho zdraví. V případě, že dojde k chybě a monitor nebude spuštěn nebo nebude jeho běh úspěšný, zašle tato služba notifikaci provozovateli systému.



Obrázek 4.4 Ukázka Telegram komunikace při propojení

4.2.1 Konfigurace

Jedním z nefunkčních požadavků, konkrétně N8, je i požadavek na konfigurovatelnost. To z důvodu, aby při změně základního nastavení, případně výkonnostních úprav nemuselo znovu docházet ke kompilování celé aplikace. Program lze nastavit souborem ve formátu YAML, který se nachází v adresáři config se jménem config.yml. Kromě typických možností, jimiž jsou údaje k API a odkazy na zdroje dat, je prostřednictvím tohoto souboru umožněno vypínat nebo zapínat vyhodnocování konkrétních zdrojů a pro každý upravit, kolik vláken by měl při vyhodnocování používat.

Další možnosti přizpůsobení běhu scraperu má samotný uživatel, který si v aplikaci zaregistruje ochrannou známku pro její následný monitoring. V souladu s funkčním požadavkem F7 je mu totiž přímo z webové aplikace umožněno měnit chování algoritmu a nastavovat jiné prahové hodnoty pro detekci kolizí u konkrétních ochranných známek.

4.2.2 Ochranné známky

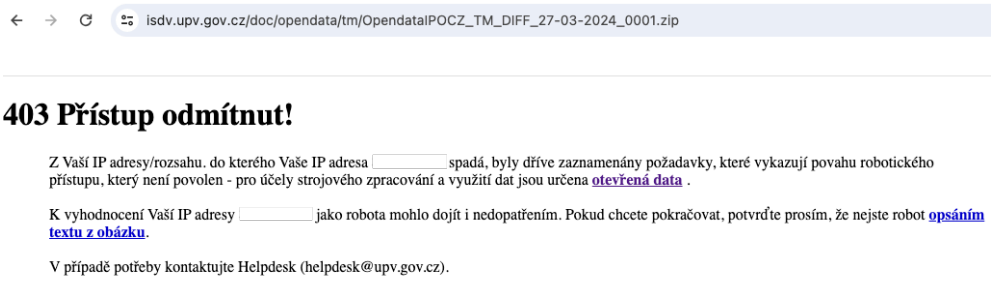
Klíčovou funkcionalitou scraperu, respektive celé aplikace je vyhodnocování kolizních přihlášek ochranných známek. Pro co nejširší záběr proto probíhá toto vyhodnocování ze dvou zdrojů. Těmi jsou český ÚPV a evropský EUIPO. ÚPV poskytuje každý všední den aktuální data o tuzemských ochranných známkách a poměrně nenáročně umožňuje detekovat nové přihlášky. EUIPO sice vyžaduje širší rozsah implementace a plánování zdrojů, na druhou stranu kromě evropských ochranných známek poskytuje i celosvětové ochranné známky s designací v EU, které získává od WIPO, viz sekce 2.2.1.2.

4.2.2.1 ÚPV

Implementaci vyhodnocování dat od ÚPV velmi zjednodušuje přístup ÚPV, který jednou publikoval celou databázi a následně zveřejňuje pouze změnové soubory, čímž pro naše potřeby usnadňuje hledání, šetří paměť a zajišťuje konzistenci.

V první řadě si scraper zjistí z API informace o posledním běhu, který se týkal dat z ÚPV. Součástí je i SHA-256 hash URL posledního souboru, který vyhodnocoval. S těmito informacemi si scraper stáhne stránku se seznamem změnových souborů a následně podle hash URL posledního souboru a hash URL těchto změnových identifikuje nezpracované, nově přidané soubory. Ty si posléze stáhne a rozbalí na disk.

Když má tyto informace již k dispozici lokálně na disku, otevře soubor, který popisuje změny. Podle jejich typu identifikuje nové přihlášky, které uloží do paměti. Podle počtu povolených vláken spustí pro každé z nich jednu vyhodnocovací funkci, která bude pracovat s $i + k$-tou stránkou seznamu známek z API, kde i je pořadí vlákna a k je počet vláken. Při počtu 4 vláken si první vlákno stáhne 1. stránku seznamu a provede porovnání se všemi přihláškami



Obrázek 4.5 Rate-limiting přístupu k otevřeným datům od ÚPV (IP adresa byla zamazána)

v paměti. Poté, co dokončí porovnání, stáhne 5. stránku seznamu a pokračuje v porovnávání. Mezitím pracuje druhé vlákno s 2. a posléze 6. stránkou atd.

Informace o právě proběhlém stažení dat a vyhodnocení kolizí jsou poté předány struktuře *API* pro jejich další zpracování.

Ačkoliv samotný ÚPV nikde užívání rate-limitingu nepřiznává, při implementaci a stahování otevřených dat, které poskytuje, došlo několikrát k zablokování přístupu pro používanou IP adresu s odkazem, že robotické zpracování běžného obsahu není povoleno a pro strojové zpracování jsou publikována otevřená data. Paradoxně je ale právě na ně v rámci zablokovaného požadavku dotazováno (obrázek č. 4.5).

4.2.2.2 EUIPO

Pro získání přístupu k Trademark Search API od EUIPO je nutné mít vytvořený EUIPO účet a schválenou aplikaci, ke které jsou vydány údaje pro komunikaci s API. Poté je možné si prostřednictvím OAuth2 protokolu a metody Client credentials nechat vystavit token a komunikovat se samotným Trademark Search API.

V dotazu je pak možné pomocí komplexního dotazovacího jazyka specifikovat, jaké ochranné známky požadujeme. Získáme tedy z API webové aplikace informace o předchozím běhu programu pro tento zdroj dat a sestavíme dotaz (kód č. 4.5) na námi požadované údaje. V něm určíme požadované stavy přihlášek a datum podání přihlášky. To musí být mladší než datum posledního běhu a zároveň starší než den, ve kterém stažení dat proběhne. Tedy při spuštění programu dnes budou analyzovány přihlášky podané nejpozději včera.

Dotaz obsahuje množství stavů přihlášek, jež se vážou jak k evropským ochranným známkám, tak k celosvětovým s designací v EU. Níže je proto pro přehlednost uveden seznam a popis vybraných stavů přihlášek, které budou systémem sledovány.

RECEIVED – Přihlášku úřad přijal, splnila formální požadavky, ale nebyl k ní ještě přiřazen pracovník úřadu.

Výpis kódu 4.5 Dotaz na Trademark Search API

```
applicationDate>2024-01-03 and applicationDate<=2024-01-01
and status=in=(
    RECEIVED,
    UNDER_EXAMINATION,
    APPLICATION_PUBLISHED,
    CANCELLATION_PENDING,
    OPPOSITION_PENDING,
    START_OF_OPPOSITION_PERIOD
)
```

UNDER_EXAMINATION – Přihláška splnila formální požadavky, byla přijata úřadem a již je přidělena pracovníkovi úřadu.

APPLICATION_PUBLISHED – Přihláška byla zveřejněna ve věstníku a veřejnost ji může namítat.

CANCELLATION_PENDING – Přihláška ochranné známky byla napadnuta a může být vymazána z rejstříku.

OPPOSITION_PENDING – Proti přihlášce byly podány námitky, ale o jejich opodstatněnosti ještě nebylo rozhodnuto.

START_OF_OPPOSITION_PERIOD – Dokumentace tento stav blíže nepopisuje[1].[29]

Po dotazu na Trademark Search API obdržíme ochranné známky a přihlášky, které vyhovují našim kritériím. Program projde všechny stránky výsledku a uloží si každou přihlášku do paměti k porovnání. V případě, že se jedná o přihlášku s obrazovou přílohou, je tato příloha stažena pro pozdější vyhodnocení.

Při vyhodnocování postupuje program stejně jako v případě dat, která získal od ÚPV. Tento proces je popsán v sekci 4.2.2.1.

Již bylo vícekrát zmíněno, že Trademark Search API implementuje rate-limiting, a je proto nutné stahování a zasílání požadavků limitovat v souladu s podmínkami užití. Za tímto účelem dochází ze strany scraperu k throttlingu požadavků, které cílí na Trademark Search API. Ve scraperu je pro toto chování používán *Ticker*, který v definovaném intervalu posílá signály do kanálu, z něhož čtou jednotlivá vlákna provádějící požadavky. Pokud přijde signál,

[1] Tento stav se podle dokumentace vztahuje pouze k mezinárodním přihláškám a ochranným známkám s designací v EU. Protože jej dokumentace ke dni citace [29] více nepopisuje a z jeho názvu se lze domnívat, že se váže k námitkovému řízení, je tento stav monitorován také.

Výpis kódu 4.6 Ukázkové použití Tickeru

```
package main

import (
    "fmt"
    "time"
)

func main() {
    throttle := time.NewTicker(time.Minute/(200 - 1))
    defer throttle.Stop()

    for i := 0; i < 100; i++ {
        <-throttle.C
        fmt.Println(i)
        // Provedení požadavku
    }
}
```

jedno z vláken jej přečte a provede požadavek na API. Demonstrace jeho užití se nachází v kódu č. 4.6.

4.2.3 Nově vzniklé společnosti

Z analýzy dostupných přístupů se jako vhodná jeví otevřená data o datových schránkách a data z RES ČSÚ. Protože v době, kdy se záznam o nové firmě objeví v jedné nebo druhé datové sadě, je již daná společnost zapsána v obchodním rejstříku a není zde lhůta pro podávání námitek, postačí pro náš účel vyhodnocovat možné kolize s delší prodlevou, pouze dvakrát měsíčně, kdy data v RES aktualizuje ČSÚ. Přístup prostřednictvím datových schránek by sice umožnil získat informace s prodlevou pouze několika hodin, ale vyžadoval by náročnější implementaci a provoz. Zároveň neexistuje relevantní důvod, proč bychom potřebovali tyto kolize vyhodnocovat tak často.

Stejně jako v ostatních případech stáhne scraper informace o posledním stažení dat z tohoto zdroje. Následně zkontroluje, zda došlo k aktualizaci CSV souboru, který obsahuje kompletní data z RES. Tato kontrola probíhá s pomocí HTTP metody HEAD, která se dotáže pouze na hlavičky odpovědi. Z obdržených hlaviček pak porovnáme datum posledního stažení s datem v hlavičce *Last-Modified*.

Pokud došlo k aktualizaci souboru, tak jej scraper stáhne a začne jej sekvenčně číst. Do paměti poté ukládá všechny společnosti, které byly aktualizovány a vyhovují sledovaným parametrům. Program tedy porovnává pouze

právnické osoby. Podle nastavení v konfiguraci tak může scraper analyzovat jen nově přidané subjekty, popř. i subjekty, u kterých došlo k aktualizaci některého z údajů. Když jsou tyto údaje ukládány do paměti, scraper ještě odstraňuje pomocí komplikovaného regulárního výrazu z názvu informaci o právní formě společnosti.

Ve chvíli, kdy jsou tyto informace uloženy v paměti, postupuje program při vyhodnocování stejně jako v případě dat, která získal od ÚPV. Tento proces je popsán v sekci 4.2.2.1.

4.2.4 Porovnávání textu

Pro porovnávání slovní reprezentace je implementován algoritmus navržený v sekci 3.5.1. Ten je v programu metodou struktury *Trademark*, která obsahuje známé informace o daném označení. Zavoláním této metody na některé ochranné známce pak dojde k porovnání s ochrannou známkou poskytnutou v argumentu metody.

Je nutné zmínit, že součást tohoto algoritmu, výpočet OSA vzdálenosti (sekce 1.3.3.1), nelze v jazyce Go implementovat naprosto přímořaře, ale je nutné brát v potaz kódování řetězců. Protože chceme porovnávat zejména české ochranné známky, které v mnohých případech obsahují diakritiku, musíme pracovat s UTF-8 kódováním. Běžné metody jako *len* nebo operátor hranatých závorek tak při práci s UTF-8 řetězci nebudou vracet správné hodnoty, protože v případě *len* dochází k počítání bytů a v případě operátoru hranatých závorek přístup na konkrétní byte v řetězci. UTF-8 řetězce však obsahují znaky, které se skládají ze sekvencí bytů[53], proto je nutné je v jazyce Go konvertovat do *slice* speciálního typu *rune*. Ten je aliasem pro typ *int32* a může tedy reprezentovat jeden Unicode znak. V případě *slice* typu *rune* pak reprezentujeme celý UTF-8 řetězec.[54] Tento rozdíl a práce s runami je zachycena na ukázce kódu č. 4.7.

Výsledná implementace navrženého algoritmu tak, jak je popsána v kapitole návrhu systému, je okomentována v příloze B.1.

4.2.5 Porovnání obrázků

I v případě porovnávání obrázků je implementován přístup již dříve přiblížený v sekci 3.5.2 a stejně jako v případě porovnávání textu je algoritmus metodou struktury *Trademark*. Díky tomu můžeme upravovat prahové hodnoty pro jednotlivá vyhodnocení tak, jak je nastavují uživatelé při vytvoření ochranné známky.

Samotný algoritmus provádí pouze kontroly, zda existují dané obrázky a následně nad nimi spouští metodu *Template matching*. Ta je obyčejnou implementací stejnojmenného přístupu, rozšířenou pouze o úpravu rozměrů obrázku a jejich kontrolu, aby nedošlo k nesplnění asercí, které kontroluje knihovna GoCV, respektive OpenCV.

Výpis kódu 4.7 Ukázka práce s typem *rune* v jazyce Go

```
package main

import (
    "fmt"
    "unicode/utf8"
)

func main() {
    retezec := "Škoda Auto"

    // Získání dély řetězce
    fmt.Println(len(retezec)) // Výsledek je 11
    fmt.Println(
        utf8.RuneCountInString(retezec),
    ) // Výsledek je 10
    // Znak Š je v UTF-8 reprezentován 2 byty
    // je tedy funkcí len započítán 2x.

    // Převod řetězce na slice run.
    sliceRun := []rune(retezec)

    fmt.Printf("%c\n", retezec[0]) // Vypíše Å
    fmt.Printf("%c\n", sliceRun[0]) // Vypíše Š
}
```

Výsledná implementace jak tohoto algoritmu (příloha B.2), tak práce s knihovnou GoCV a Template matching (příloha B.3), se nachází v přílohách této práce.

4.2.6 Komunikace s API webové aplikace

S výjimkou obrázků, které jsou uchovávány v objektovém úložišti, jsou veškerá data uložena v databázi, k níž má přístup pouze webová aplikace. Scraper s nimi může pracovat pouze za pomoci API, jež webová aplikace poskytuje. Pro práci s tímto API je tak v rámci scraperu implementována struktura *API*, která tuto komunikaci zapouzdřuje a zprostředkovává i komunikaci s objektovým úložištěm.

4.2.6.1 Report výsledků

Když pomineme dotaz na poslední běh programu, který provádí každý příkaz a byl již přiblížen v sekci 4.1.1, je zásadním prvkem při komunikaci s API report samotných výsledků. Ten pracuje ve dvou fázích. Nejprve vytvoří záznam o běhu a následně dávkově nahraje nalezené kolize, které jsou k danému záznamu přidružené. V případě, že byla v rámci nalezených kolizí detekována i obrazová kolize, je grafická reprezentace kolizní přihlášky nahrána do objektového úložiště.

4.2.6.2 Bod obnovy

V případě nynějšího stavu implementace reportování výsledků může dojít k situaci, kdy by po vytvoření záznamu o běhu selhalo z různých důvodů nahrání kolizí, čímž bychom ztratili nalezené kolize. Řešení této situace je několik a lze je implementovat jak na straně scraperu, tak na straně API webové aplikace.

Přístup, který je ve scraperu implementován a tento problém řeší, je pracovně nazván jako *bod obnovy*. V případě, že by došlo k chybě, jež by nastala mezi vytvořením záznamu o běhu a nahráním kolizí, serializuje program data o kolizích a uloží je na disk. Následně při každém spuštění program kontroluje, zda existuje soubor s těmito daty, a v případě, že ano, provede nejdříve jejich nahrání.

Za situace, kdy by nastala chyba i při vytváření bodu obnovy, je tento stav zalogován a podle nastavení aplikace je o této mimořádnosti informován administrátor prostřednictvím platformy Discord.

4.2.7 Logování

Pro porozumění chybám a neočekávaným událostem je v programu implementováno rozsáhlé logování, které propojuje několik přístupů. Pro logování se v aplikaci používá hlavní logger implementující návrhový vzor pozorovatel. Program nejdříve zaregistruje ostatní, již konkrétní implementace logování u tohoto hlavního loggeru. Pozorovatel v případě, že dojde k logování jeho prostřednictvím, notifikuje své posluchače o nastalé události (kód č. 4.8). V programu jsou nyní implementováni tři posluchači.

Souborový logger ukládá informace o chybách a událostech do souboru na disku, jehož cesta je upravitelná v konfiguračním souboru.

Stderr logger vypisuje informace o chybách a událostech na standardní chybový výstup programu.

Discord logger pak v případě zapnutého produkčního prostředí zasílá prostřednictvím webhooku zprávy o chybách a událostech do vybraného kanálu na serveru na platformě Discord.

Výpis kódu 4.8 Notifikace posluchačů hlavního loggeru

```
func (logger *Logger) Log(err error, level types.LogType) {
    if level == types.LogDebug && !logger.dev {
^^Ireturn
    }
    for _, cLogger := range logger.loggers {
        err := (*cLogger).Log(err, level)
        if err != nil {
            utils.StderrLogError(
                fmt.Errorf(
                    "%s failed to log: %s",
                    (*cLogger).Name(),
                    err
                )
            )
        }
    }
}
```

V programu může dojít k zavolání funkce *panic*. Ta označuje kritickou chybu, která nastala za běhu a měla za následek pád programu. Abychom tuto chybu zachytili a mohli ji také zalogovat, využíváme funkce *recover* (kód č. 4.9), která vrátí bližší informace o této chybě a v kombinaci s *defer* se vykoná při každém, i neočekávaném, konci běhu programu.[55]

4.3 Kontrola kvality kódu

Pro zajištění co nejkvalitnějšího kódu bez technického dluhu je jak v případě webové aplikace, tak scraperu užíváno nástrojů pro automatické kontrolování kódu oproti standardům a zároveň je využíváno statické analýzy.

Pro webovou aplikaci se využívá PHPStan pro statickou analýzu s nastavením na úroveň 7. V případě stylu kódu je použit nástroj PHP CodeSniffer, který zajišťuje, že styl kódu je v souladu s kombinací standardů PSR-12 a Slevomat Coding Standard.

V případě jazyka Go se využívá statická analýza kódu Staticcheck a Revive pro hlídání stylu kódu podle výchozího nastavení nástroje.

Samotný kód je díky tomu čitelný, bez zbytečného technického dluhu a jednotný. V případě jazyka Go jsou v rámci nastavení Revive vyžadovány komentáře pro exportované metody z balíčků a lze tedy bez obtíží vygenerovat dokumentaci pomocí různých nástrojů.

Výpis kódu 4.9 Získání chyby při pádu a její logování

```
func (app *App) Run() error {
    ...

    // vyjmutá část reprezentující získání chyby
    // a její zalogování
    defer func(logger *logger.Logger) {
        if err := recover(); err != nil {
            mainLogger.Fatal(fmt.Errorf("%s", err))
        }
    }(mainLogger)

    ...
    return nil
}
```

4.4 Pipeline

Pro udržení kvality kódu je automatizován proces jeho kontroly pomocí pipeline v rámci nástroje GitLab. Do budoucna lze pipeline rozšířit tak, aby automaticky nasazovala novou verzi do produkčního prostředí. V současné době se však sestává pouze ze čtyř částí.

docker-image v případě změny Dockerfile webové aplikace sestaví podle upravené verze nový obraz, který nahraje do registru v rámci nástroje GitLab. Tento obraz je využíván pro vývoj a při dalších částech pipeline.

dependencies stáhne závislosti aplikace, a vytvoří tak prostředí, ve kterém může být aplikace spuštěna pro automatické testování. Stažené závislosti poté předává dalším fázím a operacím formou artefaktu. V současnosti se tento krok týká pouze změny webové aplikace.

static-check je fáze, při níž dochází k samotné kontrole kódu, který byl do repozitáře nahrán. Jejím úkolem je automatické udržení kvality kódu podle definovaných standardů a bez technického dluhu. V této fázi jsou spouštěny 4 dílčí operace.

- **php-cs:** spouští nástroj PHP CodeSniffer pro kontrolu kódu podle standardů, avšak pouze v případě, že došlo ke změně PHP souboru.
- **php-stan:** spouští nástroj PHPStan pro statickou analýzu, a to pouze v případě, že došlo ke změně PHP souboru.
- **go-cs:** spouští nástroj Revive pro kontrolu kódu podle standardů, ovšem pouze za situace, kdy došlo ke změně Go souboru.

- **go-stan:** spouští nástroj Staticcheck pro statickou analýzu, a to pouze v případě, že došlo ke změně Go souboru.

test spouští operace, které pro jednotlivé části systému zahájí vyhodnocení jednotkových testů.

Kapitola 5

Testování a vybrané případy užití

Kapitola blíže analyzuje a prochází vybrané případy užití, které jsou implementovány v systému, a popisuje zvolené části, pokryté jednotkovými testy. Zároveň prezentuje výsledky testu výkonnosti programu pro vyhodnocování kolizí a probírá průměrné počty záznamů, které bude program denně zpracovávat.

5.1 Registrace

Tato sekce popisuje průchod implementovaného *UC1 Registrace*. Neregistrovaný uživatel, který vstoupí na webovou stránku, se může dostat k registračnímu formuláři díky menu, v němž je zvýrazněna položka *Registrace*. Formulář implementovaný v aplikaci je velice přímočarý – uživatel vyplňuje pouze své jméno, e-mail a heslo. Volitelně může uvést i své telefonní číslo. K němu se v současné době neváže žádná funckionalita, ale výhledově jej lze využít pro zasílání SMS notifikací. V souladu se současnou legislativou musí uživatel odsouhlasit i zásady zpracování osobních údajů. Protože aplikace zatím není nasazená, vývojová verze text tohoto dokumentu neobsahuje. Ukázkově vyplněný registrační formulář uvádí příloha C.10.

Po odeslání formuláře zkontroluje systém případnou duplicitu e-mailové adresy a shodu hesla. Pokud je vše v pořádku, odešle systém uživateli e-mail s aktivačním odkazem. Ten má platnost jen 30 minut, avšak uživatel si může nechat vygenerovat nový. Tento e-mail se neodešle okamžitě, ale je pro něj vytvořena notifikace a rozesílka této notifikace. Čeká se tedy, než dojde ke spuštění operace pro rozesílání e-mailových notifikací. Tu spustí buď nastavený cron, nebo příkaz (5.1).

```
php bin/console notifications:send --email
```
(5.1)

Aktivační e-mail, který uživatel obdrží, lze zhlédnout na obrázku č. 4.1. Po rozkliknutní tlačítka je uživatel odkázán do aplikace, kde dojde k potvrzení jeho e-mailové adresy a aktivaci účtu (obrázek č. C.9). Uživatel je zároveň přesměrován na přihlašovací stránku.

Samotný proces celé registrace vyžaduje po uživateli minimum údajů a zaneprázdní jej pouze na několik minut včetně čekání na aktivační e-mail.

5.2 Přidání ochranné známky

Tato sekce popisuje průchod implementovaného *UC6 Přidání ochranné známky* a zahrnuje i část popisu *UC13 Notifikace o kolizi*. Uživatel má po přihlášení možnost přejít přes seznam ochranných známek, dostupný na stránce *Ochranné známky*, k formuláři pro přidání nové sledované ochranné známky (obrázek č. C.11). Uživatel ve formuláři vyplní pro jednodušší identifikaci název ochranné známky a vybere její typ. Tím může být buď slovní, nebo obrazová ochranná známka. Pokud je vybrán typ slovní, nelze přidat grafickou přílohu. U obrazové však lze specifikovat slovní reprezentaci. Uživatel může přidat až 5 slovních vyjádření, která chce u jedné ochranné známky sledovat. Vyplnění čísla přihlášky má za důsledek ignorování změn s tímto číslem, které posléze program stáhne ze zdrojů dat. V sekci notifikačních kanálů může uživatel upravit, jakým způsobem chce být o možných kolizích notifikován. Poslední sekce je skrytá a uživatel si ji může zobrazit kliknutím. Jedná se o pokročilá nastavení, která upravují vyhodnocovací algoritmus pro konkrétní známku. Uživatel může u označení ignorovat diakritiku, zmenšit nebo zvětšit potřebnou Damerauovu vzdálenost pro vyhodnocení kolize a upravit práh obrazové podobnosti. Zároveň je možné upravovat detekci podřetězců.

Po odeslání formuláře zkontroluje systém vyplněná pole a nahraje případnou obrazovou přílohu do objektového úložiště. Při dalším běhu scraperu již budou pro tuto ochrannou známku kontrolovány případné kolize.

Pokud by došlo k nalezení kolize, načte systém uživatelovy preference komunikace k dané známce a vytvoří notifikaci v systému a jednotlivé rozesílky. K jejich rozeslání dojde teprve až při periodickém spuštění příslušných příkazů. Tyto notifikace se rozesílají v jazykové mutaci, kterou nastavil jako preferovanou uživatel ve svém profilu.

5.3 Jednotkové testy

Aby nedošlo v případě dalšího vývoje k neočekávaným změnám vyhodnocovacích algoritmů, jsou části, které se týkají vyhodnocování kolizí a porovnávání

jednotlivých reprezentací, pokryty jednotkovými testy. Tyto testy jsou zároveň spouštěny v rámci pipeline, a měly by zajistit hladkou integraci nového kódu.

V rámci scraperu se pro jednotkové testy používá balíček *testing*. Zvyklostí je pak psaní testů ve stejném adresáři, kde se testovaný kód nachází. Testy se nacházejí v souborech se sufixem *_test*. Metody jsou v Go exportované a mají prefix *Test*. Příkaz č. (5.2) vyhledá a spustí testy pro celý scraper.[56]

```
go test . ./...
```
(5.2)

5.4 Stažení dat a vyhodnocení kolizí

Každý den bude docházet ke stahování dat ze zdrojů a jejich následnému vyhodnocování. Jako vhodné se proto jeví změřit, jak dlouho bude tento proces trvat. Za měsíc březen 2024 eviduje podle nástroje *TMView* EUIPO 12 904 nových přihlášek evropských ochranných známek. V České republice bylo za stejné období podle stejného nástroje podáno 610 přihlášek[57]. Podle dat z RES ČSÚ v březnu 2024 vzniklo 3 199 právnických osob[58].

Březen roku 2024 měl 20 pracovních dní. Každý pracovní den by tak v průměru bylo třeba vyhodnotit zhruba 676 nových přihlášek ochranných známek a dvakrát do měsíce i data o právnických osobách.

Pro testování bylo v systému vytvořeno 1 000 sledovaných označení, pro něž program postupně stahoval data a vyhodnocoval možné kolize. Celkově bylo porovnáno 14 977 záznamů. Tedy zhruba tolik, kolik by program celkově porovnával v rámci měsíce března. Jednalo se o 12 423 záznamů z EUIPO, 1 245 z ÚPV a 1 309 z RES ČSÚ. Program běžel v Docker kontejneru, který měl k dispozici 4 GB paměti RAM a 2 jádra procesoru Intel Core i5.

Data z ÚPV byla stažena a vyhodnocena za 5 minut a 8 vteřin, v případě ČSÚ za 3 minuty a 14 vteřin a data EUIPO trvalo stáhnout a zpracovat 8 minut a 39 vteřin. Celkově program běžel 17 minut a 41 vteřin.

Kapitola 6
Možná rozšíření

Převážná většina obdobných aplikací se specializuje pouze na kontrolu kolizních přihlášek ochranných známek. Jedná se tedy o naprostý základ, který je třeba monitorovat a vyhodnocovat. Vlastníci ochranných známek však ještě potřebují uplatňovat svá práva, která jim z vlastnictví ochranné známky plynou. Chtějí zamezit jejímu užívání třetími osobami, aby nedošlo k snížení jejich zisku a k poškození jejich značky.

Již nyní spočívá výhoda vytvořené webové aplikace oproti běžně dostupným komerčním řešením v tom, že provádí kontroly i u jmen nově vzniklých právnických osob v České republice. Tím lze odhalit společnosti, které mají snahu parazitovat na značce doposud nezapsané v obchodním rejstříku, jež však již je registrovanou ochrannou známkou. Taková společnost by sice byla soudem zapsána do veřejného rejstříku, ale obchodní firma by porušovala práva vlastníka ochranné známky.

Do budoucna by však bylo žádoucí tuto výhodu ještě posílit, a aplikaci tak rozšířit o další moduly či zdroje dat a zpřesnit již implementované algoritmy a přístupy pro vyhodnocování kolizí.

6.1 Online tržiště

Vzhledem k velkému rozmachu e-commerce a e-shopů nejen v době pandemie Covid-19 se začali e-commerce giganti více přesouvat po vzoru společnosti Amazon k velkým online tržištím, přičemž nenakupují zboží od dodavatelů, aby jej následně přeprodali, ale umožňují těmto dodavatelům prodávat zboží na online tržišti pod jeho značkou. Spotřebitel si tak kupuje zboží např. od začínajícího e-shopu, ale celý proces je zastřešován již známou, silnou a důvěryhodnou značkou. V českém prostředí tento přístup využívá např. skupina Allegro nebo pro část sortimentu také online obchodník Alza.

Takové prostředí však může poskytnout ideální prostor pro prodej padělaného zboží a zboží, které porušuje práva duševního vlastnictví. Častokrát je

pak náročné zjistit, zda je zboží prodejců na těchto platformách autentické. V případě výše jmenovaných společností Allegro a Alza jsou kontrolní mechanismy nastaveny vhodně a takové zboží se na daných platformách prakticky nevyskytuje.

V poslední době došlo na českém trhu boomu tržišť prodávajících zboží převážně z Číny, odkud valná většina zboží, které porušuje duševní vlastnictví, pochází. Například se jedná o platformy Alibaba, Aliexpress, Shein, Temu a Wish. Tyto platformy jsou již spotřebiteli považovány za rizikové, protože zboží nemusí odpovídat kvalitě nebo může být padělané.

Větší problém nastává v případě společnosti Amazon, která má po světě vybudovanou silnou a důvěryhodnou značku a stejnojmenné online tržiště. Právě proto je pro značku výskyt padělků na tomto tržišti nebezpečný. Spotřebitel může díky síle značky Amazon a priori předpokládat, že nakupuje autentické zboží, a poté co obdrží padělek horší kvality, dojde tím jak k poškození značky, tak ke ztrátě na zisku.

Proto by bylo vhodné přidat funkcionalitu pro automatickou kontrolu online tržišť sledující produkty na těchto platformách a vyhledávající možné padělky, které se snaží vydávat za zboží značky chráněné ochrannou známkou, přičemž tím porušuje práva duševního vlastnictví. Vlastníci ochranných známek by tak mohli jednoduše monitorovat nakládání s jejich ochrannou známkou a případně rychle uplatnit svá práva a ochránit jak svou značku, tak samotného spotřebitele. Celý přístup by se ale nemusel omezit pouze na B2C platformy, ale mohl by zahrnout i P2P portály, jakými jsou celosvětový eBay nebo tuzemské Aukro.

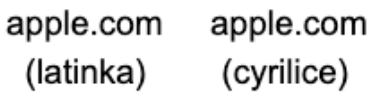
6.2 Doménová jména

V začátcích českého internetu, možná internetu obecně, docházelo poměrně běžně ke cybersquattingu. Jedná se o praktiku, kde si osoba v nedobré víře zaregistruje doménu na úkor značky nebo jména jiné osoby. Ve většině případů se snaží poté doménu prodat dotčené osobě. Z České republiky je znám případ společnosti Česká pojišťovna a. s. týkající se doménového jména ceskapojistovna.cz, který se dostal až k Vrchnímu soudu v Praze[59].

Obdobou cybersquattingu je pak typosquatting, který cílí na typické překlepy uživatelů při přepisu doménových jmen. Ukázkovým příkladem by mohla být záměna české domény seznam.cz za kyperskou doménu seynam.cy z důvodu prohození Z a Y na klávesnici.

V dnešní době však větší hrozbu představují phishingové domény, které se v uživatelích snaží pomocí různých metod evokovat pocit bezpečí známé značky. Takové přístupy zahrnují buď využívání základních technik typosquattingu a cybersquattingu, nebo také možnosti registrovat v případě různých TLD i znaky různých datových sad pomocí punnycode[60]. Tato IDN pak mohou sloužit k phishingovým útokům, které využívají homoglyfů. Příkladem může být doména apple.com zapsaná v latince a v cyrilici na obrázku č. 6.1.

V cyrilici je IDN apple.com reprezentována následovně: xn–80ak6aa92e.com. Ovšem když se prohlížeče snaží být uživateli co nejpřívětivější, zobrazí v adresním řádku již přeloženou doménu.[61]



Obrázek 6.1 Zápis domény apple.com v latince a v cyrilici v písmu Arial

Přidání detekce cybersquattingu a homografových útoků by tedy mohlo pomoci chránit nejen značku a jeji pověst, ale i její zákazníky. Jednalo by se tak o užitečnou funkci sloužící nejen pro ochranu duševního vlastnictví.

6.3 Zdroje dat

V tuto chvíli zpracovává aplikace data od ÚPV, ČSÚ a EUIPO. Tyto zdroje jsou dostatečné pro detekci kolizí tuzemských ochranných známek, ale nelze detekovat kolize z jiných států. To znamená, že pro evropskou ochrannou známku detekujeme kolizní přihlášku z České republiky a celounijní evropskou přihlášku ochranné známky, ale nedetekujeme již např. slovenskou kolizní přihlášku.

Bylo by proto vhodné aplikaci v budoucnu rozšířit o data z dalších unijních, případně i světových úřadů a organizace WIPO, a zejména získat více zdrojů ze zemí vázaných Pařížskou úmluvou[62] a Madridskou dohodou[63].

6.4 Porovnání obrázků

Systém v tuto chvíli využívá přepisu textu z obrazových známek, které následně vyhodnocuje vzhledem k možné slovní kolizi. U kombinovaných nebo pouze obrazových ochranných známek, kde se tento přepis vyskytovat nemusí, pak využívá metodu knihovny OpenCV – Template matching. Tento algoritmus, ačkoliv jeho výstupy nejsou neuspokojivé, nereflektuje již zažitou rozhodovací praxi. Zároveň se příliš soustředí na exaktní shodu šablony s místem v předloze a případě, že se na předloze, vyskytuje zvětšená část šablony, se již značně snižuje korelační koeficient.

Při testování dalších možných přístupů byl testován i SIFT. Ten často identifikoval zájmové body, které se týkaly písma, namísto jiných grafických prvků, a tím docházelo k nízkému skóre podobnosti. Řešením tohoto problému by mohla být detekce textu v obrázku a spuštění algoritmu SIFT s částmi bez textu. To však neřeší problém v případě, kdy jsou tyto grafické prvky jednoduché a nekontrastní. Důsledkem je detekce nízkého počtu zájmových bodů.

Samotné rozhodování úřadu či soudu o finální kolizi se ale zčásti zakládá na rozhodovací praxi a bere v potaz i minulé případy. Pokud by se nashromáždila data o předchozích rozhodnutích, bylo by možné natrénovat model, který by již reflektoval příslušnou rozhodovací praxi, a snížil by tak výskyt falešně pozitivních i falešně negativních výsledků.

Závěr

Cílem této práce bylo analyzovat vhodné zdroje otevřených dat a algoritmy pro vyhodnocování podobnosti českých ochranných známek a názvů nově vzniklých společností. Na základě této analýzy a porovnání již existujících řešení mělo dojít ke vzniku systému, který by monitoring těchto známek prováděl automatizovaně.

Výsledek představuje vícejazyčná webová aplikace vyvinutá v programovacím jazyce PHP, která uživatelům nabízí prostředí pro správu jejich ochranných známek, v němž si pro každé označení může uživatel upravit vyhodnocovací algoritmus tak, aby dosáhl co nejoptimálnějších výsledků. Aplikace podporuje uživatelské účty a propojení těchto účtů s aplikacemi pro okamžité doručování zpráv.

Agregaci dat a vyhodnocování možných kolizí periodicky provádí program napsaný v jazyce Go. Ten porovnává možné shody jak slovního, tak grafického vyjádření. V případě, že možnou kolizi identifikuje, informuje o jejím výskytu prostřednictvím REST API webovou aplikaci, která následně notifikuje uživatele preferovaným způsobem. Aplikace podporuje rozesílání e-mailových notifikací, Discord zpráv a Telegram zpráv.

Systém sbírá data o českých ochranných známkách z ÚPV a vyhodnocuje nově vzniklé společnosti z RES, který poskytuje a udržuje ČSÚ. Nad rámec zadání pak systém stahuje i data o evropských ochranných známkách a o mezinárodních ochranných známkách s designací v EU, které získává od EUIPO.

Práce prezentuje ukázkový průchod vybraných případů užití. Algoritmy pro vyhodnocování kolizí, které tvoří klíčovou část systému, jsou pokryty jednotkovými testy. Stav, v němž se nyní aplikace nachází, nebrání jejímu nasazení do produkčního prostředí zejména s využitím cloudových technologií.

V případě kolizí slovních ochranných známek a grafických ochranných známek, které obsahují text v latince, dokáže algoritmus s pomocí Damerauovy vzdálenosti identifikovat běžné kolize způsobené nedostatečnou rešerší při podání přihlášky ochranné známky. Detekce grafické podobnosti je náročnější a v práci použité metody mají své nevýhody. Vedle problémů při zhoršené kvalitě obrázků je největší nevýhodou absence rozhodovací praxe, která je v této

oblasti zásadní pro identifikaci podobnosti. Pokud by se podařilo shromáždit rozhodnutí úřadů v souvislosti s kolizemi a jejich námitkami, bylo by možné natrénovat model, který by identifikoval podobnost s velkou přesností.

Příloha A

Výstupy zdrojů dat

A.1 Změnový soubor databáze ÚPV

```
<?xml version="1.0" encoding="UTF-8"?>
<Transactions
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="https://isdv.upv.gov.cz/webapp/
appldoc/xsd/OpenDataTrademark_cz_v201909.xsd"
generated="2024-03-17"
>
    <Transaction operationCode="Insert">
      <TradeMarkTransactionBody>
        <TransactionContentDetails>
          <TransactionData>
            <TradeMarkDetails>
              <TradeMark>
                <ApplicationNumber>581656</ApplicationNumber>
                <ApplicationDate>2022-10-04</ApplicationDate>
                <RegistrationNumber>396249</RegistrationNumber>
                <RegistrationDate>2023-03-29</RegistrationDate>
                <PublicationDate>2022-12-21</PublicationDate>
                <ExpiryDate>2032-10-04</ExpiryDate>
                <CurrentStatusCode>6</CurrentStatusCode>
                <KindMark>Individual</KindMark>
                <MarkFeature>Figurative</MarkFeature>
                <WordMarkSpecification>
                  <MarkVerbalElementText languageCode="cs">
                    PETROL čerpací stanice
                  </MarkVerbalElementText>
                </WordMarkSpecification>
```

```
            <MarkImageDetails>
              <MarkImage sha1="...">
                <MarkImageFilename>
                    o5816560
                </MarkImageFilename>
                <MarkImageFileFormat>
                    jpg
                </MarkImageFileFormat>
                <MarkImageColourIndicator>
                    true
                </MarkImageColourIndicator>
                <MarkImageCategory>
                  <CategoryCodeDetails>
                    <CategoryCode>27.05.10</CategoryCode>
                    <CategoryCode>29.01.06</CategoryCode>
                    <CategoryCode>29.01.01</CategoryCode>
                    <CategoryCode>27.05.09</CategoryCode>
                  </CategoryCodeDetails>
                </MarkImageCategory>
              </MarkImage>
            </MarkImageDetails>
            <GoodsServicesDetails>
              <GoodsServices>
                <ClassDescriptionDetails>
                  <!--
                  Informace o třídách,
                  pro které je požadována ochrana
                  -->
                </ClassDescriptionDetails>
              </GoodsServices>
            </GoodsServicesDetails>
            <PriorityDetails>
              <Priority>
                <PriorityCountryCode>
                    CZ
                </PriorityCountryCode>
                <PriorityDate>2022-10-04</PriorityDate>
              </Priority>
            </PriorityDetails>
          </TradeMark>
        </TradeMarkDetails>
      </TransactionData>
    </TransactionContentDetails>
  </TradeMarkTransactionBody>
```

```
    </Transaction>
</Transactions>
```

A.2 Výstup EUIPO Trademark Search REST API


```
{
    "trademarks": [
        {
            "applicationNumber": "018753937",
            "markKind": "INDIVIDUAL",
            "markFeature": "WORD",
            "markBasis": "EU_TRADEMARK",
            "niceClasses": [
                29,
                31
            ],
            "wordMarkSpecification": {
                "verbalElement": "My Trademark"
            },
            "representatives": [
                {
                    "office": "EM",
                    "identifier": "10014",
                    "name": "HOGAN LOVELLS"
                }
            ],
            "applicationDate": "2023-10-15",
            "status": "UNDER_EXAMINATION"
        }
    ],
    "size": 10,
    "totalElements": 1,
    "totalPages": 1,
    "page": 0
}
```

Příloha B

Implementované algoritmy

B.1 Porovnání textu

```
func (tm1 *Trademark) CompareWording(tm2 *Trademark) bool {
    // Ignorujeme ochranné známky se shodným číslem přihlášky.
    if tm1.ApplicationNumber != ""
        && tm1.ApplicationNumber == tm2.ApplicationNumber {
        return false
    }

    for _, w1 := range tm1.Wording {
        if w1 == "" {
            continue
        }
        w1 = strings.ToUpper(w1)

        for _, w2 := range tm2.Wording {
            if w2 == "" {
                continue
            }

            w2 = strings.ToUpper(w2)

            // Odstranění diakritiky.
            if tm1.IgnoreDiacritics {
                t := transform.Chain(
                    norm.NFD,
                    runes.Remove(runes.In(unicode.Mn)),
                    norm.NFC,
                )
```

```
                w1, _, _ = transform.String(t, w1)
                w2, _, _ = transform.String(t, w2)
            }

            // Porovnání Damerauovi vzdálenosti
            // s prahem podobnosti uživatele.
            if tm1.EditDistance > 0
                && compare.DamerauDistance(w1, w2)
                <= tm1.EditDistance {
                return true
            }

            // Odstranění mezer a vyhledávání v podřetězců.
            w1 = strings.ReplaceAll(w1, " ", "")
            w2 = strings.ReplaceAll(w2, " ", "")
            if tm1.TmInCollisionWording && len(w1) > 2
                && strings.Contains(w2, w1) {
                return true
            }
            if tm1.CollisionInTmWording && len(w1) > 2
                && strings.Contains(w1, w2) {
                return true
            }
        }
    }

    return false
}
```

B.2 Porovnání obrázků

```
func (tm1 *Trademark) CompareImage(tm2 *Trademark) bool {
    // Ignorujeme ochranné známky se shodným číslem přihlášky.
    if tm1.ApplicationNumber != ""
        && tm1.ApplicationNumber == tm2.ApplicationNumber {
        return false
    }

    if tm1.ImagePath == "" || tm2.Image == "" {
        return false
    }

    if tm1.ImageThreshold > 0
        && compare.TemplateMatching(tm2.Image, tm1.ImagePath)
```

```
        >= tm1.ImageThreshold {
        return true
    }

    return false
}
```

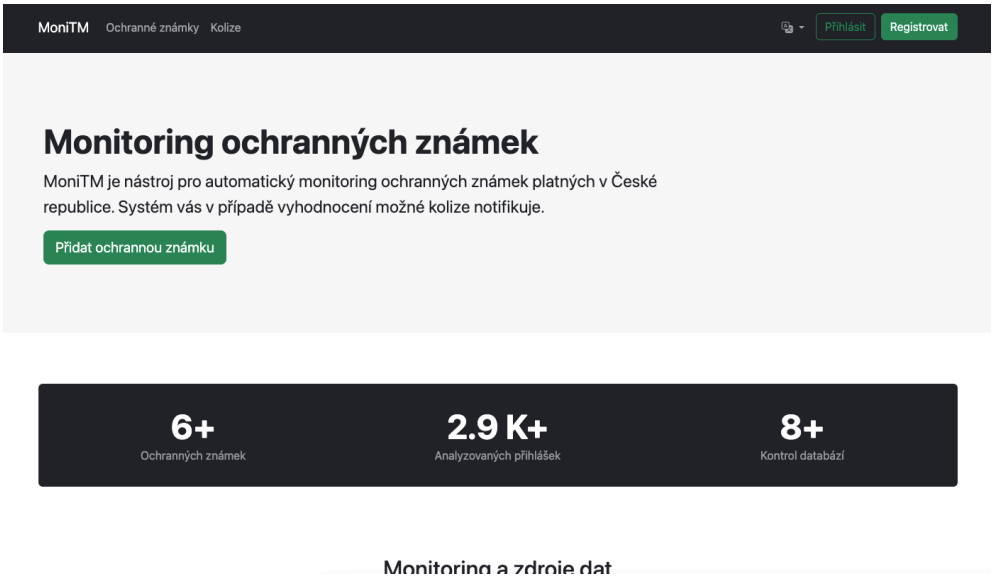
B.3 Template matching

```
func TemplateMatching(mainImage, template string) int {
    // Načtení předlohy a šablony do paměti
    img := gocv.IMRead(mainImage, gocv.IMReadGrayScale)
    defer img.Close()

    templ := gocv.IMRead(template, gocv.IMReadGrayScale)
    defer templ.Close()

    result := gocv.NewMat()
    defer result.Close()

    imgCols, imgRows := img.Cols(), img.Rows()
    templCols, templRows := templ.Cols(), templ.Rows()

    // Změna rozměrů předlohy
    if templCols >= imgCols || templRows >= imgRows {
        scaleWidth := float64(templCols+1) / float64(imgCols)
        scaleHeight := float64(templRows+1) / float64(imgRows)
        scale := math.Max(scaleWidth, scaleHeight)

        newWidth := float64(imgCols) * scale
        newHeight := float64(imgRows) * scale
        resized := gocv.NewMat()

        if newWidth == 0 || newHeight == 0 {
            return 0
        }

        gocv.Resize(
            img,
            &resized,
            image.Point{X: int(newWidth), Y: int(newHeight)},
            0,
            0,
            gocv.InterpolationDefault,
```

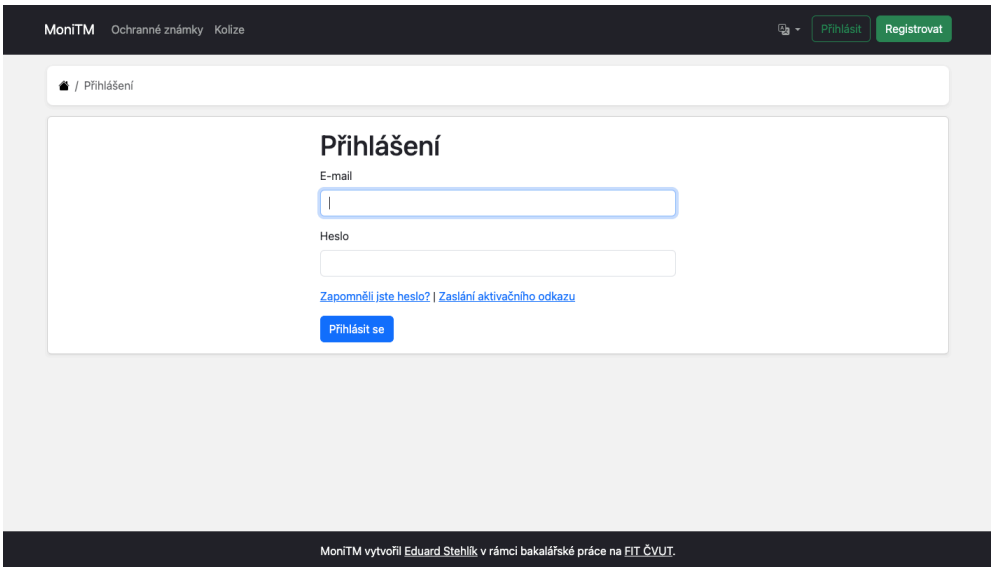
```
        )

        img.Close()
        img = resized
    }

    // Spuštění Template Matching algoritmu z knihovny GoCV
    gocv.MatchTemplate(
        img,
        templ,
        &result,
        gocv.TmCcoeffNormed,
        gocv.NewMat(),
    )

    // Získání maximální korelace
    _, maxVal, _, _ := gocv.MinMaxLoc(result)

    return int(math.Round(float64(maxVal * 100)))
}
```

Příloha C

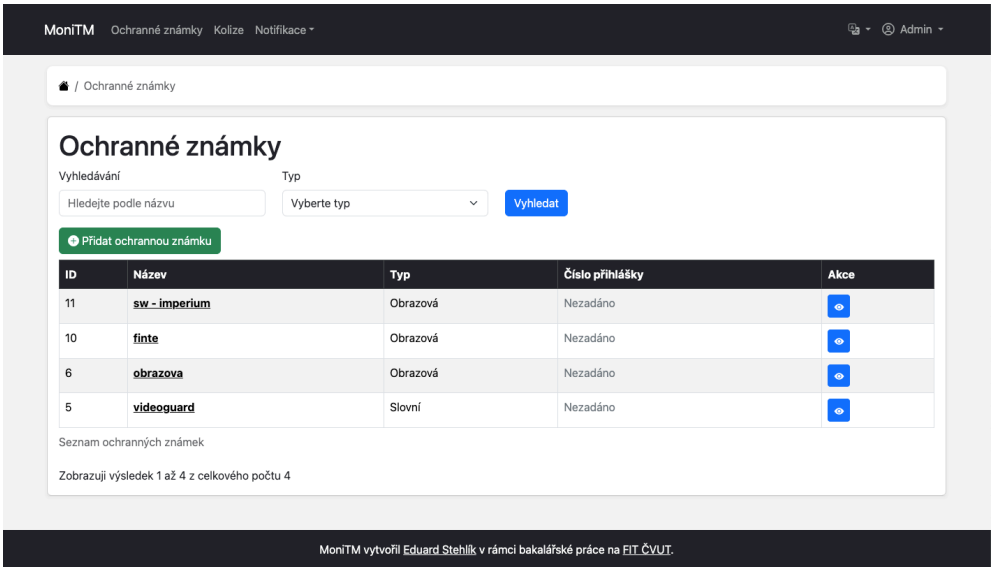
Ukázky z uživatelského rozhraní



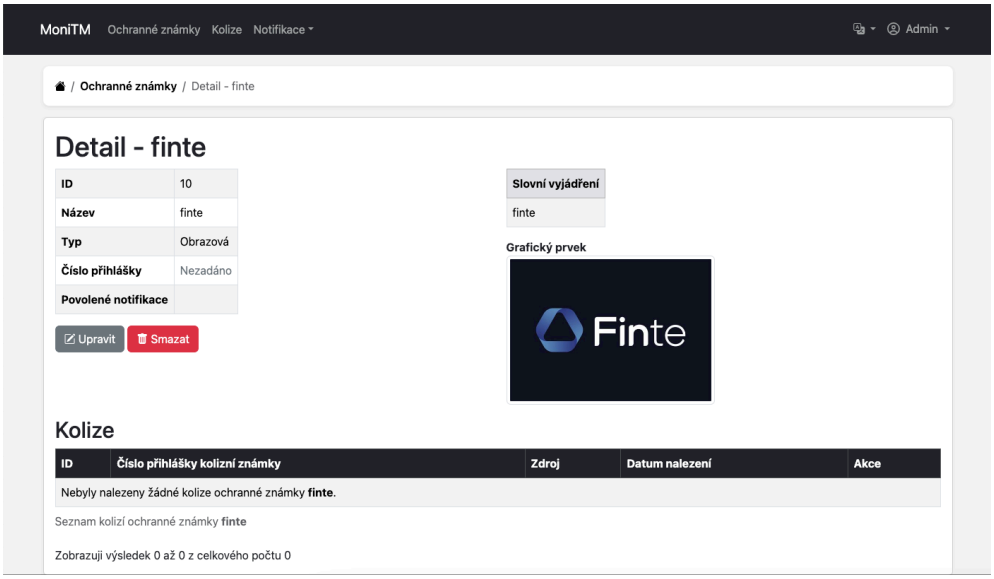
Obrázek C.1 Hlavní stránka



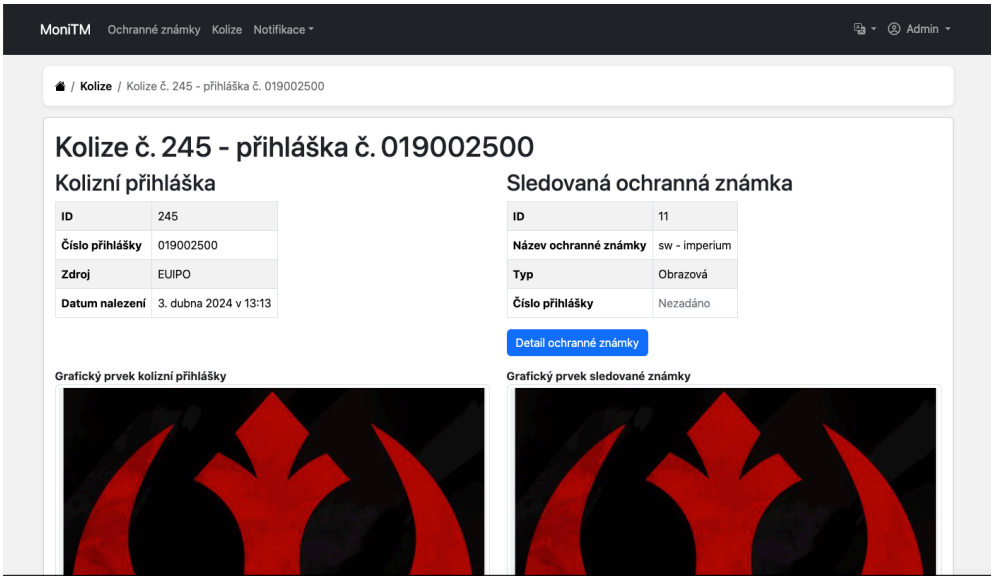
Obrázek C.2 Stránka přihlášení



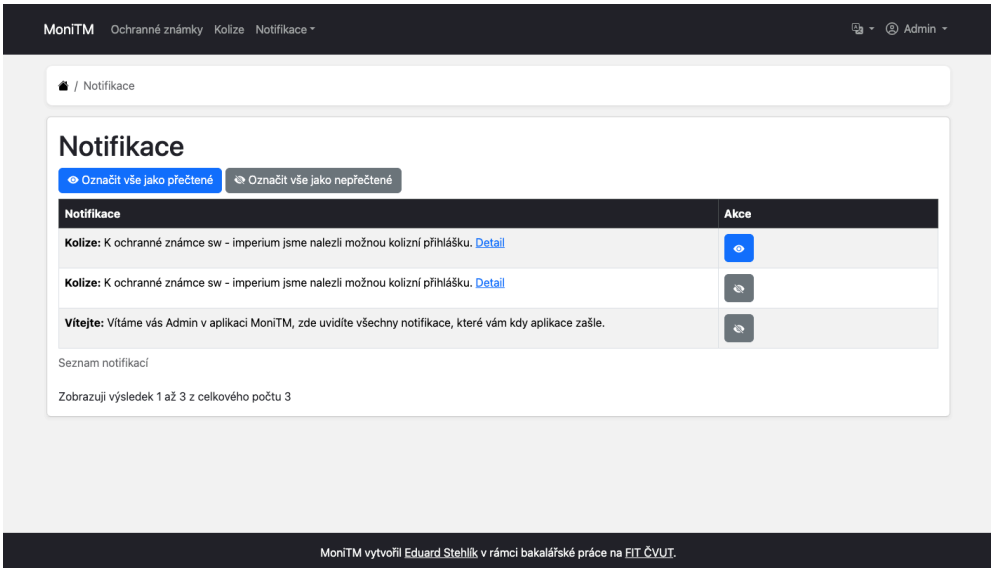
Obrázek C.3 Seznam ochranných známek



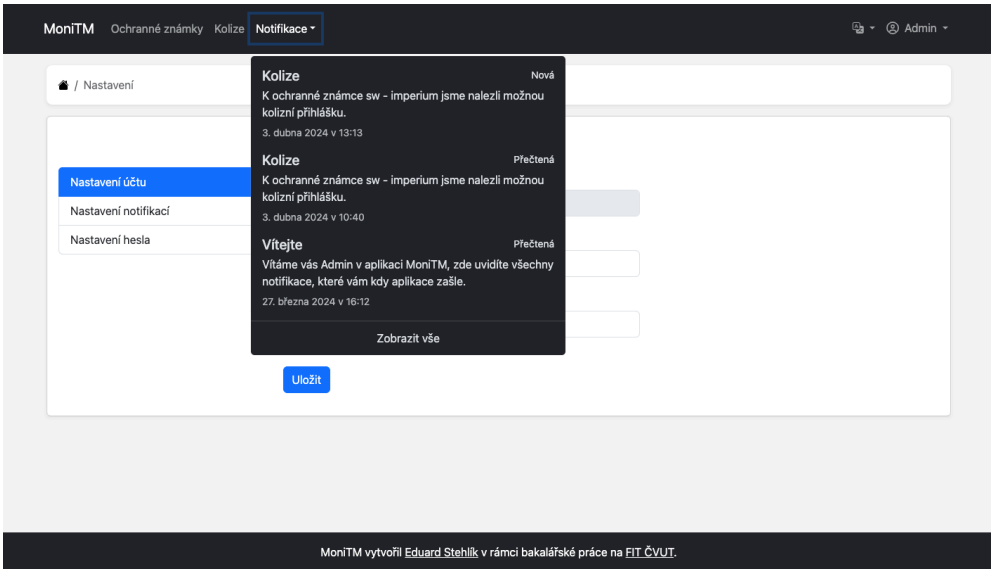
Obrázek C.4 Detail ochranné známky



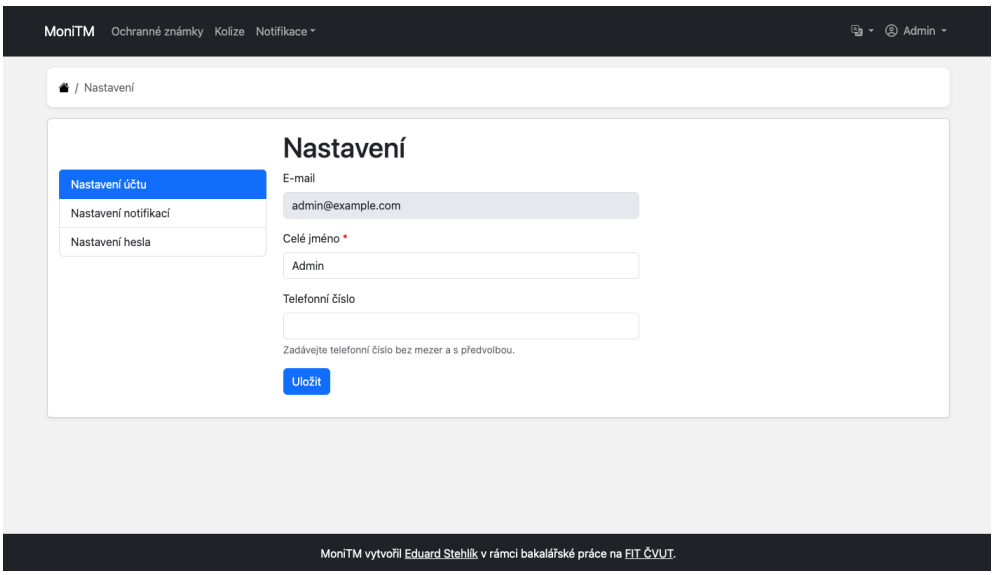
Obrázek C.5 Detail kolize



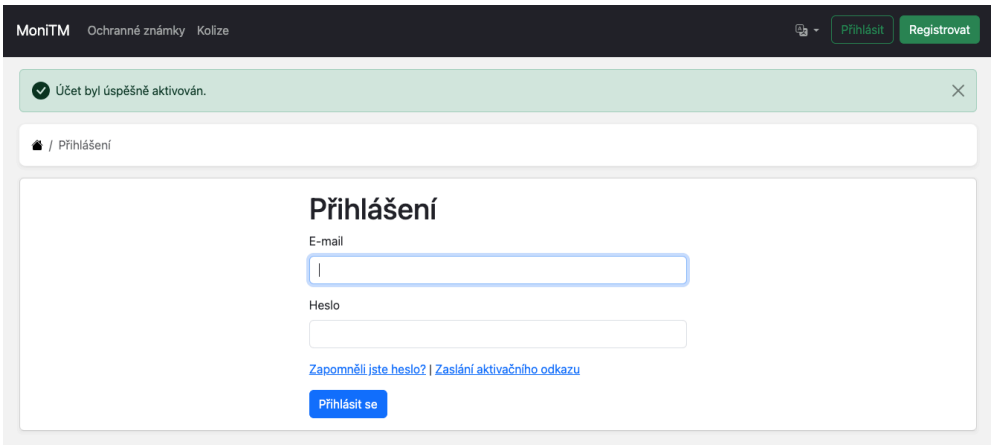
Obrázek C.6 Seznam notifikací



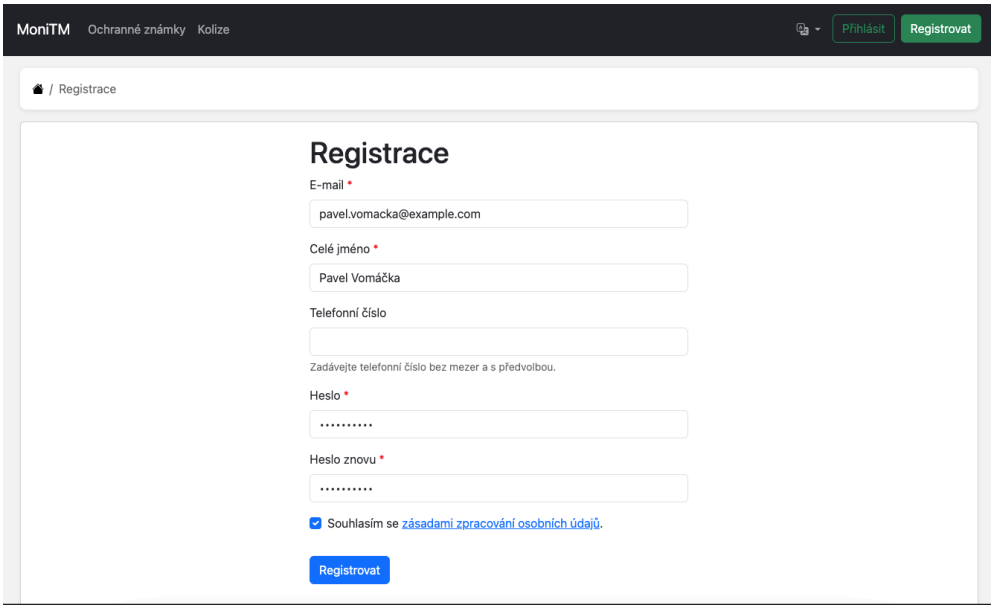
Obrázek C.7 Rozbalovací menu notifikací



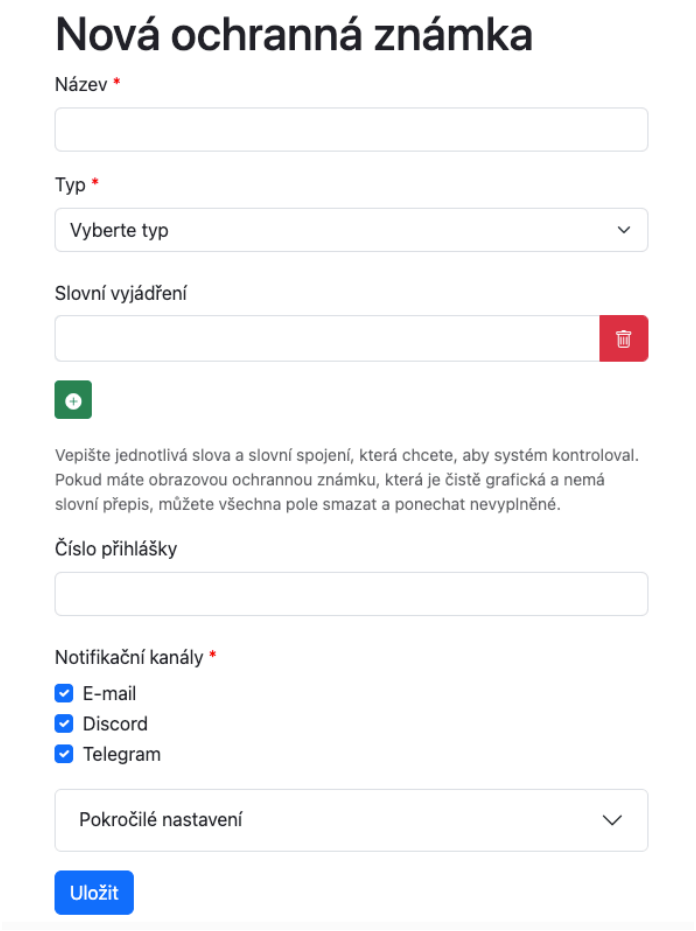
Obrázek C.8 Nastavení



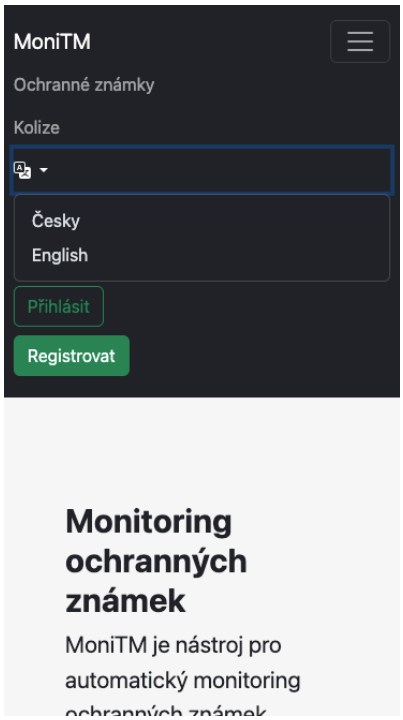
Obrázek C.9 Aktivace účtu prostřednictvím odkazu



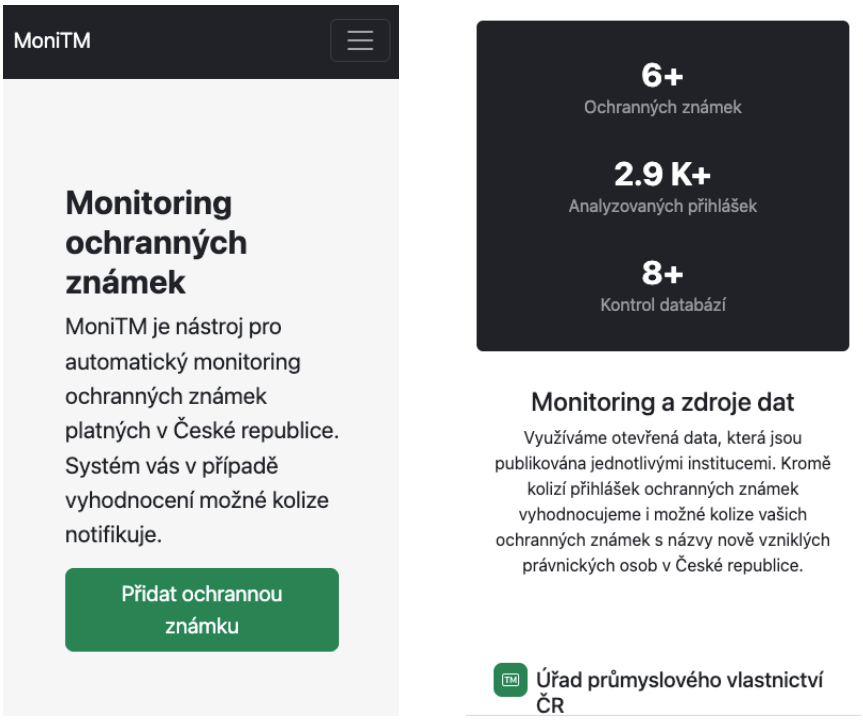
Obrázek C.10 Ukázkově vyplněný formulář registrace



Obrázek C.11 Formulář pro přidání ochranné známky



Obrázek C.12 Menu na mobilním zařízení



(a) Jumbotron hlavní stránky **(b)** Statistika na hlavní stránce

Obrázek C.13 Mobilní zobrazení hlavní stránky

Příloha D

Zprovoznění vývojové verze

Před samotným konfigurováním a zprovozněním vývojové verze je nutné získat přístup k Telegram Bot API a EUIPO Trademark Search API. Pro vývoj postačí sandbox verze Trademark Search API. Bez těchto dvou přístupů nemohou fungovat Telegram notifikace, resp. stahování dat od EUIPO. Protože jsou jednotlivé části rozděleny do kontejnerů, je potřeba před spuštěním nainstalovat Docker.

Telegram Bot – Pro získání přístupu k Telegram Bot API je nutné již mít Telegram účet. Následně je možné napsat účtu *@BotFather* příkaz */newbot*. Bot nás provede procesem vytvoření nového bot účtu a na závěr získáme API token pro komunikaci.

EUIPO API – V první řadě je nutné si vytvořit účet na stránkách EUIPO a buď v produkčním, nebo v sandboxovém prostředí jejich portálu pro vývojáře[1] si vytvořit aplikaci a přihlásit se k odběru konkrétního API, kterým je v našem případě Trademark Search API. Při vytváření aplikace je nám zobrazen privátní klíč, jímž se budeme při používání API autorizovat. Je nutné si jej pečlivě poznamenat. Následně je třeba posečkat, až dojde ze strany EUIPO k potvrzení odběru a povolení používání jejich služeb.

Konfigurace – Nakonfigurovat systém je nutné na třech místech. Odděleně konfigurujeme údaje pro komunikace mezi jednotlivými službami a nastavení pro webovou aplikaci a scraper.

1. Nejdříve je nutné ve stejném adresáři, jako je soubor *docker-compose.yml*, zkopírovat soubor .env-template do .env. Zde dochází k nastavení údajů pro MinIO objektové úložiště a PostgreSQL databázi. Vybranými proměnnými prostředí lze přepsat výchozí hodnoty v docker-compose souboru, které se zde pro PostgreSQL nacházejí. Ve vývojovém prostředí musíme změnit pouze přihlašovací údaje k MinIO.

[1] API Portál EUIPO je dostupný z https://dev.euipo.europa.eu/

2. Pro konfiguraci webové aplikace pro vývojové prostředí postačí nastavit údaje k SMTP serveru pro odesílání e-mailových notifikací a údaje k Telegram botovi. Pokud došlo ke změnám přístupů k databázi, je nutné upravit i ty.

```
# .env.local soubor v adresáři web s webovou aplikací
MAILER_DSN=smtp://ac@example.com:heslo@smtp.example.com:465
MAILER_SENDER=ac@example.com
TELEGRAM_BOT=monitm_bot # Uživatelské jméno bota
TELEGRAM_BOT_TOKEN=xxx # Token
TELEGRAM_SECRET=xxx # Secret pro Telegram webhook
# Secret může být jakýkoliv string
```

3. Poté je potřeba nakonfigurovat ještě scraper. Pro něj lze použít ukázkový konfigurační soubor v *scraper/config/config.template.yml* a zkopírovat jej do *scraper/config/config.yml*. Zde je poté nutné upravit, zda se jedná o vývojové, nebo produkční prostředí a případně přidat URL pro Discord webhook pro logování chyb a odkaz pro monitorování běhů. Dále je nutné vyplnit přístupové údaje EUIPO a údaje o připojení k webové aplikaci a MinIO. Předvyplněné údaje není třeba pro vývojovou verzi měnit. Údaje pro připojení k webové aplikaci jsou uloženy v databázi aplikace. Lze si však později nahrát ukázková data. Ta v sobě mají i jeden přístup pro scraper. Token k tomuto přístupu lze najít v *DataFixtures* webové aplikace.

Spuštění – Pokud je aplikace podle předchozích kroků správně nakonfigurovaná, měl by příkaz (D.1) sestavit obrazy a spustit všechny kontejnery aplikace.

```
docker compose up -d                                    (D.1)
```

Migrace – Při prvním spuštění je potřeba provést všechny migrace databáze, aby aplikace fungovala korektně. To lze provést příkazem (D.2) v kontejneru PHP aplikace. Zároveň je možné do databáze nahrát ukázková data pro vývoj pomocí spuštění příkazu (D.3) ve stejném kontejneru.

```
php bin/console doctrine:migrations:migrate --no-interaction    (D.2)
php bin/console doctrine:fixtures:load                          (D.3)
```

Periodické spouštění procesů – Značnou část funkcionality v aplikaci spouští čas. Pro její spouštění, ačkoliv se části nacházejí v kontejnerech, byl zvolen přístup s pomocí nástroje *cron*. Ukázkový soubor *crontab* by pak mohl vypadat následovně:

```
*/5 * * * * docker exec -d [nazev php-app kontejneru] \
php /var/www/bin/console notifications:send \
--email >/dev/null
*/5 * * * * docker exec -d [nazev php-app kontejneru] \
php /var/www/bin/console notifications:send \
--discord >/dev/null
*/5 * * * * docker exec -d [nazev php-app kontejneru] \
php /var/www/bin/console notifications:send \
--telegram >/dev/null
*/30 * * * * docker exec -d [nazev php-app kontejneru] \
php /var/www/bin/console database:clear \
--telegram >/dev/null
```

Scraper je v crontabu opomenut úmyslně. Ve vývojovém prostředí se spouští manuálně podle potřeby. Na produkčním prostředí by pak došlo k jeho zkompilování do spustitelného souboru, který by byl posléze opět spouštěn pomocí nástroje *cron*.

Registrace Telegram webhook – V případě, že uživatel zašle zprávu našemu botovi prostřednictvím Telegramu, notifikuje nás tato platforma pomocí webhooku a my následně notifikaci zpracujeme. Protože aplikace není vystavena veřejně, musíme pro zpřístupnění adresy, na kterou nás Telegram notifikuje, využít nástroj. Služba ngrok vygeneruje URL adresu a všechny požadavky, které na ni přijdou, pak přesměruje přes svoje servery na náš lokální stroj. Tím můžeme krátkodobě zpřístupnit vývojovou verzi, aby s ní mohl Telegram komunikovat.

Adresu, kterou nám poskytl ngrok, poté zaregistrujeme pro notifikace u Telegramu. Stačí k tomu zaslat POST požadavek na Telegram Bot API (D.4) s tělem popsaným na výpisku č. D.1. Pro smazání tohoto webhooku je také potřeba zaslat požadavek, ale již na jinou URL (D.5).

POST https://api.telegram.org/bot[Bot token]/setWebhook (D.4)

POST https://api.telegram.org/bot[Bot token]/deleteWebhook (D.5)

Výpis kódu D.1 Tělo požadavku pro nastavení Telegram webhook

```
{
    "url":"https://[ngrok subdoména]/webhook/telegram",
    "secret_token": "secret token z env.local"
}
```

Bibliografie

1. Zákon č. 441/2003 Sb. Zákon o ochranných známkách a o změně zákona č. 6/2002 Sb., o soudech, soudcích, přísedících a státní správě soudů a o změně některých dalších zákonů (zákon o soudech a soudcích), ve znění pozdějších předpisů, (zákon o ochranných známkách). In: *Zákony pro lidi* [online]. AION CS, 2003 [cit. 2024-04-06]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2003-441`.
2. *Mezinárodní třídník výrobků a služeb pro účely zápisu ochranných známek* [online]. 12. vydání. Praha: Úřad průmyslového vlastnictví, 2024 [cit. 2024-04-09]. Dostupné z: `https://isdv.upv.gov.cz/doc/nices/rep/2024-12_nices_vyrobky.pdf`.
3. The Open Definition. In: *The Open Definition* [online]. The Open Knowledge Foundation, 2009 [cit. 2024-04-06]. Dostupné z: `https://opendefinition.org/`.
4. Open Definition 2.1. In: *The Open Definition* [online]. Verze 2.1. The Open Knowledge Foundation, 2015 [cit. 2024-04-06]. Dostupné z: `https://opendefinition.org/od/2.1/en/`.
5. 5hvězdičková stupnice otevřených dat. In: *5-star Open Data* [online]. Hausenblas, c2012-2024 [cit. 2024-04-06]. Dostupné z: `https://5stardata.info/images/5-star-steps.png`.
6. Linked data - Design issues. In: *W3C* [online]. World Wide Web Consortium, 2006 [cit. 2024-04-06]. Dostupné z: `https://www.w3.org/DesignIssues/LinkedData.html`.
7. Zákon č. 106/1999 Sb. Zákon o svobodném přístupu k informacím. In: *Zákony pro lidi* [online]. AION CS, 1999 [cit. 2024-04-06]. Dostupné z: `https://www.zakonyprolidi.cz/cs/1999-106`.
8. MÍŠEK, J. Právní úprava otevřených dat. In: *Youtube* [online]. 2023 [cit. 2024-04-06]. Dostupné z: `https://www.youtube.com/watch?v=ezWRmZHyImA`. Kanál uživatele Digitální a informační agentura.

9. HAMMING, R. W. Error detecting and error correcting codes. *The Bell System Technical Journal*. 1950, roč. 29, č. 2, s. 147–160. ISSN 0005-8580. Dostupné z DOI: 10.1002/j.1538-7305.1950.tb00463.x.

10. LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*. 1966, roč. 10, č. 8, s. 707–710. ISSN 0038-5689.

11. SCHULZ, K.; MIHOV, S. Fast string correction with Levenshtein automata. *International Journal on Document Analysis and Recognition*. 2002, roč. 5, č. 1, s. 67–85. ISSN 1433-2833. Dostupné z DOI: 10.1007/s10032-002-0082-8.

12. DAMERAU, F. J. A technique for computer detection and correction of spelling errors. *Communications of the ACM*. 1964, roč. 7, č. 3, s. 171–176. ISSN 0001-0782. Dostupné z DOI: 10.1145/363958.363994.

13. BOYTSOV, L. Indexing methods for approximate dictionary searching: Comparative analysis. *The ACM journal of experimental algorithmics*. 2011, roč. 16, č. 1, s. 1.1–1.91. ISSN 1084-6654. Dostupné z DOI: 10.1145/1963190.1963191.

14. OOMMEN, B. J.; LOKE, R. K. S. Pattern recognition of strings with substitutions, insertions, deletions and generalized transpositions. *Pattern Recognition*. 1997, roč. 30, č. 5, s. 789–800. ISSN 0031-3203. Dostupné z DOI: https://doi.org/10.1016/S0031-3203(96)00101-X.

15. WAGNER, R. A.; FISCHER, M. J. The String-to-String Correction Problem. *J. ACM*. 1974, roč. 21, č. 1, s. 168–173. ISSN 0004-5411. Dostupné z DOI: 10.1145/321796.321811.

16. Damerau–Levenshtein distance. In: *Wikipedia* [online]. Wikimedia, 2024 [cit. 2024-04-12]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Damerau%E2%80%93Levenshtein_distance&oldid=1209413830.

17. ROOVER, C.; VLEESCHOUWER, C.; LEFEBVRE, F.; MACQ, B. Robust image hashing based on radial variance of pixels. In: *Proceedings - International Conference on Image Processing, ICIP*. 2005, sv. 3, s. III–77. ISBN 0-7803-9134-9. Dostupné z DOI: 10.1109/ICIP.2005.1530332.

18. AKSOY, M. S.; TORKUL, O.; CEDIMOGLU, I. H. An industrial visual inspection system that uses inductive learning. *Journal of Intelligent Manufacturing*. 2004, roč. 15, č. 4, s. 569–574. ISSN 0956-5515. Dostupné z DOI: https://doi.org/10.1023/B:JIMS.0000034120.86709.8c.

19. FRANGI, A. F.; PRINCE, J. L.; SONKA, M. *Medical image analysis*. San Diego, CA: Academic Press, 2023. The MICCAI Society book Series. ISBN 978-0-12-813657-7.

20. LOWE, D. G. Object recognition from local scale-invariant features. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. 1999, sv. 2, s. 1150–1157. ISBN 0-7695-0164-8. Dostupné z DOI: 10.1109/ICCV.1999.790410.

21. BRUNELLI, R. *Template matching techniques in computer vision*. Hoboken, NJ: Wiley-Blackwell, 2009. ISBN 978-0-470-51706-2.

22. SKOPAL, T. 08 - Local image features (zoom stream). In: *Youtube* [online]. 2020 [cit. 2024-04-17]. Dostupné z: https://www.youtube.com/watch?v=7Ezho_y1dzY. Kanál uživatele KSI MFF UK.

23. Watch. In: *RightHub* [online]. RightHub LTD., 2024 [cit. 2024-04-14]. Dostupné z: https://www.righthub.com/watch.

24. Monitoring ochranných známek pro dokonalou ochranu Vaší značky. In: *tramatm* [online]. TramaTM s. r. o., [b.r.] [cit. 2024-04-14]. Dostupné z: https://www.tramatm.cz/cs/trademark-watch.

25. Protect your brand with trademark monitoring. In: *Legal Zoom* [online]. LegalZoom.com, Inc., [b.r.] [cit. 2024-04-14]. Dostupné z: https://www.legalzoom.com/business/intellectual-property/trademark-monitoring-overview.html.

26. Trademark monitoring. In: *JUMP Trademarks* [online]. JUMP Trademarks, [b.r.] [cit. 2024-04-14]. Dostupné z: https://www.jumptrademarks.com/trademark/monitoring/.

27. O aplikaci. In: *Hlídač OZ* [online]. ATLAS GROUP, 2024 [cit. 2024-04-14]. Dostupné z: https://hlidacoz.cz/.

28. Must-Know Legal Tech Industry Statistics. In: *Gitnux* [online]. Gitnux, 2023 [cit. 2024-04-14]. Dostupné z: https://gitnux.org/legal-tech-industry-statistics/.

29. Trademark search (1.0.0). In: *API Portal* [online]. EUIPO, [b.r.] [cit. 2024-04-17]. Dostupné z: https://dev.euipo.europa.eu/product/trademark-search_100/api/trademark-search.

30. Global Brand Database: Frequently Asked Questions. In: *WIPO* [online]. WIPO, [b.r.] [cit. 2024-04-17]. Dostupné z: https://www.wipo.int/reference/en/branddb/faqs_branddb.html.

31. EUIPO and WIPO to strengthen cooperation in key areas. In: *EUIPO* [online]. EUIPO, 2024 [cit. 2024-04-17]. Dostupné z: https://www.euipo.europa.eu/en/news/euipo-and-wipo-to-strengthen-cooperation-in-key-areas.

32. Zákon č. 89/2012 Sb. Zákon občanský zákoník. In: *Zákony pro lidi* [online]. AION CS, 2012 [cit. 2024-04-12]. Dostupné z: https://www.zakonyprolidi.cz/cs/2012-89.

33. Zákon č. 300/2008 Sb. Zákon o elektronických úkonech a autorizované konverzi dokumentů. In: *Zákony pro lidi* [online]. AION CS, 2008 [cit. 2024-04-12]. Dostupné z: https://www.zakonyprolidi.cz/cs/2008-300.

34. Zákon č. 89/1995 Sb. Zákon o státní statistické službě. In: *Zákony pro lidi* [online]. AION CS, 1995 [cit. 2024-04-12]. Dostupné z: https://www.zakonyprolidi.cz/cs/1995-89.

35. Hypertext Preprocessor. In: *PHP* [online]. The PHP Group, 2024 [cit. 2024-04-23]. Dostupné z: https://www.php.net/.

36. Symfony, High Performance PHP Framework for Web Development. In: *Symfony* [online]. Symfony SAS, [b.r.] [cit. 2024-04-23]. Dostupné z: https://www.tmdn.org/tmview/#/tmview.

37. SKVORC, B. The Best PHP Framework for 2015: SitePoint Survey Results. In: *SitePoint* [online]. SitePoint Pty. Ltd., 2015 [cit. 2024-04-23]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/.

38. The most popular HTML, CSS, and JS library in the world. In: *Bootstrap* [online]. Bootstrap team, [b.r.] [cit. 2024-04-20]. Dostupné z: https://getbootstrap.com/.

39. PostgreSQL: The World's Most Advanced Open Source Relational Database. In: *PostgreSQL* [online]. PostgreSQL Global Development Group, 2024 [cit. 2024-04-23]. Dostupné z: https://www.postgresql.org/.

40. The Go Programming Language. In: *Go* [online]. Google, [b.r.] [cit. 2024-04-23]. Dostupné z: https://go.dev/.

41. Computer Vision using Go and OpenCV 4. In: *GoCV* [online]. The Hybrid Group, [b.r.] [cit. 2024-04-23]. Dostupné z: https://gocv.io/.

42. Open Computer Vision Library. In: *OpenCV* [online]. OpenCV team, 2024 [cit. 2024-04-23]. Dostupné z: https://opencv.org/.

43. S3 & Kubernetes Native Object Storage for AI. In: *MinIO* [online]. MinIO, Inc., 2024 [cit. 2024-04-23]. Dostupné z: https://min.io/.

44. Security. In: *Symfony* [online]. Symfony SAS, 2024 [cit. 2024-04-20]. Dostupné z: https://symfony.com/doc/current/security.html.

45. Translations. In: *Symfony* [online]. Symfony SAS, 2024 [cit. 2024-04-20]. Dostupné z: https://symfony.com/doc/current/translation.html.

46. Twig Components. In: *Symfony* [online]. Symfony SAS, 2024 [cit. 2024-04-20]. Dostupné z: https://symfony.com/bundles/ux-twig-component/current/index.html.

47. The Cache Component. In: *Symfony* [online]. Symfony SAS, 2024 [cit. 2024-04-20]. Dostupné z: https://symfony.com/doc/current/components/cache.html.

48. How to Create and Enable Custom User Checkers. In: *Symfony* [online]. Symfony SAS, 2024 [cit. 2024-04-20]. Dostupné z: https://symfony.com/doc/current/security/user_checkers.html.

49. Bots: An introduction for developers. In: *Telegram* [online]. Telegram, [b.r.] [cit. 2024-04-20]. Dostupné z: https://core.telegram.org/bots.

50. PHP Telegram Bot. In: *GitHub* [online]. noplanman, 2024 [cit. 2024-04-20]. Dostupné z: https://github.com/php-telegram-bot/core.

51. Bots FAQ. In: *Telegram* [online]. Telegram, [b.r.] [cit. 2024-04-20]. Dostupné z: https://core.telegram.org/bots/faq.

52. The Console Component. In: *Symfony* [online]. Symfony SAS, 2024 [cit. 2024-04-20]. Dostupné z: https://symfony.com/doc/current/components/console.html.

53. YERGEAU, F. UTF-8, a transformation format of ISO 10646. In: *IETF* [online]. IETF, 2003 [cit. 2024-04-22]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc3629.

54. PIKE, R. Strings, bytes, runes and characters in Go. In: *The Go Blog* [online]. Google, 2013 [cit. 2024-04-22]. Dostupné z: https://go.dev/blog/strings.

55. GERRAND, A. Defer, Panic, and Recover. In: *The Go Blog* [online]. Google, 2010 [cit. 2024-04-21]. Dostupné z: https://go.dev/blog/defer-panic-and-recover.

56. testing package. In: *Go Packages* [online]. Google, 2024 [cit. 2024-04-22]. Dostupné z: https://pkg.go.dev/testing.

57. TMView. In: *TMView* [online]. European Union Intellectual Property Network, 2024 [cit. 2024-04-23]. Dostupné z: https://www.tmdn.org/tmview/#/tmview.

58. Registr ekonomických subjektů. In: *Registr ekonomických subjektů* [online]. Český statistický úřad, 2024 [cit. 2024-04-23]. Dostupné z: https://apl.czso.cz/res/info.

59. POLČÁK, R. Vybraná judikatura – „ceskapojistovna.cz“. In: *Kolizní otázky internetových právních vztahů* [online]. Právnická fakulta, Masarykova univerzita, 2004 [cit. 2024-04-03]. Dostupné z: https://is.muni.cz/do/1499/el/estud/praf/js09/kolize/web/pages/judikatura-ceskapojistovna.html.

60. COSTELLO, A. Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA). In: *IETF* [online]. IETF, 2003 [cit. 2024-04-03]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc3492.

61. MCCARTHY, K. That apple.com link you clicked on? Yeah, it's actually Russian. In: *The Register* [online]. Situation Publishing LTD., 2017 [cit. 2024-04-03]. Dostupné z: https://www.theregister.com/2017/04/18/homograph_attack_again/.

62. Paris Convention for the Protection of Industrial Property. In: *WIPO Lex* [online]. WIPO, 1979 [cit. 2024-04-03]. Dostupné z: https://www.wipo.int/wipolex/en/text/288514.

63. Protocol Relating to the Madrid Agreement Concerning the International Registration of Marks. In: *WIPO Lex* [online]. WIPO, 2007 [cit. 2024-04-03]. Dostupné z: https://www.wipo.int/wipolex/en/text/283484.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `src` ..... zdrojové kódy implementace
  - `samples` ..... implementace algoritmů pro testovací účely
  - `scraper` ..... zdrojové kódy scraperu
  - `web` ..... zdrojové kódy webové aplikace
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF