VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in Company

2020 Marek Hanuš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Marek Hanuš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ATLAS consulting spol. s r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Radim Bača, Ph.D.**

Konzultant bakalářské práce: Mgr. Tomáš Řehák

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2020 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2020 .....

Rád bych poděkoval společnosti ATLAS consulting spol. s r.o. za to, že mi umožnila vykonání odborné praxe. Dále bych chtěl poděkovat kolegům z Data Science týmu, především Bc. Jakubovi Kólovi, vedoucímu projektu Ing. Martinovi Teleckému a svému nadřízenému Mgr. Bc. Tomáši Řehákovi za cenné rady při zpracovávání úkolů, trpělivost a vytvoření přátelského kolektivu. Také děkuji svému vedoucímu práce doc. Ing. Radimu Bačovi, Ph.D.

Abstrakt

Tato bakalářská práce popisuje činnost vykonanou během odborné praxe ve společnosti ATLAS consulting spol. s r.o., která se zabývá vývojem právních a manažerských informačních systémů. Hlavní náplní praxe bylo vyvinout služby pro získání, zpracování a komparaci dat do nové aplikace HLÍDAČ OCHRANNÝCH ZNÁMEK. Součástí tohoto úkolu bylo nutné navrhnout vhodné workflow získávání dat, databázové struktury, implementovat služby a spravovat stabilní serverové prostředí produkční aplikace.

Klíčová slova: ATLAS consulting spol. s r.o.; bakalářská praxe; ochranné známky; Python; SQL databáze; NoSQL databáze; analytika; strojové učení; zpracování dat; webová aplikace; kontejnerizace; virtualizace; DevOps

Abstract

This thesis describes the work performed during internship at ATLAS consulting spol. s r.o., a software development company focused on legal and management information systems. The main scope of practice has been to develop services for obtain, process and compare data for new application HLÍDAČ OCHRANNÝCH ZNÁMEK. Part of this task was necessary to create workflow to obtain data, design database structures, implement services and maintain stable server environment of production application.

Keywords: ATLAS consulting spol. s r.o.; bachelor practice; trademarks; Python; SQL databases; NoSQL databases; analytics; machine learning; data processing; web application; containerization; virtualization; DevOps

Obsah

Seznam použitých zkratek a symbolů

OZ	–	Ochranná známka / Trademark
MVP	–	Minimální produkt / Minimum viable product
ÚPV / IPOCZ	–	Úřad průmyslového vlastnictví / Industrial Property Office of Czech Republic
EUIPO	–	Úřad Evropské unie pro duševní vlastnictví / European Union Intellectual Property Office
WIPO	–	Světová organizace duševního vlastnictví / World Intellectual Property Organization
SaaS	–	Software jako služby / Software as a service
API	–	Application programming interface
OOP	–	Objektově orientované programování
RDBMS	–	Relační databáze / Relational Database Management System
ORM	–	Objektově relační mapování / Object-relation mapping
ML	–	Machine learning / strojové učení
FTP	–	File Transfer Protocol
CDN	–	Content Delivery Network
PyPI	–	Python Package Index (balíčkovací systém)
AWS	–	Amazon Web Services
CLI	–	Příkazový řádek / Command-line interface
VM	–	Virtuální stroj / Virtual machine
vCPU	–	Virtual central processing unit
XML	–	Extensible Markup Language (formátovací jazyk)
YAML	–	YAML Ain't Markup Language (formátovací jazyk)
SQL	–	Structured Query Language
ZIP	–	Kompresní archivní formát
TIF / TIFF	–	Tagged Image File Format (formát obrazu)
PNG	–	Portable Network Graphics (formát obrazu)
JPG / JPEG	–	Joint Photographic Experts Group (formát obrazu)

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Cílem této práce je popsat pracovní náplň a zhodnotit přínos odborné praxe, kterou jsem vykonal během bakalářského studia ve společnosti ATLAS consulting spol. s r.o.

V druhé části (2) představím samotnou společnost, její nejvýznamnější produkty, mé pracovní zařazení a firemní prostředí, ve kterém se samotná praxe odehrávala.

V následující kapitole (3) poté popíšu konkrétní produkt, na kterém jsem se po celou dobu odborné praxe podílel. Popsán je jeho účel, základní funkcionalita a analýza dalších možností jeho budoucího rozvoje.

Obsah čtvrté kapitoly (4) je věnován mé pracovní náplni, vybraným větším úkolům, na kterých jsem se v aplikaci HLÍDAČ OCHRANNÝCH ZNÁMEK podílel a dále technické detaily největších problémů, na které jsem při implementaci narazil.

Další část (5) obsahuje stručný popis nejdůležitějších technologií využitých v aplikaci, případně při jejím vývoji.

V závěrečné části (6) je zhodnocen celý průběh odborné praxe, získané zkušenosti a znalosti. Jako poslední uvádím výhody získávání praktických zkušeností oproti omezeným zkušenostem, které je možné získat jen během bakalářského studia na vysoké škole bez praxe.

2 Společnost ATLAS consulting spol. s r.o.

ATLAS consulting spol s r.o. je ryze českou společností, která se už od roku 1992 věnuje vývoji, výrobě a distribuci informačních systémů a aplikací v oblasti práva a ekonomiky. [1] Se společností ATLAS software a.s. je členem skupiny ATLAS GROUP. Společnost sídlí v budově ABC v městské části Moravská Ostrava, dále má pobočky v Praze a Brně. Aktuálně zaměstnává okolo 200 lidí. Mezi nejvýznamnější produkty patří právní informační systém CODEXIS® a dále řada manažerských softwarů ECONIX, do kterých spadají např. produkty EQUANTA®, SMLOUVY a HLÍDAČ OCHRANNÝCH ZNÁMEK. Vývojáři jsou rozděleni do malých nezávislých týmů soustředících se samostatně na daný produkt.

Obrázek 1: Logo společnosti ATLAS consulting spol. s r.o. [1]

2.1 Pracovní zařazení

Ve společnosti se angažuji již 2. rokem, v období počátku odborné praxe jsem byl zařazen do oddělení vývoje produktu HLÍDAČ OCHRANNÝCH ZNÁMEK, konkrétně na programování backendové části přípravy dat a komparací OZ. Mým primárním úkolem byla převážně implementace služeb v jazyce Python, která občas přesahovala i do jazyků PHP, Ruby, Java. Část času jsem věnoval také psaní bash skriptů, databázových procedur a případně jejich optimalizaci. Dalšími povinnostmi byla účast na pohovorech potenciálních kolegů do backendového týmu a někdy také rozhodování o jejich dalším setrvání.

Zpočátku byl mým vedoucím projektu a mentorem Radovat Kepák, ale po jeho odchodu z firmy jej nahradil Ing. Martin Telecký a přímým nadřízeným se stal Mgr. Bc. Tomáš Řehák. Součástí backendového týmu byl „ML guru“ Bc. Jakub Kól, po určitou dobu také programátoři Marek Vlčinský a Ing. Ondřej Garncarz, dále frontend PHP vývojáři Bronislav Ceh a Michal Matúš. Na analytice a návrhu se také podíleli za oddělení analýz Ing. et Ing. Gabriela Hrubešová, Ing. Veronika Štětková a vedoucí Ing. Zdeněk Vrba.

3 Produkt HLÍDAČ OCHRANNÝCH ZNÁMEK

Nová aplikace HLÍDAČ OCHRANNÝCH ZNÁMEK je poskytována jako SaaS a umožňuje uživatelům získat plnou kontrolu nad vlastními OZ a pomáhá je chránit před zneužitím konkurencí. Taktéž v připravované funkcionalitě umožňuje upozorňovat na nové přihlášky od konkurence. Logo produktu je zobrazeno na obrázku 2.

Obrázek 2: Logo produktu HLÍDAČ OCHRANNÝCH ZNÁMEK [2]

3.1 Právní důvod

Od 1.1.2019 dochází v novele zákona o OZ k zásadním změně při registraci OZ. Nově již ÚPV nebude zkoumat, zda přihláška není shodná nebo neobsahuje shodné prvky se starší OZ a povinnost se přenáší na vlastníky současných OZ, kterým je umožněno podávat námitky proti registraci. Pokud vlastník nevznese včas námitku, ÚPV OZ zapíše i přes shodnost s již exitující OZ. Vlastníci by si tedy měli pravidelně provádět monitoring zveřejněných OZ. [3]

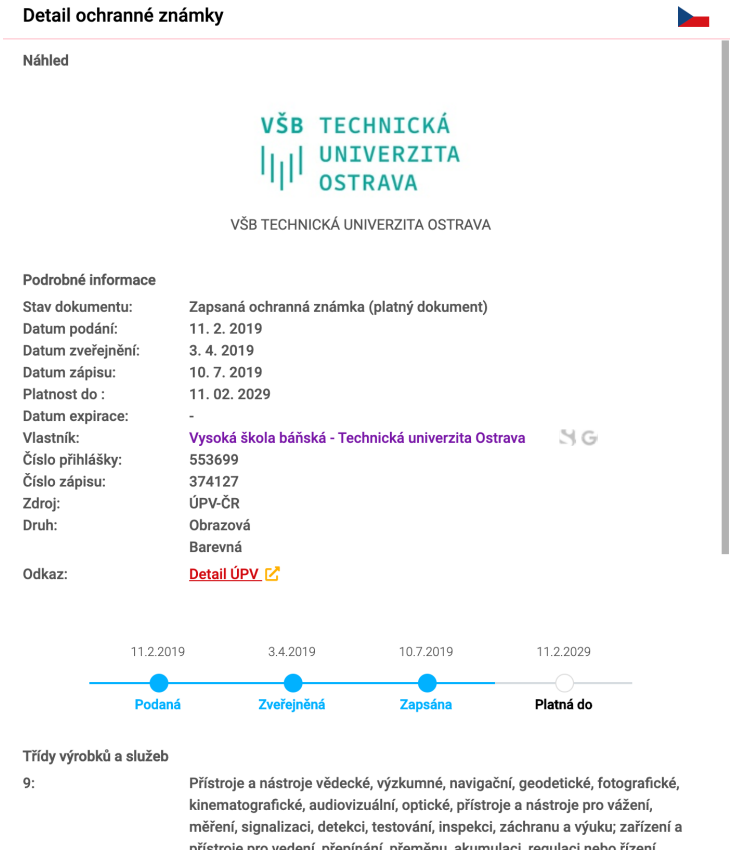
Povinnost se tedy nově přenáší na majitele OZ, aby si svou registrovanou OZ chránil před registrací jiným subjektem. Úřad vydává v týdenních intervalech věstníky, ve kterých exportuje informace o nově přihlašovaných OZ a uživatelé mají od data zveřejnění věstníku lhůtu 3 měsíce na podání námitek. Pokud tak neučiní, je OZ po 3 měsících prohlášena za registrovanou, přechází tedy do stavu zapsána a je velmi náročné takovou kolizní OZ dodatečně napadnout a obhájit si na ni svá práva. Pravděpodobnost úspěchu v soudním sporu je poté velmi nízká. Naopak v případě podání námitky v řádné lhůtě k OZ, na které je zjevná alespoň částečná podobnost, je vysoká šance na úspěch formou následného zamítnutí přihlašované OZ úřadem. Majitelům přihlašované kolizní OZ je poté většinou doporučeno ze strany patentových zástupců si takovou registraci nechat zamítnout, než řešit spor v dlouhých a nákladných soudních procesech.

3.2 Datové zdroje

Aplikace si udržuje vlastní databázi OZ, kterou automaticky denně aktualizuje z datových zdrojů českého ÚPV [4] a evropského úřadu OHIM [5], který byl v roce 2016 přejmenován na EUIPO. Do budoucna je vize integrace dat systému Madrid ze světového úřadu WIPO. Poté v souvislosti s expanzí obchodního zastoupení do nejbližších evropských států se očekává také zahrnutí dat ze zdrojů např. „Úrad priemyselného vlastníctva Slovenskej republiky“ nebo polského „Urząd Patentowy Rzeczypospolitej Polskiej.“

3.3 Funkce

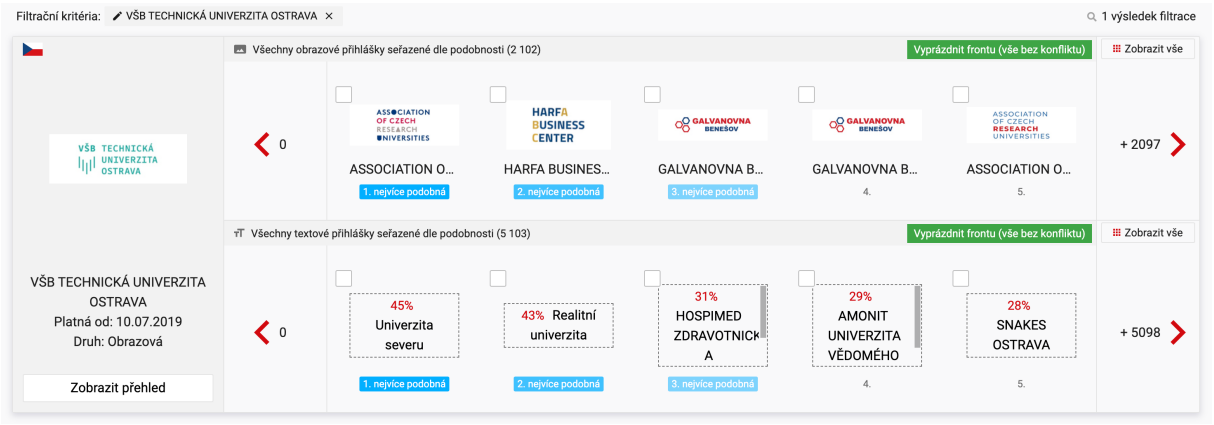
Po prvním přihlášení průvodce provede uživatele přidáním první známky do modulu *hlídané OZ*. Po zpracování komparací se uživateli zobrazí hlavní funkcionalita, a to přehledné seřazení napadnutelných OZ do fronty podle pravděpodobnosti shody. Pojem „napadnutelných OZ“ lze vysvětlit jako sjednocení všech nově přihlašovaných známek, tedy OZ ve stavech podaná nebo zveřejněná, včetně podstavů vkládání údajů, průzkum a námitky. Z toho je patrné, že uživatelé aplikace mají možnost nahlížet i mezi OZ, které ještě nebyly zveřejněny ve věstníku, ale jsou ještě ve stavu podávání přihlášky. Tento stav může trvat i několik měsíců. V budoucnu bude tato informace užitečná, pokud budu chtít sledovat nově podané OZ od konkurence. Implementace zobrazení front je znázorněna na obrázku 4. Pro každou OZ je možné zobrazit detail podobně jako na obrázku 3.



Obrázek 3: Detail OZ se všemi metadaty [6]

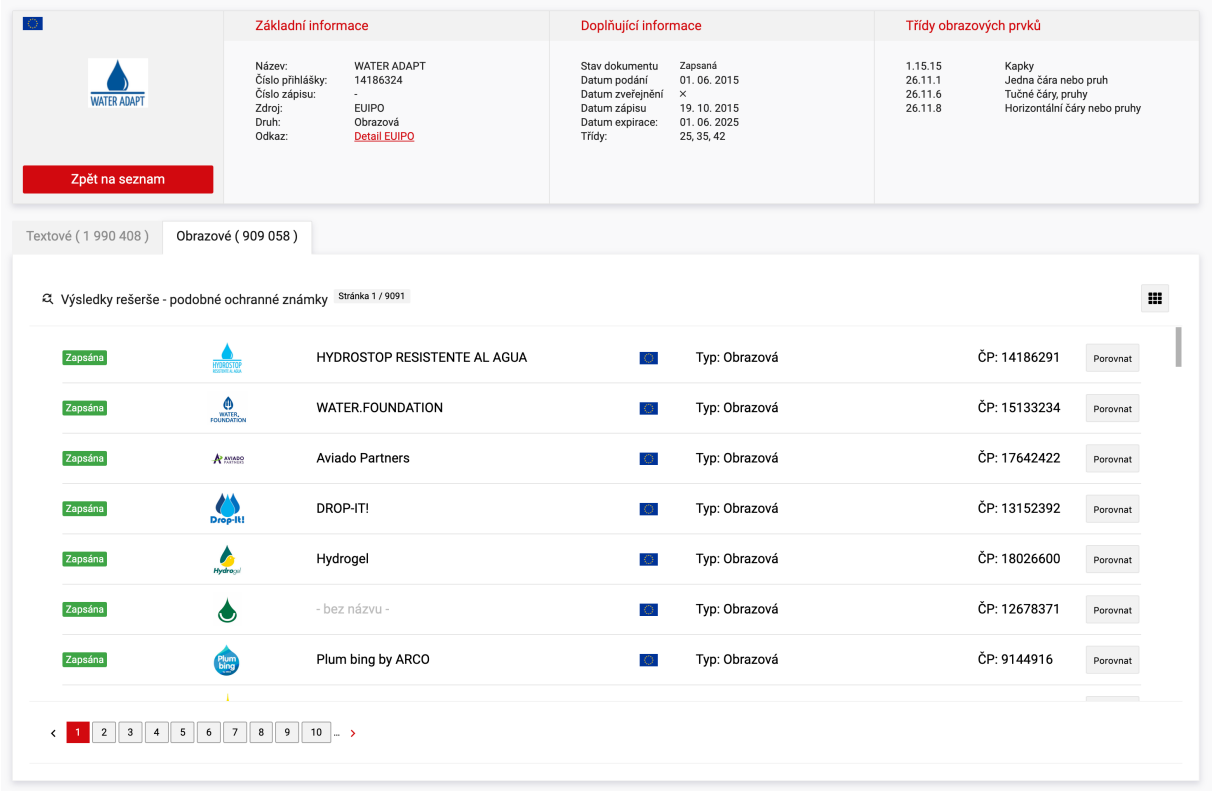
Fronty jsou v MVP implementovány dvě, jedna se seřazenými známkami podle textové a druhá podle obrazové podobnosti. Zároveň jsou do textové fronty zařazeny pouze známky obsahující textové vyjádření a do obrazové fronty obsahující zpracovatelný obrázek. Tím je také umožněno hlídat OZ typu prostorová, slovní grafická (dnes se již nepoužívá), zvuková, hologram, tvořená barvou a poziční. Fronty mohou uživatelé pravidelně vyprazdňovat, čímž se jim

zobrazují při dalším navštívení aplikace pouze OZ, které k dané známce ještě neviděli. Dále je možné v aplikaci nastavit automatické zasílání notifikace na e-mail, pokud pravděpodobnost shody překročí nastavenou hranici.



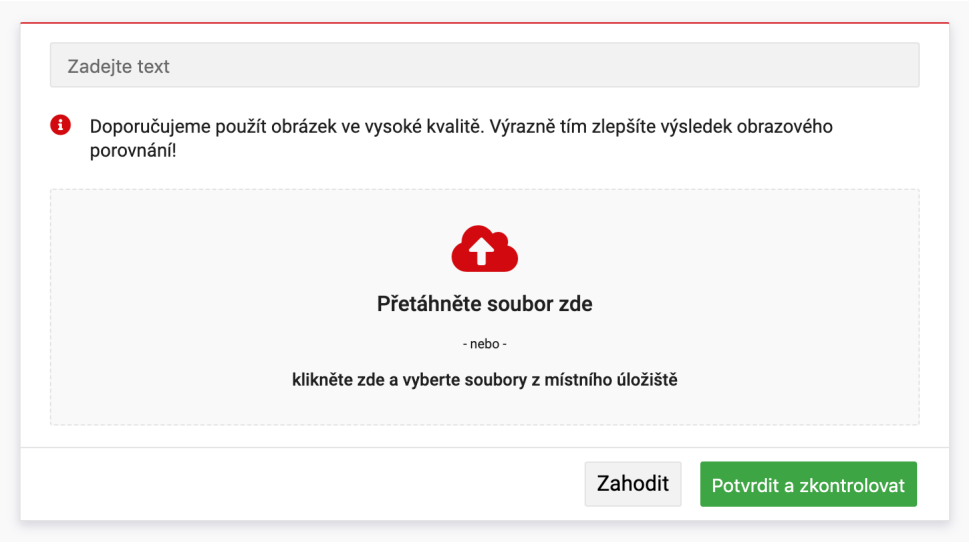
Obrázek 4: Zobrazení fronty nových přihlášek [6]

Další funkcionalitou jsou *rešerše OZ*, znázorněné na obrázku 5. Podobně jako v předchozím odstavci jde o hledání podobných OZ, ale navíc se zobrazují i OZ v ostatních právních stavech. Tedy jde o stavy např. registrovaná, zamítnutá, zrušená, ex nunc (zrušený platný dokument) a ex tunc (neplatný dokument). Dále je zobrazení rozšířeno o hledání napříč zdroji, tedy k OZ z ÚPV se zobrazí i podobné OZ z databáze EUIPO. Oproti hlídání není možné seznamy čistit, ani zasílat notifikace a je omezeno procházení na pouze prvních několik stran (vzhledem k omezení komparačních algoritmů). Pro počet rešerší není definováno žádné licenční omezení.



Obrázek 5: Zpracování rešerše OZ [6]

Rešerše OZ je dále doplněna o možnost vložení *vlastního návrhu*, kam může uživatel zadat vlastní textové vyjádření a nahrát obrázek. Pro úspěšnou komparaci je zapotřebí, aby uživatel nahrál obrázek v dobré kvalitě a bez průhlednosti (alfa kanálu). Vizualizace formuláře pro zadání dat OZ je znázorněna na obrázku 6.



Obrázek 6: Vytvoření řešerše návrhu vlastní OZ [6]

Poslední funkcí, ale zatím nedokončenou, je *hlídání konkurence* pomocí získaných dat vlastníků OZ. Uživatel si tedy bude moct definovat název společnosti nebo jeho část a systém za něj bude hlídat nově podané přihlášky daného vlastníka, změny u existujících OZ a samozřejmě automaticky zasílat e-mailové notifikace dle nastavení v aplikaci.

Další možností rozvoje je automatické hlídání uživatelem zvolené části obrazového nebo textového vyjádření OZ.

4 Pracovní náplň

V předchozí kapitole jsem popsal klíčové funkce aplikace a záměrně jsem vybral pouze ty části systému, u kterých bylo potřeba zajistit datovou vrstvu. Aby bylo možné efektivně pracovat s datovými procesy, rozdělil jsem přípravu dat na následující služby:

stahovač otevřených dat ÚPV – automatický stahovač ZIP exportů otevřených dat, při dalším spuštění stahuje inkrementálně pouze DIFF exporty.

parser otevřených dat ÚPV – služba, která je schopna rozbalit ZIP archiv, inkrementálně proiterovat nezpracované exporty, parsovat XML data a převést je do databázové struktury, uložit přílohy do centrálního úložiště včetně zamezení duplicitního uložení stejných vícekrát exportovaných příloh (většinou se jedná o obrázky typu GIF, JPG, ale vyskytují se občas MP3 a MP4 formáty). Zároveň také generuje miniatury.

stahovač otevřených dat EUIPO – inkrementální stahovač ZIP archivů OZ a anonymizovaných dat vlastníků OZ, jedná se o stahovač z FTP úložiště se zajištěním opakování pokusů při chybě, vzhledem k velmi častých chybám a opakující se nedostupnosti FTP serveru.

parser otevřených dat EUIPO – zajišťuje inkrementální rozbalovaní ZIP archivů, uložení rozsáhlých metadat do relační databáze a přenesení příloh do centrálního úložiště (převážně se jedná o JPG a TIF soubory). Jako postprocessing převádí TIFF soubory (podle formátu nikoliv přípony, která je poměrně nespolehlivá) do PNG. Také generuje miniatury.

obrazový a textový komparátor – API servery poskytující seřazené fronty identifikátorů OZ, seřazených podle pravděpodobnosti shody. Vzhledem k množství dat je nutné v nočních aktualizacích připravovat indexy, ukládat výsledků komparací do cache a vracet výsledky stránkovaně. Zajišťuje data do front v modulu hlídaní OZ a rešerší. Taktéž je schopna v reálném čase zpracovat a provést komparaci nového návrhu OZ s použitím existujícího indexu.

denní aktualizační skript – služba zajišťující automatické spuštění denní aktualizace dat, řeší správnou návaznost spouštění jednotlivých služeb, souběžnost některých služeb (pokud je to logicky možné a dovoluje to dostupná kapacita), přípravu indexů a znovunačtení použitých indexů v komparátorech.

podpůrné služby – např. proxy s cache pro obrazové a textové komparátory, CDN server s obrázky OZ, skripty na zálohování databáze, monitoring, logovací služba.

Všechny služby jsou zobrazeny v návrhu architektury ve schématu 9. Průběh denní aktualizace je zobrazen ve FlowChart schématu 8.

Získávání, zpracování informací ze zdroje ÚPV, návrh datových struktur, podpůrné služby, skripty a správa DevOps je kompletně má vlastní práce. Na zpracování dat z EUIPO jsem se

z důvodu časového tlaku podílel napůl s kolegou Markem Vlčinským. Oba komparátory jsou také má práce, ML algoritmus do komparace dodal Bc. Jakub Kól. Frontend rozhraní zpracovali PHP specialisté. Na migraci obrázků do S3 jsem se podílel návrhem a řešením některých komplikací, většinu programovacích prací řešil kolega Ing. Ondřej Garncarz.

V následujících sekcích uvedu způsob získávání informací, postupy zpracování úkolů a řešení nastalých chyb.

4.1 Porada se soudním znalcem v oboru OZ

Před samotnou implementací a současně i s ní bylo nutné provádět postupné analýzy, abychom detailně pochopili danou problematiku. Produkt je v ČR naprosto unikátní a není možné najít konkurenční systém, který by nabízel podobnou funkcionalitu.

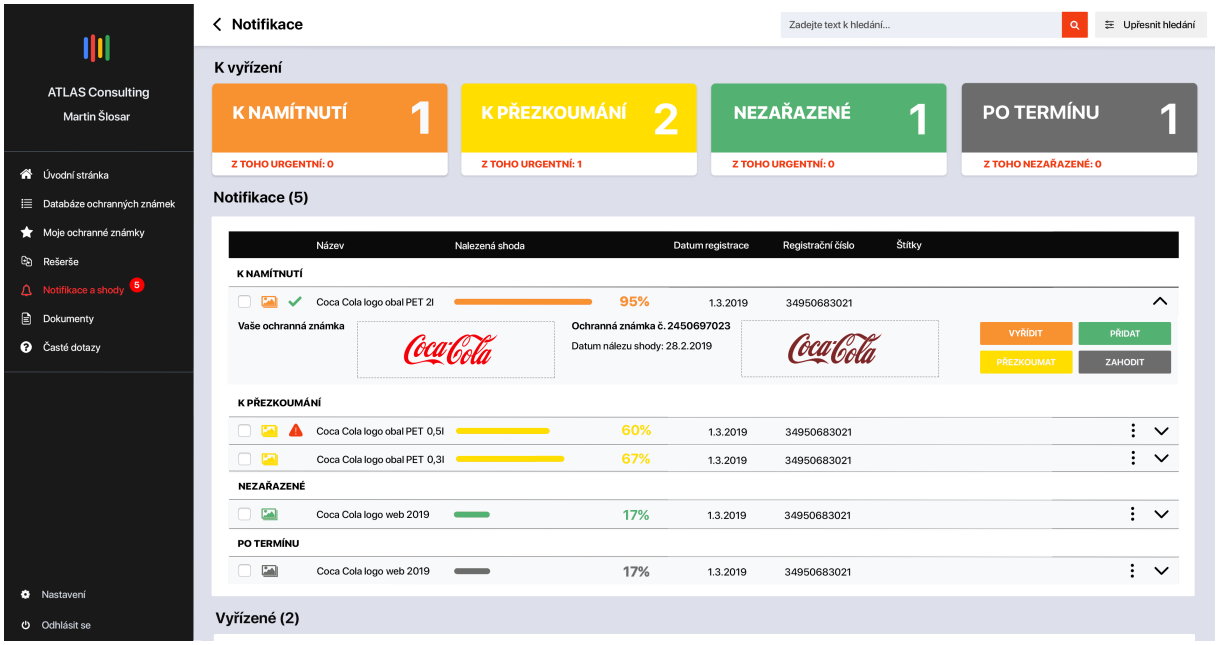
Proto jsme oslovili soudního znalce v oboru OZ, aby s námi probral způsob, jak provádí známkovou rešerši. Dále bylo cílem prodiskutovat náš návrh aplikace. Také jsme již měli připravený prototyp komparačního algoritmu.

Od soudního znalce jsme se dozvěděli, že pravidelnou rešerši provádí ručně, tedy postupně prochází národní a mezinárodní databáze a zapisuje do nich textová vyjádření OZ. Výsledky manuálně prochází a vyhodnocuje OZ, které by mohly být podobné nebo shodné.

Dle názoru soudního znalce nelze použít aplikaci jako oficiální odůvodňující dokument pro rozhodný orgán, ale pouze jen jako předběžnou rešerši. Podobnost se dále velice špatně prokazuje a bez znalostí a zkušeností soudního znalce je těžké podobnost obhájit. Také např. u léčiv, úřady řeší i chemické sloučení a léčenou nemoc. Námitky a výsledky sporů nejsou veřejně dostupné.

4.2 Nepovedený návrh

Původní návrh aplikace, zobrazený na obrázku 7, spočíval ve vytvoření rozhraní, ve kterém si uživatel označí OZ, které chce sledovat. Komparátory by v offline režimu každý den hledaly možné shody a generovaly uživateli potencionální shody, tedy by se zobrazovaly pouze dvojice OZ – hlídaná OZ oproti kolizní OZ. Nalezené možné kolize by byly seřazeny podle pravděpodobnosti sestupně. Součástí by byly i e-mailové notifikace. Webová aplikace počítala s nasazením v cloudu s offline backend službami spouštěnými periodicky v cronu. Tyto služby by byly hostovány na interní firemní virtualizaci, jelikož komparace jsou velmi náročné na HW prostředky a zároveň není nezbytně nutné zajistit 100% dostupnost těchto služeb.



Obrázek 7: Původní návrh [6]

Tento koncept nicméně narazil ve fázi realizace na hranice možností použitých algoritmů. Zejména obrazová komparace je velice subjektivní a uživatel nemusí v nabízených výsledcích podobnost najít a pochopit. Dále je téměř nemožné algoritmem rozhodnout, zda je míra kolize dostatečná, aby byla uživateli vytvořena notifikace. Po nastavení určité hranice je určitá pravděpodobnost, že algoritmus nevyhodnotí vytvoření notifikace za nezbytné a uživatel se o možné kolizní OZ nedozví a aplikace mu vůbec neumožní se k této OZ, jakkoliv dostat.

Přestože vedoucí projektu od této myšlenky nehodlal ustoupit, bylo z těchto jasných důvodů od téměř dokončeného produktu po první demo prezentaci upuštěno a začalo se hledat nové řešení. Vedoucí projektu později z této pozice odstoupil a nahradil jej zkušenější kolega Ing. Martin Telecký.

4.3 Hlídání front známek

Jedinou vyhovující variantou s přenesením rozhodnutí o kolizi na uživatele bylo vytvoření systému hlídání front. Nové přihlášky OZ jsou uživateli seřazeny podle různých algoritmů ve více frontách. Tedy k hlídané OZ se zobrazí až dvě fronty podle toho, jestli daná OZ obsahuje textové nebo obrazové vyjádření, případně obojí dohromady. Uživatel poté může frontu procházet a hledat v zobrazených textových a obrazových vyjádřeních možnou kolizní OZ, kterou si označí a přesune v systému do složky podezřelých OZ.

Tímto způsobem má uživatel možnost projít si všechny nové přihlášky a díky řadícímu algoritmu se mu zobrazují na prvních místech shody s nejvyšší pravděpodobností. V praxi tedy uživatel, pokud nenajde možnou kolizi v první 5 až 10 známkách (v určitých případech v prvních desítkách), nemusí již procházet frontu dál. Ve frontách se nachází řádově desetitisíce OZ, kde

oproti ručnímu porovnání s každou OZ ušetří aplikace uživateli množství stráveného času. Protože na českém trhu existují společnosti se stovkami OZ, je možné jim hlídání takového počtu OZ velmi zjednodušit s minimálními požadavky na personál obsluhující aplikaci.

Aby byl celý workflow efektivní a uživatel neviděl po každém spuštění aplikace stejné fronty obsazené desetitisícem OZ, je v aplikaci implementováno tlačítko pro vyčištění fronty. Reálně by ale bylo nutné v databázi uložit např. 70 000 identifikátorů OZ * 2 fronty * celkový počet hlídaných známek napříč uživateli v aplikaci. Protože by bylo náročné uchovávat takové množství informací všech odstraněných známek z front, bylo nutné hledat náhradní řešení. Proto bylo vytvořeno na základě seskupení procesů do denní aktualizace, ukládání pouze data posledního vyčištění zvlášť textové a obrazové fronty. Toto datum se vztahuje k datu poslední aktualizace datového exportu zdroje. Tedy po vyčištění fronty si systém uloží pouze datum poslední aktualizace, které získá z backendového API a při dalším načtení aplikace podle tohoto data zobrazí uživateli ve frontách pouze OZ, přidané nebo změněné po tomto datu.

Při komparaci je tedy nutné seřadit všechny neregistrované OZ (logaritmická složitost) a poté iterovat přes výsledek a porovnávat data podání přihlášky s datem posledního vyčištění fronty (lineární složitost). Kombinace těchto dvou složitostí algoritmů stále umožňuje provádět komparace a zobrazovat výsledky v reálném čase.

4.4 Zpracování dat z EUIPO

Přestože aplikace obsahovala pouze data ÚPV byl prodej licencí klientům již zahájen. Dle negativní zpětné vazby od obchodníků bylo nutné, co nejdříve zahrnout do aplikace i data evropského úřadu EUIPO.

Ačkoliv by se tento úkol zdál pouze jako zpracování podobné struktury dat jako u otevřených dat ÚPV, reálně byl vývoj mnohem složitější a nastalo spoustu komplikací. Stahování dat je možné z exportů umístěných na portálu EUIPO, nicméně tuto stránku je strojově obtížně vytěžovat. Před přístupem k seznamu souborů portál požaduje zaškrtnutí checkboxu souhlasu s podmínkami. Tento checkbox je zablokovaný, dokud uživatel neotevře nejdřív dialog s obsahem podmínek. Teprve poté je možné po zavření dialogu zaškrtnout checkbox a potvrdit formulář odesílacím tlačítkem. Na pozadí aplikace proběhne předání autorizačních klíčů a zobrazí se výpis adresáře. Po kliknutí na kterýkoliv soubor dojde k přesměrovaní na jeho adresu a předání autorizačních klíčů v parametru.

Po delší analýze možností automatizovaného skriptu na stahování těchto dat, se nám nakonec podařilo zjistit, že po přesměrování a autorizování požadavku server uživatele přesměruje na FTP server, ze kterého se provede stažení souboru. Na pozadí se předají přihlašovací údaje, které jsou zatím stále (již několik měsíců) neměnné. Implementace automatického stahovače spočívala v jednoduchém skriptu, který bude rekurzivně zpracovávat adresáře a stahovat jejich obsah do interního MinIO storage.

Úřad průběžně vydává nové exporty a po čase maže jejich starší verze, tedy je pro nás nutné si tyto soubory uchovávat a nemazat dojde-li k jejich odstranění na portálu úřadu. Exporty zabírají celkově okolo 150 GB místa na disku.

Další částí bylo napsat službu, která rozbalí ZIP soubory, zpracuje XML soubory a uloží jejich obsah do databáze. Datové soubory EUIPO obsahují mnohem více atributů a složitější strukturu XML. Aktuálně je z těchto souboru vyparsováno 3,5 milionu známek rozložených v 520 exportech. Množství unikátních známek (podle čísla přihlášky) je skoro 1,8 milionu. Zbytek jsou pouze aktualizační transakce, které obsahují kompletní strukturu dat, a nikoliv jen informaci o změně a nové hodnoty změněných atributů. Dále služba nahraje přílohy do objektového datového storage. V tuto chvíli se ukázalo jako nejrychlejší řešení uložit jejich obsah do databáze podobně jako u dat ÚPV.

Vzhledem k množství dat byla ale doba zpracování kompletního datasetu 72 hodin oproti 1,5 hodině u datasetu ÚPV. To bylo způsobeno množstvím exportovaných atributů, násobně větším počtem OZ, množstvím mnohem častějších aktualizačních transakcí a většími obrázky.

Bylo upuštěno od zpracování všech jazykových mutací Niceských tříd, kdy jsme se rozhodli spuštěný testovací skript kompletního zpracování ukončit po 6 dnech nepřetržitého běhu. Při zpracovávání je každý export obalen databázovou transakcí, což velmi snižuje čas potřebný ke zpracování. Transakce jsou vytvořeny pro každý exportní balíček, dojde-li tedy při importu k chybě, spustí se implicitně operace ROLLBACK k vrácení všech změn provedených touto transakcí. V opačném případě je transakce příkazem COMMIT potvrzena.

Na poslední chvíli před nasazením do produkce jsme si také všimli, že zhruba 58 tis. obrázků, tj. zhruba 8 %, se uživatelům nedokáže zobrazit. To bylo způsobeno obrazovým formátem TIFF, který webové prohlížeče nepodporují. Ke zpracování musela být rychle doplněna služba, která dokázala rozpoznat typ souboru a pokud se jednalo o TIFF, tak jej překonvertovat do podporovaného formátu. Detekce podle přípony souboru je sice mnohem rychlejší, ale u zmíněného datového zdroje nespolehlivá, neboť úřad exportuje i TIFF s příponou JPG. Ke konverzi byl použit nástroj ImageMagick a jako výstupní formát byl zvolen PNG, protože je stejně jako TIFF, formátem s bezztrátovou kompresí.

4.5 Neplatná instrukce

Při nasazování velkých datových změn do produkčního prostředí používáme migraci celých VM. Scénář je obvykle takový, že na testovací server se postupně nasazují nové funkce, probíhá opakované zpracování dat, průběžné testování a opravy chyb. Po dokončení implementace se práce předá na oddělení produktové podpory, která dle zadaných scénářů a s pomocí automatických testů zkontroluje správnou funkčnost kompletní aplikace. Nejjistější metodou, jak poté aplikaci přenést do produkce, je vypnout VM, naklonovat VM na produkční virtualizační cluster, tam jej zapnout a připojit uživatelská data. Finální přepnutí nové verze lze takto provést s výpadkem produkční aplikace na maximálně jednotky minut. K tomuto způsobu nasazení se za necelý rok provozu přistoupilo celkově 3x.

Bohužel ani tato metoda není úplně 100% a v jednom případě došlo k neúspěchu a navrácení původní verze. Po spuštění VM totiž jeden z kontejnerů nedokázal naběhnout. Kontejner je aplikace izolovaná v přenosném balíčku obsahující veškeré závislosti, příklad sestavení takového kontejneru je uveden ve zdrojovém kódu 3. Dle stavových informací byl kontejner ukončen chybovým kódem 132. Následovaly opakované pokusy o spuštění, otestování spouštění a restartu na testovacím serveru a poté bližší zkoumání příčiny chyby. Po podrobnější analýze jsem přišel dle manuálů na to, že se jedná o chybu SIGILL, tedy Illegal instruction. Jedna z použitých knihoven byla po aktualizaci sestavena pro procesory s novější instrukční sadou AVX2, bohužel produkční servery s procesory Intel Xeon X5670 touto instrukční sadou nedisponují.

Prvním řešením by bylo, namísto použití předkompilovaných balíčků z PyPI repozitáře, sestavit a zkompilovat balíčky ze zdrojových kódů přímo na serveru. Tohle řešení je ale velmi nepraktické a použití novějších instrukcí AVX2 pro práci s vektory má jistý důvod a jejich vynechání by mělo negativní vliv na výkon aplikace.

Další možností by bylo použít dočasně starší verzi balíčku. Jednalo by se o nejjednodušší možný zásah, jak aplikaci ihned zprovoznit. Starší verze byla kompatibilní a došlo k úspěšnému nasazení. Přesto ale toto řešení nebylo do budoucna perspektivní.

Finálním řešením bylo nakonec postupně vybalancovat VM mezi virtualizačními clustery a přenést celou VM na cluster s novějšími procesory Intel Xeon E5-2667 v4. Tím se mohl provést i upgrade na nejnovější verzi balíčků a zároveň použití novějšího procesoru v kombinaci s aktualizovaným balíčkem mělo za následek přibližně 30% zrychlení komparací v aplikaci.

4.6 Miniatury

V době největšího tlaku na co nejrychlejší vydání rozšiřující funkcionality EUIPO dat nebyl čas na optimalizaci, a proto jsme v této další fázi museli začít řešit optimalizace vzhledem k tomu, že načítání stránky hlídaných OZ mohlo trvat i desítky vteřin. Vše se odvíjelo od počtu hlídaných OZ uživatelem, nicméně se rychle začaly prodávat licence s hlídáním i stovek OZ nebo dokonce neomezené licence, přestože na to aplikace ještě nebyla připravena.

Největší bottleneck byl datový přenos. Protože při testování jsme se nacházeli v interní firemní síti, kde byly také umístěny produkční servery, byla rychlost načítání obrázků poměrně rychlá. To ale nemohli říct uživatelé, kteří k aplikaci přistupovali přes průměrně rychlý internet nebo dokonce přes mobilní datovou síť. Obchodníci také při prezentaci produktu používají mobilní data. Taktéž bylo pro nás neočekáváné zásadní zpomalení při prezentacích mimo firmu.

Byla tedy implementována služba, která načítá obrázky z databáze a postupně pomocí nástroje ImageMagick obrázky komprimuje na velikost 80 px pro nejdelší hranu a generované miniatury ukládá do oddělené databáze.

4.7 Problémy s obrázky v databázi

I přesto, že v dnešní době již databáze plně podporují uložení binárních souborů a takové řešení někteří lidé na diskusních fórech dokonce doporučují [7], došlo při nárůstu dat z EUIPO zdroje ke značným komplikacím.

Binární data jsou uložena v samostatné tabulce v datových polích typu MEDIUMBLOM s limitem velikosti 16 MB. Obsahem tabulky je pouze složený primární klíč a atribut s binárními daty.

První problém nastal při přípravě obrázků pro komparaci. Ten pomocí SQL dotazu vybírá všechny obrázky z tabulky a předává je službě, která postupně v dávkách po 64 obrázcích dopočítává „embedding vectors“ a ukládá je do MongoDB databáze. Tyto vícerozměrné vektory se poté využívají k výpočtu pravděpodobnosti shody v komparaci obrázků.

Předem uvedu, že na serverech je vypnut „swap space“. Tedy není k dispozici místo, kam by OS mohl odložit části virtuální paměti v případě nedostatku operační paměti. při plném obsazení operační paměti dříve docházelo k extrémnímu množství zápisu na relativně pomalý disk (SAS disky v SAN diskovém poli) a tím i k zásadní degradaci rychlosti aplikace. V současné době, pokud dojde k zaplnění paměti, systém násilně ukončí proces s největší spotřebou paměti (většinou se jedná o chybu skriptu nebo proces se špatně omezenou cache dat), ten se automaticky restartuje a aplikace bez problému pokračuje dál v provozu.

Při naplnění pouze českými OZ je v databázi okolo 2 GB dat, tedy se může celý výsledek dotazu uložit do paměti a poté postupně zpracovávat. S aktuálním množstvím více než 100 GB už není možné využívat operační paměť, ale je třeba použít postupný fetch záznamů po dávkách. Ten načítá postupně z databáze řádek po řádku (unbuffered query), většinou po malých dávkách a předává je aplikaci. V tuto chvílí nastává dlouho téměř neřešitelný chybový kód databáze, který nás týdny doslova nenechal v klidu spát:

```
Error Code: 2013. Lost connection to MySQL server during query
```

Celý problém spočívá v ukončení spojení ze strany MySQL serveru, pokud vyprší timeout. Bohužel, i přes veškerou snahu, konzultaci i navyšování prakticky všech dostupných limitů až na extrémní hodnoty, se nám nepodařilo zjistit, co přesně tuto chybu způsobuje. Komplikace nastaly i při testování oprav, kdy pokusné iterace padaly za běhu naprosto náhodně, jindy úspěšně proběhlo i několik pokusů za sebou.

Řešením bylo iterovat přes chunk, tedy rozdělit výsledek na malé dávky pomocí kombinace LIMIT a OFFSET. To se zdálo jako jednoduché řešení, které ale hned přineslo další potíže. Přestože příkaz OFFSET má obecně lineární složitost, při použití s chunk se složitost blíží kvadratické. Server musí načíst veškerá data z disku až po dosažení LIMIT, to vede k obrovské náročnosti na I/O. I přes použití výkonných NVMe SSD disků jsme se velice rychle dostali na zpracování v řádu desítek hodin.

Jako příklad uvedu dotaz s limit = 100 000 a offset = 0, kdy takový dotaz trvá desítky vteřin. Bohužel ten samý dotaz s offset = 2 000 000, už trvá v řádu hodin. Zjednodušený příklad dotazu je uveden ve výpisu 1.

```
SELECT trademark_id, binary_data
FROM tm_images
LIMIT 100000 OFFSET 2000000
```

Výpis 1: Původní SQL dotaz (pro názornost dotaz byl dotaz zjednodušen)

Řešení bylo nakonec velice jednoduché. Stačilo pouze vybrat záznamy z tabulky podle LIMIT a OFFSET bez binárních dat a výstupní set identifikátorů propojit s tabulkou, kde dojde k vybrání záznamů podle implicitně indexovaného primárního klíče. Dotaz je ve výpisu 2.

```
WITH cte AS (
  SELECT DISTINCT trademark_id
  FROM tm_images
  LIMIT 100000 OFFSET 2000000
)
SELECT ti.trademark_id, ti.binary_data
FROM tm_images ti JOIN cte ON ti.trademark_id = cte.trademark_id
```

Výpis 2: Řešení po optimalizaci (pro názornost dotaz byl dotaz zjednodušen)

Jako druhý problém, který zastavil plánovaný release verze do produkce, se stalo přeplňování databáze. K tomu dochází, pokud z databáze souvisle mažeme a vkládáme další data. Bohužel použitý engine InnoDB v MySQL nedokáže s odebranými daty efektivně pracovat a dochází k nekonečnému rozšiřování databáze.

Ve výchozím nastavení má MySQL od verze 8 zapnutou konfiguraci innodb_file_per_table. Zapnutí této volby bylo doporučováno jako řešení problému, v aplikaci je ale použita taktéž verze 8, takže bylo nutné hledat řešení jiné.

Další možností bylo použití příkazu OPTIMIZE TABLE <nazev_tabulky>. Ten sice standardně uvolní volné místo, v našem případě ovšem nepomohl. Systém InnoDB nepodporuje optimalizace, tedy je na pozadí provedeno kopírování dat do nového datového souboru a odstranění původního. I přesto, že nad tabulkou s binárními daty tento příkaz trvá několik hodin, nový datový soubor má nakonec téměř totožnou velikost jako původní.

Jedinou možností, jak uvolnit nevyužité místo bylo provést zálohu databáze a její obnovení. To ovšem není jednoduchý krok, vyžaduje ruční zásah a neobejde se bez několikahodinového částečného výpadku aplikace. Proto byla tato operace prováděna zhruba 1–2x měsíčně o víkendu, kdy je v aplikaci minimální provoz. Bohužel během doby výpadku mohou sice uživatelé navštívit aplikaci, ale nenačtou se obrázky, což je v této aplikaci zásadní funkcionalita.

Mezi další možné řešení, jak se vyhnout problémům s MySQL, byla migrace relační databáze do PostgreSQL. To by ale vyžadovalo upravit databázové struktury, dotazy, procedury a nebyl

by zaručen pozitivní výsledek. Komerční databáze jako např. Microsoft SQL Server nebo Oracle Database by v případě použití v produktu nepřinesly nic zásadního navíc vzhledem k vysoké nákladnosti oproti open-source databázím. Podporu nejpoužívanějších databází MySQL, PostgreSQL a SQL server jsou schopni dlouhodobě zajistit interní databázoví specialisté.

4.8 Migrace obrázků do S3

Nejpřijatelnějším dlouhodobým řešením bylo přenesení obrázků mimo databázi. Jako nejvhodnější řešení byl navržen MinIO storage, tedy open-source objektový storage kompatibilní s Amazon S3. Tohle řešení má spoustu výhod:

- Významně se sníží množství dat v relační databází a tím se zrychlí možnost zálohování, obnovení a zpracování dat
- Sníží se zátěž databázové serveru (převážně síťový provoz)
- Zrychlí se načítání obrázků uživatelům
- Do budoucna se můžou data ukládat na S3 úložištích od Amazonu a sloužit tedy jako velmi rychlá CDN s neomezenou kapacitou

Nevýhody:

- Využitá kapacita je vyšší než při uložení v databázi, kvůli uložení velkého množství malých souborů přímo na souborový systém, což je výchozí systém uložení dat MinIO serveru

Před migrací obrázků se uživatelům posílaly obrázky přímo při načítání, bylo tedy výhodné poslat uživateli v jednom requestu vyrenderované HTML a zároveň v base64 zakódované obrázky. Tohle řešení mělo ale nevýhodu v možnosti cachování obrázků u uživatele, jelikož se použité miniatury často opakují. Použití externí CDN vedlo ke znatelnému zrychlení a zefektivnění využití cache webových prohlížečů.

4.9 Statistiky dat

Součástí přípravy dat bylo také nutné zhodnotit jaká data se v databázi zobrazují, nacházet nestandardní stavy a podle toho postupovat při návrhu a rozšiřování funkcí aplikace. Mezi vybrané statistiky určené pro analytiku, které zde uvedu jsou celkové počty OZ podle zdroje v tabulce 1, počty OZ seskupené podle typu v tabulkách 2a a 2b. Dále bylo užitečné seskupení OZ podle stavu zobrazené v tabulkách 3a a 3b.

Tabulka 1: Celkové počty OZ podle zdroje [4, 5, 6]

Zdroj	Datum aktulizace	Počet OZ	Velikost příloh
ÚPV	13. květen 2020	266 538	2 198 M
EUIPO	12. květen 2020	1 810 566	65 356 M

Tabulka 2: Počty OZ rozdělené podle typu u jednotlivých úřadů [4, 5, 6]

(a) ÚPV k 13. květnu 2020

Typ	Počet OZ	Obsahuje přílohu	Obsahuje text
Slovní	123 561	0,01 %	100,00 %
Kombinovaná	107 278	99,98 %	100,00 %
Slovní grafická	18 269	99,98 %	100,00 %
Obrazová	16 384	99,67 %	48,49 %
Prostorová	1 003	99,90 %	35,79 %
Tvořená barvou	31	100,00 %	0,00 %
Se vzorem	3	100,00 %	0,00 %
Hologram	2	100,00 %	100,00 %
Poziční	2	100,00 %	0,00 %
Jiná	2	100,00 %	50,00 %
Pohybová	1	100,00 %	100,00 %
Multimediální	1	100,00 %	0,00 %
Zvuková	1	100,00 %	0,00 %

(b) EUIPO k 12. květnu 2020

Typ	Počet OZ	Obsahuje přílohu	Obsahuje text
Slovní	1 042 740	0,00 %	100,00 %
Obrazová	755 310	99,86 %	90,90 %
Prostorová	9 769	99,73 %	38,94 %
Jiná	1 027	98,93 %	30,48 %
Tvořená barvou	990	100,00 %	9,39 %
Zvuková	348	75,29 %	4,31 %
Poziční	179	100,00 %	8,94 %
Pohybová	84	8,33 %	35,71 %
Se vzorem	64	100,00 %	4,69 %
Multimediální	43	0,00 %	53,49 %
Hologram	12	75,00 %	58,33 %

Tabulka 3: Počty OZ rozdělené podle stavu u jednotlivých úřadů [4, 5, 6]

(a) ÚPV k 13. květnu 2020

Stav	Počet OZ
Zapsaná	126 386
Zaniklá	97 996
Zamítnutá/Zpětvzatá	35 524
Podaná	2 726
Zveřejněná	2 448
Zrušená/Neplatná	1 458

(b) EUIPO k 12. květnu 2020

Stav	Počet OZ
Zapsaná	1 245 931
Zaniklá	280 212
Zamítnutá/Zpětvzatá	191 475
Zveřejněná	68 811
Podaná	20 444
Zrušená/Neplatná	3 693

4.10 Vize

Dalšími možnostmi pokračování vývoje je z technického hlediska rozšíření unit testů na všechny služby, zahrnutí připravených akceptačních testů do CI pipelines a také jejich automatické spouštění po nasazení a v pravidelných intervalech běh jako health check služba na produkčním serveru. Komplexní automatické testování také velmi ulehčí migraci noční aktualizace z Luigi do Airflow, migrace všech služeb z VM do nového Kubernetes clusteru a migraci z MySQL na PostgreSQL. Také by byl vhodný refactoring komparátorů, vylepšení cachování na straně backendu a přepis REST API do GraphQL.

Z hlediska rozšiřování funkcí jde např. o zahrnutí dat vlastníků a vytvoření modulu hlídání konkurence, dále komparaci uživatelem vybrané části obrazu nebo textu v hlídání OZ. Dále také rozšíření dat o světový úřad WIPO, polský a slovenský úřad průmyslového vlastnictví.

5 Použité technologie

V této kapitole zmíním všechny nejdůležitější technologie použité v aplikaci nebo při jejím vývoji. U některých také popíšu jejich nalezené klady a zápory.

5.1 Python

Python [8] je interpretovaný programovací jazyk, který před 30 lety založil Guido van Rossum. Zaměřuje se na čitelnost, používá striktní odsazování kódu namísto závorkování. Kód je dynamicky typovaný (podporuje type hints) a používá automatickou správu paměti (garbage collector). Podporuje také OOP. V praxi se používá ve verzích 2.x (oficiálně ukončena podpora v roce 2020) a dnes již častější 3.x. Až na drobné problémy s kompatibilitou (např. práce se soubory) je plně multiplatformní. Oficiálním balíčkovacím systémem je PyPI [9]. Styl kódu je standardizován v PEP 8 [10].

5.2 PyPI

Python Package Index [9], zkráceně PyPI je balíčkovací systém pro Python. Aktuálně je považovaný za oficiální Python softwarový repozitář a obsahuje přes 227 tisíc balíčků. Obsluhuje se nástrojem PIP [11] a použité balíčky zapisují do requirements.txt souboru, také se doporučuje v souboru uvádět i použité verze.

Nástoj PIP oproti Yarn [12] (JavaScript) nebo například Composer [13] (PHP) poskytuje pouze základní funkce a neumožňuje např. automatizovanou správu requirements.txt souboru nebo závislostí. Mezi nejvýznamnější použité knihovny patří:

SQLAlchemy [14] – Open-source sada nástrojů pro práci s SQL a ORM.

Alembic [15] – Sada nástrojů pro správu databázových migrací.

MySQLClient [16], PyMySQL [17] – Knihovny zajišťující komunikaci s MySQL serverem. MySQLClient, vytvořený jako fork MySQLdb1 [18], využívá API systémové knihovny libmysqlclient, která je napsaná v čistém C a díky tomu je v určitých případech i několikanásobně rychlejší. Instalace knihovny ale může být problematická, neboť např. existuje balíček default-libmysqlclient-dev pro Ubuntu 18.04LTS [19] a Debian 10 [20], ale v Alpine je dostupná pouze alternativa mariadb-connector-c-dev [21]. Druhou variantou je PyMySQL, která je napsána v čistém Pythonu, tedy nevyžaduje žádnou instalaci systémových balíčků, nicméně použití čistého Pythonu má negativním vliv na rychlost.

PyMongo [22] – Oficiální balíček pro práci s NoSQL databází MongoDB.

Boto 3 [23], Minio [24] – Boto 3 je knihovna pro práci s AWS službami, konkrétně S3. MinIO je open-source objektový storage, kompatibilní s S3, ke kterému byla vytvořena přidružená knihovna Minio. Jedná se tedy o alternativu k Boto 3 zajišťující pouze funkcionalitu

dostupnou v MinIO serveru. Obě knihovny umožňují pracovat jak s AWS S3, tak s MinIO serverem a mají podobnou deklaraci.

Flask [25] – Webový framework poskytující základní funkcionalitu webovému serveru. Vhodný pro jednoduché API nebo s pomocí šablonovacího systému Jinja2 [26] k renderování HTML. Mapování objektů, validace formulářů, nahrávání souborů je možné doinstalovat pomocí rozšíření.

Gunicorn [27] – Gunicorn je WSGI HTTP server pro Unix-like operační systémy. Pomocí správy workers procesů umožňuje zpracovávat více požadavků současně.

Luigi [28] – Knihovna vytvořená firmou Spotify, která se zabývá zpracováním komplexních pipelines a dávkových úkolů.

tqdm [29] – Rychlý a rozšiřitelný progress bar používaný při práci v CLI prostředí.

pytest [30] – Framework pro psaní automatický testů. Ideální pro jednoduché unit testy.

Faker [31] – Balíček pro generování náhodných dat, vhodný pro použití např. v automatických testech.

Memory Profiler [32] – Modul určený k monitorování spotřeby paměti. Při kombinaci s knihovnou matplotlib je možné vizualizovat spotřebu paměti v časovém průběhu do grafu.

5.3 PHP

PHP [33] je populární skriptovací jazyk určený pro vývoj webových aplikací. Mezi nejznámější frameworky patří Symfony, Laravel, a v česku Nette, vytvořený Davidem Grudlem. Styl kódu je definován v PSR [34] doporučeních.

5.4 MySQL

MySQL [35] je open-source RDBMS. Současný majitel Oracle Corporation také nabízí v placených edicích [36] rozšiřující funkce, např. replikaci a škálování v clusteru včetně podpory 24x7. Poté co byla společnost odkoupena firmou Sun Microsystems (dnes již zvaná Oracle Corporation), byl vytvořen dodnes rozvíjený fork MariaDB [37]. MySQL využívají největší softwarové společnosti provozující Facebook, Twitter, YouTube a další.

5.5 MongoDB

MongoDB [38] je dokumentově orientovaná databáze, klasifikovaná jako NoSQL. Pro práci s dokumenty využívá JSON objekty. MongoDB vyvíjí společnost MongoDB Inc. Dnes je velmi populární a používají ji společnosti Uber, Google, IBM a další.

5.6 Redis

Redis [39] je open-source in-memory úložiště datových struktur. Využívá se jako databáze, cache nebo zprostředkovatel zpráv. Podporuje uložení řetězců, hashů, listů a spoustu dalších datových struktur. Má vestavěnou replikaci, vysokou dostupnost a partitioning v clusteru.

5.7 MinIO

MinIO [40] je cloudové objektové úložiště kompatibilní s technologií Amazon S3. Implementuje server, webového klienta a množství SDK pro jazyky Go, Java, JavaScript, Python, Haskell. Využit je ve společnostech Apple, Boeing, Disney a dalších.

5.8 Git

Nejpopulárnějším nástrojem pro distribuci a verzování změn zdrojového kódu je Git [41]. Základními koncepčními prvky jsou větve, commity, merge commity a tagy. Všechna data se ukládají do repozitářů a umožňují synchronizaci s repozitářem na serveru. Veřejnými Git repozitáři jsou GitHub [42], GitLab [43] a BitBucket [44]. GitLab jako jediný umožňuje free cloud i on-premise instalaci (provoz na vlastní infrastruktuře). Pokud je v jednom repozitáři sdruženo více projektů, pak se takový repozitář nazývá monorepo. Pro práci s větvemi se doporučuje využívat GitFlow [45] branching model. Git se používá pro verzovaní kódu Linuxového kernelu a ve společnostech Facebook a Google.

5.9 GitLab CI/CD

GitLab CI/CD [46] je nástroj pro automatické navázání vývoje na testování (Continuos Integration, Continuos Delivery) a nasazení (Continuos Deployment). Konfigurace se zapisuje v YAML formátu do .gitlab-ci.yml souboru.

Pipelines – Rozdělení zpracování do jednotlivých úkolů (jobs) zařazených ve skupinách (stages), které se za sebou řetězí (pipelines). Pokud dojde k chybě, další stage již nepokračuje.

Job artifacts – Job artifacts je jeden nebo více automaticky vygenerovaných souborů při zpracování jobu. Tyto soubory se nahrají z GitLab Runner do GitLab rozhraní a jsou k dispozici po omezenou dobu ke stažení. Používají se například pro code coverage report.

Schedule pipelines – Plánované úlohy spouštěné pravidelně v definovaném čase. Využívá se často například pro nightly builds.

Pipelines for Merge Requests – Generování pipeline při vytvoření merge requestu. Umožňuje také automatické vytvoření fork repozitáře, kde dojde k úspěšnému merge commitu a pipeline se vygeneruje ze zdrojových kódů po merge requestu.

GitLab Runner – Worker pro zpracování GitLab CI/CD jobs. Podporuje současný běh více jobs. Umožňuje instalaci na operační systémy GNU/Linux, macOS a Windows.

CI services – Pokud je běh některé služby závislý např. na databázi nebo objektovém úložišti, je možné tyto služby linkovat a GitLab Runner automaticky před spuštěním definovaných úkolů spustí tyto závislosti a také je odebere po dokončení.

5.10 Sentry

Sentry [47] je služba pro správu chybových událostí. Umožňuje běh v cloudu i on-premise instalaci. Ideální pro prioritizaci, přiřazování členů týmu k událostem, automatické seskupení opakujících se událostí a snadné odkazování např. v GitLab.

5.11 Prometheus & Grafana

Prometheus [48] zpracovává metriky generované aplikacemi a umožňuje zasílat upozornění při překročení hodnot, např. diskové kapacity. Grafana [49] zajišťuje vizualizaci dat do grafů.

5.12 Docker

Docker [50] je sada služeb poskytující prostředí k zabalení a přenášení software. Jednotlivé balíčky se nazývají kontejnery a jsou izolovány pomocí virtualizace na úrovni operačního systému. Tím Docker poskytuje základ k automatizaci vývoje, testování a distribuce aplikací. Aktuálně podporuje nativně platformu Linux, ale lze jej také používat i na macOS/Windows. Tyto nenativní platformy využívají paravirtualizaci, který má ale negativní dopad na výkon běžících aplikací. Do produkčního prostředí jsou vhodné pouze distribuce Dockeru pro Linuxové OS, jako např. RHEL, SLES, Ubuntu a CentOS.

Hlavním balíčkem je Docker Engine, obsahující Docker daemon a poskytující nezbytné prostředí pro běh a správu kontejnerů. Ty se skládají z Docker kontejnerů a Docker images, rozdělených do jednotlivých vrstev. Sestavení image probíhá ze souboru nazvaném Dockerfile. Docker registry je repozitář pro ukládání Docker images. Mezi veřejné repozitáře patří např. Docker hub [51], případně je možné provozovat privátní Docker registry.

Dalším nástrojem je Docker compose pro správu multi-kontejnerových aplikací, jejichž definice se zapisuje do YAML souborů, typicky nazvaných docker-compose.yml. Manipulace s kontejnery probíhá přes docker-compose CLI rozhraní.

Docker Swarm je mód umožňují provoz a správu kontejnerů v clusteru. Ovládání probíhá pomocí docker swarm CLI utility.

5.13 VMware vSphere

VMware vSphere [52] je sada nástrojů pro virtualizaci od společnosti VMware.

Základem je VMware ESXi [53] enterprise hypervisor, poskytující nástroje pro nasazení a správu virtuální serverů. Virtualizace je poskytována přímo jako systémová služba, tedy je označována jako nativní (typ 1). Využívá BusyBox shell a vlastní kernel založený na Linuxu. Dále obsahuje webové rozhraní umožňující plnohodnotnou správu VM, jejich snapshotů a zobrazení statistik vytížení procesoru, paměti, sítě a disku. Poskytován je i ve free verzi s omezením na 8 vCPU pro každou VM.

Při integraci v clusteru s vCenter server [54] umožňuje provozovat službu vMotion, tedy migraci VM mezi jednotlivými nodes (fyzickými servery) bez nutnosti restartu VM a s výpadkem v řádu milisekund. Další funkcí je HA zajišťující automatický restart VM na jiném node v případě HW chyby fyzického serveru. Díky těmto funkcím je vhodný pro nasazení do provozu pro náročné aplikace v produkčním prostředí.

6 Závěr

Během řešení úkolů v rámci odborné praxe jsem využil především obecné teoretické znalosti získané během studia VŠB. Převážně se jednalo o algoritmizaci, složitosti algoritmů, teoretické informatiky, návrhové vzory, komunikačních technologií, SQL databází a samozřejmě programování.

Protože reálné aplikace jsou mnohem komplexnější než školní projekty, byly pro mě přínosem cenné rady od zkušenějších kolegů.

Oproti obyčejnému studiu mi praxe přinesla spoustu praktických zkušeností, mezi které patří například dělení rozsáhlých úkolů na menší dílčí části a jejich uvádění ihned do testovacího provozu. Dále tvorba harmonogramu a plánování odhadů času implementace. Velmi přínosná byla zkušenost s technologiemi, ke kterými jako student nepřijdu vůbec do styku. Také je v praxi velice důležitá práce v týmu, vzájemná komunikace, dělení úkolů a zastupitelnost.

Literatura

1. *ATLAS consulting spol. s r.o.* [online] [cit. 2020-03-28]. Dostupné z: https://atlasconsulting.cz/.
2. *Hlídač ochranných známek – Mějte své ochranné známky pod kontrolou* [online] [cit. 2020-05-13]. Dostupné z: https://hlidacoz.cz/.
3. *Novela zákona o ochranných známkách | Právní prostor* [online] [cit. 2020-05-10]. Dostupné z: https://www.pravniprostor.cz/clanky/rekodifikace/novela-zakona-o-ochrannych-znamkach.
4. *Export Opendata IPOCZ Trademark* [online] [cit. 2020-05-14]. Dostupné z: https://isdv.upv.cz/webapp/webapp.opendata.tm.
5. *Open Data - EUIPO - Europa* [online] [cit. 2020-05-14]. Dostupné z: https://euipo.europa.eu/ohimportal/en/open-data.
6. *Hlídač ochranných známek, ATLAS consulting spol. s r.o.* [online] [cit. 2020-04-14]. Dostupné z: https://hlidacoz.atlasconsulting.cz/.
7. *Storing Images in DB - Yea or Nay? - Stack Overflow* [online] [cit. 2020-05-13]. Dostupné z: https://stackoverflow.com/a/22804.
8. *Welcome to Python.org* [online] [cit. 2020-04-13]. Dostupné z: https://www.python.org/.
9. *PyPI · The Python Package Index* [online] [cit. 2020-04-13]. Dostupné z: https://pypi.org/.
10. *PEP 8 – Style Guide for Python Code | Python.org* [online] [cit. 2020-04-13]. Dostupné z: https://www.python.org/dev/peps/pep-0008/.
11. *pip - The Python Package Installer — pip 20.0.2 documentation* [online] [cit. 2020-04-13]. Dostupné z: https://pip.pypa.io/en/stable/.
12. *Home | Yarn - Package Manager* [online] [cit. 2020-04-13]. Dostupné z: https://yarnpkg.com/.
13. *Composer* [online] [cit. 2020-04-13]. Dostupné z: https://getcomposer.org/.
14. *SQLAlchemy - The Database Toolkit for Python* [online] [cit. 2020-04-13]. Dostupné z: https://www.sqlalchemy.org/.
15. *Alembic - SQLAlchemy* [online] [cit. 2020-04-13]. Dostupné z: https://alembic.sqlalchemy.org/en/latest/.
16. *PyMySQL/mysqlclient-python: MySQL database ... - GitHub* [online] [cit. 2020-04-13]. Dostupné z: https://github.com/PyMySQL/mysqlclient-python.
17. *PyMySQL/PyMySQL: Pure Python MySQL Client - GitHub* [online] [cit. 2020-04-13]. Dostupné z: https://github.com/PyMySQL/PyMySQL.

18. *farcepest/MySQLdb1: MySQL database connector for ... - GitHub* [online] [cit. 2020-04-13]. Dostupné z: https://github.com/farcepest/MySQLdb1.

19. *Ubuntu – Details of package default-libmysqlclient-dev in bionic* [online] [cit. 2020-04-13]. Dostupné z: https://packages.ubuntu.com/bionic/default-libmysqlclient-dev.

20. *Debian – Details of package default-libmysqlclient-dev in buster* [online] [cit. 2020-04-13]. Dostupné z: https://packages.debian.org/buster/default-libmysqlclient-dev.

21. *mariadb-connector-c-dev - Alpine Linux packages* [online] [cit. 2020-04-13]. Dostupné z: https://pkgs.alpinelinux.org/package/edge/main/x86/mariadb-connector-c-dev.

22. *PyMongo - MongoDB API* [online] [cit. 2020-04-13]. Dostupné z: https://api.mongodb.com/python/current/.

23. *Boto 3 Documentation — Boto 3 Docs 1.12.39 documentation* [online] [cit. 2020-04-13]. Dostupné z: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html.

24. *Python Client API Reference - MinIO* [online] [cit. 2020-04-13]. Dostupné z: https://docs.min.io/docs/python-client-api-reference.html.

25. *Welcome to Flask — Flask Documentation (1.1.x)* [online] [cit. 2020-04-13]. Dostupné z: https://flask.palletsprojects.com/en/1.1.x/.

26. *Jinja — Jinja Documentation (2.11.x)* [online] [cit. 2020-04-13]. Dostupné z: https://jinja.palletsprojects.com/en/2.11.x/.

27. *Gunicorn - Python WSGI HTTP Server for UNIX* [online] [cit. 2020-04-13]. Dostupné z: https://gunicorn.org/.

28. *spotify/luigi: Luigi is a Python module that helps you ... - GitHub* [online] [cit. 2020-04-13]. Dostupné z: https://github.com/spotify/luigi.

29. *tqdm/tqdm: A Fast, Extensible Progress Bar for ... - GitHub* [online] [cit. 2020-04-13]. Dostupné z: https://github.com/tqdm/tqdm.

30. *pytest: helps you write better programs — pytest documentation* [online] [cit. 2020-04-13]. Dostupné z: https://docs.pytest.org/en/latest/.

31. *Welcome to Faker's documentation! — Faker 4.0.2 ...* [online] [cit. 2020-04-13]. Dostupné z: https://faker.readthedocs.io/en/master/.

32. *pythonprofilers/memory_profiler: Monitor Memory ... - GitHub* [online] [cit. 2020-04-13]. Dostupné z: https://github.com/pythonprofilers/memory_profiler.

33. *PHP.net* [online] [cit. 2020-04-13]. Dostupné z: https://www.php.net/.

34. *PHP Standards Recommendations - PHP-FIG* [online] [cit. 2020-04-13]. Dostupné z: https://www.php-fig.org/psr/.

35. *MySQL* [online] [cit. 2020-04-13]. Dostupné z: https://www.mysql.com/.

36. *MySQL Editions - MySQL* [online] [cit. 2020-04-13]. Dostupné z: https://www.mysql.com/products/.

37. *MariaDB Foundation - MariaDB.org* [online] [cit. 2020-04-13]. Dostupné z: https://mariadb.org/.

38. *MongoDB: The most popular database for modern apps* [online] [cit. 2020-04-13]. Dostupné z: https://www.mongodb.com/.

39. *Redis* [online] [cit. 2020-05-15]. Dostupné z: https://redis.io/.

40. *MinIO | High Performance, Kubernetes-Friendly Object Storage* [online] [cit. 2020-04-13]. Dostupné z: https://min.io/.

41. *Git SCM* [online] [cit. 2020-04-13]. Dostupné z: https://git-scm.com/.

42. *The world's leading software development platform · GitHub* [online] [cit. 2020-04-13]. Dostupné z: https://github.com/.

43. *GitLab* [online] [cit. 2020-04-13]. Dostupné z: https://about.gitlab.com/.

44. *Bitbucket | The Git solution for professional teams* [online] [cit. 2020-04-13]. Dostupné z: https://bitbucket.org/product.

45. *Introducing GitFlow* [online] [cit. 2020-04-13]. Dostupné z: https://datasift.github.io/gitflow/IntroducingGitFlow.html.

46. *GitLab CI/CD - GitLab Documentation* [online] [cit. 2020-04-13]. Dostupné z: https://docs.gitlab.com/ee/ci/.

47. *Sentry: Application Monitoring and Error Tracking Software* [online] [cit. 2020-04-13]. Dostupné z: https://sentry.io/welcome/.

48. *Prometheus - Monitoring system & time series database* [online] [cit. 2020-04-13]. Dostupné z: https://prometheus.io/.

49. *Grafana: The open observability platform | Grafana Labs* [online] [cit. 2020-04-13]. Dostupné z: https://grafana.com/.

50. *Docker: Empowering App Development for Developers* [online] [cit. 2020-04-13]. Dostupné z: https://www.docker.com/.

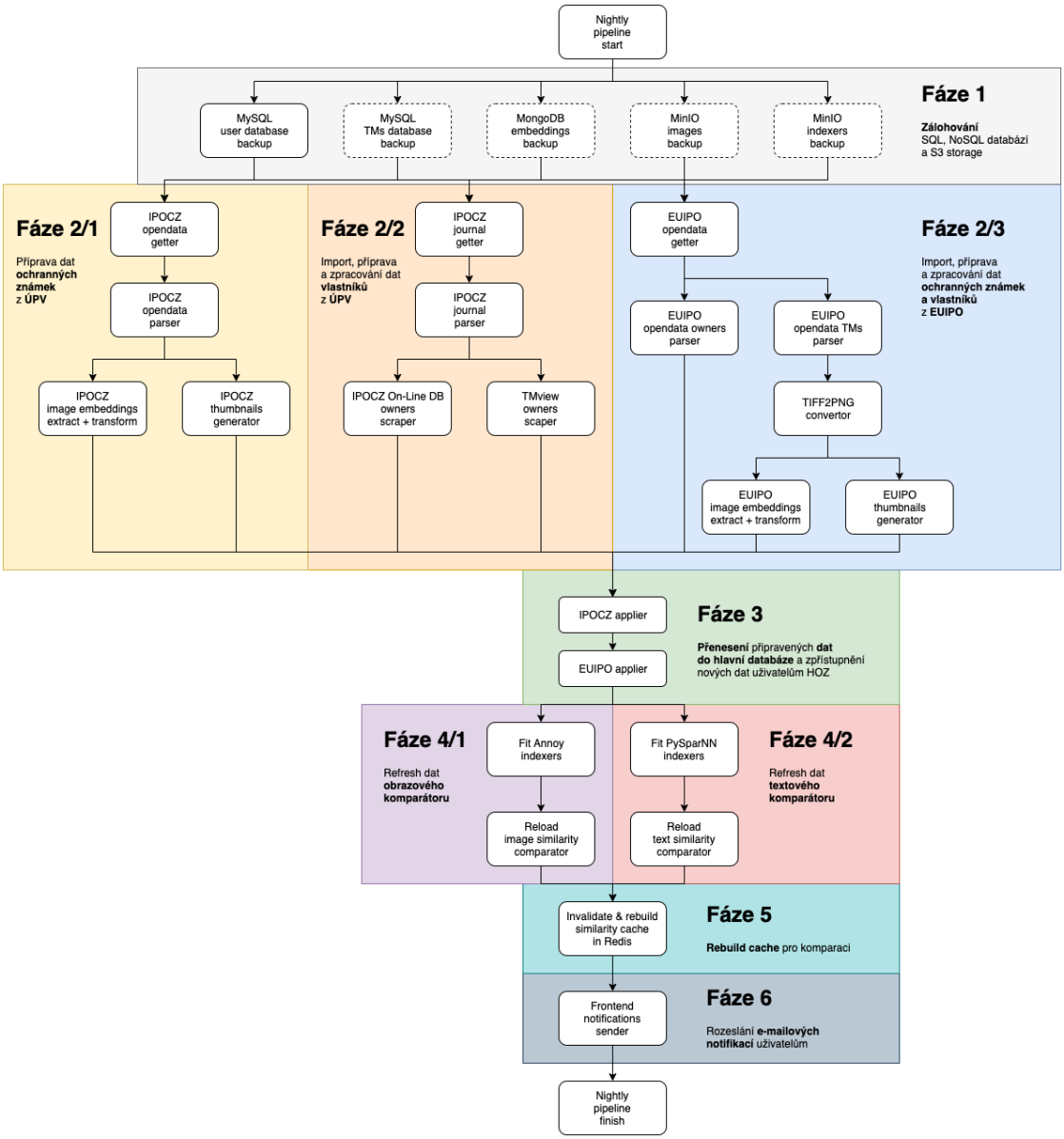
51. *Docker Hub* [online] [cit. 2020-04-13]. Dostupné z: https://hub.docker.com/.

52. *What is vSphere? | Server Virtualization Software | VMware | CZ* [online] [cit. 2020-04-13]. Dostupné z: https://www.vmware.com/cz/products/vsphere.html.

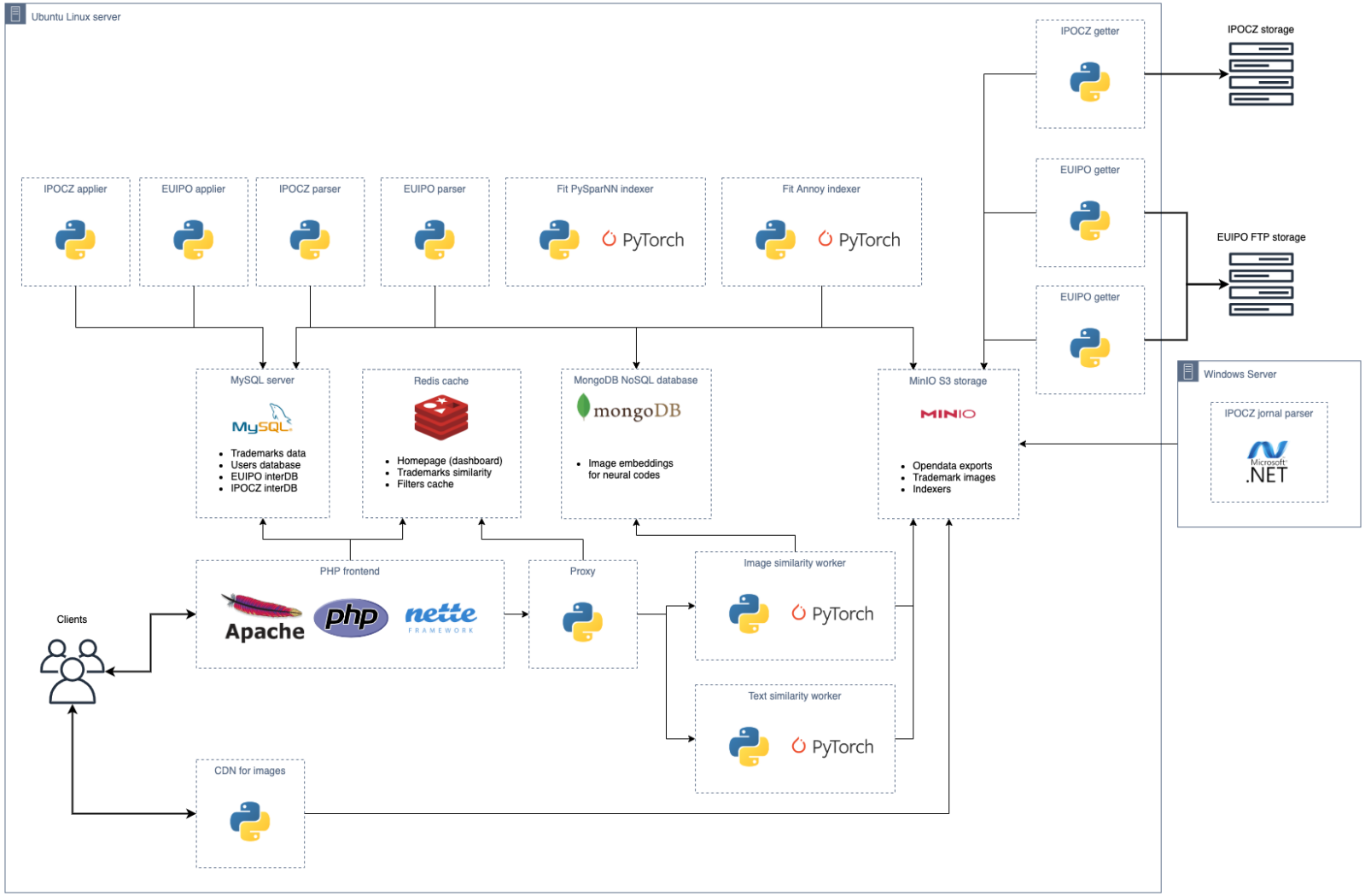
53. *ESXi | Bare Metal Hypervisor | VMware | CZ* [online] [cit. 2020-04-13]. Dostupné z: https://www.vmware.com/cz/products/esxi-and-esx.html.

54. *Server Management Software - vCenter Server | VMware | CZ* [online] [cit. 2020-04-13]. Dostupné z: https://www.vmware.com/cz/products/vcenter-server.html.

A Architektura a procesy



Obrázek 8: Proces denní aktualizace dat



Obrázek 9: Schéma architektury aplikace

B Kontejnerizace

```
FROM python:3.8-alpine

RUN apk update && apk add --no-cache gcc mariadb-connector-c-dev musl-dev
    imagemagick libmagic

WORKDIR /app

COPY core/requirements.txt /app/core/
COPY ipocz_parser/requirements.txt /app/ipocz_parser/

RUN pip install --no-cache-dir -r core/requirements.txt -r ipocz_parser/
    requirements.txt

COPY core /app/core
COPY ipocz_parser /app/ipocz_parser

CMD ["python", "-m", "ipocz_parser.app"]
```

Výpis 3: Příklad sestavení kontejneru pro službu IPOCZ parser