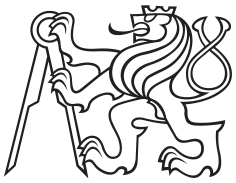
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídící techniky

Databázová podpora sbírek

Tomáš Neumaier

Školitel: Doc. Ing. Ivan Jelínek, CSc.
Leden 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Neumaier** Jméno: **Tomáš** Osobní číslo: **420033**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Systémy a řízení**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Databázová podpora sbírek

Název bakalářské práce anglicky:

Database support for collections

Pokyny pro vypracování:

1. Proveďte rešerši současných aplikací databázové podpory katagolizace sbírek
2. Navrhněte výchozí strukturu databáze modulární aplikace
3. Navrhněte a implementujte další kolekce, které vyžadují odlišný přístup v katalogizování
4. Navrněte systém správy kolekcí pro moderátory kolekcí
5. Navrhněte uživatelský interface a systém ovládání mobilní aplikací, speciálně pro tablety, pro standardního uživatele
6. Do mobilní aplikace implementujte externí systém pro obrazového rozpoznání položek kolekcí
7. Navrhněte vhodný způsob testování a aplikaci otestujte, výsledky vyhodnoťte

Seznam doporučené literatury:

[1] Ajzele, Branko. Mastering PHP 7. 1. vydání. Pack Publishing, 2017. ISBN 9781785882814
[2] Rahman, Mizanur. PHP 7 Data Structures and Algorithms. 1. vydání. Pack Publishing 2017. ISBN 9781786463890
[3] Welling, Luke and Thompson, Laura. PHP and MySQL Web Development. 5. vydání. Addison-Wesley Professional, 2016. ISBN 0321833899
[4] Phillips, Bill and Hardy, Brian. Android Programming: The Big Nerd Ranch Guide. 3. vydání. Big Nerd Ranch Guides, 2017. ISBN 0134706056
[5] Meier, Reto. Professional Android. 4. vydání. Wrox, 2018. ISBN 1118949528 (4. vydání vychází v lednu)

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Ivan Jelínek, CSc., kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **16.01.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

doc. Ing. Ivan Jelínek, CSc. podpis vedoucí(ho) práce	prof. Ing. Michael Šebek, DrSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

Chtěl bych poděkovat svému dědovi, Františkovi Hrbáčkovi, který umožnil nafocení předmětů z jeho sbírky, kde vracení do původního stavu mu zabralo mnoho hodin. Chtěl bych poděkovat všem, kteří se více či méně podílely na testování aplikace a svými postřehy umožnily její vylepšování. Chtěl bych poděkovat své přítelkyni, Zuzaně Štenclové, za pomoc při plnění obsahu. A chtěl bych také poděkovat vedoucímu práce, doc. Ing. Ivanovi Jelínkovi, CSc., za trpělivost při vedení 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.

Praha, leden 2019

Abstrakt

Bakalářská práce se zabývá způsobem zpracování a katalogizace sbírek menšího či středně velkého charakteru. Hlavní správa kolekce je vedena v modulární webové aplikaci, kde každý katalog má svoji vlastní stránku, vyvíjené v PHP s použitím frameworku Nette a databáze MySQL. Webovou aplikaci doplňuje mobilní aplikace pro rychlé vyhledávání mimo domov, zejména v situacích jako jsou návštěvy burz či trhů, kde možnost nahlédnout do své sbírky může být klíčová ...

Klíčová slova: Nette, PHP, MySQL, Java, Android, OpenCV

Školitel: Doc. Ing. Ivan Jelínek, CSc.

Abstract

This Bachelor thesis deals with way of processing and cataloging collections of smaller or moderate character. User's management of collection is located in modular web application, where each catalog has its own webpage, developed in PHP with use of Nette framework and MySQL database. Web application is complemented by mobile application used for quick search outside home, especially in situations such as auctions or markets where the possibility of previewing owned collection might be crucial ...

Keywords: Nette, PHP, MySQL, Java, Android, OpenCV

Title translation: Database support for collections

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

1.1 Úvod

"Sběratelství je koníček – hobby, který spočívá v získávání specifického druhu předmětů (popřípadě i zážitků s nimi spojených), založeném na specifickém zájmu sběratele a jejich uchovávání ve sbírce. Tyto sbírky bývají většinou kvalitně roztříděné, zkatalogizované a atraktivně vystavené."[1]

Úvodní odstavec o sběratelství ze stránky Wikipedie obsahuje jednu nepřesnost. I když sběratelé své předměty třídí a katalogizují, je jenom málo sběratelských oborů, které by se mohly pyšnit ucelenými sběratelskými katalogy. Většina práce připadá na samotného sběratele a případná snaha o hlubší proniknutí do oboru je spíše prací archiváře. Výjimku představují pouze největší sběratelské školy, především numismatika a filatelie, které mají své propracované online katalogy. StampWorld, katalog známek, má přes 300 tisíc uživatelů, Numista, katalog mincí, má aktuálně přes 100 tisíc uživatelů. Australská pošta však odhaduje, že je na světě přes 20 milionů sběratelů známek. [2]

Ve sběratelství známek a minci se točí velké množství peněz. Nejdražší známka na světě, 1 centová známka z Britské Guyany z roku 1856, se vydražila v roce 2014 za 9.5 milionů dolarů [3]. Nejdražší mince, 1 americký dollar z roku 1794, se v roce 2013 vydražil za 10 milionů dolarů. [4]. Většina sběratelských oborů se však nachází v jiných cenových relací. V rámci této práce je použita sbírka sokolských odznaků a medailí. Nejdražší vydražená sokolská medaile z roku 1924 se vydražila za 75 tisíc korun [5], to je zhruba průměrná měsíční mzda programátora. Koncovou cenu vždy tvoří nabídka a poptávka. U menších sběratelských oborů je zkrátka poptávka nižší a i když i nabídka může být značně omezená, menší množství sběratelů vždy znamená menší množství peněz v daném oboru. Čistě z komerčního hlediska se tak nevyplatí tvořit katalogy pro menší sběratelské obory, protože vstupní náklady se jen těžko rozprostřou mezi omezené množství sběratelů.

Tato práce však cílí přesně na tyto typy sbírek, které nejsou dostatečně populární, přesto jsou zajímavé. Chce napomoci sběratelům s orientací v problematice a umožnit snadnější vstup nových sběratelů do oboru. Díky jednotné modulární platformě pro kolekce se sníží množství nutných zásahů

programátora a na sběratelích tak zůstane pouze samotné vytváření katalogu - pokud jej již ovšem nevytvořil jiný sběratel před nimi.

1.2 Podobné aplikace

1.2.1 StampWorld.com

Zřejmě největší online katalog známek na světě a možná i největší sběratelský katalog vůbec. K lednu 2019 má přes 300 tisíc uživatelů a v databázi přes 650 tisíc různých typů známek [6]. V katalogu je možné si u známky vyhledat téměř všechny možné informace včetně ceny s prokliky na prodejce. Katalog je členěný na současné státy s možností výběru zaniklých států, které se přes území současných rozprostíraly.

StampWorld umožňuje i vytváření vlastního katalogu a tím evidenci vlastněních známek. Je však vidět, že na této funkci aplikace nestojí. Z hlavního katalogu si není možné přidat známku do sbírky a je nutné použít odkazu „moje známky“ z uživatelského profilu. Tam je možné si vytvářet různé katalogy nejen s rozdílnými známkami, ale i funkcemi typu „chtěl bych“. Při editaci katalogu se načtou všechny známky, pro výběr konkrétních je nutné použít vyhledávání. Vzhledem k výpisu jedné známky na řádek a celkem devatenácti řádků na stránku bez možnosti změny zobrazení pak, alespoň z mého pohledu, trvá tvorba vlastního katalogu dlouho. Neumožňuje navíc přidávat počty vlastněných předmětů a dokonce ani vlastní poznámku. Pomyslným vrcholem je, že katalog stránky nepropisuje informace z vlastních katalogů, což je z mého pohledu největší chybou.

StampWorld nemá mobilní aplikaci a ani internetová stránka není pro mobilní zařízení optimalizovaná.

1.2.2 Numista.com

Numista je katalogem mincí, aktuálně obsahuje přes 130 tisíc položek. Má přehledný katalog členěný podle států, kde stejné mince různých ročníků jsou seskupeny do sebe. Umožňuje i propracované vyhledávání podle libovolných parametrů.

Na rozdíl od StampWorld má propracovaný uživatelský katalog. Katalogu předchází interaktivní mapa s rozmístěním vlastněních mincí do jednotlivých států. Rovněž v rámci globálního katalogu je vidět, které mince člověk vlastní. Jediným problémem je, že se mince ukazuje jako vlastněná, i když je vlastněn pouze jeden ročník. Celkově však působí katalog dobrým a přehledným dojmem, který ještě umocňuje možnost si nechat položky vyexportovat.

Numista nemá mobilní aplikaci, nicméně internetová stránka je alespoň responsivní pro použití v mobilních telefonech. Funkcionalita je alespoň částečně přizpůsobena mobilním telefonům, i když je zřejmé, že se jedná pouze o responsivní variantu desktopové verze.

1.2.3 Guldiner Light

> *"Filozofie programu Guldiner Light je, aby program nekomunikoval s internetem (což vyžadují skuteční sběratelé, kteří si cení své sbírky) a tím zabezpečil bezpečnost dat, které jsou do programu vloženy. Část zákazníků dokonce pro správu sbírky používá počítač, který vůbec nepřipojují k internetu."*[7]

Česká komerční aplikace pro Windows od studia Pteranodon Soft. Jak je z citace zřejmé, jedná se čistě o offline aplikaci. Používá šablony na předměty pomocí kterých se nastavují pole. Tyto šablony umožňuje i editovat. V případě, že by uživatel chtěl, může si data překopírovat mezi počítači. Aplikace mi osobně nepřijde uživatelsky přívětivá, zřejmě se však na ni dá zvyknout. Hlavní nevýhodou vyplývající z podstaty programu je nutnost si katalog tvořit sám. Byť autor umožňuje ze svých stránek stáhnout katalogy československých mincí, jedná se opět o úzký segment numismatiky, pro kterou se dá využít i Numista zmíněný výše. Sběratel je tak opět odkázán sám na sebe a je otázkou, zda existují nějací sběratelé, kteří by mezi sebou sdíleli data vytvořená v tomto programu. Další nevýhodou, i když zde funkcí, je pouhá dostupnost offline a tím nedostupnost katalogu z libovolného místa.

1.2.4 Papír

Aby můj děda měl přehled o vlastněných odznacích, nosil sebou na burzu katalogy. Ty vycházejí ze dvou publikací "Dějiny československé tělovýchovy v odznacích", konkrétně z 5. a 6. dílu. Původní publikace se mi již nepodařilo dohledat a musím vycházet pouze z informací uvedených na okopírované obálce. Ta udává, že publikace vyšly v devadesátých letech 20. století v pouhém nákladu padesáti kusů. Aby si dokázal udržet přehled, musel si vyznačovat vlastněné odznaky a rozšiřovat katalogy o odznaky, které se mu podařilo získat, ale v katalogu se o nich nenacházela zmíňka. Katalog je k dnešnímu dni soubor čtyř šanonů o celkové váze čtyři kilogramy. Horší je jejich neskladnost, do obyčejného batohu se nevejdou. A navíc, i když katalogy rozšířil o velké množství informací, tyto informace jsou pouze na papíru a není jak je předat dále. Každý další sběratel tak musí jít po stejných stopách, znovu a znovu od počátku. V těchto svazcích, v potřebě přesunout i sběratelství historických předmětů do nového tisíciletí, leží počátky myšlenky této práce.

Z uvedených katalogů má zřejmě nejblíže optimálnímu stavu Numista. Jednoduchý, přímý přístup, který uživatele nezatěžuje, s předpřipraveným katalogem. Chybějící mobilní aplikaci slušně zastupuje responsivně vytvořená webová prezentace. Je zřejmé, že spousta funkcionality je dělána na míru mincím a tato funkcionalita se s nutností universálnosti modulární aplikace tolik neslučuje, přesto ovládání se zdá být vhodnou inspirací.

Kapitola 2
Implementace

2.1 Použité technologie

2.1.1 PHP

Hypertext preprocessor, mnohem známější pod zkratkou PHP, je nejrozšířenější[8] jazyk pro tvorbu internetových stránek, který vychází z programovacích jazyků C a Java. Vznikl už v roce 1995, poslední major verze s číslem sedm je z roku 2015. Aktuální minor verze 7.3 vyšla 6. prosince 2018. Značnému rozšíření pomohlo i poměrně jednoduchá a benevolentní pravidla pro psaní kódu. Aplikace v PHP lze psát procedurálně, objektově, kombinovaně či tzv. špagetově (aplikace je jeden soubor s posloupností if-else stuktur), jazyk je dynamicky typovaný, PHP dokonce samo provádí datové konverze, kde je mu to umožněno. Striktní chování přibylo až od poslední major verze. Práce je psána na verzi 7.2.

Benevolentnost zápisu kódu pak vedla k velmi značným rozdílům v kvalitě zápisu kódu mezi začínajícími programátor a mezi profesionály. Snaha ke zlepšení kvality kódu především open source knihoven, aplikací a jiných nástrojů vedla ke vzniku organizace PHP Standards Group. Jejich hlavním cílem bylo unifikování postupů nejen pro psaní kódu, ale i pro obecné práce s knihovnami tak, aby se knihovny třetích stran mohly bez větších úprav nasadit na různé projekty [21]. PHP Standards Group v roce 2013 vydala dvě PSR (PHP Standard Recommendation) s čísly 1 a 2, které se zabývají způsobem zápisu zdrojového kódu. Tato práce se řídí novější PSR – 2, která sama doplňuje PSR – 1. [22]

2.1.2 Nette

Nette je český open source framework pro tvorbu webových aplikací psaných v PHP. Nette patří mezi nejpoužívanější frameworky v České republice[9]. Zahraničnímu přesahu brání především pomalejší překlad anglické dokumentace, byť se v tomto směru mnohé zlepšilo. Nevýhodou může být pro někoho taky absence odborné literatury. To je však dle mého názoru vynahrazeno kvalitní českou dokumentací. Práce využívá Nette ve verzi 2.4, která je přechodovou verzí mezi 2.3 a plánovanou major verzí 3[10].

2.1.3 Doctrine 2

Doctrine je ORM (Object-Relation Mapping) framework pro práci s databázovou vrstvou. Jeho hlavním úkolem je umožnit práci s daty jako s objekty. Jedna databázová tabulka pak zpravidla odpovídá jedné třídě, jednotlivé sloupce tabulky jsou pak vlastnosti třídy. Výhodou Doctrine je, že skrývá před aplikací použitou databázovou aplikaci díky použití vlastního jazyka DQL vycházejícího z jazyka SQL, který si knihovna překládá do jazyka dle použitého typu připojení. Díky tomu je možné aplikaci psanou s použitím Doctrine přepojit na jinou databázovou aplikaci bez nutnosti úprav v dotazech. V práci je použita knihovna Kdyby/Doctrine, která integruje Doctrine 2 do frameworku Nette. Doctrine 2 v práci nahrazuje databázovou knihovnu Dibi z frameworku Nette.

2.1.4 Composer

Composer je nástroj pro správu knihoven a jiných závislostí projektu psaných v PHP. Jeho úkolem je umožňovat snadnou instalaci závislostí a jejich aktulizaci. Záležitost aktulizace závislostí je díky composeru otázkou jednoho příkazu v příkazové řádce.

2.1.5 MySQL

MySQL je systém pro správu relačních databázi. Vychází z jazyka SQL a dále jej rozšiřuje. Práce využívá MySQL ve verzi 5.6.

2.1.6 Java

Java je objektově orientovaný jazyk vycházející z jazyka C++. K prosinci 2018 se jedná o nejpopulárnější programovací jazyk. [12] V projektu je nativní Java používána pro scripty pracující s knihovnou OpenCV. Verze pro Android je používaná v mobilní aplikaci.

2.1.7 OpenCV

OpenCV je multiplatformní open source knihovna pro strojové vidění. Je k dispozici pro Windows, Linux, MacOS, iOS a Android. Má rozhraní pro C++, Python a Javu. Obsahuje téměř 60 modulů. Práce využívá knihovnu OpenCV ve verzi 3.4 a zejména modul features2d. Pro potřeby práce musela být zkompilovaná s přepínačem OPENCV_ENABLE_NONFREE pro povolení detektoru SURF, který je patentován, ale pro nekomerční účely lze použít.

2.1.8 Další knihovny a rozšíření

Níže ve stručnosti uvádím další knihovny a rozšíření, které jsou v práci použity.

jQuery

Javascriptová knihovna pro lepší práci s javascriptem

JSTree

Javascriptová knihovna používaná pro efektivnější správu stromu kategorií

SweetAlert2

Javascriptová knihovna pro hezčí dialogová okna

Bootstrap

Bootstrap je front-end framework obsahující podpůrné nástroje pro tvorbu stránek. Ačkoli obsahuje i javascriptové knihovny, v rámci práce jsou používány především CSS prvky, a to zejména stylizace formulářů a responsivní rozmístění prvků.

Nette.ajax.js

Javascriptová knihovna pro lepší práci s Ajaxem při frameworku Nette.

Kdyby/Translation

PHP knihovna určená pro překlady textů. I když je práce psaná pouze v češtině, knihovna je využívána zejména pro překlady konstant při výpisech textů.

Imagick

Knihovna a soubor aplikací pro zpracování obrazu. V rámci balíčku PECL (PHP Extension Community Library), což jsou stáhnutelná rozšíření do PHP, je k dispozici interface pro komunikaci s knihovnou z prostředí PHP.

Picasso

Android knihovna pro zjednodušení práce s obrázky.

2.2 Databázová struktura aplikace

2.2.1 Entity

K pochopení databázové vrstvy je nedříve nutné vydefinovat základní databázové objekty, které může modulární aplikace využívat.

Modul

Modul je objekt, který ohraničuje výčet používaných dat. Jeho hlavním úkolem je identifikace sebe a uchovávání informací o seznamu přiřazených kategorií. Jelikož může být žádoucí, aby se např. kategorie nacházela ve více modulech (lze si představit, že československé známky by se nacházely v modulu věnovaném československé historii a zároveň v modulu věnovaném známkám obecně), nelze podřazené entity přímo přiřadit modulu. Modul by tak měl být nezávislý na svém obsahu.

Navázaná aplikace by měla mít základní modul, ze kterého budou ostatní moduly dědit. To umožní vytvořit sadu universálních metod, díky čemuž zakládání nových modulů může být opravdu otázkou krátké chvíle. V nejextrémnější variantě, ke které směřuje i tato práce, pak modul může existovat pouze na databázové úrovni, přičemž pro zobrazování se využijí standartní universální metody.

Kategorie

Úkol kategorie je členit předměty na menší, lépe uchopitelné skupiny. Kategorie se mohou různě řetězit. Pro lepší přenášení dat mezi moduly je vhodné umožnit sdílení celé kategorie.

Šablona

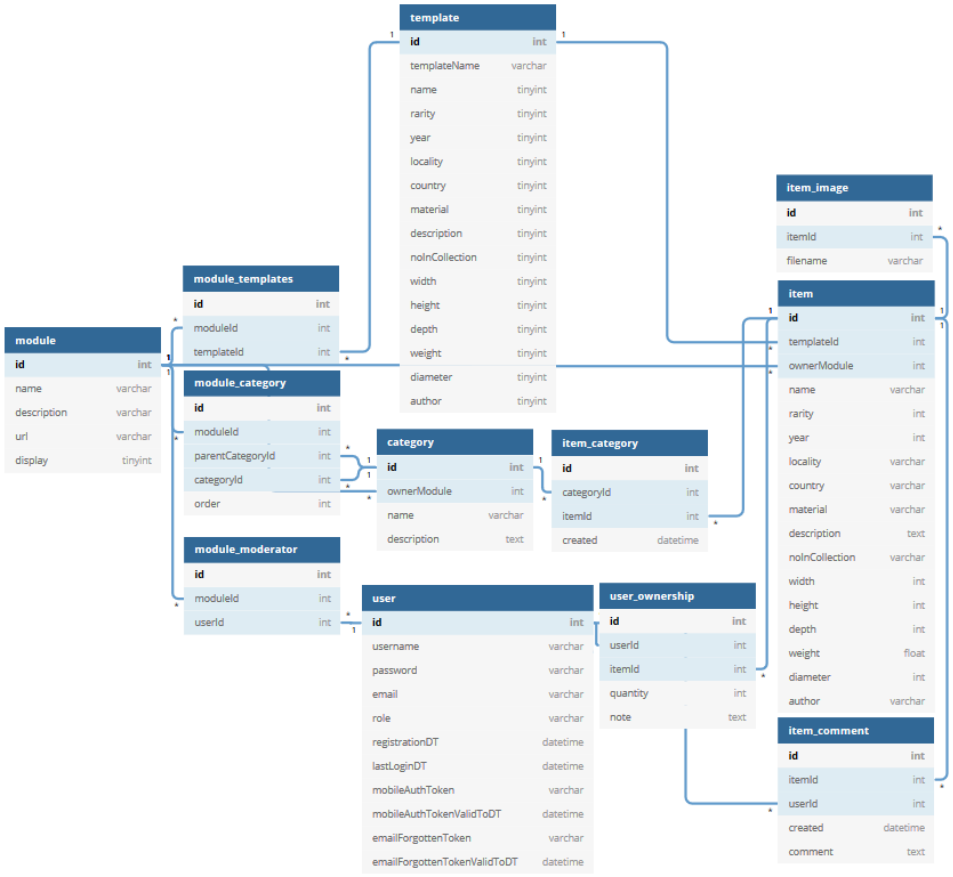
Aby bylo možné definovat předmět a jeho vlastnosti a ty pak dynamicky vypisovat, je nutné vytvořit šablonu předmětu. Předáváním šablon se aplikace může rozhodnout, jakým způsobem danou šablonu vypíše a zároveň umožňuje sdružit stejné parametry napříč různými typy předmětů. Příkladem by mohla být situace, kdy by uživatel chtěl sledovat předměty z drahých kovů bez ohledu na jejich typ.

Předmět

Úkolem předmětu je uchovávat informace o předmětu sbírkové povahy. U mince chceme například znát její hodnotu, rok či případně materiál. Jiné informace budeme chtít vědět o známce, kde jiný materiál než papír neočekáváme. Je zřejmé, že musí existovat vazba na šablonu. Rovněž se dá předpokládat existence podřazených objektů, jako je například množina přiřazených obrázků.

Uživatel

Úloha uživatele je držet osobní údaje uživatelů. Kromě obligátních údajů jako jsou email či heslo, je to také informace o přiřazených tokenech (emailový pro obnovu hesla, mobilní pro ověření totožnosti komunikujícího uživatele přes API) či o vlastnictví předmětů. Možnosti označit vlastnictví a naopak zobrazit si pouze chybějící předměty, možnosti vyhledat si předmět a dostat



Obrázek 2.1: Databázové schéma

informaci o tom, zda-li jej vlastním či nikoli, jsou nejdůležitější vlastnosti katalogu.

2.2.2 Implementace databázové vrstvy

Na obrázku 2.1 je zobrazeno schéma databáze. Jedná se o kořenový strom, kde kořenem je modul se svojí tabulkou module. V modulu jsou definovány použité šablony použitím pomocné tabulky module_templates. Účelem je pouhé usnadnění výběru šablon při vytváření předmětu, aby moderátor nebyl zatěžován šablonami, které ve svém modulu nevyužije. Seznamu moderátorů je určen provazbou přes tabulku module_moderator. Rozdílem oproti atributu user.role je v tom, že atribut role definuje globální chování, kdežto module_moderator definuje roli moderátor pouze v rámci konkrétního modulu. Pokud by měl uživatel nastavené user.role na hodnotu moderátor, stal by se moderátorem ve všech modulech. Seznam použitých kategorí v modulu pak určuje tabulka module_category. V ní se navíc definuje rodičovská kategorie (opět pouze v rámci modulu) a pořadí. O správné řízení těchto atributů, zejména pořadí, se stará webová aplikace.

Uživatele jsou uloženi v tabulce user. Uživatel je sdílený napříč moduly a

účelně neexistuje žádná provazba na modul. Kromě základních parametrů jako jsou přihlašovací jméno, heslo, email a další obsahuje tabulka atributy mobileAuthToken, mobileAuthTokenValitToDt, emailForgottenToken a emailForggotenTokenValitToDT. První dva jmenovaný slouží API. MobileAuthToken je token, který získá uživatel po přihlášení se přes funkci API login. Druhý atribut je platnost, do kdy je považován token za platný. Webová aplikace, jejíž součástí je API rozhrání, při každém validním dotazu s použitím autorizačního tokenu posune platnost tokenu o další dva dny. Email tokeny jsou obdobou mobilních, slouží však pro funkci obnovení zpomenutého hesla. Na uživatele je navázáno vlastnictví předmětů pomocí tabulky user_ownership. Ta kromě vazby na uživatele a předmět obsahuje ještě atributy quantity a note. Quantity je počet vlastněných předmětů, note je nepovinná uživatelská poznámka. Záznamů, které shodně ukazují na stejného uživatele i stejný předmět, může být více.

Předměty jsou uchovávány v tabulce item. Při ideálním používání aplikace ze strany moderátorů by každý předmět měl pouze jeden záznam (tedy by napří moduly nevznikaly duplikáty). V takovém případě se všechny informace předmětu propíšou do všech modulů, kam jsou předměty přiřazeny, včetně například záznamů o vlastnictví. Uživatel by tak při navštívení nového modulu viděl vlastnictví vyplněná v ostatních modulech. Na předměty navazují obrázky v tabulce item_image a komentáře v tabulce item_comment.

2.3 Webové aplikace

Webová aplikace je jádrem celého systému. Veškeré logické výpočty a nápočty jsou řízeny z ní. Počínaje samotnou logikou pro webovou aplikaci, přes přípravu dat pro mobilní aplikaci až po přípravu klasifikátorů pro knihovnu OpenCV, vše je řízeno z webové aplikace.

2.3.1 Základní členění

Jelikož je webová aplikace nejrozsáhlejší, v úvodu zmíním základní strukturu webové aplikace. Níže uvedené odstavce pomohou lépe pochopit sekce následující.

V root adresáři projektu se nachází několik složek. Temp je systémová složka pro dočasně vygenerované soubory. Log je adresář pro logovací zprávy. Důležité jsou zejména chybové hlášky, které se sem ukládají pro pozdější zpracování programátorem. Adresář opencv obsahuje předpřipravené obrázky pro funkci rozpoznávání obrazu. Adresář vendor obsahuje použité PHP knihovny třetích stran. V adresáři www se nachází vše, k čemu má přístup uživatel, tedy například javascriptové scripty či styly v CSS. Na tento adresář je také směřován server. Samotná vlastní aplikace se však nachází v adresáři app.

Složka se dělí na dvě základní, Core a Modules. V Core se nachází všechny základní třídy, zejména ty pro práci s databází. Ve složce Modules se pak nachází všechny moduly. Výchozí modul je Base, který obsahuje základní nástroje pro ostatní moduly. Je psán takovým způsobem, že sám o sobě

je dostatečným pro vypisování jakékoli kolekce. Pokud tedy nejsou žádné speciální požadavky na způsob výpisu kolekce, nemusí se ani vytvářet žádné soubory pro daný modul. Tento případ ukazuje modul PlayGround, jehož definice je pouze v databázi. Dalšími složkami a tedy moduly jsou Sokol a MagicGathering. Jedná se o moduly kolekcí, přičemž druhý jmenovaný rozšiřuje Base modul větší měrou. Poslední jsou moduly API a Cron, které mají své specifické úkoly a jsou neveřejné, nicméně vzhledem k logice aplikace musí být i tyto ne-sbírkové moduly umístěny v sekci Modules. Není však pro ně nezbytné, aby dědily od modulu Base.

Nette, stejně jako Android, je založené na architektuře MVC, neboli Model-View-Controller. Základní myšlenkou je oddělení aplikační logiky (model), renderovací části (view) a části pro komunikaci s uživatelem (controller). Controllery zde zastupují Presentery, které se nachází v Modules. View zastupují šablony, které jsou vždy u komponent, ke kterým patří. Ať to jsou presentery nebo například formuláře umístěné samostatně v Core\Components\Forms. Model samotný je pak dále členěn.

Databázová vrstva je dělena na několik úrovní. Tou nejnižší jsou entity v Core\Entity. Ty představují samotné databázové tabulky. Nad každou entitou stojí právě jeden repositář. Úkolem repositáře je sestavovat samotné databázové dotazy. Tuto funkci nemá žádná jiná vrstva, žádná jiná třída DQL dotazy nesestavuje. Všechny repositáře se nacházejí v Core\Repository. Nad repositářem může být fasáda, neboli třída Facade. Tyto třídy mají za úkol skrýt podobné repositáře před vyšší vrstvou. Typická je řada Item (Item, ItemCategory, Comment, Image), kde celá tato skupina je zastoupena třídou ItemFacade. Ta pak rozhoduje, která volání delegovat kterému repositáři. Nejvyšší vrstvou jsou služby, neboli services. Tyto služby představují ucelený funkční blok. Ve službách je prováděna logika, zde dochází ke zpracování uživatelských vstupů z jedné strany a výstupu z databáze ze strany druhé. Příkladem je CollectionService, která zaštiťuje metody pro práci se sbírkou.

2.3.2 Inicializace

Při návštěvě uživatele je uživatel, bez ohledu na použitou subdoménu, veden do adresáře www, který nedělá nic jiného, než načte a spustí bootstrap v adresáři app. Boostrap slouží k autoloadingu externích php knihoven a k inicializaci samotné aplikace. Nette ve svém výchozím chování indexuje všechny třídy použité v projektu. Vzhledem k modulárnímu systému však toto není žádoucí a loading je upraven tak, že se načítá vždy pouze vybraný modul a Base modul. Největší výhodou této úpravy je, že moduly jsou na sobě plně nezávislé - v případě, že jeden modul způsobuje pád aplikace, ostatní běží nedotčeny. Hlavním důvodem plného oddělení modulů však byly problémy s Dependency Injection. Při požadování služby pouze na základě informace, že implementuje určitý interface či dědí z určitě abstraktní třídy, by mohla existovat pouze jedna taková služba napříč celou aplikací. Pokud by existovaly dvě služby implementující například interface I, DI by se nemohlo rozhodnout, kterou službu načíst a aplikace by spadla. Oddělením modulů je tento problém omezen pouze na konkrétní modul.

V bootstrapu se dále připravuje třída Settings, která je typu Singleton. Jejím úkolem je držet informace o zvoleném modulu napříč aplikací. Využití je především v repositářích, které díky zvolenému postupu nemusí od vyšších vrstev vyžadovat informaci o zvoleném modulu. Stejně tak jsou vyšší vrstvy nezávislé na zvoleném modulu. Například služba CategoryService zodpovědná za práci s kategoriemi a předměty vůbec nezjišťuje, v jakém modulu se aktuálně nachází. Jedinou výjimkou jsou speciální moduly API a Cron.

Před spuštěním aplikace se ještě načítá router. Úkolem routeru je vytvářet SEO - příznivější odkazy. Router zajišťuje, že požadavky uživatele budou doručeny správným presenterům. Nette nemá žádný fallback v případě absence routeru, router je tak pro tento úkol nezbytný. Každá třída, která implementuje interface IModuleRouter, je automaticky přidaná do souboru rout aplikace. Modul si tak může nastavovat vlastní adresy a přesměrování podle potřeb.

O nastavení správného Controlleru a View, tedy v případě Nette presenteru a šablony, se stará BasePresenter modulu Base, který je bezpodmínečným předkem všech presenterů z modulů. V případě, že objeví daný prvek v modulu, načte jej, jinak načítá výchozí presentery a šablony z modulu Base.

2.3.3 Moduly

Base

Základním modulem je modul Base. Obsahuje všechny potřebné funkce k zobrazení modulu, přičemž každou funkci můžou moduly libovolně upravovat. Modul se skládá z presenterů, šablon a routeru. Vše ostatní (například formuláře, databázové vrstvy, pomocné třídy ad.), je umístěno v Core. Předkem pro všechny presentery je BasePresenter. Ten má dva hlavní úkoly. Prvním je vybrání správné šablony. Pokud se ve složce načítaného modulu nachází odpovídající šablona, je primárně určena ke zpracování ona, v opačném případě se použije výchozí šablona z modulu Base. Druhým úkolem je načítání globálních prvků jako je například menu nebo vyhledávání.

Zvláštností je pouze presenter ImagePresenter, který má jedinou metodu, renderCategoryImage. Metody render jsou přístupové metody, tedy metody přímo získávající uživatelské vstupy a zároveň metody zodpovědné za zobrazení stránky. Jejím úkolem je zobrazit obrázek dle předaného ID. Tento údaj je předán společně s rozměry třídě ImageHandler. Ta buď načte a zobrazí obrázek požadovaných rozměrů, existuje-li, nebo načte původní, zmenší jej, uloží a zobrazí. Zároveň nastaví hlavičku ETag, která je otiskem obrázku a slouží jako jeho identifikátor pro cachovací funkce prohlížeče. V případě, že se změní obrázek, změní se md5 otisk, tím se změní ETag a obrázek se načte znova. Tento způsob zpracování obrázků je velmi šetrný k serveru jak z pohledu přenášených dat, tak z pohledu výpočetní náročnosti.

API a Cron

Speciální moduly. Kvůli logice aplikace musí být všechny vnější přístupy vedeny do modulů, i když se nejedná o moduly katalogové povahy. Oba

obsahují pouze MainPresenter a router, který směřuje všechny přístupy v těchto modulech do MainPresenteru.

Modul API slouží jako vstupně výstupní brána pro komunikaci s mobilní aplikací. Modul, respektive presenter sám, předpřipravuje pro mobilní aplikaci data. To kupříkladu znamená, že na dotaz ohledně kategorií v modulu nevrací strom, ale seřazené kategorie v poli. Místo, aby vracel předmět samotný, vrací i napočtené údaje o vlastnictví či seznamy obrázků. Mobilní aplikace má tak zjednodušené zpracování dat a je snížené množství dotazů, které si mezi sebou mobilní a webová aplikace musí k zobrazení výsledků vyměnit.

Cron modul je modul pro automatické úlohy. Aktuální jedinou automatickou úlohou je předpříprava obrázků pro knihovnu OpenCV. Úloha načte obrázky předmětů a pokud ještě neexistují jejich předpřipravené sady, vytvoří jejich kopii, zmenší je na maximální rozměr 480x480 px, odstraní šum a celý obrázek převede do černobílé kombinace. Takto předpřipravená data se ukládají do složky opencv. Operace probíhá jednou denně v noci, nově vytvořené předměty jsou tak dostupné k obrazovému rozpoznávání až následující den.

Katalogové moduly

V rámci ukázky práce byly implementovány a datově naplněny tři moduly. Prvním modulem je Sokol, který rozšiřuje Base modul pouze v minimu funkcí. Jedná se o funkční katalog sokolských odznaků a medailí, ukázku toho, jak by katalog mohl skutečně vypadat. Všechny fotky pochází ze skutečné sbírky. Tento modul byl během vývoje primárním a zvláště na něm se provádělo testování. Tento modul využívá šablon předmětů Odznak a Medaile. Obě šablony se liší jenom ve dvou parametrech, nicméně funkce rozpoznávání obrazu bere z medaile dvě fotografie pro testování (u odznaku jenom jednu), jelikož předpokládá, že medaile je focena z rubu i líce.

Druhým modulem je modul Magic the Gathering. Modul ukazuje možnosti nadstavby nad výchozím modulem. Grafická podoba stránky je upravena pomocí volně stažitelné šablony Darkly [15]. Data jsou stažena pomocí třídy Module\MagicGathering\Scripts\EditionDataDownloader, který stahuje seznam kategorií a položek přes API stránky ScryFall. [16] Pro zobrazení speciálních symbolů, které jsou v popisech karet, je přidán vlastní filtr. [17] Ten všechny speciální výrazy nahrazuje obrázky. Vidět je to u karet v modulu Playground, které tímto filtrem nedisponují. Modul používá šablonu karty, která obsahuje navíc parametr číslo v pořadí, podle kterého lze řadit kategorie.

Třetím a zároveň nejmenším modulem je modul PlayGround. Tento modul nemá žádné vlastní soubory, využívá pouze služeb BaseModulu. Slouží především k ukázce sdílení předmětů, kdy jsou do modulu nasdíleny předměty a kategorie ostatních modulů. Jelikož jde o tytéž předměty jako v ostatních modulech, propisují se i informace jako je například vlastnictví předmětu.

2.3.4 Vlastní vývoj modulu

K základnímu zprovoznění modulu stačí přidat záznam do tabulky modules. Jméno slouží zároveň jako identifikátor modulu, podle jména se řídí name-

space tříd modulů a v app/Modules/<jméno modulu> se hledají všechny soubory modulu, pokud existují. Při navštívení url zadaném v záznamu tabulky, a samozřejmě pokud je daná doména nasměrována na server s touto prací, je automaticky načten výchozí modul stejně jako v případě ukázkového PlayGround modulu. Pokud je požadováno, aby se v rámci modulu vytvářely nové záznamy předmětů, je potřeba přidat ještě záznamy do tabulky module_templates, která povoluje používání vyjmenovaných šablon v modulu. Pokud tyto záznamy přidány nebudou, můžou moderátoři modulu pouze přidávat předměty z ostatních modulů. Pro moderátorství modulu je také nutné přidat záznamy o moderátorech do tabulky module_moderator.

Základní modul je opravdu základní a pro pokročilejší úpravy dat v modulu, jako je například úprava textových symbolů v modulu Magic the Gathering na grafické, je nezbytné založit nový modul i souborově. V modulech se zpravidla nacházejí tři typy souborů: presentery, šablony a router.

Šablony stránek, pokud mají nahradit šablonu z modulu Base, musí zrcadlit umístění šablon modulu Base. Pokud se šablona nachází v Base/templates/<složka>/<šablona>, potom se musí nahrazující šablona nacházet v <název modulu>/templates/<složka>/<šablona>. Tímto dokáže BasePresenter zodpovídající za správné načítání šablon načíst šablonu z jiného modulu.

Presentery musí podědit odpovídající presentery z Base modulu. Tím se zajistí provázání akcí. Všechny odkazy v šablonách jsou totiž vedeny do modulu Base. Presenter v modulu však může tuto akci převzít, i když na něj odkaz nevede. K tomu je potřeba ještě správně nakonfigurovat router. Ten musí přesně říci, které presentery modul obsahuje. Router modulu je totiž při sestavování všech rout předřazen výchozímu Routeru modulu Base, takže při vyhodnocování přesměrování uživatele se primárně nahlédne k routerům modulu. Ač to možná zní složitě, zápis je ve skutečnosti jednoduchý. Tento kód přidává routu, která říká, že všechny akce vedené na presenter Item (zodpovídá za zobrazení detailu předmětů) budou převzaty modulem MagicGathering.

```
$env = SettingsEnvironment::getInstance();
$router = new RouteList('Module:MagicGathering');
$router[] = new Route(
        '//{$env->getModuleUrl()}/item/<action>',
        ["presenter" => "Item"]
);
```

Modul však může registrovat i vlastní služby či vytvářet vlastní formuláře. Pokud existuje konfigurační soubor modulu, který se musí explicitně nacházet na Modules/<jméno modulu>/config/config.module.neon, je tento soubor nahrán a zpracován jako všechny jiné konfigurační soubory. Jelikož se soubor připojuje jako poslední, má možnost změnit v rámci svého běhu i parametry celé aplikace.

2.4 Mobilní aplikace

Celé katalogizování pozbývá smyslu, pokud jej člověk nemůže využít ve chvílích, kdy jej potřebuje nejvíce. V prostředí domova je zpravidla možné si okamžitě zjistit, zda-li sbírkový předmět mám či nemám, nenachází-li se v trezoru mimo domácnost. Kritická chvíle nastává v situacích, kdy se člověk nenachází doma a má krátké rozhodovací okno. Příkladem můžou být nejrůznější burzy či návštěvy starožitnictví. V takových chvílích člověk často dělá chyby a kupuje věci, které již má, domnívajíc se opak. Či přesně obráceně, nekoupí něco, co nemá.

Ač je v dnešní době poměrně dobrá možnost procházet webové stránky v mobilu či tabletu, dedikovaná aplikace je vždy lepší. Zároveň by však mobilní aplikace neměla suplovat existenci webové aplikace – něco takového není žádoucí a pouze by to způsobovalo nutnost dvojité implementace nových funkcionalit. Klíčové vlastnosti mobilní aplikace tak jsou především možnost procházet svoji sbírku a možnost vyhledávat nad kolekcí.

Mobilní aplikace je psaná v jazyce Java s Android SDK. Pro snížení množství přenášených dat ze serveru je na straně serveru, tedy webové aplikace, prováděno předzpracování dat a komprese obrázků. Aplikace je vyvíjena pro zařízení s API alespoň 19, tedy zařízení s Android 4.4 z roku 2013 [19] a novější. Aplikace vychází z výukových příkladů a doporučeních knihy Android Programming.[18] Pro vývoj aplikace jsem použil oficiální Android Studio. Android Studio má několik způsobů zobrazení struktury projektu. Ve všech dalších textech se budu odkazovat na výchozí zobrazení *Android*, které vhodně strukturuje projekt, ale neodpovídá adresářové struktuře projektu.

Zobrazení *Android* od sebe odděluje třídy psané v jazyce Java a zdroje. Zdroje jsou všechny ostatní soubory. Jde o obrázky, jazykové soubory, soubory s informacemi o rozložení prvků (layouty), zvukové soubory a další. Všechny tyto soubory se nachází v adresáři res, kde jsou dále děleny do subadresářů podle svého typu. Nejdůležitější je složka layout, která obsahuje xml soubory s informacemi o rozložení prvků. Android kvůli svému použití na zařízeních s velmi odlišnými rozměry používá relativní pozicování prvků. Například kontejner LinearLayout definuje, že všechny prvky v kontejneru jsou v něm řazeny buď horizontálně nebo vertikálně. Všechny prvky tak jsou v kontejneru relativně vůči sobě.

Java třídy se nacházejí v adresáři java. Roli presenterů z Nette zde zastávají třídy Activity. Activity jsou třídy zodpovědné za zpracování uživatelských vstupů jako jsou doteky, změny orientace zařízení, přepnutí aplikace. Při běhu aplikace je vždy aktivní pouze jedna aktivita. Druhou skupinou aktivních tříd jsou fragmenty. Fragmenty jsou třídy, kterým aktivity delegují práci. Slouží k oddělení logických celků, aby nemusela být veškerá logika pouze v aktivitách. Výhodou fragmentů je také možnost jejich znovupoužití ať v rámci jiných aktivit, nebo jako potomka jiného fragmentu.

O držení dat se stará singleton třída DataBank. Ta pod sebou uchovává odvezené třídy známých tříd z webové aplikace - Module, Category, Item a Ownership. Data se ze serveru získávají lazy-loadingem, což znamená, že

potřebná data jsou stahována až tehdy, kdy jsou potřeba. Obsah kategorie se například stáhne až tehdy, kdy je potřeba kategorii vykreslit. O každou datovou úroveň se stará jeden příslušný fragment, který zároveň zodpovídá i za její vykreslování. O seznam modulů se kupříkladu stará ModuleListFragment, o obsah kategorie CategoryFragment. Výjimkou z toho je pouze singleton třída User, která stojí mimo třídu DataBank a celá se inicializuje ihned po přihlášení.

Aplikace je programována hierarchicky. Výchozí aktivitou je LoginActivity. Jelikož primárním úkolem aplikace je zobrazovat uživateli, které předměty vlastní, nemá mobilní aplikace příliš velkého významu bez uživatelovo přihlášení se. Přihlášení je tedy povinné. Při úspěšném přihlášení se do mobilu uloží přihlašovací údaje a při dalším spuštění aplikace jsou přihlašovací údaje předvyplněny. Při každé akci je na serveru obnovován autorizační token, který je posílán s každým datovým požadavkem, s platností na další dva dny.

Po úspěšném přihlášení se uživatel ocitne na aktivitě ModuleListActivity, která vypisuje seznam dostupných modulů. Z modulů uživatel pokračuje na kategorie, z kategorií na předměty, z předmětů se spouští ještě galerie. Stranou stojí pouze vyhledávání přístupné z menu. Lze vyhledávat pomocí textu nebo experimentálně pomocí fotografie. O obsluhu se stará SearchAktivity s dvěma fragmenty, SearchTextFragment a SearchPhotoFragment. Výsledek hledání se zobrazuje na SearchResultActivity, která pro vypsání výsledků používá CategoryFragment, primárně určený k výpisu předmětů z kategorie.

Kapitola 3
Ovládání

3.1 Ukázková verze

Ukázkové moduly jsou ke dni 1. 1. 2019 přístupné na následujících adresách. Všechny zmiňované funkce jsou k dispozici, uživatelé se po registraci dostanou do role registered, nemají tedy přístup k moderátorským funkcím. Mobilní aplikace není k dispozici pro stažení z internetu. Je možné použít projekt z přílohy a pomocí Android Studia jej nahrát do mobilu. Ukázky níže budou z modulu Sokol.

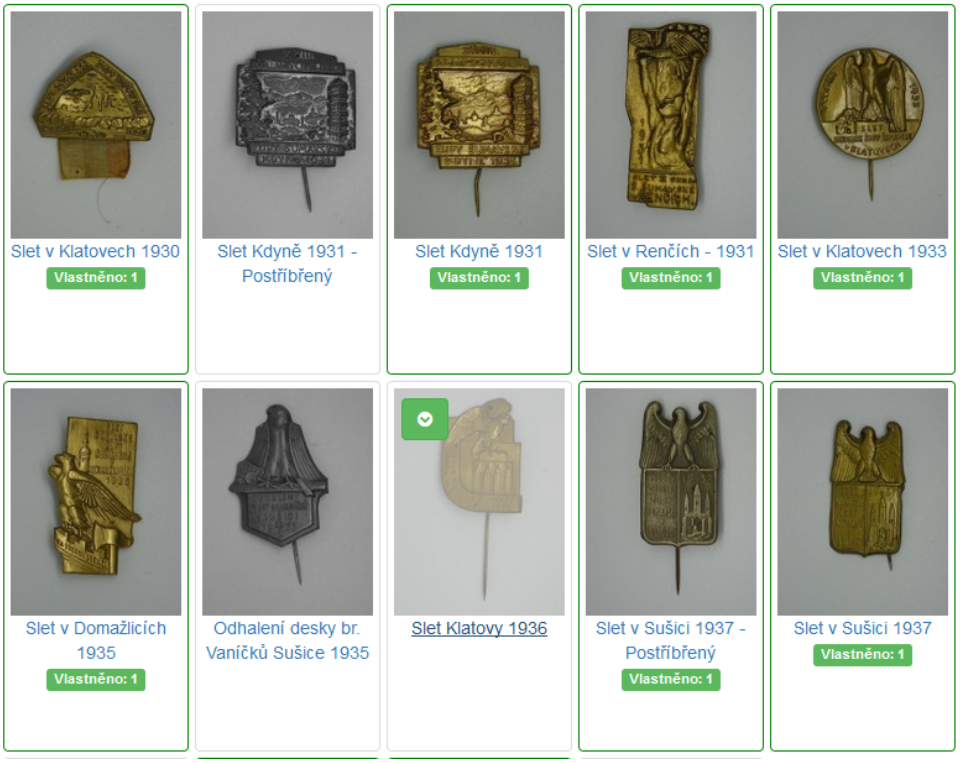
- http://sokol.cataloger.cz
- http://magic-the-gathering.cataloger.cz
- http://playground.cataloger.cz

3.2 Uživatelský pohled

3.2.1 Webová aplikace

Návštěvník stránky si může bez omezení prohlížet všechen obsah. Na hlavní stránce jsou vidět poslední přidané předměty. V přehledu kategorie se liší zobrazení přihlášeného a nepřihlášeného uživatele. Registrovaný uživatel má na rozdíl od nepřihlášeného graficky zvýrazněno, které předměty vlastní. Kromě barevného ohraničení a štítku se registrovanému uživateli ukazují nevlastněné předměty černobíle. Tento grafický prvek umožňuje zrychlení orientace se v katalogu. V případě potřeby může uživatel najet myší na předmět, který se následně vykreslí barevně. Nepřihlášenému uživateli se všechny předměty vykreslují barevně. Na obrázku 3.1 jsou vidět černobíle nevlastněné předměty, barevně s orámováním vlastněné předměty a předmět, odznak s titulkem Slet Klatovy 1936, po najetí myší. Tlačítko v levém horním rohu tohoto odznaku slouží pro rychlé přidání odznaku do sbírky, tento proces znázorňuje obrázek 3.2.

Aplikace nelimituje uživatele v množství vlastnických záznamů. Může si je tak rozdělit například podle stavu či počtu držených a zapůjčených kusů.



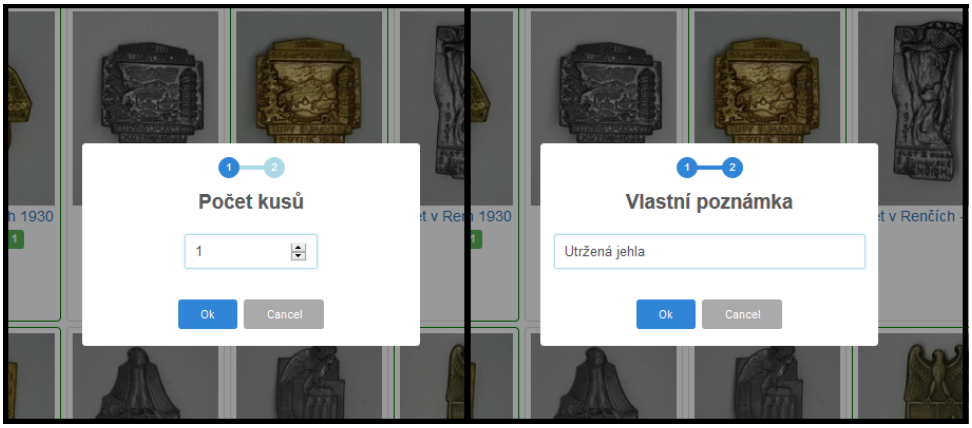
Obrázek 3.1: Zobrazení předmětů v kategorii přihlášeného uživatele

Editace záznamů je přímočará, stejná jako vytváření záznamu z přehledu kategorie. Na kartě předmětu jsou ještě další tři záložky. Popis, vlastnosti a diskuze. Popis je výchozí a zde by se nacházely zajímavosti k předmětu. Vlastnosti jsou specifické parametry definované šablonou. Z parametrů jsou v ukázkové verzi vyplněny pouze rok a město, ostatní parametry z časových důvodů vyplněny zpravidla nejsou. Diskuze je záložka, kam mohou uživatelé přidávat své postřehy. Karta předmětu je zobrazena na obrázku 3.3

3.2.2 Mobilní aplikace

Mobilní aplikace slouží primárně k listovaní katalogem vlastněných předmětů a tomu byla aplikace přizpůsobena. Aplikace rozšiřuje zadání práce a přidává podporu i menších zařízení, zejména mobilních telefonů. I když je jistě lepší ovládání tabletem, ignorovat celý segment mobilních telefonů nepokládám za dobrý nápad. Fotky z mobilní aplikace jsem vyfotil v emulátorech Android Studia.

Po spuštění aplikace je uživatel vyzván k přihlášení. Jelikož mobilní aplikace je primárně rozšířením webové aplikace pro umožnění procházení své sbírky, není přidána podpora nepřihlášených uživatelů. Po přihlášení musí uživatel vybrat modul, ve kterém se chce pohybovat. To načte seznam kategorií a vypíše je. Nachází-li se uživatel na tabletu, má obrazovku rozdělenou na dvě části, v levé má kategorie a v pravé předměty dané kategorie. Pokud je uživatel na mobilu, je nucen se proklikávat v pořadí modul -> kategorie



Obrázek 3.2: Přidání předmětu do uživatelské sbírky

-> přehled předmětů v kategorii -> detail předmětu. Mezi předměty se pak může pohybovat posunem do boku pomocí ViewPageru. [20]

Uživatel na tabletu je ve větší výhodě. Po celou dobu pohybu v modulu mu zůstává v levé části menu kategorií 3.4, takže se může snadněji přepínat mezi kategoriemi. Výjimkou je pouze vyhledávání a galerie, obě aktivity se zobrazují přes celou obrazovku. Detail předmětu 3.5 je podobný tomu z webové aplikace, vynechána je pouze diskuze.

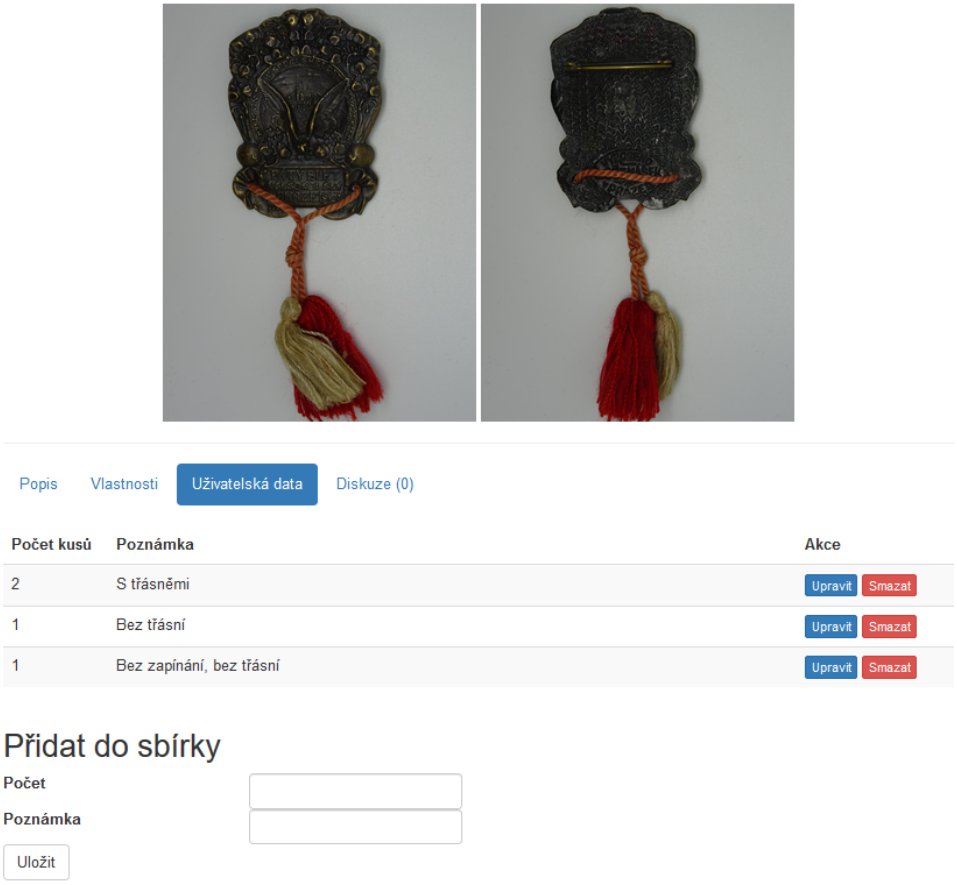
Kromě zobrazování položek umožňuje aplikace i vyhledávání. Klasické pomocí textů. Text se porovnává s názvem, popisem, městem a rokem. Druhou možností je vyhledávání pomocí fotografie. Ta se odešle na server, kde se zpracuje a pošle se výsledek zpět. Avšak jelikož je ukázový server málo výkonný, je vhodnější používat textové vyhledávání, pokud je to možné. Obrazové vyhledávání je tak především pro předměty, které nemají text a kde tak nelze využít textového vyhledávání.

3.3 Moderátorský pohled

Úkolem moderátora je především tvořit kolekci předmětů ať již jejich vytvářením nebo přebíráním z jiných modulů, konfiguraci modulu na starost nemají. Po přihlášení uživatele v modulu, ve kterém je moderátorem, se mu zobrazí moderátorský panel. V něm má k dispozici tři položky

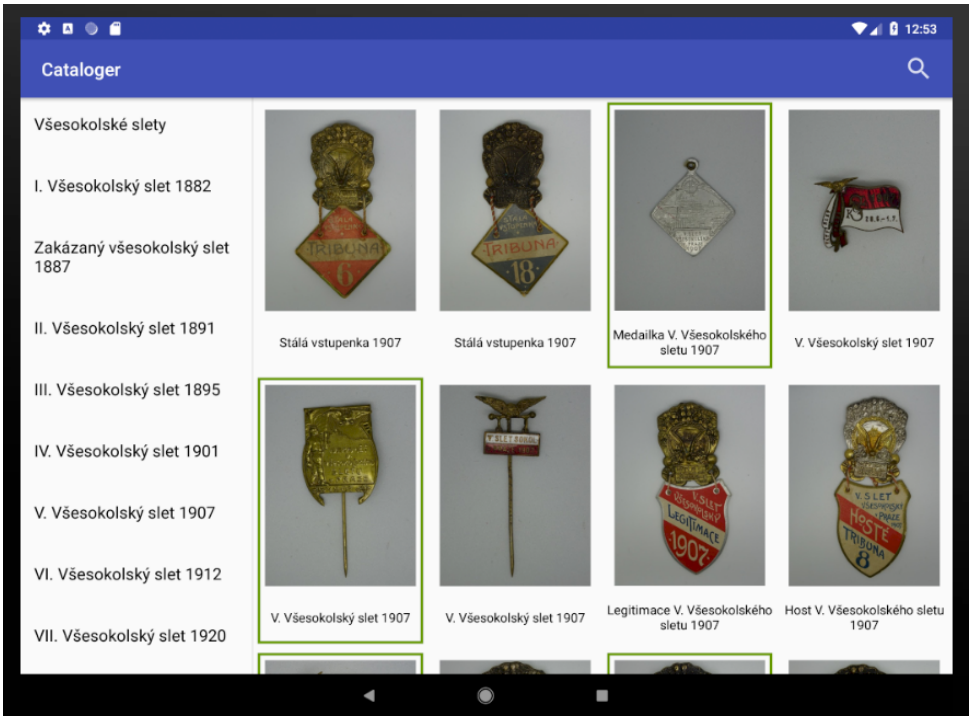
Ve správě kategorií může tvořit kategorie a sestavovat strom kategorií. Ten je řešen systémem drag & drop, což znamená, že kategorie lze přesouvat pouhým přetahováním myší. Pořadí, ve kterém se kategorie při ukládání nachází, je přeneseno i do výpisu. Žádné automatické řazení kategorií například podle jména implementováno není. Úprava kategorie se provádí přímo na profilu kategorie. Popis je pro moderátory zobrazen stejně jako pro uživatele, ale ve skutečnosti se jedná o textareu, do které stačí psát a úpravy potvrdit. 3.6

Moderátor má možnost předmět vytvořit nebo převzít z jiného modulu. Při přebírání musí znát ID předmětu. To se zobrazuje v url a předpokládá se, že případní moderátoři by byli s touto informací obeznámeni. Předmět se vytváří

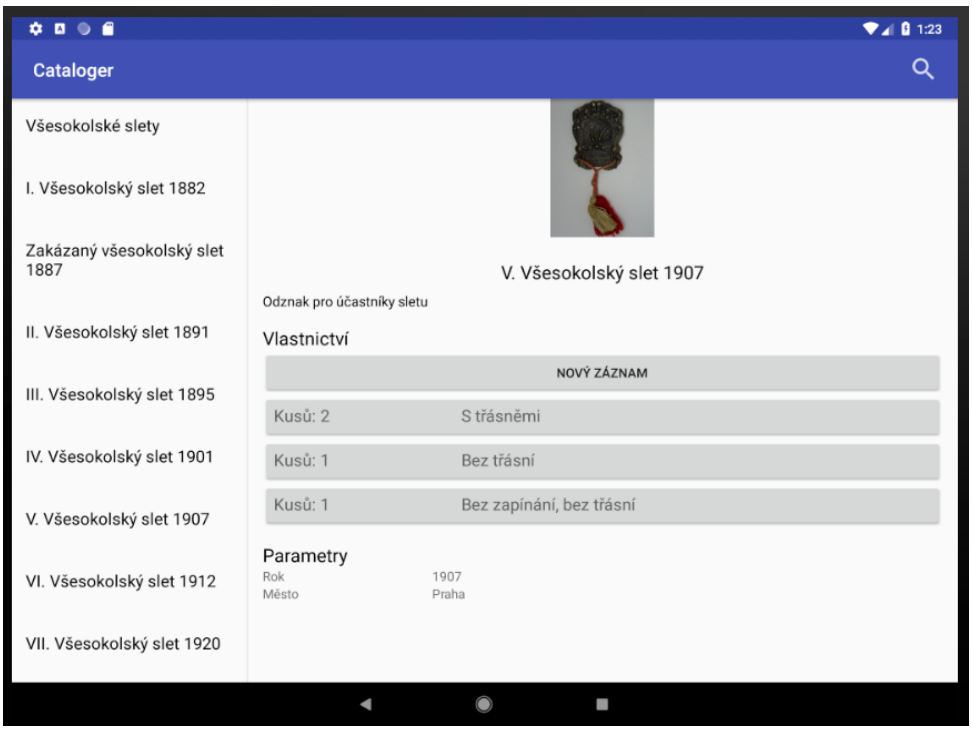


Obrázek 3.3: Detail předmětu

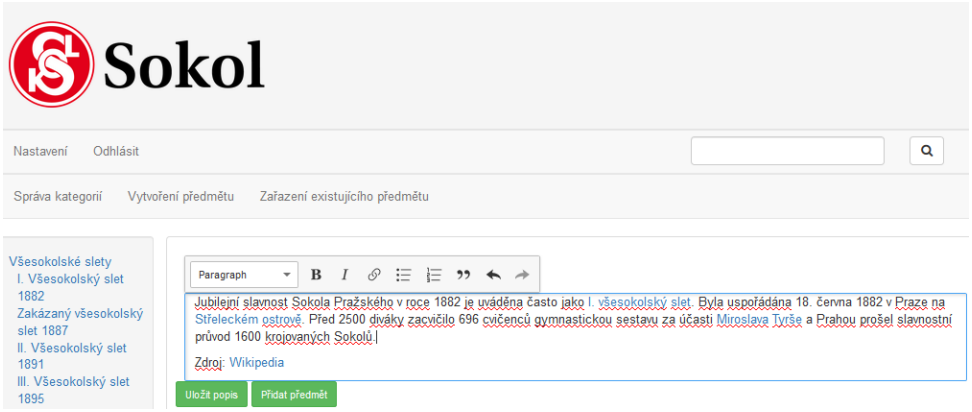
na záložce vytvoření předmětu. Při navštívení stránky vidí uživatel pouze dva seznamy, kategorie a šablony. Moderátor musí nejdříve vybrat šablonu ze seznamu a potvrdit ji. Na základě výběru šablony se poté zobrazí příslušná pole. Pole, která nejsou na šabloně povoleny, se nevypisují. K editaci předmětu se přistupuje z předmětu samotného. Moderátorovi se zobrazí navíc záložka pro editaci. Editace probíhá ve stejném formuláři jako vytváření předmětu s výjimkou polí kategorie a šablona, které jsou v editaci zakázané. Navíc se však zobrazují všechny kategorie všech modulů, ve kterých se předmět nachází. Kromě toho se zde nachází i přehled přiřazených obrázků, které se dají i smazat. Při smazání obrázku aplikace odebere rovněž všechny vygenerované miniatury.



Obrázek 3.4: Výpis předmětů v kategorii na tabletu



Obrázek 3.5: Detail předmětu na tabletu



Obrázek 3.6: Úprava popisu kategorie se zobrazenou moderátorskou lištou

Kapitola 4

Rozpoznávání obrazu

4.1 Úvod

Účelem obrazového rozpoznávání je vyhledat požadovaný předmět bez nutnosti jeho hlubší znalosti uživatelem. Tento způsob je preferován především v situaci, kdy předmět neobsahuje texty a vyjádření slovy jeho podstaty je minimálně obtížné. V takovém případě je obrazové vyhledávání takřka jedinou možností.

Pro funkci obrazového rozpoznávání byla zvolena knihovna OpenCV ve verzi 3.4. OpenCV je jednou z nejpoužívanějších knihoven pro počítačové vidění používaných mimo jiné firmami Google, Yahoo, Microsoft či Intel. [14] Jeho předností je veliké množství funkcí a nástrojů včetně implementace do dalších jazyků. Ačkoli je původně psaný v C++, je dostupná i knihovna v jazyce Java, kterou tato práce využívá.

Původní myšlenkou bylo naučit systém rozpoznávat jednotlivé kategorie předmětů. Jelikož v rámci vývoje jsem se zaměřil na odznaky, velmi rychle jsem se dostal do problému definice samotného odznaku. Při pohledu na různé odznaky se člověk dostává do obtíží hledání společných znaků. Odznak může být různých tvarů, materiálů i rozměrů. I v případě, že bychom vzali do úvahy pouze odznaky z dob před rokem 1948, i zde narážíme na velmi různé tvary, jak ilustruje obrázek 4.1. Jediným skutečným společným znakem tak zůstává přítomnost jehly, která však nemusí na čelní straně být nutně vidět. Z tohoto důvodu jsem se rozhodl změnit přístup a místo identifikování skupin identifikovat samotné předměty.

4.2 Implementace

Pro práci jsem zvolil modul features2d a vycházel jsem ze série tutoriálů od Any Huamán.[23] Tyto články pracují s descriptorem SURF (Speeded up robust features), který není dodáván ve standartní distribuci s OpenCV a jeho použití vyžaduje vlastní kompilaci celé knihovny s explicitním povolením tohoto descriptoru. Důvodem je jeho patentová ochrana a možnost jeho použití pouze pro nekomerční účely, což této bakalářské práci vyhovuje.

SURF modul však obsahuje zvláštní chybu, kde každé následující volání

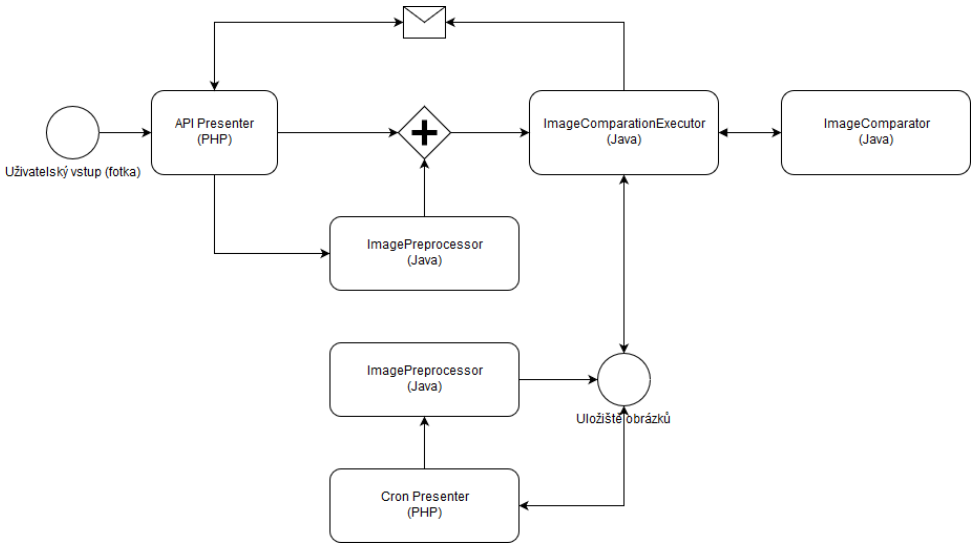
Obrázek 4.1: Ukázky různých tvarů odznaků

nad stejným vstupem generuje rozdílné výsledky. Odhaduji, že tento modul obsahuje nějakou skrytou statickou proměnnou. Na internetu se dají dohledat dotazy na chování tohoto modulu v jazyce Java, odpovědi jsem však již nenalezl. Pro vyřešení tohoto problému jsem tak musel aplikaci pro rozpoznávání rozdělit na dvě. První aplikace provádí samotné rozpoznávání, jejím vstupem jsou obrázky k porovnání, výsledkem je obodování. Druhá aplikace zodpovídá za její volání, předává seznam obrázků a přebírá výsledky. Nevýhodou je zpomalení celého procesu, bohužel chyby z opakovaného volání SURF detektoru byly příliš závažné, než aby mohly být opomíjeny.

V celém procesu rozpoznávání obrazu jsou zapojeny zejména tři miniaplikace napsané v jazyce Java a dva scripty v PHP s několika dalšími podpůrnými třídami. Schéma zapojení aplikací znázorňuje obrázek 4.2.

Předpříprava dat

Obrázky předmětů sloužící po porovnávání s hledaným předmětem z obrázku jsou připraveny předem. Cron modul invokovaný cron akcí operačního systému načte všechny předměty a pokud ještě nemají vytvořený snímek pro rozpoznávání obrazu, vytvoří kopii hlavního obrázku předmětu, zmenší jej na maximálních 480x480 px a zavolá aplikaci ImagePreprocessor, která z



Obrázek 4.2: Schéma zapojení komponent pro zpracování obrazu

obrázku odstraní šum a následně jej převede do černé a bílé barvy. Takto upravené obrázky jsou uloženy na disk.

Uživatelský vstup

Jelikož je funkce rozpoznávání obrazu k dispozici pouze z mobilní aplikace, je uživatelský vstup přijat vždy modulem API a jeho presenterem. Ten zkontroluje authentizační údaje a správnost obdržení obrázku (obrázek je z mobilní aplikace odesílán zkonvertovaný do formátu base64 [24]). V případě kladného vyhodnocení obou bodů je obrázek upraven v aplikaci ImagePreprocessor. Upravený obrázek společně s dalšími údaji převezme ImageComparationExecutor. Jeho úkolem je především spouštět ImageComparator a odečítat z něj výsledky. Vstupem ImageComparatoru je testovaný obrázek a porovnávaný obrázek, který byl již dříve předpřipraven v modulu Cron. Výstupem ImageComparatoru je plocha, ve které by se měl nacházet porovnávaný obrázek, přenásobená o počet shodných bodů nalezených v obou obrázcích. V případě prvního údaje se přepokládá, že největší nalezená plocha bude patřit hledanému předmětu. Druhý násobitel slouží především k omezení false - positive chyb, kdy malé množství společných bodů mohlo způsobit chybné určení obrázku. Algoritmus předpokládá, že dva podobné obrázky budou mít významné množství společných bodů. Tato bodová ohodnocení sbírá ImageComparationExecutor, který je ukládá do souboru, jehož umístění bylo předáno z API presenteru. Po vytvoření tohoto souboru jej přečte čekající API presenter, přeparsuje výsledky a vrátí jako svůj výsledek až 100 předmětů, které dosáhly nejvyššího ohodnocení.

Kapitola 5
Testování aplikace

Testování práce jsem rozdělil na tři části, webovou aplikaci, mobilní aplikaci a samostatně rozpoznávání obrazu. Webovou aplikaci jsem díky snadnému sdílení zpřístupnil dalším sběratelům, kteří byli ochotni aplikaci otestovat. Všichni uživatelé byli v roli sběratelů a měli za úkol testovat stránku z pohledu běžných návštěvníků. Kromě nich aplikaci testovala i moje přítelkyně, která plnila daty sokolský modul a vytvořila v něm více než 700 položek. Z jejích poznatků se vylepšoval moderátorský pohled.

Druhou částí byla mobilní aplikace. Vzhledem k obtížnému sdílení rozpracovaných aplikací na platformě Android jsem se rozhodl tuto část testovat sám zpracováním své sbírky z mobilu.

Třetí samostatnou částí je modul na rozpoznávání obrazu, kde jsem testoval odezvu systému na nafocené snímky odznaků. Fotky byly vyfoceny mobilem, ale samotné testování probíhalo již v počítači.

5.1 Testování webové aplikace

Uživatelé přišli s několika připomínkami, z nichž většinu jsem implementoval. Dvě jsem zamítnul, ač jedna z nich mi přišla zajímavá, kvůli náročnosti vývoje jsem ji do práce nakonec nezahrnul. Uvádím ji alespoň v sekci Možnosti dalšího vývoje 5.4. V tabulce 5.1 uvádím seznam připomínek a stav realizace.

5.2 Testování mobilní aplikace

V mobilní aplikaci se mi podařilo odladit téměř všechny chyby. Aplikace řeší správně i *Rotation problem*, který spočívá ve ztrátě dat při změně orientace zařízení. Odladil jsem stavy, kdy se uživatel snaží provést akci s již expirovaným tokenem. V takovém případě je vrácen na aktivitu s loginem. Na tuto aktivitu se rovněž již nedá vrátit po přihlášení používáním tlačítka zpět.

Stále však existují chyby, které se občas objeví, ale nedokážu je reprodukovat a tedy ani vyřešit. Tyto chyby se objevily pouze v emulátoru, ale nedá se vyloučit, že se neobjeví i na fyzických zařízeních.

Nevýhodou rovněž zůstává občasná pomalejší odezva ukázkového serveru. To je bohužel situace daná hardwarem serveru. Řešením by mohla být nějaká

Požadovaná funkce	Můj komentář	Stav realizace
Přidání kategorie na detail předmětu	Vzhledem k tomu, že předmět se může i v rámci jednoho modulu nacházet ve více kategorií, musí se vypsat všechny možné varianty. Nedá se určit, ze které kategorie uživatel přišel.	Implementováno
Přidání možnosti na obnovení hesla	Opomenutá funkce	Implementováno
Změna názvu tlačítka "Přiřadit předmět"	Jedná se o tlačítko pro přiřazení existujícího předmětu do kategorie. Mělo být snadno zaměnitelné s tlačítkem "Vytvořit předmět". Tlačítko jsem přejmenoval na "Zařazení existujícího předmětu".	Implementováno
Možnost smazat vlastnictví přímo tlačítkem	Před úpravou se vlastnictví mazalo snížením počtu na nulu (což stále funguje). Přidal jsem ale tlačítko, které maže vlastnictví přímo	Implementováno
Přesunutí uživatelských tlačítek do pravé části	I když podle mého pozorování musím uznat, že většina stránek má tyto prvky v pravé části, ponechávám je v levé. V pravé části se nachází pole pro vyhledávání, které má přednost. Tlačítka pro přihlášení či následně pro nastavení nebudou využívány tak často, aby se kvůli nim měl měnit design.	Zamítnuto
Možnost tvorby uživatelských modulů	Nápadu se věnuji samostatně	Odloženo

Tabulka 5.1: Seznam obdržených připomínek

forma cachování dat a ukládání je u uživatele, při každém přihlášení by se však musela stejně udělat nějaká forma synchronizace dat a je otázkou, kolik času by toto ušetřilo. Největším časovým konzumentem jsou fotky, avšak podle mého názoru jsou lidé ochotnější počkat o sekundu dvě déle, než u sebe v mobilu skladovat stovky až tisíce fotek z aplikace.

- Několikrát jsem pozoroval, že se kategorie překreslí přes jinou. K takovému chování by však nemělo docházet, jeden fragment kategorie je při přepnutí vždy nahrazen jiným. Při přepnutí na jinou kategorii a zpět se kategorie vykreslí správně.

- Jedinkrát jsem v poslední verzi mobilní aplikace narazil na to, že aplikace spadla. Při prokliku z kategorie na předmět aplikace spadla, protože prý předmět nebyl inicializován. Předměty se však inicializují při načtení kategorie a bez inicializace předmětu by nemohl být ani zobrazen samotný předmět v kategorii. Při opětovném spuštění a používání aplikace se mi chybu nepodařilo reprodukovat.

5.3 Testování systému pro rozpoznání obrazu

Pro testování jsem zvolil 15 náhodných odznaků. Tyto odznaky jsou stejného typu, jako ty v databázi, avšak nejedná se o totožné kusy. Některé odznaky vypadají téměř identicky, jiné však nemají například stuhu. Chybějící stuha je běžná vada odznaků, která by v ideálním případě neměla mít na porovnávání vliv. Odznaky jsem vyfotil za různých světelných podmínek různě kvalitními přístroji, následně jsem fotky porovnával s databází 793 fotek ze 735 předmětů. Všechny fotky jsou přiloženy na DVD, uvedená ID odpovídají názvům souborů.

První test byl proveden kompaktním fotoaparátem Sony DSC HX-60, kterým byly foceny i samotné předměty v modulech, ve světelném boxu s umělým osvětlením. Tento test představuje referenční laboratorní podmínky. V rámci testu všechny předměty kromě jednoho skončily na prvním místě, tedy byly správně určeny. Chybně je určena medailka "Župní slet Boskovice", kde kvůli jednotvárné textuře předmětu nedošlo k optimální konverzi obrázku v rámci preprocessingu. Test shrnuje tabulka 5.2

ID	Počet obrázků	Pořadí	Lepší než [%]
-	14	1	100%
15	1	58	92,7%
-	**15**	**4,8**	**99,5%**

Tabulka 5.2: Výsledky prvního testu systému pro rozpoznávání obrazu

Pro druhý test jsem zvolil mobilní telefon nižší třídy Samsung Galaxy J5 z roku 2016. Fotoaparát příliš dobrých kvalit nedosahuje, ty jsem se však v tomto testu pokoušel vyvážit světelným boxem. Výsledky shrnuje tabulka 5.3 a hodnoty jsou nepřekvapivě horší než v předchozím bodě. Medailka z předchozího testu se propadla až na 437 místo. Překvapivě však skončil odznak s id 8, "Otevření sokolovny v Praze - Vokovice". Tento odznak mi svojí strukturou složitý nepřipadá. Na prvním místě se ostatně umístil odznak velmi podobný, jenom s jiným textem. I podle grafického zobrazení se, alespoň ze subjektivního pohledu, podařilo správně mapovat body. Přesto je výsledek, jaký je. Předpokládám, že kdyby došlo k novému nafocení, bude ohodnocení a tím i umístění lepší. Úspěšnost je však stále vysoká, i přes dvě chyby je téměř 90%.

Pro třetí test jsem nastavil i špatné světelné podmínky, odznaky byly foceny v kuchyni za oranžového světla lampy, kterou jsem ještě účelně stínil. Šum na těchto fotkách je vysoký, mnohé texty nejsou ani čitelné. Tyto podmínky představují situaci, ve které by se fotit ani nemělo. Výsledky tohoto testu

ID	Počet obrázků	Pořadí	Lepší než [%]
-	10	1	100%
14	1	4	99,5%
12	1	66	91,7%
9	1	166	79%
15	1	437	44,9%
8	1	517	34,8%
-	**15**	**80**	**89,9%**

Tabulka 5.3: Výsledky druhého testu

shrnuje tabulka 5.4. Mnohé z obrázků na posledních místech jsou v situaci, kdy kvůli šumu po práci ImagePreprocessoru nezbylo z obrázku kromě ohraničení nic moc. Přesto je překvapením, že se umístily až na konci, tedy že většina ostatních obrázků k porovnání má k testovanému blíže. Jedná se o zajímavé chování, jeho pochopení by mohlo vést ke zlepšení výpočtů z nekvalitních snímků.

ID	Počet obrázků	Pořadí	Lepší než [%]
-	7	1	100%
13	1	12	98,5%
14	1	165	79,2%
9	1	495	37,6%
15	1	700	11,7%
3	1	708	10,7%
10	1	738	6,9%
7	1	741	6,6%
8	1	759	4,3%
-	**15**	**288**	**63,7%**

Tabulka 5.4: Výsledky třetího testu

Je zřejmé, že úspěšnost vyhodonocení snímku stojí a padá na kvalitě zařízení a scény samotné. Předměty, které se odlišovaly od předlohy například stuhou či zašpiněním, byly alespoň v prvním testu určeny správně. Je zjevné, že při pořízení kvalitních snímků dokáže aplikace data solidně vyhodnocovat. Při nekvalitních snímcích je úspěšně určen každý druhý snímek, což jsou stále lepší výsledky, než které jsem od testu očekával. Je však otázkou, jak vyřešit umístění pěti odznaků na pozicicích 700+, protože i kdyby docházelo k porovnávání pouze obrysu předmětu a vše ostatní by pohltil preprocessor šumu, měly by se tyto předměty nacházet výše.

5.4 Možnosti dalšího vývoje

I když jsem s mobilní aplikací spokojen, je zde jedno úskálí, kterým je autentizace. Ideální by byla implementace OAuth2 protokolu. Jelikož se jedná o rozsáhlou problematiku, nakonec jsem ji do finální verze aplikace nezahrnul.

Systém pro rozpoznávání obrazu je vždy možno vylepšovat. Jedním z nápadů by mohlo být i rozpoznávání textu. Problém však je dlouhotrvající výpočet. Bez snížení času zpracování snímku se není kam příliš posouvat. V tuto chvíli mi taky není známá žádná knihovna, která by dokázala číst text po obvodu medailí či obecně nacházet text v libovolném směru bez nutnosti rotování s obrázkem.

Zajímavý nápad z testování byl možnost uživatelských modulů, kde by si uživatel mohl tvořit svoji sbírku stejně, jako když moderátor spravuje modul. Nad nápadem jsem přemýšlel, ale jeho implementace by pravděpodobně zabrala příliš času. Navíc je nutné vyřešit otázku bezpečnosti. Nepřikláněl bych se však k možnosti vytváření předmětů, protože tím by každý uživatel pouze předměty vytvářel pro sebe místo snahy je čerpat z modulů, což jde proti duchu práce. Nehledě na zřejmou datovou zátěž. Umím si však představit situaci, kdy si uživatel vytvoří kategorii, do které vybere pouze předměty, které ho zajímají. Abych uvedl příklad, jeden z mých kolegů sbírá pouze tématiku Opavska, z celého modulu Sokol ho tak zajímá jenom necelá desítka odznaků z tohoto okresu.

Vhodnou funkcí by bylo také diskuzní forum. Vzhledem k náročnosti jenom samotné logiky však nebylo zahrnuto do finální verze. Stejně jako vše ostatní by i forum muselo být modulární. Je zřejmé, že v sekci "Československo"by se mělo propisovat i fórum modulu Sokol. Avšak jak jsem již řekl, vytvořit skutečně dobré fórum je v dnešní době, kdy jsou uživatelé navyklý na nějaký standard, velmi náročný úkol. A špatné fórum zase nemá smysl tvořit, takové totiž uživatele spíše ještě odradí.

Kapitola 6

Závěr

Téma této práce jsem tvořil jako bič na sebe. Udělat modulární aplikace pro kolekce jsem chtěl už pár let, nebyl jsem ovšem schopen se do projektu přinutit. Věděl jsem, že vytvoření modulární aplikace je časově velice náročný úkol, který je navíc nutné promýšlet dopředu. Chybná implementace dílčích částí by vedla k obrovským problémům v případě dalšího vývoje. Ostatně i v rámci této práce docházelo k některým rozsáhlým změnám kvůli udržitelnosti kódu. Navíc jsem nebyl ochoten se smířit s jednoduchou stránkou, která by dokázala implementovat pouze jednu kolekci. I když můj sběratelský obor je poměrně úzce zaměřen, byl bych rád, kdyby další moduly přibývaly, vzali si je na starost další lidi a tvořili veřejné online katalogy pro všechny. Co mě během vývoje překvapilo je, jak snadno se, při znalosti Javy, programuje pro Android. I když celý ekosystém nepůsobí pro nováčky přátelským dojmem, pro naprogramování funkční aplikace stačí méně, než by člověk čekal. Je však nutné přiznat, že bez publikací by byl vývoj mnohem náročnější.

V rámci práce se podařilo vytvořit skutečně modulární webovou aplikaci. I když mám v hlavě další a další nápady, kam práci směřovat, bylo nutné se držet rozsahem při zemi. Tyto myšlenky však slouží jako nadstavba na to nejdůležitější - funkční kostru, kterou lze již nyní použít na takřka libovolné sbírky.

Podařilo se vytvořit i mobilní aplikaci, která sice neoplývá designem, je však funkční a plně vykonává činnost, pro kterou byla stvořena, totiž práce se sbírkou mimo domov, mimo přístup ke sbírce i k počítači.

Podařilo se vytvořit obstojné zpracování obrazu. Co se nepodařilo, je zpracování obrazu v rozumném čase. Aktuální průměrná doba zpracování jednoho požadavku je při 800 snímcích k porovnání kolem deseti minut. A to je daleko, daleko za rozumnou dobu, po kterou se dá čekat. Navíc při zpracování dochází k extrémnímu vytížení procesoru. Při více souběžných požadavcích by čekací lhůta byla ještě delší. Variantou je upgrade hardwaru, možná by též šlo zmenšit obrázky. Každé zmenšování je však na úkor přenosti. Řešením by též mohlo být využití grafické karty namísto procesoru při zpracování obrazu. Je to ostatně úkol, ve kterém by grafické karty měly excelovat. Toto řešení jsem však vzhledem k ukázkovému serveru, který je jenom virtuálním serverem bez grafické karty, použít nemohl.

Literatura

[1] *Sběratelství* [online], poslední aktualizace 15. 2. 2018 [cit. 1. 1. 2019], Wikipedie. Dostupné z:
`https://cs.wikipedia.org/wiki/Sb%C4%9Bratelstv%C3%AD`

[2] *How Many People Collect Stamps* [online], 25. 8. 2016, Apfelbaum Inc. Dostupné z:
`https://www.apfelbauminc.com/blog/post/how-many-people-collect-stamps/`

[3] *Nejdražší známka stojí 192 milionů. První majitel ji prodal za pár šilinků* [online], 15. 6. 2014, iDNES. Dostupné z:
`https://zpravy.idnes.cz/drazba-nejvzacnejsi-znamky-one-cent-dbs-/zahranicni.aspx?c=A140618_064609_zahranicni_skr`

[4] *10 Rarest and Most Valuable Coins in the World* [online], 16. 10. 2016, the spruce crafts. Dostupné z:
`https://www.thesprucecrafts.com/rarest-and-most-valuable-coins-768161`

[5] *LiveBid - Aukce 86* [online], 19. 5. 2018, LiveBid. Dostupné z:
`https://livebid.cz/auction/aurea_86/detail/161#`

[6] *About Us* [online], StampWorld. Dostupné z:
`https://www.stampworld.com/en/about/`

[7] *Úvod* [online], Pteranodon Soft. Dostupné z:
`http://www.pteranodonsoft.cz/`

[8] *Usage statistics and market share of PHP for websites* [online], Web Technology Surveys. Dostupné z:
`https://w3techs.com/technologies/details/pl-php/all/all`

[9] ČÁPKA, David *Úvod do Nette frameworku pro PHP* [online], ITNetwork. Dostupné z:
`https://www.itnetwork.cz/php/nette/zaklady/uvod-do-php-frameworku-nette`

[10] GRUDL, David *Pojďte otestovat Nette 2.4 RC* [online], Nette Framework forum. Dostupné z:
`https://forum.nette.org/cs/26250-pojdte-otestovat-nette-2-4-rc`

[11] *Composer* [online], poslední aktualizace 4. 7. 2018 [cit. 1. 1. 2019], Wikipedie. Dostupné z:
`https://cs.wikipedia.org/wiki/Composer`

[12] *TIOBE Index for December 2018* [online], poslední aktualizace 12. 2018 [cit. 31. 12. 2018], TIOBE. Dostupné z:
`https://www.tiobe.com/tiobe-index/`

[13] *ETag* [online], poslední aktualizace 6. 2018, Mozilla. Dostupné z:
`https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/ETag`

[14] *About* [online], cit. 1. 1. 2019, OpenCV. Dostupné z:
`https://opencv.org/about.html`

[15] *Free themes for Bootstrap* [online], navštíveno 1. 1. 2019, Bootswatch. Dostupné z:
`https://bootswatch.com/3/`

[16] *API Documentation* [online], navštíveno 1. 1. 2019, ScryFall. Dostupné z:
`https://scryfall.com/docs/api/`

[17] *Latte filtry* [online], navštíveno 1. 1. 2019, Nette. Dostupné z:
`https://latte.nette.org/cs/filters`

[18] PHILLIPS Ben, STEWART Chris a MARSICANO Kristin. *Android Programming: The big nerd ranch guide, 3rd edition.* Indianapolis: Pearson Technology Group, 2017. 695 s. ISBN 978-0134706054

[19] *Android KitKat* [online], poslední aktualizace 23. 6. 2018 [cit. 1. 1. 2019], Wikipedie. Dostupné z:
`https://cs.wikipedia.org/wiki/Android_KitKat`

[20] *Slide between fragments using ViewPager* [online], poslední aktualizace 20. 12. 2018, navštíveno 1. 1. 2019, Android Developers. Dostupné z:
`https://developer.android.com/training/animation/screen-slide`

[21] *Frequently Asked Questions* [online], navštíveno 1. 1. 2019, PHP-FIG. Dostupné z:
`https://www.php-fig.org/faqs/`

[22] *PSR-2: Coding Style Guide* [online], navštíveno 1. 1. 2019, PHP-FIG. Dostupné z:
`https://www.php-fig.org/psr/psr-2/`

[23] *2D Features framework* [online], navštíveno 1. 1. 2019, OpenCV. Dostupné z:
`https://docs.opencv.org/3.4/d9/d97/tutorial_table_of_content_features2d.html`

[24] *Base64* [online], poslední aktualizace 22. 8. 2018 [cit. 1. 1. 2019], Wikipedie. Dostupné z:
`https://cs.wikipedia.org/wiki/Base64`

Příloha A

Seznam použitých zdrojových kódů a dalších materiálů

Všechny mnou vytvořené zdrojové kódy se v případě PHP aplikace nacházejí v root adresáři app. U metod, které jsem převzal, uvádím vždy tag @author s uvedením buď autora, nebo alespoň stránky, odkud kód pochází. Stejně postupuji v případě mobilní aplikace. Dále uvádím seznam zdrojů ostatních materiálů, které jsem doposud neuvedl.

- Core\Components\Image\ImageHandler::renderImage - Část věnovaná cachování obrázků a odeslaných hlaviček
- Core\Components\System\System::executeCommand - Spouštění příkazů a čtení výsledků
- Core\Components\Settings\Settings::isMobileDevice - Detekce mobilních zařízení
- Core\Utils\Utils::rrmdir - Recursivní mazání adresáře a jeho obsahu
- Module\Base\Presenter\BasePresenter::buildCategoryTree - Tvorba stromu kategorií
- cz.cataloger.cataloger.general.PhotoProcessor::rotate a rotateImage - Detekce, v jaké orientaci mobilu byl snímek vyfocen, a jeho rotace před odesláním k vyhledávání fotkou.
- cz.cataloger.imagecomparator.ImageComparator::run - Detekce objektu dle tutorialu features2d [23]
- Obrázky pro modul Magic the Gathering pocházejí ze serveru http://scryfall.com.
- Obrázky symbolů používaných v popisech karet modulu Magic the Gathering pocházejí z projektu Manamoji Discord od vývojářů ScryFall.
- Placeholder symbol načítání, který se zobrazuje v mobilní aplikaci před načtením fotky předmětu, pochází ze serveru http://icons8.com

Příloha B

Přílohy na DVD

- Databáze - SQL script pro vytvoření databáze a nahrání obsahu, obsah je použit z ukázkové aplikace
- Mobilní aplikace - Celý projekt Android studia pro mobilní aplikaci
- Testování obrazu - Použité fotografie v testu pro rozpoznávání obrazu včetně referenčních fotografií nahraných v aplikaci
- Webová aplikace - Webová aplikace bez dat z modulu Magic Gathering, která lze vytvořit pomocí třídy EditionDataDownloader, zato s výslednými snímky ImagePreprocessoru ve složce opencv
- Zpracování obrazu - Zdrojové kódy aplikací ImagePreprocessor, ImageComparator a ImageComparationExecutor a jejich zkompilované soubory jar.