VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH INFORMAČNÍHO SYSTÉMU
INFORMATION SYSTEM DESIGN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Eduard Stankovič**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Miloš Koch, CSc.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Student:	**Eduard Stankovič**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
Vedoucí práce:	**doc. Ing. Miloš Koch, CSc.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává bakalářskou práci s názvem:

Návrh informačního systému

Charakteristika problematiky úkolu:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza problému
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Na základě firemní strategie a potřeb firmy připravit návrh řešení nového informačního systému

Základní literární prameny:

BASL, Josef a Roman BLAŽÍČEK. Podnikové informační systémy: podnik v informační společnosti. 3. aktualiz. a dopl. vyd. Praha: Grada, 2012. 323 s. ISBN 978-80-247-4307-3.

GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. Podniková informatika. 2. přeprac. a aktualiz. vyd. Praha: Grada. 2009, 496 s. ISBN 978-80-247-2615-1.

MOLNÁR, Zdeněk. Efektivnost informačních systémů. 2. rozš. vyd. Praha: Ikar, 2000. 178 s. ISBN 80-247-0087-5.

SCHWALBE, Kathy. Řízení projektů v IT. Brno: Computer Press, 2007. 720 s. ISBN 978-80-251-1-26-8.

SODOMKA, Petr a Hana KLČOVÁ. Informační systémy v podnikové praxi. 2. aktualiz. a rozš. vyd. Brno: Computer Press, 2010. 501 s. ISBN 978-80-251-2878-7.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2016/17

V Brně dne 28.2.2017

L. S.

doc. RNDr. Bedřich Půža, CSc. ředitel	doc. Ing. et Ing. Stanislav Škapa, Ph.D. děkan

ABSTRAKT

Táto bakalárska práca je zameraná na návrh prototypu informačného systému, slúžiaceho na evidenciu zbierok minerálov. Jeho hlavným cieľom je navrhnúť aplikáciu na ktorej sa bude dať vyskúšať koncept evidencie zbierok online.

ABSTRACT

This bachelor thesis is focused on prototype proposal of information system focused on cataloging mineral collections. Its main objective is to design application on which it will be possible to try concept of catalogizing collections online.

KĽÚČOVÉ SLOVÁ

Informačný systém, katalóg, databáza, php, laravel, javascript, angularjs

KEYWORDS

Information system, catalogue, database, php, laravel, javascript, angularjs

BIBLIOGRAFICKÁ CITÁCIA

STANKOVIČ, E. Návrh informačního systému. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2017. 63 s. Vedoucí bakalářské práce doc. Ing. Miloš Koch, CSc..

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 31. května 2017 .....

podpis studenta

POĎAKOVANIE

Veľmi rád by som poďakoval vedúcemu mojej bakalárskej práce doc. Ing. Milošovi Kochovi, CSc., za jeho užitočné rady a pripomienky pri spracovaní tejto bakalárskej práce.

OBSAH

ÚVOD

V minulosti sa ľudia pri vedení rôznych katalógov spoliehali na tlačené kópie a neskôr na kópie súborov. To však prinášalo nevýhody spojené s ich správou. Neúplné údaje, nutnosť spojovať rôzne súbory a záznamy dokopy a podobne.

V dnešnej dobe, keď je internet súčasťou našich životov a jeho dostupnosť je na vysokej úrovni, preto je práve toto médium tým ideálnym médiom na evidenciu rôznych katalógov. Tieto katalógy sú dostupné z celého sveta, kde je prístup na internet a človek sa už nemusí viac starať, či si zbalil svoj poznámkový blok alebo či si skopíroval aktuálny súbor s evidenciou.

V tejto bakalárskej práci sa zameriam na návrh prototypu takéhoto online katalógu, ktorý by mal zberateľom minerálov pomôcť s evidenciou a zdieľaním ich zbierky.

CIEĽ PRÁCE

Cieľom tejto bakalárskej práce bude navrhnúť prototyp informačného systému na evidenciu zbierok minerálov. Navrhnutý systém by mal slúžiť na overenie konceptu spravovania zbierok minerálov prostredníctvom internetu. Systém by mal umožňovať jednoduchšiu evidenciu zbierok, mal by byť jednoduchý na použitie a mal by užívateľovi dávať možnosť prezentovať svoju zbierku kdekoľvek vo svete, za použitia moderných mobilných alebo desktopových zariadení s prístupom na internet.

Celý systém bude vytvorený za použitia moderných webových technológií. Serverová časť bude vytvorená v jazyku PHP za použitia frameworku Laravel. Aplikácia, ktorú bude obsluhovať užívateľ, tzv. frontend, bude vytvorený v jazykoch JavaScript, CSS a HTML, a to za použitia frameworku AngularJS a Material Design for AngularJS.

1 TEORETICKÉ VÝCHODISKÁ

V tejto kapitole vymedzím znalosti potrebné na vytvorenie prototypu aplikácie. Predovšetkým sa zameriam na použité technológie. Taktiež priblížim niektoré vybrané návrhové vzory.

1.1 Použité technológie

V tejto časti popíšem použité technológie a postupy, ktoré som použil pri návrhu prototypu.

1.1.1 HTML

HTML je hlavným značkovacím jazykom používaným na prenos informácií prostredníctvom webu. Pôvodne bol tento jazyk vyvinutý za účelom zobrazovania vedeckých dokumentov, avšak jeho štruktúra mu dovolila sa vyvinúť do dnešnej podoby. [1]

V posledných rokoch prešiel jazyk HTML veľkou zmenou, keď bola vydaná jeho nová verzia HTML5. Táto verzia zo sebou prinášala mnoho nových značkovacích prvkov ako taktiež aj špecifikáciu, ako sa majú prehliadače správať k zle napísaným stránkam. [2]

Medzi novinky, ktoré verzia 5 priniesla partia napríklad: [2]

- Validácia formulárov
- CANVAS a SVG tagy pre uľahčenie práce s grafickými prvkami
- Tagy AUDIO a VIDEO pre natívnu podporu prehrávania multimediálneho obsahu, bez nutnosti pluginov tretích strán
- Nové typy tagu INPUT definujúce jeho rôzne hodnoty a ich validáciu

V minulosti po zverejnení HTML5 bola jeho podpora slabá a rôzne prehliadače si ju interpretovali inak, v dnešnej dobe je HTML5 podporovaný v plnom rozsahu na všetkých moderných verziách prehliadačov [3]

1.1.2 CSS

CSS je štýlovací jazyk slúžiaci na opísanie zobrazovania dokumentov napísaných v jazykoch HTML a CSS. Jazyk opisuje ako sa majú dané elementy zobraziť na obrazovke, papieri, čítačkách obrazoviek, či na iných médiách. [4]

V súčasnosti používanými verziami sú CSS2 a CSS 3, kde CSS3 je spätne kompatibilná s verziou 2. Vo verzii CSS3 prišlo k predstaveniu mnohých nových funkcií, z ktorých väčšina je implementovaná vo všetkých moderných prehliadačoch. [5]

1.1.2.1 Responzívny design

So zvyšujúcou sa popularitou mobilných zariadení umožňujúcich prístup na internet, bolo nutné obsah webových stránok prispôsobiť týmto zariadeniam. Bolo toho dosiahnutého pomocou operátora media-query. [6]

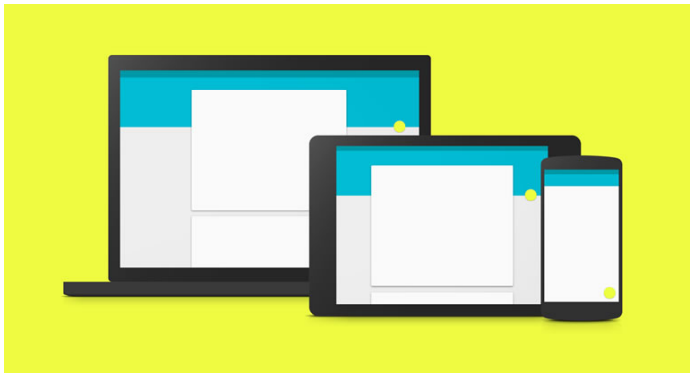
1.1.2.2 Frameworky

Nové funkcie v CSS3 a ich dobrá dokumentácia umožnili vznik takzvaných CSS frameworkov. Jedná sa o sadu predpripravených štýlov pre určité HTML elementy a taktiež rôzne systémy zobrazovania dát. [7]

Medzi najpoužívanejšie frameworky patrí Twitter Bootstrap, Pure CSS, Gumby a Semantic UI framework. [8]

1.1.3 Material design

Material design je jazyk pre tvorbu designov. Bol vytvorený spoločnosťou Google pre ich novú verziu operačného systému Google Android a to v lete roku 2014. Napriek tomu, že špecifikácia tohto jazyka sa zameriava predovšetkým na mobilné dotykové zariadenia, je ho možné použiť a pre vývoj webov. [9]



Obr. 1: Príklad material designu a jehoadaptácia skrz rôzne zariadenia [9]

1.1.4 JavaScript

JavaScript je najpopulárnejší jazyk vo webovom a open source ekosystéme. Podľa octoverse.github.com, ktorý svoje dáta čerpá z verzovacej platformy github.com, je prvý v počte otvorených pull requestov. [10]

Jazyk JavaScript vznikol v roku 1995 pod záštitou vtedy najpopulárnejšieho prehliadača Netspace Navigator. Prvým pomenovaním jazyka bolo Mocha, neskôr toho istého roku, sa Netscape rozhodol zmeniť tento názov na LiveScript. Jazyk mal mať syntax podobnú vtedy populárnemu jazyku Java. Vďaka licencovaní obchodnej značky Java od spoločnosti Sun, spoločnosti Netscape, mohol byť jazyk LiveScript premenovaný na JavaScript. [11, str. 2]

V roku 1997 bol organizáciou ECMA štandardizovaný pre obecné použitie, čím sa JavaScript stal nadstavbou jazyka ECMA so zameraním na webové prehliadače. Od svojho vzniku prešiel mnohými zmenami a jeho súčasná podoba je postavená na ECMA 2016. [13, str. 2]

1.1.5 NodeJs

Node.js je asynchrónny, udalosťami poháňaný JavaScriptový runtime. Je navrhnutý pre vytváranie škálovateľných aplikácii. [12]

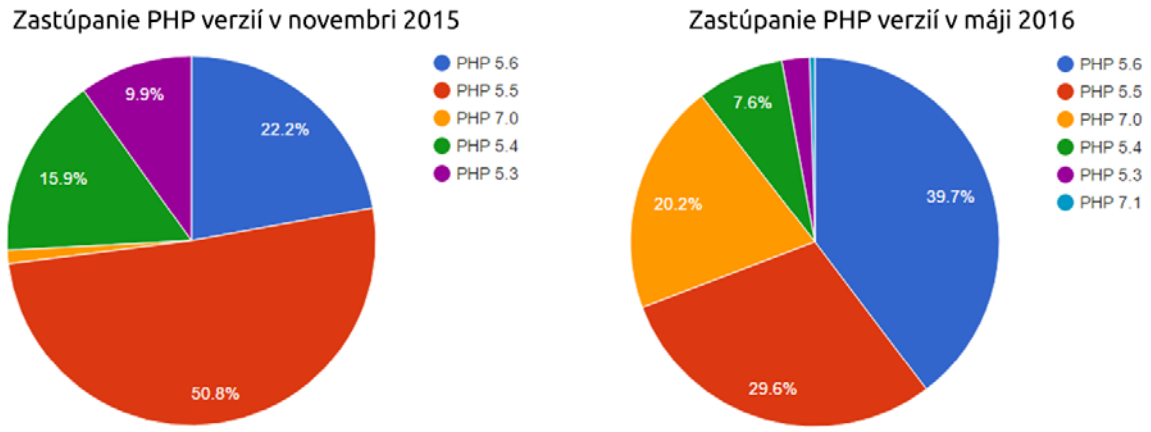
1.1.6 PHP

PHP je interpretovaný, skriptovací jazyk slúžiaci na tvorbu webových stránok s dynamickým obsahom.

Jazyk PHP ako ho poznáme dnes bol odvodený od jazyka PHP/FI vytvoreného v roku 1994 Rasmusom Lerdorfom. Dôvodom pre jeho vytvorenie bolo sledovanie návštev v jeho online resumé. V roku 1995 bol kód jeho balíka nazývaného PHP Tools zverejnený, čo umožnilo rôznym vývojárom opravovať chyby a zdokonaľovať funkcionalitu. [14]

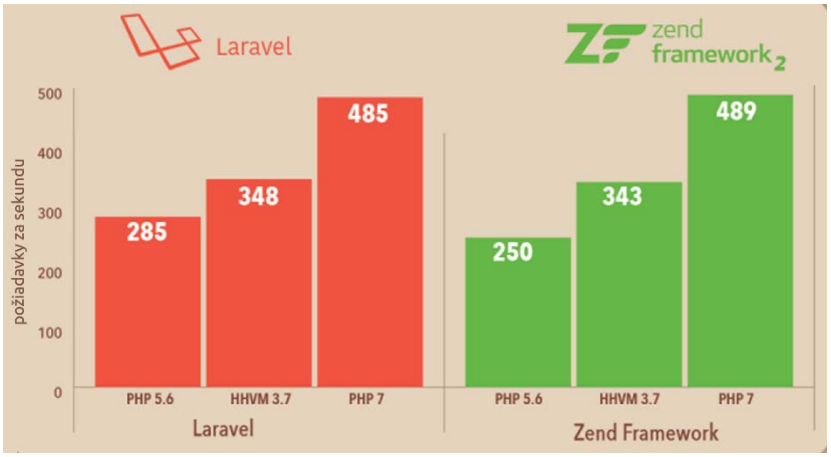
PHP 3 bola prvá verzia pripomínajúca súčasný stav jazyka. Táto verzia tiež predstavila základy Objektovo orientovaného programovania. Nástupcom bola verzia PHP 4 vydaná v roku 1999. Táto verzia bola postavená na Zend Engine. Priniesla nové vlastnosti ako napríklad http session, možnosť podpory viacerých webových serverov, output buffering a bezpečnejší prenos a spracovanie užívateľských vstupov. [14]

Po dlhom vývoji bolo v roku 2004 vydané PHP5. Implementovalo novú verziu ich jadra Zend Engine 2 s novým objektovým modelom, umožňujúcim písať moderné aplikácie. [14]



Obr. 2: Využitie jednotlivých verzií PHP v roku 2015 a 2016 (preložené z [34])

Poslednou hlavnou verziou jazyka je PHP7, ktoré prináša niekoľko zaujímavých vlastností. Hlavnou z nich je prechod na nový engine nazývaný PHP#NG, ktorý zdvojnásobil rýchlosť spracovania skriptov. [15]



Obr. 3: Porovnanie výkonu rôznych verzií PHP v dvoch rôznych frameworkoch (preložené z [35])

Ďalšou veľkou novinkou bolo implementovanie deklarácie typov. PHP vo svojich predchádzajúcich verziách nepodporovalo deklarovanie typov premenných a návratových hodnôt funkcií. To viedlo často k rôznym chybám, ktoré sa ukázali až za runtimeu. [15]

```php
function addOld($a, $b) {
    return $a + $b;
}
echo addOld(5, 'a');
```

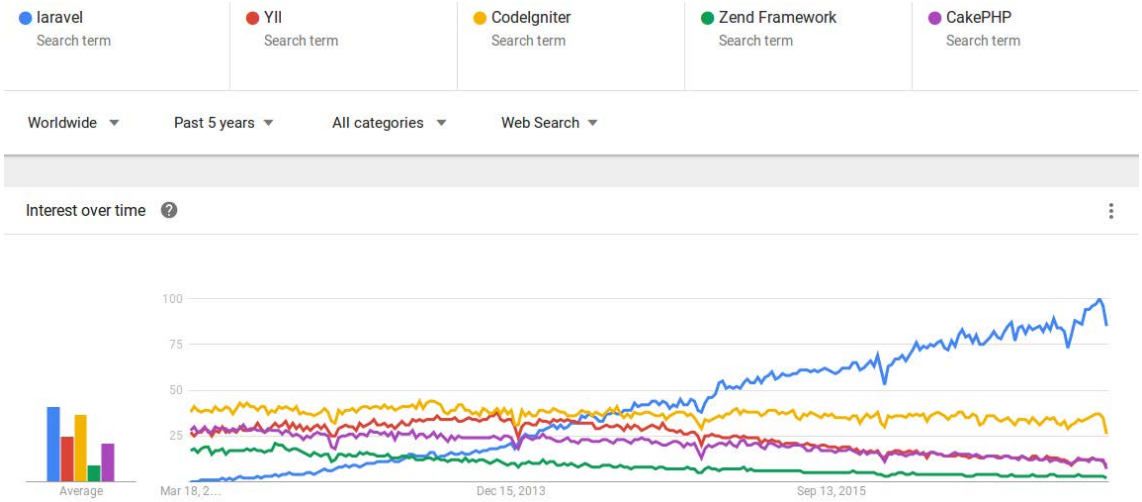
```php
function add(int $a, int $b) {
    return $a + $b;
}
echo add(5, 'a');
```

Obr. 4: ukážka rozdielu medzi zápisom funkcie v php 5.x a php 7.x [vlastné spracovanie]

Na obr. 4 vidíme kód zo starého PHP vľavo a vpravo kód PHP 7. Kód vľavo sa spustí bez problému avšak vráti hodnotu 5. Kód vpravo vyhodí chybovú hlášku o nedodržaní dátového typu parametru.

1.1.7 Laravel

Laravel je najpopulárnejší PHP framework. Medzi jeho hlavné stránky patrí jednoduchá syntax, ORM (Objektovo relačné mapovanie) Eloquent a testovateľnosť. Taktiež medzi hlavné dôvody jeho prvenstvo patri aj obrovská komunita a ekosystém vybudovaný okolo frameworku. [18] [19]



Obr. 5: Trend obľúbenosti Laravelu a iných frameworkov [18]

Prvá verzia frameworku Laravel bola vydaná v roku 2011. Medzi jej vlastnosti patrila vlastná implementácia ORM nazvaného Eloquent, routovanie, modulový systém pre rozšírenia, validácia, autentizácia. [16, str. 24]

Vývoj frameworku sa pohyboval veľmi rýchlo keď v roku 2012 bola vydaná verzia 3. Táto verzia predstavila koncept controllerov, testovania IoC kontajneru, databázových migrácií a relácií medzi jednotlivými objektami v ORM. Taktiež bol predstavený CLI modul, umožňujúci generovanie kódu čím urýchlil vývoj. [16, str. 24]

Laravel 4 bol veľkou zmenou oproti predchádzajúcim verziám. Predstavil novú štruktúru, implementoval sa do Composer ekosystému, čo je manažér balíkov, a knižníc pre PHP. Boli predstavené nové vlastnosti ako Queues, Mail komponenta, a plnenie databázy, tzv database seeding. [16, str. 25]

Posledná hlavná verzia Laravel 5 bola vydaná vo februári 2015. Scheduler bol jednou z noviniek. Umožňoval zjednodušený zápis CRON úloh. Ďalšími novinkami bola podpora naviazanosti tried na tzv. interfaces, podpora spracovania formulárových requestov a podpora dotenv balíka, pre ľahšiu správu vývojárskych a produkčných prostredí. [16, str. 25]

1.1.7.1 Eloquent ORM

Eloquent ORM poskytuje elegantný a jednoduchý spôsob ako komunikovať s databázou. Každá databázová tabuľka je v ňom reprezentovaná vlastným modelom, čo je vlastne trieda rozširujúca základnú triedu Eloquentu.

Model dovoľuje, získavanie, vytváranie mazanie a upravovanie záznamov v databáze. Taktiež podporuje možnosť nastaviť relácie medzi objektami a uľahčiť tak získavanie údajov medzi viacerými tabuľkami a to bez napísania čo len riadku SQL kódu. [33]

1.1.7.2 Elixir

Elixir bol novinkou predstavenou vo verzii Laravel 5. Poskytuje čisté a rozumné rozhranie pre kompiláciu rôznych CSS a JS preprocesorov, čo umožňuje jednoduchú správu štýlov a skriptov v projekte. [17]

1.1.8 AnguarJS

AngularJS je framework pre dynamické webové aplikácie. Dovoľuje používať jazyk HTML a následne ho rozšíriť o dynamickú funkcionalitu. Angular disponuje tzv. Dependency injection a obojsmerným previazaním dát, tzv. two-way databindingom. Čo redukuje množstvo kódu potrebného pre vytvorenie podobného správania. Funkcionalitu rôznych elementov rozširuje pomocou direktív, čo je vlastne vlastný HTML element previazaný s určitým JavaScript kódom. [20]

Medzi kľúčové vlastnosti frameworku patrí

- Obojsmerné previazanie dát
- Jednoduchá možnosť pre cyklenie určitých častí HTML dokumentu
- Podpora pre validáciu formulárov
- Pridávanie nového správanie pre HTML elementy

1.1.9 Redis

Redis je open source riešenie. Je to vlastne in-memory sklad (dáta sú uložené v pamäti a po vypnutí procesu sa stratia), slúžiaci ako databáza, cache a predavač správ. Podporuje rôzne datové štruktúry od stringov, cez hashe, listy až po bitmapy. [21]

1.1.10 Socket.io

Socket IO je knižnica umožňujúca komunikáciu v reálnom čase. Skladá sa zo servera bežiacom pod NodeJS a z Javascriptovej knižnice vloženej v HTML stránke. Ako transportnú vrstvu využíva spojenie cez tzv. web sokety, ku ktorým pridáva špeciálne metadata slúžiace na identifikáciu paketov a prenášaných správ. [22]

Soket.io je používaný hlavne na výmenu údajov medzi aplikáciou a serverom . Obvyklým príkladom využitia tejto technológie je chat. V implementácii chatu pomocou PHP sa prehliadač v určitom časovom intervale pýta serveru, aké zmeny boli vykonané od posledného requestu. Následne tieto zmeny premietne do klientskej aplikácie. Tento prístup si vyžaduje obrovské množstvo requestov, čo vedie k zaťaženiu servera a nižšej rýchlosti komunikácie. [23]

Pri použití SocketIO sa nadviaže spojenie, ktoré je udržiavané. Počas tohoto spojenia si môže server a klient vymieňať informácie. Napríklad pri použití SocketIO v chate, klient sa pripojí na kanál, následne odošle správu. Táto správa sa spracuje serverom, a ten upozorní všetkých pripojených klientov o tejto správe. [23]

1.1.11 Rest API

V roku 1993 spoluzakladateľ webového servera Apache http, Roy Fielding, sa začal zaoberať problémom škálovateľnosti webových aplikácii. Vymysleli 6 kategórií podmienok, ktoré mali pomôcť riešiť tento problém. [24, str. 2]

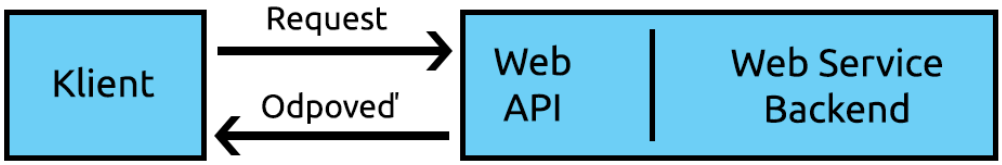
Medzi kategórie patrili:

- Client-server architektúra
- Jednotné rozhranie
- Vrstevnatosť
- Cache

- Bez-stavovosť
- Code on Demand

V roku 2000, Fielding vymenoval a popísal Webové návrhové štýly vo svojej dizertačnej práci nazvanej Representional State Transfer (REST). Táto práca vychádzala z kategórií, ktoré definoval v 1993. [24, str. 5]

Webové služby poskytujú svoje funkcie a dáta externým aplikáciám pomocou aplikačného rozhrania (API) a dovoľujú im si navzájom vymieňať dáta. API je vlastne tvárou alebo prostredníkom medzi webovým serverom a užívateľskou aplikáciou. [24, str. 5].



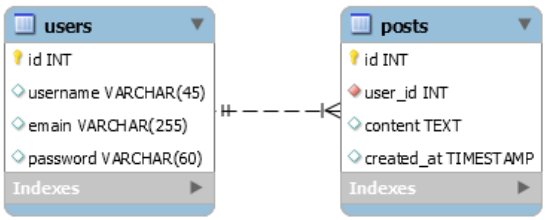
Obr. 6: Diagram zobrazujúci komunikáciu medzi klientom a serverom (vlastné spracovanie podľa [24, str. 6])

Návrh REST API je dnes bežným spôsobom implementácie API rozhraní v moderných službách. Medzi najpoužívanejšie patrí napríklad Google API, Twitter API alebo Amazon API. [25]

1.1.12 MySQL

MySQL je najpopulárnejšie open source SQL databázové riešenie. Za jeho vývoj a podporu zodpovedá spoločnosť Oracle. [26]

MySQL databáza ukladá údaje v oddelených tabuľkách. V týchto tabuľkách sú uložené údaje medzi ktorými je následne možné vytvárať tzv. relácie ako napríklad 1:1 či 1:N. [26]



Obr. 7: ER diagram väzby 1:N [vlastné spracovanie]

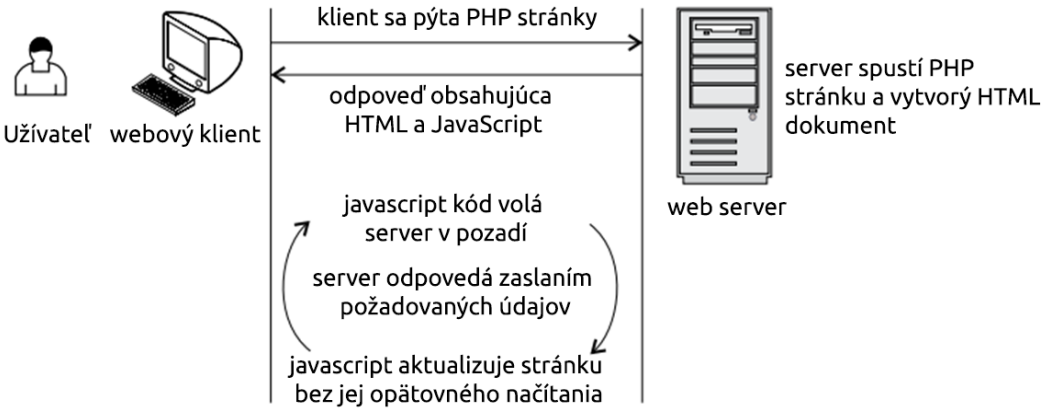
Na obr. 7 je možné vidieť schému pozostávajúcu z dvoch tabuliek, tabuľky users a posts. V tabuľke users sa uschovávajú údaje o užívateľoch ako napríklad meno a email. Taktiež je ku každému záznamu vygenerované unikátne ID, ktoré nám pomáha identifikovať užívateľa.

V tabuľke posts sa nachádzajú údaje o správe, ktorú užívateľ napísal a dátum kedy ju napísal a unikátne ID správy. Na to aby bolo možné identifikovať, ktorú správu ktorý užívateľ napísal je potrebné to tabuľky posts vložiť ďalší údaj, a to user_id, obsahujúce id autora z tabuľky users.

Jednoduchým SQL príkazom `SELECT * FROM posts JOIN users ON users.id = posts.user_id;` vieme vybrať všetky správy s ich príslušným autorom. Tento prístup nám umožňuje v databáze jednoducho vytvárať zložitejšie DB štruktúry a pristupovať k dátam spojeným reláciami.

1.1.13 Ajax

Ajax je moderný nástroj používaný na vytváranie užívateľsky prívetivých webových stránok. AJAX je skratkou pre Asynchronous Javascript and XML, s dôrazom na slovo asynchronous, čo znamená asynchrónny. V podstate sa jedná o techniku umožňujúci volať server na pozadí a získať dodatočné údaje či meniť údaje a to bez nutnosti znovu načítať zobrazený obsah v prehliadači. [27, str. 8][27, str. 14]



Obr. 8: Schéma komunikácie za použitia technológie Ajax (preložené z [27, str. 14])

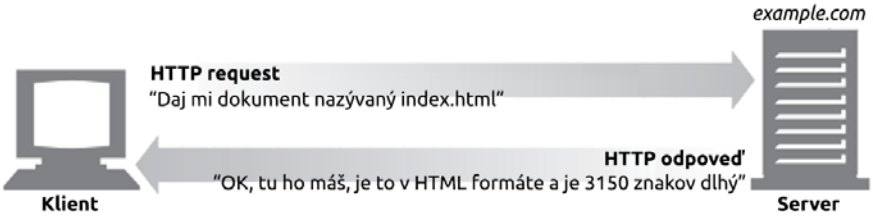
Vďaka tejto vlastnosti a mnohým ďalším je Ajax vhodnou technológiou na komunikáciu webového klienta s REST API bežiacej na serveri. [28]

1.1.14 HTTP server

Obsah webových stránok je uložený na webových serveroch. K tomuto obsahu sa dá pristupovať pomocou HTTP protokolu. Keď užívateľ vo webovom prehliadači navštívi stránku example.com, pošle sa http request.

```
GET /index.html HTTP/1.1
Host: example.com
```

Následne sa server pokúsi nájsť daný súbor (v tomto prípade index.html). V prípade, že je úspešný pošle daný súbor späť prehliadaču pomocou http odpoveď spolu s ďalšími metadátami ako napríklad typ obsahu, dĺžka a podobne. [29, str. 4]



Obr. 9: Diagram znázorňujúci komunikáciu medzi klientom a serverom (preložené z [29, str. 4])

1.1.14.1 Apache

Apache http server bol vytvorený Robertom McCoolom v roku 1995 a neskôr bol vyvíjaný pod dohľadom Apache Software Foundation. Jedná sa o najpopulárnejší webový server. [30]

Apache dokáže spracovávať ak statický obsah tak aj dynamický obsah a to bez použitia externých komponent. Spolieha sa na interné moduly, ktoré sa dajú jednoducho zakázať, či povoliť. [30]

1.1.14.2 Nginx

Nginx bol vyvíjaný Igorom Sysoevom od roku 2002, zverejnený bol v roku 2004. Jeho hlavnou úlohou bolo riešiť problém tisícov súbežných spojení na server. Od tej doby tento webový server získal veľkú priazeň vďaka svojej jednoduchosti a možností škálovania. [30]

Nginx na rozdiel od Apache nemá možnosť spúšťať dynamický obsah. Na spracovanie rôznych requestov, musí spúšťať daného interpreta, ktorý sa postará o spracovanie requestu a pošle obsah späť Nginx serveru. Tento prístup má veľkú výhodu a to v tom,

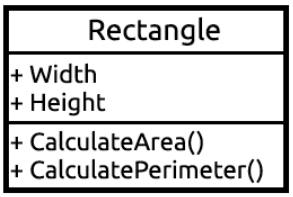
že dokáže statický obsah poskytovať priamo, bez nutnosti spúšťať žiadne ďalšie služby a moduly.[30]

1.1.15 OOP

OOP známe ako Objektovo Orientované Programovanie je spôsob myslenia a návrhu aplikácií. Jeho úlohou je maximalizovať znovu-použiteľnosť napísaného kódu a tým minimalizovať náklady na údržbu zdrojových kódov. [31, str. V]

Objektovo orientované programovanie vychádza z toho, že každý program je simuláciou skutočného alebo virtuálneho sveta. Napríklad účtovnícky program simuluje správanie spoločnosti a jej zákazníkov vzhľadom na obstarávanie objednávok, zásob, účtovania a platieb. [32, str. 10]

Obr. 10: Vizuálna reprezentácia objektu rectangle (vlastné spracovanie [32, str. 4])



Obr. 11: UML diagram objektu rectangle (vlastné spracovanie [32, str. 9])

Na obr. 10 je vidieť 2D útvar, obdĺžnik a jeho jednotlivé vlastnosti. Na obr. 11 je vidieť UML diagram obsahujúci jeho vlastnosti a dve metódy, ktoré počítajú jeho obsah a obvod.

```
<?php
Class Rectangle {

        private $height;
        private $width;
```

```
        public function __construct(float $height, float $width){
                $this.height = $height;
                $this.width = $width;
        }

        public function calculateArea(): float{
                return $this.height * $this.width;
        }

        public function calcualtePerimeter(): float {
                return ($this.height + $this.width) * 2;
        }
}
$rectangle = new Rectangle(10, 20);
echo $rectangle->calculateArea();
```

Na ukážke kódu je možné vidieť konkrétnu implementáciu triedy Rectangle z UML diagramu (obr. 11) v jazyku PHP. Danú triedu je následne nutné inštancovať príkazom `$rectangle = new Rectangle(10, 20);` a následne vieme na vzniknutom objekte volať jeho metódy a atribúty. Napríklad príkaz `echo $rectangle->calculateArea();` vypíše hodnotu 200.

2 ANALÝZA

V tejto časti prestavím produkt a budem sa venovať analýze požiadaviek klienta na funkcionalitu informačného systému. Taktiež sa budem venovať analýze konkurenčných projektov.

2.1 Predstavenie produktu

V dnešnej dobe existuje mnoho online informačných systémov, ktoré svojou funkcionalitou pomáhajú bežným ľudom uľahčovať prekážky, s ktorými sa či už v bežnom živote alebo v ich práci stretávajú. Minsets.net je SaaS (stránka ako služba) aplikácia, ktorá uľahčuje katalogizáciu zbierok minerálov bežným zberateľom ale aj dealerom. Dealeri často oslovujú svojich potencionálnych zákazníkov na rôznych mineralogických burzách či výstavách, Minsets.net chce ich pôsobenie preniesť aj na internet, kde prostredníctvom rozhrania webových stránok môžu pohodlne spravovať svoju katalóg a s minimálnou námahou zdieľať svoje minerály, či už priamym odkazom, alebo cez HTML kód vložený na ich webové stránky.

2.2 Popis funkcionality

V tejto časti popíšem funkcionalitu, ktorú má výsledný prototyp informačného systému spĺňať.

2.2.1 Vzorka

Vzorka bude základným prvkom v informačnom systéme. Ku každej vzorke bude možné nahrať neobmedzený počet fotografií a z nich vybrať jednu, ktorá ju reprezentuje. Každá vzorka bude obsahovať názov, vlastný identifikačný reťazec volený užívateľom lokalitu a obsahujúce minerály.

2.2.2 Hlavný katalóg

V hlavným katalógu budú zobrazené všetky vzorky minerálov uložené v databáze daného užívateľa. Tieto údaje budú prezentované dvoma spôsobmi, image view a table view.

2.2.2.1 Image view

V tomto zobrazení sa bude klásť dôraz na zobrazenie hlavného obrázku danej vzorky. Toto zobrazenie bude reprezentované dlaždicami obsahujúcimi obrázok, identifikačné číslo vzorky, názov vzorky a lokalitu, z kadiaľ daná vzorka pochádza.

2.2.2.2 Table view

Ako už z názvu vyplýva table view bude reprezentovaný tabuľkou. Toto zobrazenie bude klásť dôraz na zobrazované údaje, čo bude umožňovať jednoduchšiu orientáciu vo veľkom množstve vzoriek. V tomto zobrazení bude taktiež prítomný aj náhľad hlavného obrázku vzorky.

2.2.3 Sety

Úlohou tejto funkcionality bude zoskupovanie vzoriek minerálov, či už za účelom katalogizácie alebo za účelom zdieľania týchto setov. Jednotlivé sety by mali byť reprezentované image viewom, kde bude možné zvoliť vzorku reprezentujúcu danú skupinu. Každá skupina bude mať možnosť byť zdieľaná a to prostredníctvom vygenerovaného odkazu. Pre každú zdieľanú skupinu bude možné nastaviť, ktoré atribúty priradených vzoriek budú môcť byť zdieľané.

2.2.4 Storage

Úlohou tejto funkcionality bude uľahčiť evidenciu vzoriek vo vzťahu k miestu ich uskladnenia. Storage by mal mať stromovú štruktúru. V tejto štruktúre bude možné každý storage voľne presúvať a s ním aj všetky priradené vzorky. Taktiež by mal umožňovať jednoducho vidieť všetky vzorky uložené v danom storage.

2.2.5 Lokality

Každá vzorka bude mať špeciálny atribút lokalita. Tento atribút bude možné naplniť akýmkoľvek údajom. Taktiež by mal obsahovať našepkávanie lokalít z databázy webu mindat.org.

2.2.6 Vlastné polia atribútov

Každá vzorka bude obsahovať preddefinované atribúty ako napríklad, názov, minerály, lokalita, popis či poznámka. Správca systému by mal mať možnosť na požiadanie

špecifickému užívateľovi vytvoriť ďalšie atribúty. Tieto atribúty by mali byť nasledujúcich typov:

- String
- Price
- GPS
- Number
- Text
- URL
- Boolean
- Dimension
- Weight
- Date
- Time

2.2.7 Filtrovanie podľa atribútov

Informačný systém by mala umožňovať filtrovať vzorky podľa všetkých dostupných atribútov vrátane tých, ktoré boli vytvorené na mieru pre užívateľa. Filter by sa mal týkať iba vzoriek a nie storage a skupín, a mal byť aktívny na všetkých obrazovkách, až po dobu, keď ho užívateľ automatický neresetuje.

2.2.8 Import

Keďže sa jedná o katalóg, medzi jeho funkcionalitu by mal patriť import CSV súborov, obsahujúcich potrebné údaje.

2.2.9 Užívateľské rozhranie

Užívateľské rozhranie má byť intuitívne pre užívateľa. Malo by byť rozdelené do logických celkov riešiacich jednotlivú problematiku ako napríklad hlavný katalóg, sety, storage a podobne. Keďže v dnešnej dobe sa na web pristupuje väčšinou z mobilných zariadení, je nevyhnutné aby web bol prispôsobený týmto zariadeniam a podporoval všetky moderné webové prehliadače.

2.3 Analýza konkurencie

V tejto časti priblížim konkurenčné projekty s podobným, alebo rovnakým zameraním.

2.3.1 Colido.de

Colido.de je nemecký startup zameriavací sa na zberateľov minerálov. Medzi značné výhody ich riešenia patrí profesionálne navrhnutý design a dlhodobé pôsobenie. Ďalšou výhodou je ich spolupráca s najväčším mineralogickým časopisom v Európe, s časopisom Lapis od ktorého čerpajú aktuálnu databázu minerálov. Nezanedbateľnou výhodou je ich nízka cena za platený plán.

Medzi nevýhody ich riešenia patrí značné obmedzovanie platených funkcií, a to aj po zaplatení prémiového poplatku. O navýšenie rôznych kvót je treba pristupovať individuálne a ich prístup tak nie je transparentný, čo môže odradiť potencionálnych zákazníkov.

2.3.2 Mineraldesk.com

Mineraldesk je projektom so sídlom v Juhoafrickej Republike. Ich riešenie je postavené na flash aplikácii, ktorú je nutné inštalovať. Medzi výhodu ich riešenia patrí možnosť spravovať zbierku v režime offline. Taktiež obsahuje rôzne zložité filtrovacie funkcie. Najväčšou výhodou je priama spolupráca so serverom mindat.org ktorý im poskytuje prístup do najväčšej databázy minerálov a lokalít.

Medzi nevýhody patrí samotný off-line režim, kde synchronizácia je značne zabugovaná. Ďalšou nevýhodou je nutnosť inštalovať dodatočný software na spustenie ich aplikácie. Nezanedbateľnou nevýhodou je aj ich webové rozhranie, ktoré je postavené na platforme Wordpress čo predstavuje možnosť potencionálneho využitia známych exploitov a tým získanie ich databázy.

3 VLASTNÝ NÁVRH RIEŠENIA

Táto kapitola sa zaoberá návrhom prototypu aplikácie, obsahujúcej základné funkcie popísané v predchádzajúcej kapitole.

3.1 Vývojový stack

Pred tým, než začnem s návrhom riešenia, si definujem v akom prostredí bude aplikácia bežať. Na základe toho vyberiem technológie, na ktorých postavím prototyp aplikácie.

3.1.1 Použité webové technológie

Aplikácia sa vyvíja od začiatku, preto nie som ovplyvňovaný žiadnymi faktormi pri výbere použitých technológií.

- Databáza: ako najvhodnejšie a najdostupnejšie riešenie pripadá databáza MariaDB čo je vlastne nadstavbou nad MySQL.
- Backend: aplikácia pobeží na PHP za použitia frameworku Laravel. Na backende bude tiež použitý soketový server na real time komunikáciu so frontendom. Tento server bude napísaný v javascripte a spúšťaný na NodeJS.
- Frontend: Ako frontend framework som zvolil AngularJS, ktorý som volil hlavne kvôli jeho popularite, veľkej komunite a výbornej dokumentácii. Ako podpornú UI knižnicu som zvolil Material Design for AngularJS a Bootstrap 3.

3.1.2 Server

Aplikácia bude hostovaná na vlastnom virtuálnom serveri a bude mať všetky jeho prostriedky vyhradené pre seba. Ako poskytovateľa služieb vyberám Hukot.cz a jeho variantu Normálka, ktorá poskytuje nasledovné parametre:

- CPU XEON E5/E3 / Opteron 6200/6300 (dual core)
- 2 GB RAM
- 40 GB SSD
- 1Gbps pripojenie
- Neobmedzený prenos údajov

Operačný systém nainštalovaný na servery bude Ubuntu 16.04 LTS. Server bude spravovaný nástrojom nazývaným Laravel Forge, ktorý umožňuje správu servera, deployovanie kódu a všetky základné úlohy spojené so správou PHP aplikácií na VPS. Laravel Forge automaticky na server nainštaluje nasledovné podporné aplikácie:

- PHP 7.1
- Nginx
- NodeJS
- MariaDB
- Redis Server

Na uskladňovanie fotografií nepoužijem lokálny disk virtuálneho servera, pretože v budúcnosti by to mohlo priniesť značné komplikácie pri zálohovaní servera. Namiesto toho použijem cloudovú službu AWS S3 Storage prevádzkovanú spoločnosťou Amazon.

3.2 Návrh databázovej schémy

V tejto sekcii vypíšem zoznam entít vystupujúcich v databáze, ich relácií a následne ich atribútov. Nakoniec zostavím ER diagram.

3.2.1 Zoznam entít

Ako prvý krok pri návrhu databázy si určíme, ktoré entity budú v našej databáze vystupovať.

Entita	**Popis**	**Alias**	**Tabuľka**
Užívateľ	Používateľ aplikácie	User	users
Vzorka	Vzorka minerálu patriaca užívateľovi	Specimen	specimens
Fotografia	Fotografia danej vzorky	Photo	photos
Lokalita	Lokalita nálezu vzorky	Locality	localities
Minerál	Minerál vyskytujúci sa na danej vzorke	Mineral	minerals
Set	Kolekcia vzoriek vytvorená užívateľom	SpecimenSet	specimen_sets
Farba	Farba vyskytujúca sa na vzorke	Color	colors
Storage	Úložisko, v ktorom sa daná vzorka nachádza	Storage	storages

Vlastné atribúty	Atribúty vzoriek vytvorené užívateľom	PropertyType	user_property_types
Hodnota atribútu	Hodnoty atribútov vzoriek	SpecimenProperty	specimen_properties
Zdieľaná položka	Položky zdielané užívateľom	SharedItem	Shared_items
Import	Záznam o importe vykonanom užívatelom	Import	imports

Tab. 1: Zoznam entít [vlastné spracovanie]

V zozname entít môžeme vidieť entity Vzorka, Vlastné atribúty a Hodnoty atribútov, ktorých vzťah by som priblížil. Každý užívateľ si vie vytvoriť N atribútov s dopredu definovaným typom. Každá vzorka môže obsahovať tieto atribúty. Hodnoty daných atribútov reprezentuje entita Hodnota atribútov, ktorá je špecifická pre každú vzorku.

3.2.2 Zoznam vzťahov

Po identifikácii entít určím vzťahy, ktoré medzi nimi vznikajú. Každému vzťahu určím o aký typ relácie sa jedná.

Entita	Typ relácie	Vzťah	Entita
User	1:N	má	Specimen
User	1:N	nahral	Photos
User	1:N	vytvoril	SpecimenSet
User	1:N	Spustil	Import
User	1:N	Zdieľa	SharedItem
User	1:N	Definoval	PropertyType
Specimen	1:N	Má	Photo
Specimen	1:N	Nachádza sa v	Storage
Specimen	1:N	Pochádza z	Locality
Specimen	1:N	Má	SpecimenProperty
Specimen	1:N	Bol importovaný v	Import
Specimen	N:M	Má	Mineral
Specimen	N:M	Má	Color
Specimen	N:M	Patrí do	SpecimenSet
PropertyType	1:N	Má	SpecimenProperty
SpecimenSet	N:M	Môže zdieľať	PropertyType

Tab. 2: Zoznam vzťahov [vlastné spracovanie]

V tabuľke (Tab. 2) je možné vidieť niekoľko relácií N:M. Tieto relácie sa v databáze nedajú priamo vytvoriť a musia byť rozložené na relácie 1:N. Toho dosiahneme vytvorením pomocnej tabuľky medzi dvoma entitami, ktorá bude obsahovať 2x cudzí kľúč, každý prepojený s primárnym kľúčom tabuliek entít, ktorých sa daná relácia týka.

Napríklad pri relácii Specimen <> Color vznikne tzv. pivot table (spojovacia tabuľka) s názvom color_specimen. Ďalšie pivot tabuľky budem vytvárať podľa nasledujúceho pravidla: Názov tabuliek zmením na jednotné číslo, zoradím podľa abecedy a spojím ich pomocou podtržítka. Jedinou tabuľkou, ktorej sa to nebude týkať bude tabuľka medzi reláciou SpecimenSet <> PropertyType a to hlavne kvôli dĺžke názvu tabuľky a nedostatočnej reprezentácii názvu relácie.

Entita 1	Entita 2	Pivot tabuľka
Specimen	Mineral	mineral_specimen
Specimen	Color	color_specimen
Specimen	SpecimenSet	specimen_set_specimen
SpecimenSet	PropertyType	specimen_set_shared_properties

Tab. 3: Zoznam pivotných tabuliek [vlastné spracovanie]

3.2.3 Atribúty tabuliek

V tejto kapitole opíšem všetky tabuľky a to, aké údaje sa v nich budú nachádzať. V každej tabuľke sa bude nachádzať atribút id, ktorý v tabuľkách nižšie nebude zobrazený. Jeho účelom je unikátna identifikácia záznamov. Bude sa jednať o AUTO INCREMENT pole, o ktorého hodnoty sa bude starať databáza samotná.

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
name	Celé meno užívateľa	STRING	255	Nie
email	Emailová adresa	STRING	255	Nie
password	Hashované heslo užívateľa	STIRNG	60	Nie
aws_region	Region kde sa majú ukladať fotografie	STRING	15	Nie

Tab. 4: Zoznam atribútov tabuľky users [vlastné spracovanie]

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
user_id	ID užívateľa ktorý vlastní vzorku	INT	10	Nie
locality_id	ID lokality z ktorej vzorka pochádza	INT	10	Áno
storage_id	ID skladu v pod ktorý je vzorka priradená	INT	10	Áno
import_id	ID importu, v ktorom bola vzorka importovaná	INT	10	Áno
title	Názov vrozky	STRING	255	Áno
custom_id	Vlastný identifikátor vzorky	STRING	255	Áno

Tab. 5: Zoznam atribútov tabuľky specimens [vlastné spracovanie]

Názov	Popis	Typ	Dĺžka	Null
user_id	ID užívateľa ktorý nahral fotografiu	INT	10	Nie
specimen_id	ID Vzorku ku ktorému je fotografia priradená	INT	10	Áno
region	ASW región, na ktorom je fotografia nahraná	STRING	15	Nie
file_name	Vygenerovaný názov súboru, uloženého v AWS S3	STRING	255	Nie
file_size	Veľkosť súboru v kB	INT	10	Nie
is_main	Určuje, či je fotografia hlavnou fotografiou vzorky	INT	1	Áno
original_file_name	Pôvodný názov fotografie pred nahraním na server	STRING	255	Nie
original_width	Pôvodná šírka obrázku v pixeloch	INT	5	Nie
original_height	Pôvodná výška obrázku v pixeloch	INT	5	Nie

Tab. 6: Zoznam atribútov tabuľky photos [vlastné spracovanie]

Názov	Popis	Typ	Dĺžka	Null
user_id	ID užívateľa ktorý vytvoril set	INT	10	Nie
name	Názov setu	STRING	255	Nie
description	Popis setu	TEXT		Áno
is_shared	Informuje, či je set zdieľaný	INT	1	Áno

Tab. 7: Zoznam atribútov tabuľky specimen_sets [vlastné spracovanie]

Názov	Popis	Typ	Dĺžka	Null
specimen_set_id	ID setu v ktorom sa vzorka nachádza	INT	10	Nie
specimen_id	ID vzorky	INT	10	Nie

Tab. 8: Zoznam atribútov tabuľky specimen_set_specimen [vlastné spracovanie]

Názov	Popis	Typ	Dĺžka	Null
name	Názov farby	STRING	40	Nie
hex_value	HEX hodnota farby	STRING	6	Nie

Tab. 9: Zoznam atribútov tabuľky colors [vlastné spracovanie]

Názov	Popis	Typ	Dĺžka	Null
color_id	ID farby	INT	10	Nie
specimen_id	ID vzorky	INT	10	Nie

Tab. 10: Zoznam atribútov tabuľky color_specimen [vlastné spracovanie]

Názov	Popis	Typ	Dĺžka	Null
user_id	ID užívateľa, ktorý pridal lokalitu	INT	10	Áno
parent_id	ID rodičovskej lokality	INT	10	Áno
name	Názov lokality	STRING	255	Nie
full_path	Cesta rodičovských elementov	STRING	255	Áno
level	Level vnorenia záznamu	INT	4	Áno

mindat_id	ID lokality odpovedajúcej v databáze mindat.org	INT	10	Áno
gps_long	GPS dĺžka	DOUBLE	13,10	Áno
gps_lat	GPS šírka	DOUBLE	13,10	Áno
approved	Určuje, či je lokalita oficiálne uvedená v mindat.org databáze	INT	1	Nie

Tab. 11: Zoznam atribútov tabuľky localities [vlastné spracovanie]

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
name	Názov minerálu	STRING	255	Nie
formula	Chemický vzorec minerálu	TEXT		Áno
mindat_id	ID odpovedajúce záznamu v databáze mindat.org	INT	10	Áno
approved	Určuje, či je minerál oficiálne potvrdeným minerálom	INT	1	Nie

Tab. 12: Zoznam atribútov tabuľky minerals [vlastné spracovanie]

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
mineral_id	ID minerálu	INT	10	Nie
specimen_id	ID vzorky	INT	10	Nie

Tab. 13: Zoznam atribútov tabuľky mineral_specimen [vlastné spracovanie]

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
parent_id	Id rodiča	INT	10	Áno
user_id	ID užívateľa, ktorému patrí daná skladovacia jednotka	INT	10	Nie
lidx	Ľavý index	INT	10	Nie
ridx	Pravý index	INT	10	Nie
depth	Hĺbka vnorenia daného záznamu	INT	4	Nie
name	Názov skladovacej jednotky	STRING	255	Nie
specimen_count	Počet vzoriek v danej skladovacej jednotke	INT	10	Nie

Tab. 14: Zoznam atribútov tabuľky storages [vlastné spracovanie]

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
user_id	ID užívateľa, ktorý vytvoril import	INT	10	Nie
original_file	Názov originálneho, užívateľom nahraného CSV súboru	STRING	255	Nie
saved_file	Cesta k užívateľom nahranému originálnímu CSV súboru na serveri	STRING	255	Nie
chunk_size	Množstvo riadkov CSV súboru spracovaných v jednej iterácii	INT	2	Nie
estimated_rows	Počet záznamov k importu	INT	5	Nie
iterations_done	Počet už ukončených iterácií nad daným importom	INT	5	Nie

iterations	Celkový počet iterácií potrených na spracovanie daného súboru	INT	5	Nie
is_done	Reprezentuje, či daný import bol už dokončený	INT	1	Nie

Tab. 15: Zoznam atribútov tabuľky imports [vlastné spracovanie]

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
user_id	ID užívateľa, ktorý definoval daný atribút	INT	10	Nie
name	Názov atribútu	STRING	255	Nie
type	Typ	STRING	10	Nie
active	Určuje, či je daný atribút aktívny	INT	1	Nie
order_value	Poradie v akom sa má daný typ zobrazovať v užívateľskom rozhraní	INT	5	Áno

Tab. 16: Zoznam atribútov tabuľky user_property_types [vlastné spracovanie]

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
user_property_type_id	ID užívateľom definovaného atribútu	INT	10	Nie
specimen_set_id	ID setu	INT	10	Nie

Tab. 17: Zoznam atribútov tabuľky specimen_set_shared_properties [vlastné spracovanie]

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
user_id	ID užívateľa, ktorý daný set zdielal	INT	10	Nie
shareable_id	ID zdieľanéj položky	INT	10	Nie
shareable_type	Typ zdielanej položky	STRING	255	Nie
hash	Hash, použítý jako identifikátor použitý v adrese na zdielanie	String	60	Nie

Tab. 18: Zoznam atribútov tabuľky shared_items [vlastné spracovanie]

Názov	**Popis**	**Typ**	**Dĺžka**	**Null**
user_property_type_id	ID užívateľom definovaného atribútu	INT	10	Nie
specimen_id	ID vzorky	INT	10	Nie
string_value	Hodnota atribútu, pokiaľ je jej typ string	STRING	255	Áno
price_value	Hodnota atribútu, pokiaľ je jej typ price	DECIMAL	10,2	Áno
price_currency	Mena atribútu, pokiaľ je jej typ price	STRING	3	Áno
gps_lat	Hodnota GPS výšky atribútu, pokiaľ je jej typ gps	DOUBLE	13,10	Áno
gps_long	Hodnota GPS šírky atribútu, pokiaľ je jej typ gps	Double	13,10	Áno
number_value	Hodnota atribútu, pokiaľ je jej typ number	Double	8,2	Áno

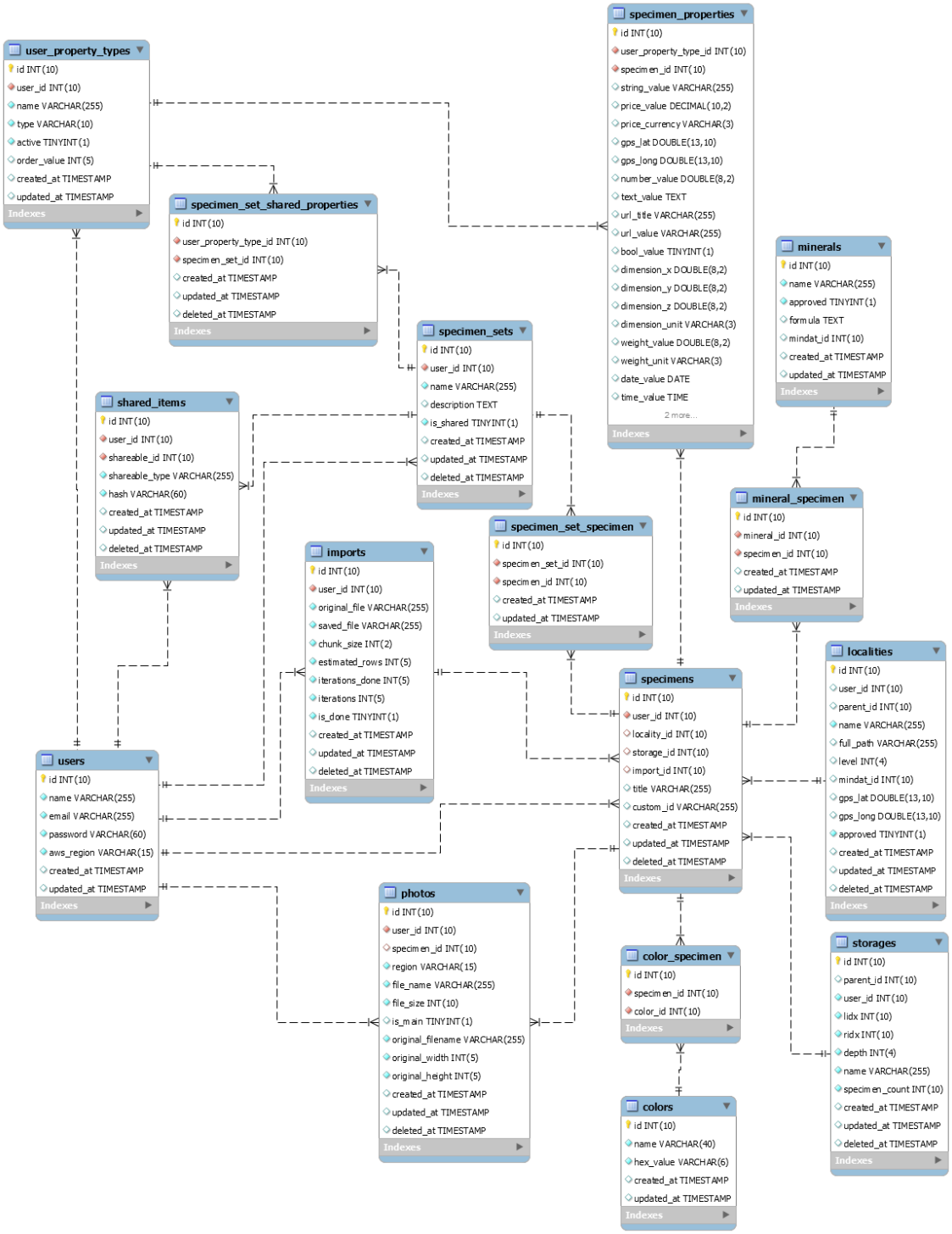
text_value	Hodnota atribútu, pokiaľ je jej typ text	Text	1000	Áno
url_title	Hodnota názvu odkazu atribútu, pokiaľ je jej typ url	STRING	255	Áno
url_value	Hodnota http odkazu atribútu, pokiaľ je jej typ url	STRING	255	Áno
bool_value	Hodnota atribútu, pokiaľ je jej typ bool	INT	1	Áno
dimension_x	Hodnota rozmeru X, pokiaľ je jej dimension	DOUBLE	8,2	Áno
dimension_y	Hodnota rozmeru Y, pokiaľ je jej dimension	DOUBLE	8,2	Áno
dimension_z	Hodnota rozmeru Z, pokiaľ je jej dimension	DOUBLE	8,2	Áno
dimension_unit	Názov jednotky atribútu, pokiaľ je jej dimension	STRING	3	Áno
weight_value	Hodnota atribútu, pokiaľ je jej typ weight	DOUBLE	8,2	Áno
weight_unit	Jednotka atribútu, pokiaľ je jej typ weight	STRING	3	Áno
date_value	Hodnota atribútu, pokiaľ je jej typ date	DATE		Áno
time_value	Hodnota atribútu, pokiaľ je jej typ time	TIME		Áno

Tab. 19: Zoznam atribútov tabuľky specimen:properties [vlastné spracovanie]

3.2.4 ER diagram

Zoznam atribútov a ich vlastností spolu so zoznamom entít použijem k vytvoreniu ER diagramu, ktorý bude reprezentovať databázu prototypu aplikácie.

Pri niektorých tabuľkách som pridal atribúty created_at, updated_at a deleted_at. Tieto atribúty nesú dátový typ TIMESTAMP, čo je vlastne čas a dátum. Prvotným nastavením týchto polí je NULL. Tieto polia slúžia na determinovanie, kedy bol daný záznam vytvorený, upravený a zmazaný. Označovaním záznamov ako zmazaných umožníme spätnú obnovu dát nastavením atribútu deleted_at späť na NULL, čo uľahčí obnovovanie zmazaných údajov.



Obr. 12: ER diagram databázy [vlastné spracovanie]

3.3 Príprava ORM vrstvy

V tejto časti sa budem venovať vytvoreniu a namapovaniu entít z databázy s triedami v PHP frameworku Laravel (modelmi). Nebudem popisovať vytvorenie každého modelu ale popíšem vytvorenie modelov na dvoch príkladoch, podľa ktorých budú vytvorené ostatné modely.

3.3.1 Specimen

V tomto bode ukážem akým spôsobom som definoval Specimen model, ktorý bude zodpovedný za komunikáciu medzi PHP aplikáciou a SQL databázou. Každý model musí rozširovať základný model ORM vrstvy Eloquentu, ktorý sa stará o všetku logiku generovania SQL príkazov. Popíšem vybrané metódy a atribúty danej triedy.

```php
class Specimen extends Model {

    public $timestamps = true;

    use SoftDeletes;

    protected $dates = ['deleted_at'];
    protected $fillable = array('user_id', 'locality_id', 'storage_id', 'title', 'custom_id');

    protected $with = ['minerals', 'photos', 'colors', 'locality', 'storage', 'properties'];

    public function minerals()
    {
        return $this->belongsToMany(App\Mineral::class);
    }

    public function storage()
    {
        return $this->belongsTo(App\Storage::class);
    }

    public function locality()
    {
        return $this->belongsTo(App\Locality::class, 'locality_id');
    }

    public function photos()
    {
        return $this->hasMany(App\Photo::class, 'specimen_id');
    }

    public function colors()
    {
        return $this->belongsToMany(App\Color::class);
    }

    public function user()
    {
        return $this->belongsTo(App\User::class);
    }

    public function sets()
    {
        return $this->belongsToMany(App\SpecimenSet::class, 'specimen_set_specimen');
    }

    public function properties()
    {
        return $this->hasMany(SpecimenProperty::class);
    }
}
```

Obr. 13: Zdrojový kód Specimen modelu [vlastné spracovanie]

V triede Specimen som nastavil atribút $timestamp = true. V prípade, že je tento atribút nastavený na true, tak v prípade, že sa tento model vytvorí, či upraví, budú v SQL príkaze automaticky nastavené stĺpce created_at a updated_at.

Príkaz use SoftDeletes; a $dates = ['deleted_at']; vravia modelu, aby pri generovaní SQL príkazu pridal na jeho koniec podmienku, ktorá obmedzuje výber iba na hodnoty, kde deleted_at je nastavené na NULL. Taktiež, ak na daný objekt zavoláme metódu delete() nebude permanentne vymazaný iba označený ako vymazaný.

Príkaz $fillable=['user_id', ...]; modelu vraví, že tieto atribúty môžu byť menené metódami create() a update().

Atribút $with určuje, ktoré z implementovaných relácií budú volané stále pri vytvorení tohto objektu. Toto je zvlášť výhodné pokiaľ vieme, že vo väčšine prípadov, keď budeme volať tento model, chceme spolu s ním získať aj jeho ďalšie relácie.

Metódy znázornené na obrázku (Obr. 13), sú vlastne reprezentáciami relácií na úrovni databázy. Metóda photos() vracajúca $this->hasMany(App\Photo::class, 'specimen_id'); vytvorí na základe modelu Photo SQL príkaz, ktorý vyberie všetky záznamy z tabuľky namapovanej na tento model, ktorých specimen_id atribút je rovný hodnote id atribútu objektu, na ktorom voláme metódu photos(). Táto relácie reprezentuje väzbu typu 1:N.

Podobne fungujú aj iné relácie. Relácia belongsTo() je reprezentáciou väzby N:1 a relácia belongsToMany() je reprezentáciou väzby N:M.

3.3.2 Storage

Model Storage musí podporovať funkcionalitu, kde sa bude jeden záznam odkazovať na záznam v tej istej tabuľke a tým sa vytvorí stromová štruktúra. K vyriešeniu tohoto problému použijem programovací vzor nazývaný Nested Set Model. Aby som nemusel implementovať celú logiku tohoto vzoru, použijem balíček etrepat/baum, rozširujúci základnú funkcionalitu Eloquent ORM modelu.

```php
class Storage extends Node {

    protected $table = 'storages';
    public $timestamps = true;

    use SoftDeletes;

    protected $dates = ['deleted_at'];

    protected $parentColumn = 'parent_id';

    protected $leftColumn = 'lidx';

    protected $rightColumn = 'ridx';

    protected $depthColumn = 'depth';

    protected $scoped = array('user_id');

    protected $fillable = array('name', 'comma_path', 'specimen_count');

    protected $guarded = ['lidx', 'ridx', 'depth'];

    public function specimens()
    {
        return $this->hasMany(Specimen::class);
    }

    public function user()
    {
        return $this->belongsTo(User::class, 'user_id');
    }
}
```

Obr. 14: Implementácia modelu Storage [vlastné spracovanie]


```
{
  "name": "root",
  "children": [{
    "name": "Stand #1",
    "children": [{
      "name": "Stand #1.1",
      "children": []
    }, {
      "name": "Stand #1.2",
      "children": [{
        "name": "Stand #1.2.1",
        "children": []
      }, {
        "name": "Stand #1.2.2",
        "children": []
      }, {
        "name": "Stand #1.2.3",
        "children": []
      }]
    }, {
      "name": "Stand #1.3",
      "children": []
    }]
  }]
}
```


Obr. 15: Možný výsledok výpisu modelu Storage [vlastné spracovanie]

Implementácia tohto balíčka nám umožní po zavolaní nasledujúceho príkazu, získať vnorené pole záznamov znázornené na obrázku obr 15.

```
Storage::select('name')->first()->getDescendantsAndSelf()->toHierarchy();
```

3.3.3 Ostatné modely

Ostatné modely vytvorím a namapujem za použitia techník spomenutých v predchádzajúcich bodoch.

3.4 Návrh REST API

V tejto kapitole sa budem venovať návrhu aplikačného rozhrania, ktoré bude používané na komunikáciu servera a javascriptovej aplikácie.

3.4.1 Endpointy

V tejto časti definujem endpointy (koncové body). Všetky endpointy budú vracať objekt vo formáte JSON a so správne nastavenou http hlavičkou Content-Type: application/json. Tento JSON objekt bude atribútmi odpovedať atribútom jednotlivých modelov, ktoré bude endpoint vracať.

URI	Http metódy	Popis
auth/login	POST	Príma prihlasovacie údaje od užívateľa a vracia JWT token
auth/registration	POST	Príma prihlasovacie údaje a vytvára nového užívateľa
color	GET	Vráti kolekciu všetkých dostupných farieb
import	POST	Vykoná import súboru priloženého v requeste
locality/{search}	GET	Vráti kolekciu lokalít, obsahujúcich hodnotu {seach} vo svojom názve
mineral/{search}	GET	Vráti kolekciu minerálov, obsahujúcich hodnotu {search} vo svojom názve
property	GET	Vráti kolekciu užívateľských typov prihláseného užívateľa
property	POST	Vytvorí nový užívateľský typ prihlásenému užívateľovi
set	GET	Vráti kolekciu setov prihláseného užívateľa
set	POST	Vytvorí nový set prihlásenému užívateľovi a vráti ho
set/{id}	PATCH	Upraví set s id = {id} a vráti zmenený objekt
set/{id}	DELETE	Odstráni set s id ={id} a vráti zmazaný objekt
specimen	GET	Vráti kolekciu vzoriek patriacu prihlásenému užívateľovi

specimen	POST	Vytvorí novú vzorku
specimen/{id}	PATCH	Upraví vzorku s id = {id} a vráti zmenený objekt
specimen/{id}	DELETE	Odstráni vzorku s id = {id} a vráti zmazaný objekt
specimen/{id}/sets	PUT	Vzorke s id = {id} prepíše sety do ktorých patrí a vráti zmenený objekt
specimen/{id}/storage	PUT	Vzorke s id = {id} prepíše storage pod ktorý patrí a vráti zmenený objekt
storage	GET	Vráti kolekciu objektov storage prihláseného užívateľa
storage	POST	Vytvorí nový storage prihlásenému užívateľovi a vráti ho
storage/{id}	PATCH	Upraví storage s id = {id} a vráti zmenený objekt
storage/{id}	DELETE	Odstráni storage s id = {id} a jeho potomkov a vráti zmenený objekt
storage/{id}/move	POST	Zmení umiestnenie storage s id = {id} a vráti zmenený objekt
upload	POST	Nahrá fotografiu a vráti jej objekt

Tab. 20: Zoznam API endpointov [vlastné spracovanie]

3.4.2 Implementácia endpointov

V tejto časti na vybranom endpointe ukážem akým spôsobom vracia údaje. Podobným spôsobom sú definované ďalšie endpointy.

3.4.3 Route definícia

Ako prvú vec som s vytvoril definíciu endpointu, ktorá bude vravieť routeru frameworku Laravel, ako zaobchádzať s requestami, prichádzajúcimi na danú adresu.

```
Route::group([
    'prefix' => 'v0',
], function(){
    Route::group([
        'middleware' => ['jwt.auth']
    ], function() {
        Route::get('specimen', [
            'uses' => 'SpecimenController@index'
        ]);
    });

});
```

Obr. 16: Zápis route GET specimen [vlastné spracovanie]

Kód na Obr. 16 robí to, že najprv pred všetky routy definované v danej skupine nastaví prefix v0. Tento prefix je vlastne verzia našej API, je to užitočné do budúcnosti, ak by sme chceli API meniť na novšiu verziu a mať prístup aj k starej.

Následne je použitá skupine definujúca middleware, ktorý obsahuje logiku, stojacu za tým, či je užívateľ prihlásený alebo nie. Ak nie vráti užívateľovi patričnú správu a nepustí ho k zdrojom nachádzajúcim sa v danej skupine.

Najviac vnorený príkaz je definícia endpointu samotného. Názov funkcie get určuje, o akú http metódu sa jedná. Prvým argumentom funkcie je URI adresa na ktorú bude reagovať. V tomto prípade specimen. Druhým argumentom je pole konfigurácie daného routu. V tomto prípade nastavujeme iba controller, ktorý sa má starať o vybavenie daného requestu

3.4.4 Controller

Rozšírenie triedy Controller, v tomto prípade SpecimenController sa bude starať o spracovanie requestov, ktoré konzumujú URI specimen. Jeho implementáciu si ukážeme na spracovaní GET requestu endpointu specimen.

```
class SpecimenController extends Controller
{
    protected $auth;

    public function __construct(Guard $auth)
    {
        $this->auth = $auth;
    }

    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $specimens = this->auth->user()->specimens;

        return $specimens->toArray();
    }

}
```

Obr. 17: Definícia controlleru SpecimenController [vlastné spracovanie]

Na obrázku Obr. 16 som definoval, že endpoint GET specimen používa SpecimenController@index čo v tomto prípade znamená, že na controlleri SpecimenController zavolá metódu index.

Keďže endpoint GET specimen má vracať kolekciu vzoriek, patriacu prihlásenému užívateľovi a o kontrolu prihlásenia sa mi postaral middleware definovaný v routovaciom súbore, môžem v constructore triedy inicializovať premennú, ktorej bude pripísaný autentifikačný servis frameworku Laravel.

Následne som definoval samotnú metódu index. V nej som vytvoril premennú $specimens, ktorej som pripísal hodnotu aktuálne prihláseného užívateľa získaného zavolaním funkcie $this->auth->user();. Táto funkcia nám vracia model User, ktorý som implementoval podobným spôsobom ako model Specimen popísaným na Obr. 13. To nám umožňovalo na danom objekte zavolať atribút specimens, ktorý sa automatický zavolá metódu specimens() a naplní sa jej výsledkom, čo v tomto prípade bude kolekcia všetkých vzoriek patriacich prihlásenému užívateľovi. Na konci vrátim premennú $specimens, ktorú ešte pred vrátením zmením na pole, zavolaním funkcie toArray().

3.5 Návrh užívateľského rozhrania

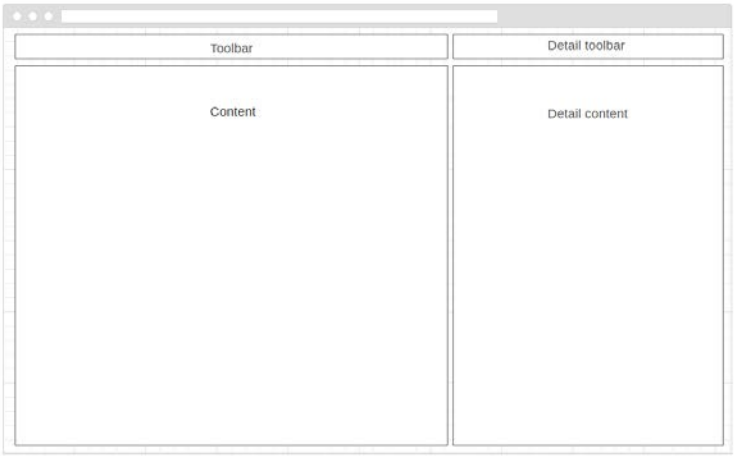
V tomto bode predstavím náčrty užívateľského rozhrania. Taktiež popíšem jeho základné správanie, ktoré mi pomôže pri následnej implementácii.

3.5.1 Všeobecné rozloženie

Základné rozloženie stránky (Obr. 18) som navrhlo tak, aby navigačné prvky boli umiestnené po bokoch okna. Tým mi ostalo veľa miesta pre obsah.

Obr. 18: Všeobecné rozloženie prvkov aplikácie [vlastné spracovanie]

Navigácia sa nachádza na ľavej strane okna, v prípade že sa jedná o mobilné zariadenie je táto navigácia skrytá. V hornej časti sa nachádza toolbar, ktorý v sebe nesie informácie o aktuálnej časti aplikácie ktorá sa používa. Content je vyhradený pre obsah.

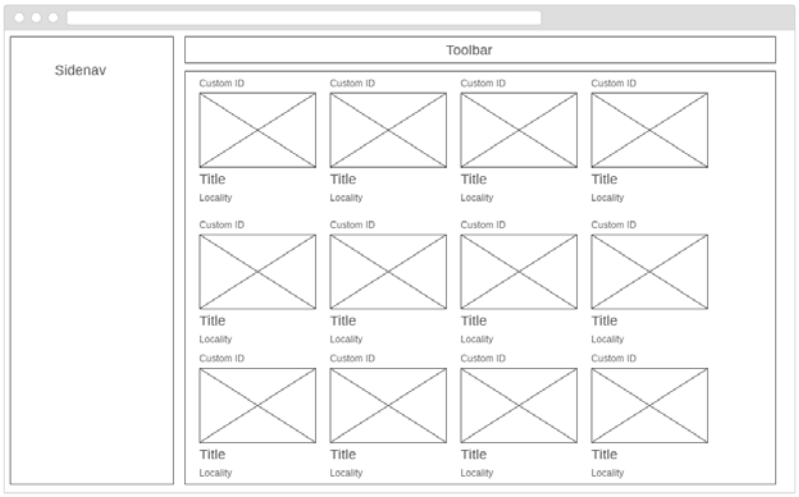


Obr. 19: Rozloženie s detailom objektu [vlastné spracovanie]

Ďalším základným rozložením (Obr. 19) je rozloženie s detailom objektu. V tomto rozložení sa z pravej strany vysunie bočný panel, ktorý vytlačí obsah doľava. Taktiež sa úplne schová bočné menu na ľavej strane čím ostane viac miesta pre obsah.

3.5.2 Dashboard

Táto obrazovka (Obr. 20) bude slúžiť na zobrazovanie všetkých vzoriek v zbierke.



Obr. 20: Rozloženie zobrazenia dashboard [vlastné spracovanie]

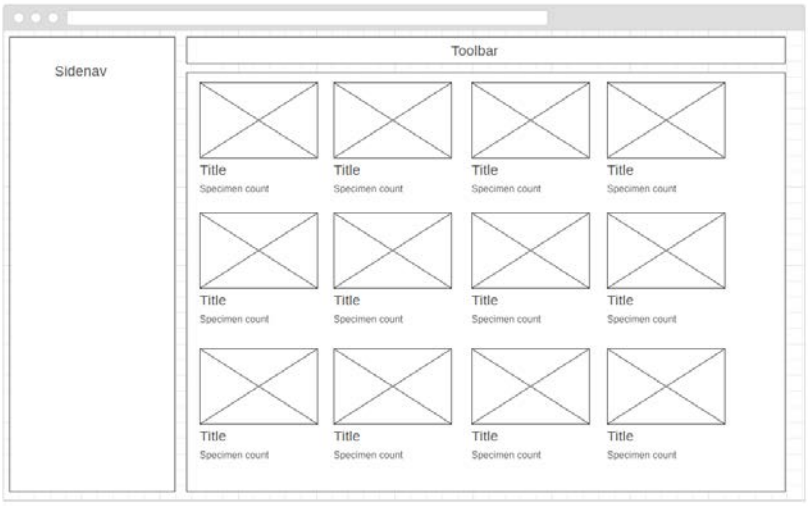
Pri návrhu rozloženia, som kládol dôraz na zobrazovaný obrázok minerálu a základných informácií o vzorke. Rozhodol som sa o grid rozloženie reprezentované malými dlaždicami. V každej dlaždici zaberá väčšinu miesta obrázok. Medzi ďalšie zobrazované údaje patrí užívateľsky definované id, ktoré môže slúžiť ako evidenčné číslo vzorky, názov vzorky a lokalita, z ktorej vzorka pochádza.

Ďalším možným rozložením je okrem rozloženia grid, je rozloženie tabuľky. Toto rozloženie obsahuje všetky atribúty vzorky spolu s miniatúrnym obrázkom.

Po kliknutí na jednotlivé dlaždice sa sprava vysunie panel s detailmi o danej vzorke.

3.5.3 Sety

Táto obrazovka (Obr. 21) bude obsahovať zoznam setov definovaných užívateľom. Rozloženie bude implementované rovnako ako pri dashboarde. Avšak na rozdiel od dashboardu bude obsahovať iba obrázok, názov vzorky a počet vzoriek v sete.

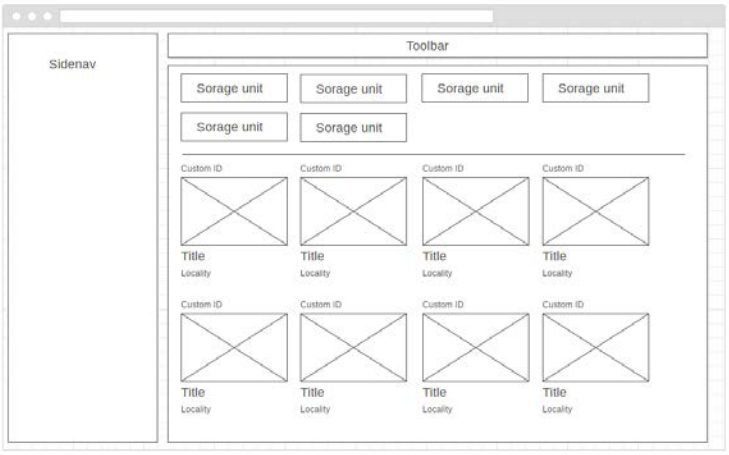


Obr. 21: Rozloženie obrazovky sety [vlastné spracovanie]

Po kliknutí na set sa otvorí obrazovka rovnaká ako pri dashboarde, ktorá bude obsahovať iba vzorky patriace do otvoreného setu.

3.5.4 Storage

Úlohou tejto obrazovky (Obr. 22) bude zobraziť skladovacie jednotky a ich obsah. Aby užívateľ mal okamžitý prehľaď čo sa v danej jednotke nachádza rozhodol som sa rozdeliť oblasť pre obsah na dve časti.



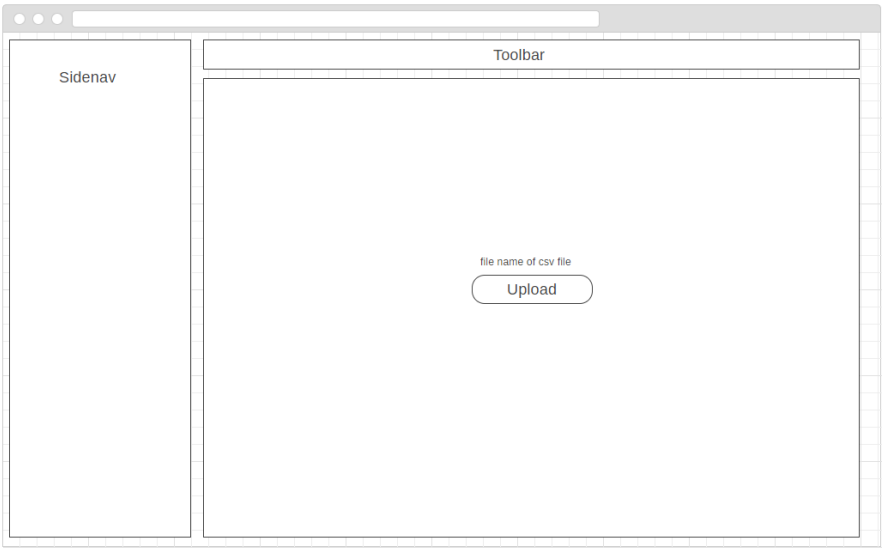
Obr. 22: Rozloženie obrazovky storage [vlastné spracovanie]

V hornej časti bude zobrazovaný zoznam skladovacích jednotiek. Tieto jednotky budú predstavovať jednotky, ktoré sú priamym potomkom aktuálne otvorenej jednotky. Po kliknutí na jednotku, sa otvorí táto jednotka s rovnakým rozložením.

V spodnej časti budú zobrazené vzorky nachádzajúce sa v tejto skladovej jednotke a všetkých jednotkách ktoré sú v nej umiestnené.

3.5.5 Import

Táto obrazovka bude v prototype aplikácie jednoduchá, bude obsahovať iba formulár na nahranie CSV súboru.



Obr. 23: Rozloženie obrazovky import [vlastné spracovanie]

3.5.6 Filter

Keďže podľa zadania, má byť filter aktívny až pokiaľ sa neresetuje, rozhodol som sa že bude aj viditeľný. Ako umiestnenie filtra som vybral ľavé menu (Obr. 24). Menu som rozdelil na dve časti z ktorých je jedna vždy skrytá. Na prepínanie medzi týmito časťami som sa rozhodol použiť tlačidlá umiestnené v hornej časti obrazovky.

Filter samotný obsahuje zoznam atribútov a vstupných polí, podľa ktorých je možné filtrovať. V spodnej časti som pridal tlačidlá pre aplikovanie filtra a jeho reset.

Obr. 24: Rozloženie bočného panelu filtra [vlastné spracovanie]

3.6 Návrh obrazoviek aplikácie

V tejto kapitole sa budem venovať inicializácii aplikácii a spôsobu získavania údajov zo serveru jednej z obrazoviek. Postup použitý pri vytváraní danej obrazovky môže byť použitý pri všetkých ďalších.

3.6.1 Inicializovanie aplikácie

Ako prvé som si vytvoril súbory index.html, ktorý bude slúžiť ako vstupný bod našej aplikácie. Ten načíta potrebné súbory a inicializuje AngularJS framework.

```html
<!DOCTYPE html>
<html lang="en" ng-app="MinSets" ng-controller="MainCtrl">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Mineralo.GY</title>

    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

    <!-- Bootstrap -->
    <link href="build/css/vendor.css" rel="stylesheet">
    <link href="build/css/app.css" rel="stylesheet">
</head>
<body>
    <md-content ui-view></md-content>

    <script src="build/js/vendor.js"></script>
    <script src="build/js/config.js"></script>
    <script src="build/js/app.js"></script>
</body>
</html>
```

Obr. 25: Zdrojový kód index.html [vlastné spracovanie]

Na obr. 25 je vidieť, že vkladám ďalšie súbory ako napríklad vendor.js, app.js či app.css. Tieto súbory sú generované nástrojom Laravel Elixir. Ten mi umožní spojiť všetky externé moduly Nastavenie Elixir (Obr. 25), ktoré som sa rozhodol používať do jedného súboru a tým zníži zaťaženie siete a serveru.

Celá aplikácia sa inicializuje kódom ng-app='MinSets', ktorý načíta angular modul definovaný v načítaných javascript súboroch.

```
var elixir = require('laravel-elixir');

elixir(function(mix) {
    mix.less([
        'main.less'
    ], 'public/catalogue/build/css/app.css');

    mix.styles([
        ...
        '/angular-material/angular-material.css',
        ...
    ], 'public/catalogue/build/css/vendor.css', 'bower_components');


    mix.scripts([
        '../resources/assets/js/app.js',
        '../resources/assets/js/components/**/*.js'
    ], 'public/catalogue/build/js/app.js', 'bower_components');

    mix.scripts([
        ...
        'jquery/dist/jquery.js',
        'flexslider/jquery.flexslider-min.js',
        'angular/angular.min.js',
        'angular-material/angular-material.min.js',
        'angular-ui-router/release/angular-ui-router.js',
        ...
    ], 'public/catalogue/build/js/vendor.js', 'bower_components');
});
```

Obr. 26: Nastavenia Elixir [vlastné spracovanie]

3.6.2 Definovanie routovania aplikácie

Každá z našich obrazoviek bude viazaná na špecifickú URL adresu. Aby AngularJS vedel ako správne vykresliť dané adresy, musím podobne ako to bolo v prípade backendu definovať endpointy.

Na obr. 27 je ukážka kódu, ktorý definuje endpoint collection s url adresou collection. K tomu to endpointu je priradený obsah. Tento obsah načítava html súbor obsahujúci template dashboardu a spojité ho s controllerom CollectionController.

```
var app = angular.module('MinSets', [
    'ngMaterial',
    'ngSanitize',
    'ngMessages',
    'restangular',
    'ui.router',
    'angular-flexslider',
    'md.data.table',
    'LocalStorageModule',
    'duScroll',
    'ngCollection',
    'thatisuday.dropzone',
    'ui.bootstrap',
    'ui.tree',
    'ui-notification',
    'ngFileUpload',
]);

app.config(function($stateProvider, $urlRouterProvider){
    $stateProvider.state('collection', {
        url: 'collection',
        controller: 'CollectionController',
        templateUrl: 'build/js/views/collection.html'
    });

    $urlRouterProvider.otherwise('/collection');
});
```

Obr. 27: Príklad zápisu routovania v AngularJS [vlastné spracovanie]

3.6.3 Controller

V controlleri (Obr. 28) je objektu $scope, ktorý je prístupný v html súbore, nastavený atribút specimens. Následne je volaná metóda služby SpecimenService, ktorá spúšťa AJAX request na server. V bloku then(...) je definovaná logika, ktorá sa má vykonať po úspešnom dokončení requestu. V tomto prípade pripíše odpoveď do $scope.specimens.

```
function CollectionlController($scope, $rootScope, SpecimenService){
    $scope.specimens = []
    SpecimenService.getAll().then(function(data) {
        $scope.specimens = data;
    });
};

angular.module('MinSets').controller(CollectionlController, CollectionlController);
```

Obr. 28: Zápis objektu ColectionController [vlastné spracovanie]

3.6.4 Service

Servisy budú slúžiť na získavanie údajov zo serveru. Budú využívať základnej AngularJS knižnice na komunikáciu s http servermi.

```
var SpecimenService = function($http){
    this.getAll = function(){
        return $http.get('v0/specimens');
    };
};

angular.module('MinSets').service('SpecimenService', SpecimenService);
```

Obr. 29: Definícia SpecimenService [vlastné spracovanie]

Kód na obr. 29 vytvára SpecimenService a definuje na ňom metódu getAll(), ktorá volá asynchrone API adresu v0/specimens.

3.7 Finálny vzhľad aplikácie

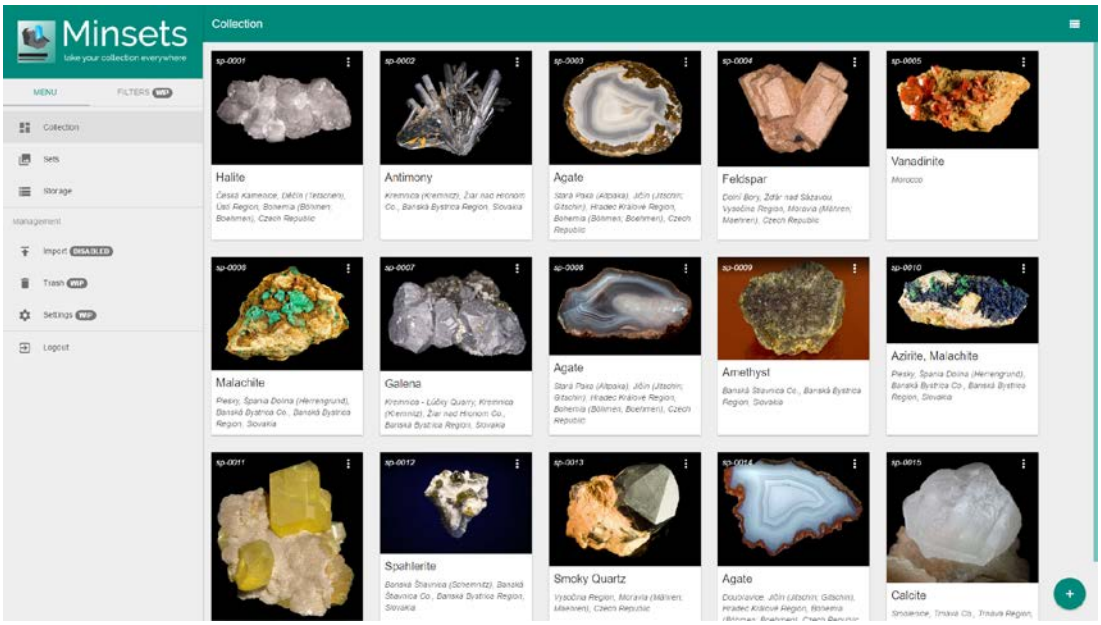
V tejto kapitole predstavím výslednú aplikáciu. Ukážem obrázky vybraných obrazoviek a popíšem spôsob akým sa správajú.

3.7.1 Dashboard

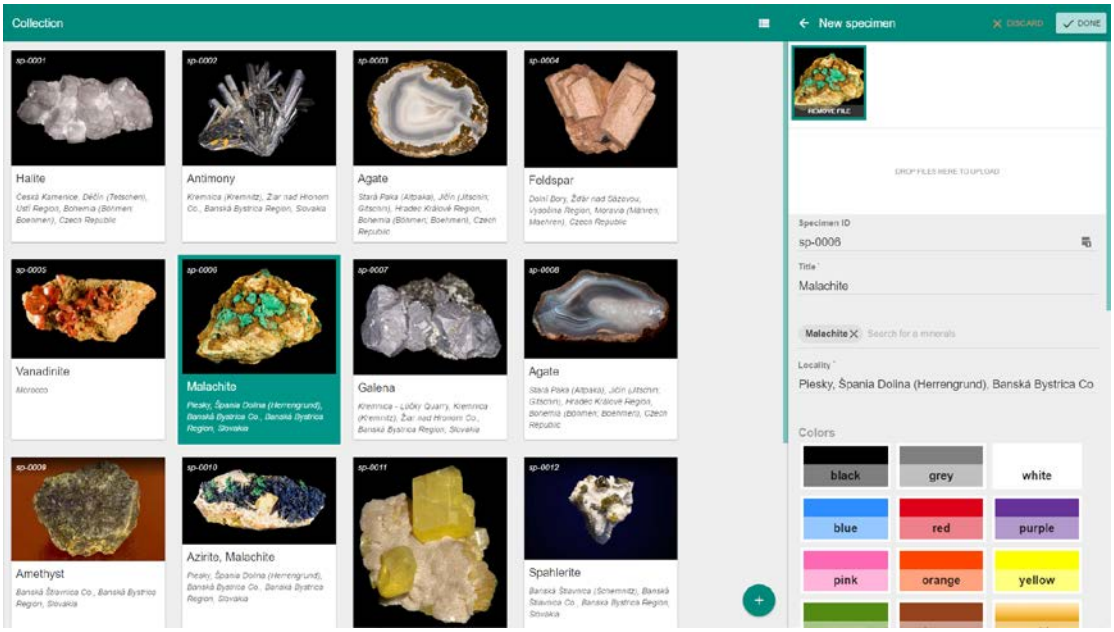
Dashboard mal byť miestom, z kadiaľ uvidím svoju zbierku v podobe umožňujúcu rýchly prehľad nad informáciami, ale aj v podobe, kde bude hraď dominantnú úlohu obrázok (Obr. 30) . Táto obrazovka je prvou obrazovkou, s ktorou sa užívateľ stretne a v základnom nastavení ponúkne práve prehľad obrázkov, ktorý je možné prepnúť na table view pomocou tlačidla v pravom hornom rohu.

Oproti návrhu rozloženia bolo jednotlivým dlaždiciam so vzorkami pridané tlačidlo. Toto tlačidlo je umiestnené v pravom hornom rohu každej dlaždice a po jeho kliknutí sa otvorí kontextové menu. Jednou z položiek kontextového menu je aj editácia (Obr. 31).

Editácia využíva rozloženia kde sa ľavé bočné menu skryje a z pravej strany sa vysunie panel, s možnosťou editácie.



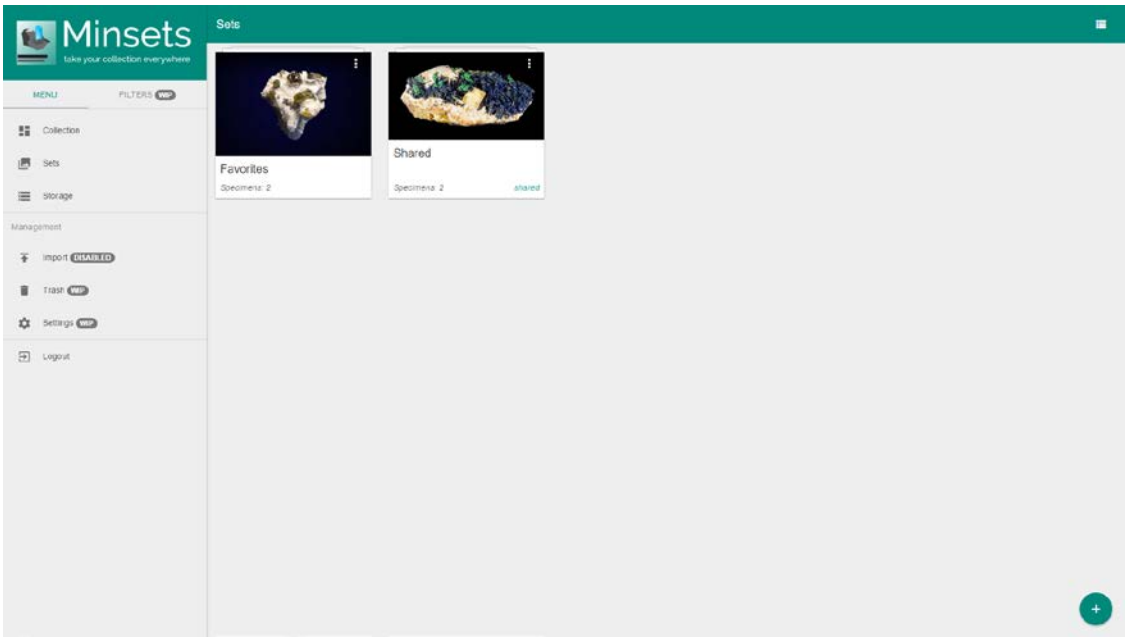
Obr. 30: Dokončená verzia obrazovky dashboard [vlastné spracovanie]



Obr. 31: Dokončená verzia obrazovky s editáciou vzorky [vlastné spracovanie]

3.7.2 Sety

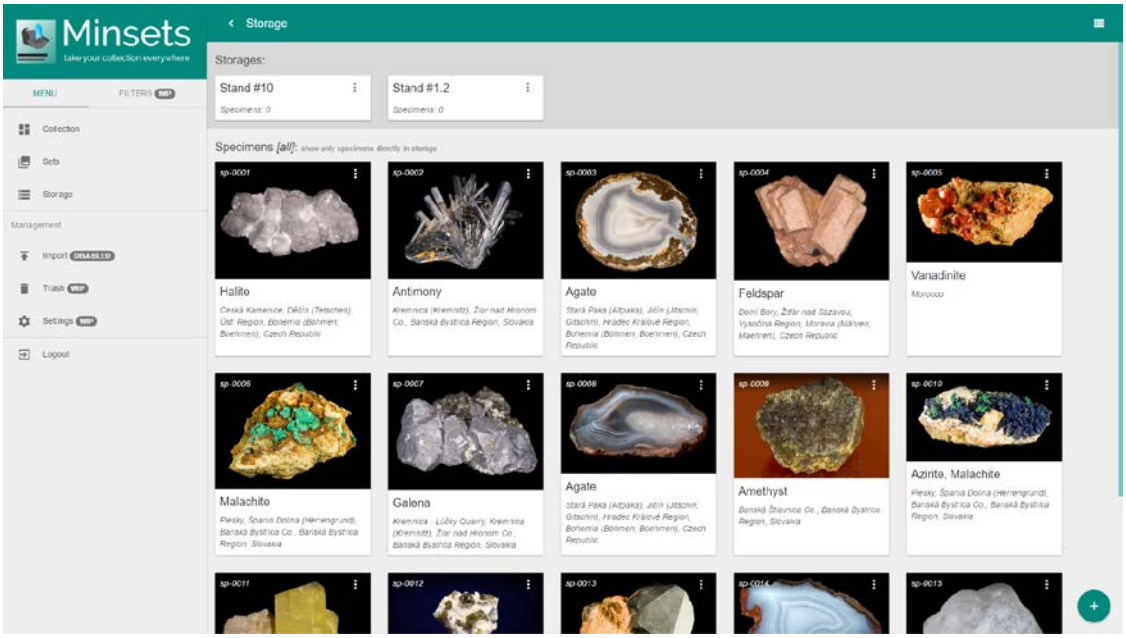
Obrazovka poskytujúca prehľad setov (Obr. 31) je dosť podobná obrazovke dashboardu. Aby bolo možné vizuálne rozlíšiť dlaždicu setu a dlaždicu vzorky, pridal som dlaždiciam setov efekt, ktorý tvorí dojem, že sa za nimi nachádza ďalšia dlaždica. Taktiež každý set v prehľade jasne indikuje, či je alebo nie je zdieľaný.



Obr. 32: Dokončená obrazovka setov [vlastné spracovanie]

3.7.3 Storage

Podľa návrhu mala obrazovka storage zobrazovať ak jednotlivé úložné priestory, tak aj minerály nachádzajúce sa v nich. Priestor, kde sú zobrazené jednotlivé úložné priestory som potreboval vizuálne oddeliť od dlaždíc so vzorkami. K tomu som zvolil farebné odlíšenie pozadia.



Obr. 33: Dokončená obrazovka storage [vlastné spracovanie]

3.8 Ekonomické zhodnotenie

Zberatelia minerálov často na evidenciu vzoriek používajú tabuľky písané v programoch ako sú Excel, poprípade evidenciu zbierky zapisujú do kníh. Avšak v prípade, že chcú svoju zbierku prezentovať na internete, poprípade chcú svoju zbierku upravovať na cestách je nutné so sebou nosiť kópiu Excel súboru alebo knihu samotnú. Moje riešenie odstraňuje nutnosť so sebou nosiť rôzne médiá a vystačí si len s moderným zariadením, akým je smartphone, či tablet, a internetovým pripojením.

Navrhnutý systém by mal užívateľovi priniesť nasledujúce prínosy:

- Možnosť spravovania zbierky kdekoľvek s prístupom na internet
- Príjemnejšie grafické rozhranie
- Možnosť definovania vlastných parametrov
- Možnosť prezentovať svoju zbierku potenciálnym kupcom s minimálnou námahou
- Lepší prehľad nad skladom, kde má vzorky uložené

Správa a prevádzkovanie webových aplikácii môže byť niekedy nákladná. Veľkú úlohu tam hrá náročnosť aplikácie na hardwarové požiadavky a počet návštevníkov.

Náklady pri správe navrhnutej aplikácie môžeme rozdeliť na fixné a variabilné. Fixné náklady budú zahŕňať náklady na prenajatie domény a VPS servera. Tieto náklady sa v budúcnosti môžu zmeniť avšak teraz budem predpokladať, že v najbližší rok zmene nedôjde.

Služba	Mesačný poplatok	Ročný poplatok
Doména	-	10€
VPS server	5.7€	68.4€
Laravel Forge	18€	216€
Spolu	-	303.4€

Tab. 21:Zoznam fixných nákladov [vlastné spracovanie]

Variabilné náklady sa budú odvíjať od počtu užívateľov využívajúcich náš systém. Budem predpokladať, že každý užívateľ má vyhradený 1GB ukladacieho priestoru pre jeho fotografie. Jeho obrázky budú predstavovať 500MB prenesených dát zo serverovne AWS S3 a spraví 1000 requestov na servery AWS S3 za mesiac.

Počet užívateľov	Mesačný poplatok	Ročný poplatok
1	0.05€	0.6€
100	11.5€	138€
1000	115.81€	1389.72€

Tab. 22: Zoznam variabilných nákladov [vlastné spracovanie]

Uvedené odhadované variabilné náklady platia iba za predpokladu, že každý z užívateľov bude využívať službu s maximálnym vyťažením, čo sa nedá predpokladať. Ak k službe bude pristupovať občasne a nebude nahrávať a prenášať veľké množstvo dát, tak náklady značne klesnú.

ZÁVER

Cieľom tejto bakalárskej práce bolo navrhnúť prototyp webovej aplikácie slúžiacej na evidenciu zbierok minerálov.

Počas tvorby tohto prototypu som navrhol databázový model, v ktorom som zohľadnil všetky požiadavky na tento systém. Následne som navrhol API rozhranie, cez ktoré bude webová aplikácia komunikovať so serverom. Nakoniec som navrhol užívateľské rozhranie systému tak aby boli všetky ovládacie prvky dobre dostupné

Aplikácia bude slúžiť na osvedčenie konceptu spravovania zbierok minerálov cez internet a bude využívaná ľuďmi, ktorí sa prihlásia k uzatvorenému beta testovaniu.

ZOZNAM POUŽITÝCH ZDROJOV

[1] HTML Standard. *Web Hypertext Application Technology Working Group* [online]. [cit. 2017-05-21]. Dostupné z: https://html.spec.whatwg.org/multipage/introduction.html

[2] HTML 5. *Jak psát web* [online]. [cit. 2017-05-21]. Dostupné z: https://www.jakpsatweb.cz/html/html-5.html

[3] HTML5 Browser Support. *W3Schools Online Web Tutorials* [online]. [cit. 2017-05-21]. Dostupné z: https://www.w3schools.com/html/html5_browsers.asp

[4] CSS. *Mozilla Developer Network* [online]. [cit. 2017-05-21]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS

[5] CSS3 Introduction. *W3Schools Online Web Tutorials* [online]. [cit. 2017-05-21]. Dostupné z: https://www.w3schools.com/css/css3_intro.asp

[6] A Brief History of Responsive Web Design. *Integrated Digital Marketing Agency - Synecore* [online]. [cit. 2017-05-21]. Dostupné z: http://engage.synecoretech.com/marketing-technology-for-growth/bid/204297/A-Brief-History-of-Responsive-Web-Design

[7] What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design. *Awwwards - Website Awards - Best Web Design Trends* [online]. [cit. 2017-05-21]. Dostupné z: https://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html

[8] 11 nejlepších HTML a CSS frameworků pro webdesignéry. *Interval.cz | Svět Internetu, Technologií a Bezpečnosti* [online]. [cit. 2017-05-21]. Dostupné z: https://www.interval.cz/clanky/11-nejlepsich-html-a-css-frameworku-pro-webove-designery

[9] What is Google's Material Design? *Design Inspiration & Ideas from Envato* [online]. [cit. 2017-05-21]. Dostupné z: https://envato.com/blog/introduction-material-design

[10] The state of the Octoverse 2016. *GitHub* [online]. [cit. 2017-05-21]. Dostupné z: https://octoverse.github.com/

[11] ANTANI, Ved. *Mastering JavaScript.* Birmingham: Packt Publishing, 2016. ISBN 978-1-78528-134-1.

[12] About Node.js. *Node.js* [online]. [cit. 2017-05-21]. Dostupné z: https://nodejs.org/en/about/

[13] ZAKAS, Nicholas C. *Professional JavaScript for Web Developers*. 3rd edition. Indianapolis: John Wiley & Sons. ISBN 978-1-118-02669-4.

[14] History of PHP. *PHP: Hypertext Preprocessor* [online]. [cit. 2017-05-21]. Dostupné z: http://php.net/manual/en/history.php.php

[15] PHP 7: deset věcí, které o něm potřebujete vědět. *Interval.cz | Svět Internetu, Technologií a Bezpečnosti* [online]. [cit. 2017-05-21]. Dostupné z: https://www.interval.cz/clanky/php-7-deset-veci-ktere-o-nem-potrebujete-vedet

[16] STAUFFER, Matt. *Laravel: Up and Running*. Sebastopol: O'Reilly Media, 2016. ISBN 978-1-491-93608-5.

[17] Laravel Elixir. *Laravel - The PHP Framework For Web Artisans* [online]. [cit. 2017-05-21]. Dostupné z: https://laravel.com/docs/5.0/elixir

[18] Why laravel is best php framework in 2017*? Best Web Design And Mobile Application Development Company* [online]. [cit. 2017-05-21]. Dostupné z: http://www.amarinfotech.com/why-laravel-is-best-php-framework-in-2016.html

[19] 10 nejlepších PHP frameworků pro vývojáře. *Interval.cz | Svět Internetu, Technologií a Bezpečnosti* [online]. [cit. 2017-05-21]. Dostupné z: https://www.interval.cz/clanky/10-nejlepsich-php-frameworku-pro-vyvojare/

[20] What Is AngularJS? *AngularJS — Superheroic JavaScript MVW Framework* [online]. [cit. 2017-05-21]. Dostupné z: https://docs.angularjs.org/guide/introduction

[21] Introduction to Redis. *Redis* [online]. [cit. 2017-05-21]. Dostupné z: https://redis.io/topics/introduction

[22] Socket.io. *GitHub* [online]. [cit. 2017-05-21]. Dostupné z: https://github.com/socketio/socket.io

[23] Get Started: Chat application. *Socket.IO* [online]. [cit. 2017-05-21]. Dostupné z: https://socket.io/get-started/chat/

[24] MASSÉ, Mark. *REST API Design Rulebook.* Sebastopol: O'Reilly Media, 2012. ISBN 978-1-449-31050-9.

[25] Top 10 Web APIs - Bridging Today's Technology. *Digital Asset Management Software – Webdam* [online]. [cit. 2017-05-21]. Dostupné z: https://webdam.com/blog/top-10-web-apis_bridging-todays-technology/

[26] What is MySQL? *MySQL :: Developer Zone* [online]. [cit. 2017-05-21]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

[27] BRINZAREA-IAMANDI, Bogdan, Cristian DARIE a Audra HENDRIX. *AJAX and PHP Building Modern Web Applications.* Second Edition. Birmingham: Packt Publishing, 2009. ISBN 978-1-847197-72-6.

[28] Ajax and REST, Part 1. *IBM - United States* [online]. [cit. 2017-05-21]. Dostupné z: https://www.ibm.com/developerworks/library/wa-ajaxarch/

[29] GOURLEY, David a Brian TOTTY. *HTTP: The Definitive Guide.* Sebastopol: O'Reilly Media, 2002. ISBN 978-1-56592-509-0.

[30] Apache vs Nginx: Practical Considerations. *DigitalOcean: Cloud computing designed for developers* [online]. [cit. 2017-05-22]. Dostupné z: https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations

[31] HILLAR, Gastón C. *Learning Object-Oriented Programming.* Birmingham: Packt Publishing, 2015. ISBN 978-1-78528-963-7.

[32] PECINOVSKÝ, Rudolf. *OOP - Learn Object Oriented Thinking and Programming*. Řepín - Živonín: Academic Series, 2013. ISBN 978-80-904661-9-7.

[33] Eloquent: Getting Started. *Laravel - The PHP Framework For Web Artisans* [online]. [cit. 2017-05-22]. Dostupné z: https://laravel.com/docs/5.4/eloquent

[34] PHP Versions Stats - 2016.1 Edition. *Jordi Boggiano* [online]. [cit. 2017-05-22]. Dostupné z: https://seld.be/notes/php-versions-stats-2016-1-edition

[35] Turbocharging the Web with PHP 7. *Zend the PHP Company* [online]. [cit. 2017-05-22]. Dostupné z: https://www.zend.com/en/resources/php7_infographic

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK