

Aplikace pro evidenci a správu čtyřlístků

Mgr. Ladislav Nagy



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Mgr. Ladislav Nagy**
Osobní číslo: **A22053**
Studijní program: **B0613A140020 Softwarové inženýrství**
Forma studia: **Kombinovaná**
Téma práce: **Aplikace pro evidenci a správu čtyřlístků**
Téma práce anglicky: **Application for Recording and Managing Four-Leaf Clovers**

Zásady pro vypracování

- Navrhněte webovou aplikaci dle tématu zadání.
- Zvolte vhodné technologie pro implementaci navrženého řešení.
- Implementujte navržené řešení pomocí vybraných technologií.
- Implementovanou aplikaci nasadte online.
- Vhodně prezentujte výsledky práce.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. BEAULIEU, Alan. Learning SQL: Generate, manipulate, and retrieve data. Sebastopol, CA O'Reilly, 2022.
2. BEVACQUA, Nicolas. Javascript Application Design: A build first approach. Shelter Island, NY Manning Publications Company, 2015.
3. FERRARI, Luca and PIROZZI, Enrico. Learn postgresql: Use, manage, and build secure and scalable databases with postgresql 16. Birmingham, UK Packt Publishing Ltd, 2023.
4. KLEPPMANN, Martin. Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. Sebastopol, CA O'Reilly, 2021.
5. OSMANI, Addy. Learning javascript design patterns. Sebastopol, CA O'Reilly Media, 2013.

Vedoucí bakalářské práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **27. října 2024**
Termín odevzdání bakalářské práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem řádně citoval; v případě publikace výsledků budu uveden jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.
- jsem při tvorbě této práce použil nástroj generativního modelu AI ChatGPT (chatgpt.com) k vygenerování počátečního návrhu kódu a ke gramatickým korekturám. Po použití tohoto nástroje jsem provedl/a kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne

.....

podpis autora

ABSTRAKT

Bakalářská práce představuje webovou aplikaci Čtyřlístkotéka, která byla vytvořena v jejím rámci a slouží k evidenci, správě a veřejné prezentaci sbírky čtyřlístků doktora Petra Žáčka. Praktická část práce se zaměřuje na podrobný popis zdrojového kódu aplikace, zatímco teoretická část poskytuje akademickou reflexi použitých klíčových technologií. Aplikace byla vyvinuta převážně v jazyce JavaScript s využitím prostředí Node.js.

Klíčová slova: webová aplikace, JavaScript, Node.js, evidence, databáze

ABSTRACT

The bachelor's thesis presents the web application Čtyřlístkotéka, which was created to manage and publicly present the four-leaf clover collection of Dr. Petr Žáček. In the practical part of the thesis, the source code of the application is described in detail, while the theoretical part provides an academic overview of the key technologies used. The application was developed primarily using the JavaScript language and the Node.js environment.

Keywords: web application, JavaScript, Node.js, record management, database

Děkuji doktoru Petru Žáčkovi nejen za odpovědi na mé otázky během tvorby webové aplikace a psaní této práce, ale také za příležitost vytvořit pro něj aplikaci Čtyřlístkotéka. Děkuji také mému bratrovi Petru Nagyovi a mé přítelkyni Michaelle Řezníčkové za jejich podporu během studia.

OBSAH

ÚVOD.....	11
I TEORETICKÁ ČÁST	13
1 ARCHITEKTURA MIKROSLUŽEB.....	14
2 PROGRAMOVACÍ JAZYK JAVASCRIPT.....	15
3 PROGRAMOVACÍ JAZYK PYTHON.....	18
4 DATABÁZOVÉ SYSTÉMY	20
4.1 POSTGRESQL	20
4.2 JAZYK SQL	21
5 ZÁKLADY BEZPEČNOSTI	22
II PRAKTICKÁ ČÁST.....	24
6 ARCHITEKTURA APLIKACE.....	25
6.1 SCHÉMA DATABÁZE	26
6.2 SERVER.JS	27
6.3 UPLOADDATA.JS	29
6.4 CONVERT.JS.....	31
6.5 QUERIES.JS.....	32
6.6 PDFGENERATOR.PY.....	34
6.7 ADRESÁŘ PUBLIC	37
6.7.1 MAIN.HTML	38
6.7.2 RECORDS.HTML.....	39
6.7.3 LIST.HTML.....	42
6.7.4 STATISTICS.HTML.....	44
6.7.5 LOGIN.HTML	48
6.7.6 UPLOAD.HTML	49
7 NAsAZENÍ APLIKACE	57
ZÁVĚR	58
SEZNAM POUŽITÉ LITERATURY.....	61
SEZNAM OBRÁZKŮ	63
SEZNAM PŘÍLOH.....	64

ÚVOD

Tato bakalářská práce představuje webovou aplikaci Čtyřlístkotéka, která vznikla za účelem digitalizace sbírky čtyřlístků doktora Petra Žáčka a zároveň umožňuje její správu a veřejnou prezentaci. Práce je praktického charakteru, důraz je tedy kladen na popis zdrojového kódu aplikace, nikoliv na teoretickou reflexi metodologie tvorby webových aplikací a v teoretické části práce proto pouze stručně reflektujeme použité technologie.

V teoretické části práce nejprve stručně vysvětlujeme principy architektury mikroslužeb, jež aplikace využívá (viz kapitolu 1 Architektura mikroslužeb). Následuje akademicko-teoretický popis programovacího jazyka JavaScript a vývojového prostředí Node.js, v němž je napsána převážná část kódu (viz kapitolu 2 Programovací jazyk JavaScript). Protože část funkcionality zajišťuje skript napsaný v jazyce Python, je i tento jazyk stručně popsán v samostatné kapitole. Ačkoliv je i popis tohoto jazyka převážně akademicko-teoretický, je kladen větší důraz na popis jeho praktického využití (viz kapitolu 3 Programovací jazyk Python). Kapitola 4 Databázové systémy slouží nejen jako úvod ke dvěma následujícím podkapitolám, ale poskytuje také ucelený, byť stručný úvod do databázových systémů a nastíněna je i jejich základní typologie. Podrobněji je rozebrána architektura systému řízení báze dat PostgreSQL, který aplikace využívá (viz kapitolu 4.1 PostgreSQL), a naopak velmi stručně je v samostatné podkapitole popsán jazyk SQL (viz kapitolu 4.2 Jazyk SQL). Teoretickou část práce uzavírá přehled nejčastějších typů útoků na webové aplikace podle J. P. Muellera (viz kapitolu 5 Základy bezpečnosti).

Převážnou část praktické části práce tvoří kapitola 6 Architektura aplikace a její podkapitoly, které popisují samotný zdrojový kód aplikace. V úvodu k této kapitole jsou nejprve stručně představeny soubory, ze kterých se aplikace skládá a které podrobněji popisujeme v následujících podkapitolách. V přílohách k bakalářské práci nalezneme pouze soubory, které v samostatných podkapitolách popisujeme – automaticky generované soubory, stejně jako stažené ikony a fonty, k práci nepřikládáme.

V první podkapitole podrobně popisujeme strukturu obou tabulek, ze kterých se skládá databáze, kterou aplikace využívá (viz kapitolu 6.1 Schéma databáze). V následujících pěti podkapitolách postupně popisujeme soubory napsané v jazyce JavaScript a soubor napsaný v jazyce Python. Tyto podkapitoly začínají vždy stručným přehledem importovaných modulů a knihoven a pokračují postupným vertikálním popisem souborů. V podkapitole 6.2 server.js je popsán soubor, který představuje vstupní bod aplikace. Na konci podkapitoly

je také blíže vysvětlen proces autentizace a autorizace. V následující podkapitole popisujeme postup nahrávání fotografií a extrakci dat, jejich formátování a ukládání do databáze (viz kapitolu 6.3 `uploadData.js`). Pokračujeme popisem souboru, který slouží ke konverzi fotografií z formátu HEIC do formátu JPEG (viz kapitolu 6.4 `convert.js`). V poslední podkapitole, která se věnuje popisu souboru napsaném v jazyce JavaScript, jsou popsány funkce, které pomocí SQL dotazů zjišťují statistické údaje o aktuálním stavu databáze aplikace (viz kapitolu 6.5 `queries.js`). Soubor napsaný v jazyce Python, který slouží ke generování PDF dokumentu s vybranými údaji z databáze, popisujeme v podkapitole 6.6 `pdfGenerator.py`.

V poslední podkapitole popisujeme adresář, ve kterém se nachází soubory psané v jazyce HTML a které tvoří uživatelské rozhraní aplikace (viz kapitolu 6.7 Adresář public). Jednotlivé soubory jsou podrobně popsány v samostatných oddílech, jež jsou doplněny ilustrativními obrázky. V prvním oddíle popisujeme soubor, který tvoří domovskou stránku aplikace (viz kapitolu 6.7.1 `main.html`). Následuje oddíl popisující soubor, jenž vytváří stránku Záznamy, která slouží k prezentaci údajů k jednotlivým čtyřlístkům (viz kapitolu 6.7.2 `records.html`). Oddíl 6.7.3 `list.html` popisuje, jak je vytvořena stránka Seznam záznamů, která v několikastránkové tabulce představuje celou sbírku čtyřlístků. Další oddíl je věnován popisu souboru, který prezentuje statistické údaje na stránce Statistiky na veřejné části webu (viz kapitolu 6.7.4. `statistics.html`). Oddíl 6.7.5 `login.html` představuje přihlašovací stránku do administrátorské části webu, která je podrobně popsána v oddíle 6.7.6 `upload.html`.

Poslední kapitola závěrečné práce popisuje nasazení aplikace na internet (viz kapitolu 7 Nasazení aplikace). Webovou aplikaci Čtyřlístkotéka lze navštívit na adrese www.ctyrlistkoteka.cz; optimalizace pro mobilní zařízení dosud chybí.

I TEORETICKÁ ČÁST

1 ARCHITEKTURA MIKROSLUŽEB

Do kontrastu bývají často stavěny dva základní typy architektury aplikací, a to monolitická architektura a architektura mikroslužeb. V monolitické architektuře plní všechny funkce jedna aplikace. Naproti tomu v architektuře mikroslužeb v podstatě spolupracuje více menších, vzájemně nezávislých aplikací, přičemž každá z nich plní co nejmenší množství funkcí. I proto, že je změna v monolitické architektuře složitější, je dnes dávana přednost architektuře mikroslužeb, i když vzájemné propojení jednotlivých komponent a jeho údržba bývá složitější a časově náročnější [1, s. 93; 2, s. 7].

Návrh konkrétní architektury začíná identifikací základních funkcí, které má systém plnit, a uživatelských akcí, jimiž budou tyto funkce využívány. Na tomto základě jsou navrženy jednotlivé komponenty a způsob jejich vzájemné komunikace. Většina komunikace mezi mikroslužbami probíhá buď formou požadavku a odpovědi (request-response), nebo prostřednictvím asynchronních front (asynchronous/queue) [1, s. 93].

Pro interakci jednotlivých mikroslužeb je zapotřebí API (z anglického Application Programming Interface neboli rozhraní pro programování aplikací). API poskytuje bezpečný přístup k datům a funkcím jiné aplikace bez nutnosti přímého přístupu ke zdrojovému kódu nebo databázi. To je možné díky definování předem povolených možností [1, s. 227; 3, s. 52].

2 PROGRAMOVACÍ JAZYK JAVASCRIPT

Aplikace, která byla v rámci bakalářské práce vytvořena, je až na jednu funkcionalitu napsána v jazyce JavaScript. Z tohoto důvodu je vhodné tento jazyk krátce popsat z akademicko-teoretického hlediska.

JavaScript je vysokoúrovňový interpretovaný jazyk s dynamickou typovou kontrolou, vhodný jak pro objektově orientované, tak pro funkcionální programování. Místo tradiční třídové dědičnosti používá JavaScript dědičnost prototypovou – v jazyce JavaScript ve skutečnosti třídy neexistují a objekty dědí z jiných objektů. Nejdůležitější roli hrají v jazyce JavaScript funkce. Funkce může být využita jako proměnná nebo jako objekt a lze ji například předat jako argument jiné funkci [1, s. 207; 4, s. 8–13; 5, s. 1].

Jazyk JavaScript byl vytvořen v roce 1996 [6, s. 2]. David Flanagan uvádí, že syntaxe jazyka JavaScript sice vychází ze syntaxe jazyka Java, jedná se ale o plně samostatný jazyk, k jehož používání není znalost jazyka Java nutná [5, s. 1]. Chris Northwood dokonce poznamenává, že dnes má jazyk JavaScript s jazykem Java společného jen velmi málo a jeho pojmenování bylo v podstatě marketingovým tahem využívajícím zvyšující se popularitu jazyka Java v dané době [1, s. 159]. Jazyk JavaScript původně vznikl, jak již název napovídá, jako skriptovací jazyk, dnes se však jedná o robustní a efektivní univerzální (či obecný z anglického termínu *general-purpose*) jazyk [5, s. 1]. Současný JavaScript je v různých verzích standardizován organizací ECMA (European Computer Manufacturers Association, neboli Evropská asociace výrobců počítačů) pod názvem ECMAScript [1, s. 159]. Jazyk JavaScript je snadno osvojitelný a podle Erica Elliotta je i nejvhodnějším vstupním jazykem do světa programování. Tento názor podporuje i fakt, že k použití JavaScriptu není zapotřebí instalace žádných komplexních vývojových prostředí [7, s. 1].

Eric Elliott také označil JavaScript za nejdůležitější programovací jazyk [7, s. 1]. Společně se značkovacím jazykem HTML (Hypertext Markup Language), který specifikuje obsah webových stránek, a jazykem CSS (Cascading Style Sheets tj. kaskádové styly), který definuje jejich vzhled, tvoří JavaScript – zajišťující interaktivitu a chování stránek – základ vývoje webových aplikací. Drtivá většina moderních webových stránek JavaScript využívá a všechny moderní současné webové prohlížeče obsahují jeho interpret nebo překladač. Ve většině moderních prohlížečů je JavaScriptový kód kompilován a vysoce optimalizován, díky čemuž se jeho běhová rychlost velmi přibližuje výkonu kódu napsanému v jazycích C nebo C++. Pro komunikaci s uživatelským rozhraním používá JavaScript ve webovém

prohlížeči rozhraní DOM (Document Object Model, neboli objektový model dokumentu) [1, s. 207; 5, s. 1; 7, s. 2].

Dynamické webové stránky využívají skriptování obsahu buď na straně serveru (server-side scripting), kdy je webová stránka nejprve vygenerována na serveru a poté odeslána uživateli, nebo skriptování obsahu na straně klienta (client-side scripting), kdy je stránka sestavována přímo v zařízení uživatele na základě přijatého kódu. Využívání jazyků pro skriptování na straně serveru, jako je například jazyk PHP, přináší bezpečnostní výhody (například skrytý zdrojový kód) a výkonnostní výhody, neboť zatěžuje server, nikoliv zařízení uživatele. Naopak jazyky pro skriptování na straně klienta, jako je JavaScript, využívají pro zpracovávání kódu webový prohlížeč a nevyžadují tedy explicitní webový server. Výkonnost je při skriptování na straně klienta kompenzována nižší potřebou komunikace a přenosu dat mezi zařízením (klientem) a serverem. Jazyky pro skriptování na straně klienta také zjednodušují vývoj webových aplikací, neboť mohou běžet v prohlížeči pouze lokálně bez potřeby připojení k internetu [3, s. 149–150].

JavaScript byl zpočátku využíván výhradně na straně klienta, neboť v prohlížeči běžel příliš pomalu. Vydání vykreslovacího jádra prohlížeče (web browser engine) V8 společností Google však JavaScript zrychlilo, díky čemuž je dnes možné jej použít i na straně serveru a je možné jej využít i například k vývoji desktopových aplikací, softwaru pro drony nebo systémů v oblasti Internetu věcí (Internet of Things, IoT). Používání jednoho jazyka pro vývoj frontendových i backendových částí aplikace zjednodušuje práci vývojářů a zrychluje celý vývojový proces. S možností využívat stejný jazyk na klientské i serverové straně se objevila i možnost sdílení kódu mezi klientem a serverem, což vedlo ke vzniku nového typu aplikací – nazývaných izomorfní či univerzální aplikace – které mohou běžet jak na straně klienta, tak na straně serveru. Dnes je JavaScript na straně klienta dominantní a prakticky všudypřítomný, ale díky Node.js se uplatňuje i na straně serveru [1, s. 160; 4, s. 7–16; 6, s. 1; 7, s. 1].

Node.js je otevřený software využívající událostmi řízenou architekturu s asynchronními vstupně-výstupními (I/O) operacemi, které probíhají na pozadí. To umožňuje vytváření škálovatelných a efektivních webových serverů a podporu vývoje real-time aplikací za využití webových soketů (web sockets). Node.js nabízí rozhraní pro práci s nejznámějšími relačními databázovými systémy, včetně PostgreSQL, a podle Ethana Browna revitalizoval koncept NoSQL databází. Společně s Node.js je často využíván minimalistický a flexibilní webový framework Express [1, s. 207; 4, s. 16; 6, s. 2–6].

JavaScript podporuje modulární architekturu, přičemž modul je znovupoužitelná softwarová komponenta. Eric Elliott doporučuje vnímat moduly jako malé a nezávislé aplikace, které jsou plně funkční a testovatelné. Moduly by měly být co nejmenší a nejjednodušší a plnit pouze jednu specifickou funkci. Rovněž se nedoporučuje, aby mezi sebou jednotlivé moduly přímo interagovaly. Jednou z největších výhod modulů je jejich samostatnost – změna kódu jednoho modulu zpravidla neovlivní ostatní části systému a jednotlivé moduly lze snadno znovu použít i v jiných aplikacích [7, s. 71–72].

3 PROGRAMOVACÍ JAZYK PYTHON

Jak již bylo zmíněno v předchozí kapitole, jedna z funkcionalit naší webové aplikace není napsána v jazyce JavaScript. Tato funkcionalita, jež je podrobně popsána především v kapitole 6.6 pdfGenerator.py, je implementována v jazyce Python. Z tohoto důvodu je vhodné se u tohoto jazyka stručně zastavit.

Python je univerzální (nebo také obecný z anglického general-purpose) programovací jazyk, který je často definován jako objektově orientovaný skriptovací jazyk (object-oriented scripting language). Označení „univerzální“ znamená, že jej lze využít pro širokou škálu činností – od vývoje webových aplikací, přes tvorbu počítačových her až po robotiku. Tento fakt je v určitém kontrastu k označení „skriptovací“, které v tomto případě neodkazuje na tradiční skriptovací jazyky jako jsou Perl nebo Tcl, ale spíše vyjadřuje jednoduchost jazyka a jeho vhodnost pro rychlé psaní menších programů. Python je však zároveň dobře škálovatelný a i díky silné podpoře objektově orientovaného programování – která například umožňuje opětovné použití již napsaného kódu – je vhodný i pro vývoj robustních aplikací. Jazyk Python podporuje kromě objektového přístupu také procedurální programování [8, s. 3–15].

Jazyk Python byl navržen tak, aby jeho syntaxe byla jednoduchá, čitelná a úsporná, což jej činí příjemným pro vývojáře. Kvůli těmto vlastnostem bývá Python občas označován jako „spustitelný pseudokód“. Zároveň z něj tyto vlastnosti dělají ideální nástroj pro rychlý vývoj, neboť zvyšuje produktivitu programátorů. Vývoj dále urychluje skutečnost, že se jedná o jazyk interpretovaný – není tedy nutná kompilace, což však zároveň představuje jeho největší slabinu, neboť je zpravidla pomalejší než kompilované jazyky jako jsou C nebo C++ [8, s. 3–16].

Ve skutečnosti je však kód v jazyce Python nejprve kompilován do meziformátu nazývaného bajtkód (bytecode), nikoliv přímo do strojového kódu. Tato skutečnost nejen vysvětluje jeho interpretovanou povahu, ale také jeho multiplatformnost. Samotný interpreter Pythonu je napsán v jazyce ANSI C a kód je proto obvykle ve stejné podobě spustitelný na různých zařízeních (např. v operačním systému Linuxu i v operačním systému Windows) [8, s. 4–14].

Jazyk Python je vyvíjen jako software s otevřeným zdrojovým kódem, disponuje robustní kolekcí knihoven a je snadno kombinovatelný s jinými programovacími jazyky.

Python dále nabízí například dynamickou typovou kontrolu a automatickou správu paměti. Tvůrcem jazyka je Guido van Rossum [8, s. 4–15].

4 DATABÁZOVÉ SYSTÉMY

Coronel, Morris a Rob ve svém shrnutí první kapitoly knihy *Database Systems: Design, Implementation and Management* připomínají, že znalost přesných, relevantních a včasných dat je klíčová pro správné rozhodování, které je nezbytné pro přežití. Data definují jako „surová“ (raw) fakta a informace jako výsledek jejich zpracování, který objasňuje jejich význam. Data jsou obvykle ukládána do databáze, jejíž implementace a správa vyžaduje systém řízení báze dat (DBMS z anglického Database Management System) [9, s. 25].

Kroenke, Auer, Vandenberg a Yoder doplňují naznačený model – uživatel, DBMS a databáze – o čtvrtou komponentu: databázovou aplikaci. Ta zprostředkovává komunikaci mezi uživatelem a DBMS, přičemž na základě SQL dotazů umožňuje číst a měnit obsah databáze. Databázovou aplikaci může tvořit jeden nebo více programů. Systém řízení báze dat je počítačový program, který slouží k vytváření, zpracovávání a správě jednotlivých databází pomocí jazyka SQL. Databáze je pak definována jako kolekce propojených tabulek a dalších struktur. Uživatel je ten, kdo databázi využívá a prohlíží, vkládá či upravuje data [10, s. 12].

Na trhu v současnosti převažují databázové systémy využívající relační, objektově-relační nebo NoSQL modely. Oproti tomu databáze založené na síťovém a hierarchickém modelu v podstatě vymizely. NoSQL (zkratka anglického Not Only SQL, tedy „nejen SQL“) databáze kladou důraz na výkon na úkor zamezení chyb a jedná se například o databáze sloupcové, dokumentové, grafové nebo typu klíč-hodnota. Relační databáze fungují na principu vzájemných relací mezi daty a využívají CRUD operace (Create, Read, Update, Delete – vytvoř, přečti, uprav, smaž) a zpravidla i jazyk SQL. Objektově-relační databáze pak rozšiřují relační model o podporu konceptů z objektově-orientovaného programování, jako je například dědičnost (inheritance) [11, s. 2–7].

4.1 PostgreSQL

PostgreSQL je svobodný a otevřený multiplatformní systém řízení báze dat (DBMS) s širokou škálou funkcí. Často bývá označován jako objektově relační DBMS [11, s. 31; 12, s. 11].

PostgreSQL využívá klient-server architekturu, ve které může mnoho klientů přistupovat k serveru s uložištěm dat přes síť využívající TCP/IP protokoly nebo UNIXové sokety. Aplikace klientů mohou pro přístup používat rozhraní ODBC (Open Database

Connectivity), neboť architektura PostgreSQL neumožňuje aplikacím přímý přístup k datům uloženým v databázi, ale pouze prostřednictvím databázového serverového procesu. PostgreSQL server je možné rozdělit na čtyři subsystémy: správce klientských procesů (process manager), zpracovatele dotazů (query processor), utility (utilities) a správce uložště (storage manager) [11, s. 36; 12, s. 13–14].

PostgreSQL nativně podporuje OLTP (online transactional processing) operace a při využití dostupných rozšíření i OLAP (online analytical processing) operace, které zahrnují například ETL (extract, transform, and load) proces. PostgreSQL vyhovuje principům ACID, a je tedy poměrně odolný i vůči selhání hardwaru [11, s. 32–34].

PostgreSQL vzniklo z relační databáze Ingres, kterou vytvořil Michael Stonebraker na Kalifornské univerzitě v Berkeley. Později byl vyvinut relační databázový server postgres, obohacený o podporu komplexních datových typů a objektově relační model, který byl následně rozšířen o podporu jazyka SQL, čímž vznikl systém Postgres95. Číslo 95 bylo nakonec nahrazeno SQL a výslednému produktu byl otevřen zdrojový kód [11, s. 31–32; 12, s. 12–13].

4.2 Jazyk SQL

Jazyk SQL vznikl z jazyka SQUARE vytvořeného společností IBM na základě návrhu jazyka DSL/Alpha, který byl vytvořen doktorem E. F. Coddem pro manipulaci s daty v relačních tabulkách. Původní název jazyka SQL byl SEQUEL. V pozdějších edicích uvádějí autoři Kroenke, Auer, Vandenberg a Yoder jazyk SQL jako pátou součást databázového systému, neboť se jedná o mezinárodně uznávaný jazykový standard pro práci s databázemi [10, s. 30; 13, s. 7].

Jazyk SQL využívá princip deklarativního programování – proces, kterým je získáván požadovaný výsledek, tedy není specifikován programátorem, ale je zvolen samotným databázovým systémem, konkrétně jeho částí nazývanou anglicky optimizer. SQL však není plnohodnotný programovací jazyk, a při vývoji aplikací je proto potřeba jej integrovat do jiného jazyka, často za použití podpůrného API [13, s. 9].

5 ZÁKLADY BEZPEČNOSTI

Pravděpodobnost využití určitého typu útoku závisí na architektuře napadané aplikace [14, s. 2]. Dle Hoffmana otevřely komplexnější a modernější aplikace nový prostor pro možnosti útoků a aplikace podobné té, kterou jsme v rámci této bakalářské práce vytvořili, jsou zranitelnější než monolitické aplikace bez možnosti uživatelské interakce [15, s. 187].

Mueller uvádí následující typy útoků, kterými jsou webové aplikace ohrožovány nejčastěji [14, s. 2–5]:

- Přetečení bufferu (buffer overflow), tedy zaslání nadměrného množství dat, které ohrozí paměť zařízení obsahující spuštěný kód. Hrozbě lze předejít kontrolou a omezením množství zasílaných dat.
- Injektáž kódu (code injection), kdy je do komunikace mezi serverem a uživatelem vložen škodlivý kód. Ochranu představuje například šifrování přenosu dat a používání protokolu HTTPS.
- Skriptování napříč sítí (cross-site scripting, XSS), tedy vložení škodlivého kódu do výstupního datového toku aplikace, který je následně vykonán v prohlížeči uživatele.
- Injekce příkazu operačního systému (operating system command injection), kdy dochází k modifikaci příkazů operačního systému využívaných aplikací. Je vhodné minimalizovat jejich použití.
- Únos spojení (session hijacking), tedy vniknutí do komunikace mezi serverem a jiným uživatelem.
- SQL injection, tedy neoprávněná modifikace SQL dotazu.
- Pevně zadané autentizační údaje (uložení přihlašovacích údajů přímo v kódu neboli hardcoded authentication), nedostatečné ověření identity (nedostatečně podrobná autentizace) či příliš benevolentní autorizace (nadměrně široká přístupová práva).
- Zpřístupnění skrytých adresářů kvůli neomezování vstupu speciálních znaků (např. lomítek).
- Změna parametrů odesílaných na server, které by správně měly být validovány a šifrovány.
- Používání relativních cest k souborům, knihovnám, API nebo frameworkům, které mohou být pozměněny.

- Nedostatečná kontrola na server nahrávaných dat.
- Zastaralý způsob šifrování přenášených dat.

Podle Hoffmana by měl být největší důraz kladen na zajištění bezpečnosti přenosu dat [15, s. 188].

II PRAKTICKÁ ČÁST

6 ARCHITEKTURA APLIKACE

Aplikace nese funkční název „heic-metadata-extractor“, což je pozůstatek z první fáze jejího vývoje. Je implementována v jazyce JavaScript a využívá jeden podpůrný skript napsaný v jazyce Python. Aplikace dále využívá objektově-relační databázový systém PostgreSQL.

V kořenovém adresáři aplikace se nachází celkem devět souborů a pět dalších adresářů (podsložek). Vstupním bodem aplikace je soubor „server.js“, který plní roli serveru. Hlavní funkcionality aplikace jsou dále zajištěny třemi soubory v jazyce JavaScript – „uploadData.js“, „convert.js“ a „queries.js“ – které jsou společně s podpůrným skriptem „pdfGenerator.py“, napsaným v jazyce Python, a souborem „schema.sql“, který slouží k případnému automatickému vytvoření databázové struktury, podrobně popsány v jednotlivých podkapitolách. Sedmým souborem je soubor „.env“, který slouží k uložení citlivých údajů v podobě systémových proměnných. Automaticky generovaný soubor „package.json“ obsahuje základní informace o aplikaci a definuje seznam závislostí (knihoven), které aplikace potřebuje. Soubor umožňuje jednoduchou instalaci všech závislostí příkazem `npm install`. Robustnější, rovněž automaticky generovaný, soubor „package-lock.json“ zajišťuje, aby se aplikace vždy instalovala se stejnými verzemi všech balíčků, čímž garantuje konzistentní prostředí.

Balíčky a knihovny uvedené v souboru „package.json“ jsou při instalaci staženy do adresáře „node_modules“, který se automaticky vytvoří při spuštění příkazu `npm install`. Jakýkoliv ruční zásah do tohoto adresáře je nežádoucí. Adresář „public“ obsahuje šest HTML souborů, pět ikon ve formátu SVG a jeden obrázek ve formátu PNG, které společně tvoří webové uživatelské rozhraní aplikace. Tyto soubory jsou podrobně popsány v kapitole 6.7 Adresář public. Adresář „uploads“, který se v případě potřeby vytvoří při nahrávání dat do aplikace automaticky, slouží k ukládání nahrávaných fotografií. Složka „fonts“ obsahuje dva soubory ve formátu TrueType font (.ttf), které definují běžné a tučné písmo používané při generování PDF souborů z databáze aplikace. Pro tento účel je důležitý také poslední adresář „maps“, do něhož může administrátor prostřednictvím webového rozhraní ukládat mapy s místy nálezů čtyřlístků. Vygenerovaný PDF soubor „records_report.pdf“ je následně uložen také do kořenového adresáře aplikace.

6.1 Schéma databáze

Schéma databáze je definováno v souboru „schema.sql“ a slouží k případnému automatickému vytvoření databázové struktury. Pro potřeby naší aplikace byla zvolena relační databáze PostgreSQL.

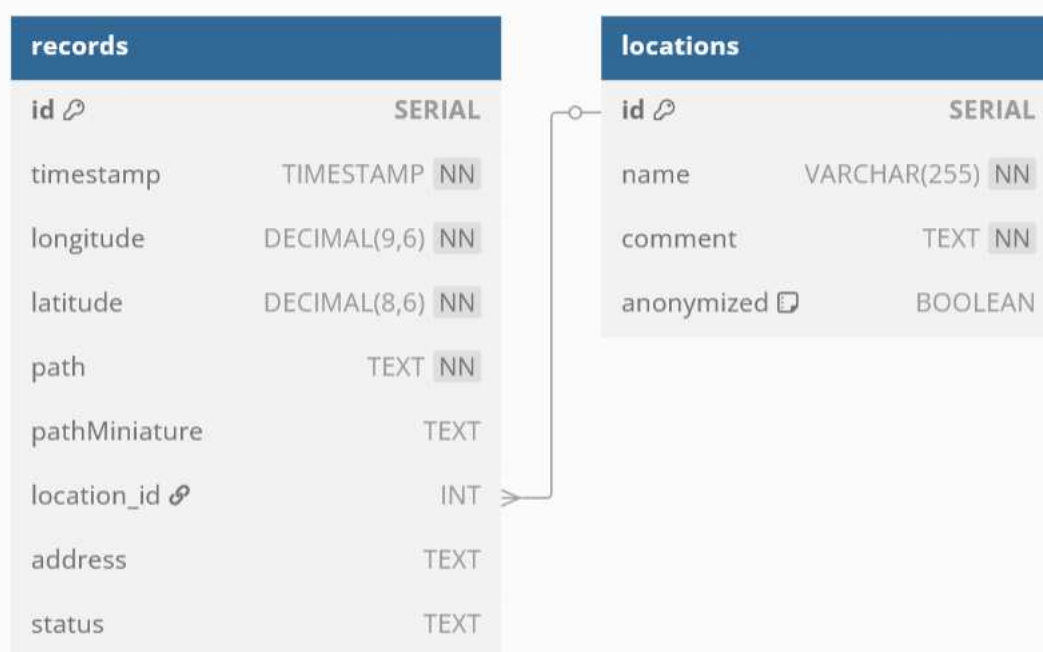
Po vytvoření samotné databáze a navázání spojení s ní jsou vytvořeny dvě tabulky. Tabulka s názvem „records“ slouží k ukládání informací o nalezených čtyřlístcích, zatímco tabulka s názvem „locations“ uchovává údaje o místech, kde byly čtyřlístky nalezeny.

Tabulka „locations“ slouží výhradně k osobním účelům majitele stránek. Majitel si totiž vlastním způsobem eviduje lokality, v nichž čtyřlístky nachází. V tabulce proto nalezneme čtyři sloupce. Sloupec „id“ je automaticky generovaný primární klíč a zároveň identifikátor dané lokace. Sloupec „name“ představuje název lokace tak, jak ji pojmenoval majitel, a je definován datovým typem VARCHAR o rozsahu 255 znaků. Sloupec nesmí být u žádného záznamu prázdný a pro každý záznam musí být unikátní. Neprázdný musí být i sloupec „comment“, který je definován datovým typem TEXT a který slouží pro ukládání komentářů k lokacím. V tomto sloupci však, na rozdíl od sloupce „name“, nemusí být pro každý záznam unikátní hodnota. Poslední sloupec „anonymized“ je datového typu BOOLEAN s výchozí hodnotou false a určuje, zda má být daná lokace anonymizována. Lokace je za anonymní označena nejčastěji tehdy, pokud by její znalost návštěvníkem stránek mohla zasahovat do soukromí majitele.

Tabulka „records“ je tvořena devíti sloupci. Stejně jako v případě tabulky „locations“ slouží jako primární klíč automaticky generovaný identifikátor záznamu uložený ve sloupci „id“. Ve sloupci „timestamp“, příhodného datového typu TIMESTAMP, je uložena časová stopa pořízení fotografie, jež je automaticky zjištěna při nahrávání snímku (viz kapitolu 6.3 uploadData.js). Sloupec „timestamp“ nesmí být prázdný, stejně jako další tři sloupce „longitude“, „latitude“ a „path“. Ve sloupci „longitude“ je datovým typem DECIMAL s přesností devíti číslic, z toho šesti za desetinnou čárkou, uvedena zeměpisná délka místa, kde byla fotografie pořízena. Ve sloupci „latitude“ je pak obdobným způsobem, avšak s pochopitelně nižší přesností osmi číslic, uložena zeměpisná šířka místa. I tyto dva sloupce jsou vyplněny automaticky na základě informací uvedených ve fotografii – majitel stránek totiž používá mobilní telefon, který do EXIF metadat snímků automaticky ukládá i GPS údaje. Do sloupce „path“ je automaticky uložena cesta k uložené fotografii (viz kapitolu 6.3 uploadData.js), přičemž sloupec je definován datovým typem TEXT. Obdobným

způsobem je do sloupce „pathMiniature“ uložena cesta k administrátorem poskytnuté miniatuře fotografie. Sloupec „pathMiniature“ však nemusí být při nahrávání záznamu, na rozdíl od sloupce „path“, vyplněn. Sloupec „location_id“ slouží jako cizí klíč odkazující na tabulku „locations“ a musí být majitelem při nahrávání souboru specifikován v jeho názvu (viz kapitolu 6.3 uploadData.js). Do sloupce „address“ je automaticky uložena adresa odvozená ze souřadnic (i tento proces je podrobněji popsán v kapitole 6.3 uploadData.js). Sloupec „address“ je rovněž datového typu TEXT. Do posledního sloupce „status“ je uložen stavový příznak záznamu. Příznak musí být, stejně jako identifikátor lokace, specifikován majitelem v názvu souboru a sloupec je omezen na akceptování pouze osmi povolených hodnot – 'V', 'D', 'Z', 'N', 'L', 'G', 'S' a 'J' – jejichž význam je objasněn v kapitole 6.3 uploadData.js.

Soubor „schema.sql“ je ve své celistvosti uveden v příloze P VI: Soubor „schema.sql“. Model databáze je k vidění na obrázku 1 Model databáze.



Obrázek 1 Model databáze

(zdroj: vlastní)

6.2 server.js

Vstupním bodem aplikace je soubor „server.js“. Na jeho začátku jsou tradičně importovány potřebné moduly; konkrétně se jedná o webový aplikační framework Express, modul path pro práci s cestami v systému souborů, modul fs umožňující čtení a zápis souborů, klienta

pg pro připojení k PostgreSQL databázi a práci s ní a modul `child_process` umožňující spuštění externích procesů. Dále je importován middleware `body-parser` pro zpracování těla HTTP požadavků a middleware `cors` pro povolení Cross-Origin Resource Sharing. Z knihoven jsou dále využity `jsonwebtoken` pro tvorbu a ověřování JSON Web Tokenů, `bcrypt`, která slouží k bezpečnému hashování hesel, a `express-rate-limit`, která omezuje maximální počet pokusů o přihlášení do administrátorské části webu za určitý časový interval. Nakonec jsou importovány modul `dotenv` pro práci s konfiguračními soubory `„.env“` a vlastní moduly `./uploadData` pro práci s pomocnými funkcemi a `./queries` obsahující SQL dotazy.

Samotný kód začíná vytvořením instance Express aplikace, definováním serverového portu a nastavením konfiguračního objektu pro připojení k PostgreSQL databázi. Následně jsou definovány cesty k veřejně dostupným stránkám, které jsou uloženy v adresáři `„public“` a podrobněji popsány v kapitole 6.7 Adresář public, a definována je také cesta k adresáři `„uploads“`, jenž slouží pro ukládání nahraných fotografií. V kódu jsou dále volány funkce `uploadHEIC` a `uploadLocation` pro nahrávání dat, které jsou popsány v kapitole 6.3 `uploadData.js`.

Následuje větší část kódu sloužící k získávání dat z databáze. Nejdříve je definována cesta pro získání záznamů z tabulky `„locations“` (viz kapitolu 6.1 Schéma databáze) a poté jsou definovány dvě velmi podobné cesty pro získání záznamů z tabulky `„records“` (viz kapitolu 6.1 Schéma databáze): jedna vrací konkrétní záznam podle zadaného identifikátoru (ID) a druhá vrací všechny dostupné záznamy. V obou případech jsou z tabulek navraceny všechny informace o záznamech. Údaje týkající se polohy – tedy zeměpisná šířka a délka, adresa, název lokace a komentář k lokaci – jsou v případě její anonymizace upraveny přímo v rámci SQL dotazu (místo skutečných hodnot se předají souřadnice 0.0000 nebo text typu `„Adresa anonymizována“`). Výsledná data jsou následně zpracována pomocnými funkcemi popsány v kapitole 6.3 `uploadData.js` a ve formátu JSON předána do veřejné části webové aplikace.

Dále je v kódu definována cesta pro spuštění externího skriptu v jazyce Python s názvem `„pdfGenerator.py“`, který blíže popisují v kapitole 6.6 `pdfGenerator.py`, cesta pro získání souřadnic konkrétního záznamu dle jeho identifikátoru a cesta pro ukládání snímku mapy ke konkrétnímu záznamu. Snímek mapy je ve formátu PNG uložen pomocí modulu `fs` do speciálního adresáře `„mapy“` a jeho název odpovídá identifikátoru záznamu.

Následuje větší část kódu sloužící k editaci záznamů v obou databázových tabulkách prostřednictvím webového rozhraní. Pro každou tabulku jsou vytvořeny dvě API cesty – jedna pro načtení konkrétního záznamu podle jeho identifikátoru a druhá pro jeho editaci. Stejně jako v předchozích částech kódu je i zde kontrolována správnost zadaných údajů a je zajištěna správa chyb.

V kódu jsou dále volány všechny funkce spouštějící SQL dotazy pro získání statistických údajů o aktuálním stavu databáze, které jsou detailně popsány v kapitole 6.5 queries.js.

V poslední části kódu zajišťujeme bezpečný přístup do administrátorské části webového rozhraní. Nejprve jsou do konstant načteny citlivé údaje ze systémových proměnných – `SECRET_KEY` pro podepisování JSON Web Tokenů a `ADMIN_PASSWORD_HASH`, což je hash administrátorského hesla uloženého v bezpečné podobě. Dále je nastavena podpora Cross-Origin Resource Sharing (CORS) pro zpracování požadavků a body-parser pro dekodování dat odesílaných prostřednictvím HTTP formulářů. Pro zajištění ochrany proti útokům hrubou silou je nastaven maximální počet pokusů o přihlášení za daný čas – konkrétně 5 pokusů za 15 minut. Jako další je definována veřejně dostupná cesta k HTML stránce s přihlašovacím formulářem (viz kapitolu 6.7.5 login.html). Poté je uživatelem zadané heslo porovnáno s hashovaným administrátorským heslem, jež je uloženo v systémové proměnné, a shodují-li se, je uživateli vygenerován JSON Web Token sloužící k potvrzení jeho identity s expirací 1 hodiny. Zároveň je ověřený uživatel automaticky přesměrován na stránku s uživatelským rozhraním pro administrátorské činnosti, která je bez platného autentizačního tokenu nepřístupná.

Zcela na konci kódu je zajištěno spuštění celé aplikace na definovaném serverovém portu.

Soubor „server.js“ má i s komentáři jej vysvětlujícími celkem 485 řádků a je k dispozici v příloze P I: Soubor „server.js“.

6.3 uploadData.js

Soubor „uploadData.js“ slouží, jak již název napovídá, ke zpracovávání nahrávaných fotografií a k následnému uložení dat do databáze. Na jeho začátku jsou importovány následující moduly a knihovny: modul `fs` umožňující čtení a zápis souborů, modul `path` pro práci s cestami k souborům a adresářům, knihovna `multer` pro zpracování souborů

v rámci HTTP požadavků, modul `pg` pro komunikaci s PostgreSQL databází, knihovna `exif-parser` pro extrakci metadat z nahrávaných fotografií, modul `node-fetch` pro odesílání HTTP požadavků a vlastní modul `convert` obsahující funkci `convertHeicToJpeg`, kterou blíže komentujeme v kapitole 6.4 `convert.js`.

Na začátku kódu definujeme cestu k adresáři „uploads“, do něhož jsou za pomoci knihovny `multer` ukládány konvertované fotografie. Pokud tento adresář neexistuje, je pomocí modulu `fs` automaticky vytvořen.

Dále je definována asynchronní funkce `uploadHEIC`, která nejprve rozlišuje originální fotografie od případných miniatur (viz kapitolu 6.7.6 `upload.html`) na základě údajů získaných pomocí níže definované funkce `parseFileName`. Ta rozděljuje název nahrávaného souboru na jeho funkční části a extrahuje z nich potřebné informace. V případě, že název souboru neodpovídá očekávanému formátu, funkce vyvolá chybu. Funkce `uploadHEIC` následně volá externí funkci `convertHeicToJpeg` ze souboru „convert.js“, která převádí formát fotografie z HEIC do JPEG (funkce je podrobněji popsána v kapitole 6.4 `convert.js`), a poté volá níže definovanou funkci `insertMetadata`. Ta z fotografie extrahuje metadata, upraví je pomocí níže definovaných funkcí `epochToIso8601` a `reverseGeocode` a všechna získaná data následně uloží do tabulky „records“ v naší PostgreSQL databázi (viz kapitolu 6.1 Schéma databáze). Funkce `epochToIso8601` jednoduše převádí z fotografie získané unixové časové razítko nejprve na milisekundy a poté do formátu ISO 8601, což je požadovaný standard pro ukládání časových údajů v datovém typu `TIMESTAMP`. Funkce `reverseGeocode` zase využívá reverzní geokódování (zpětné vyhledávání adresy) prostřednictvím veřejného API služby Nominatim, která je součástí projektu OpenStreetMap [16], a převádí GPS souřadnice na textovou adresu. Neobsahuje-li fotografie potřebná metadata, jsou do tabulky „records“ k danému záznamu uloženy výchozí hodnoty: datum 1. 1. 2000 a souřadnice 0.0000 pro zeměpisnou délku i šířku.

Po definici asynchronní funkce `uploadLocation`, která – jak již název napovídá – slouží k ukládání lokací do databázové tabulky „locations“ (viz kapitolu 6.1 Schéma databáze), jsou až do konce souboru definovány pomocné funkce určené pro formátování dat z databáze pro jejich zobrazení na veřejné část webu. Jako první je definována funkce `formatTimestamp`, která nejdříve převádí datum a čas uložený v databázi požadovaném formátu ISO 8601 do pro uživatele čitelnější podoby a ten následně rozděljuje na funkční části, tedy sekundy, minuty, hodiny, dny, měsíce a roky. Ke dnům poté přidává slovní

označení dne v týdnu (například pondělí) a číselné označení měsíců převádí na jejich slovní variantu. Funkce nakonec vrací finálně zformátovaný řetězec ve tvaru: „neděle 2. června 2024 v 11:54:06“.

Následuje definice funkce `decimalToDMS`, která jednoduše převádí decimálně uvedené koordináty na formát DMS, tedy stupňů, minut a vteřin. Tato funkce je následně využita v dále definované funkci `formatCoordinates`, která souřadnice doplňuje o zeměpisné směrovky. Výstupem funkce je naformátovaný textový řetězec následujícího formátu: „17° 39′ 25.5″ V“.

Jako poslední je definována funkce `translateStatus`, která převádí jednoznakový kód statusu uložený v databázi na jeho plnovýznamovou textovou reprezentaci. Jedná se o následující seznam: hodnota „V“ je převedena na „V pořádku“, hodnota „D“ je převedena na „Darován“, „Z“ je převedeno na „Ztracen“, „N“ je převedeno na „Nevyfocen“, „L“ je převedeno na „Bez zadané lokace“, „G“ je převedeno na „Byl mi darován“, „S“ je převedeno na „Status není určitelný“ a hodnota „J“ je převedena na „S jinou fotografií“. Pokud hodnota neodpovídá žádnému definovanému statusu, je na veřejnou část webu vykreslen text „Status neznámý“.

Zcela na konci souboru jsou všechny funkce (a konstanty) exportovány jako součást modulu. Soubor „`uploadData.js`“ má i s komentáři jej vysvětlujícími celkem 293 řádků a je k dispozici v příloze P II: Soubor „`uploadData.js`“.

6.4 `convert.js`

Soubor „`convert.js`“ obsahuje pouze definici jedné funkce, nejdříve jsou ale importovány potřebné moduly, a to modul `fs` pro čtení a zápis souborů, modul `path` pro práci s cestami a modul `child_process`, který umožňuje spouštění externích nástrojů z prostředí Node.js.

Následně je v souboru definována funkce `convertHeicToJpeg`, která za pomoci open-source balíku softwarových nástrojů ImageMagick převádí fotografie z formátu HEIC do formátu JPEG. Funkce jako vstup přijímá cestu k původnímu HEIC souboru, který byl uložen do adresáře „`uploads`“, a na jejím základě vygeneruje novou cestu s příponou `.jpg`, kam bude uložen výstupní soubor. Převod je realizován spuštěním systémového příkazu `magick convert`, který vytvoří JPEG verzi fotografie. Po úspěšném dokončení převodu je původní HEIC soubor automaticky smazán, čímž se minimalizuje zbytečné zatížení

uložiště. V případě jakékoli chyby při převodu nebo mazání je chyba zachycena a předána zpět volajícímu skriptu.

Funkce je implementována jako **Promise**, což umožňuje její použití pomocí klíčového slova **await** v rámci asynchronních funkcí. Při úspěšném dokončení vrátí funkce cestu k nově vytvořenému JPEG souboru. Na konci souboru je funkce exportována v rámci modulu. Použití této funkce předpokládá, že je v systému správně nainstalován nástroj ImageMagick a že je dostupný příkazem **magick**.

Celý soubor má i s komentáři pouhých 31 řádků a je uveden v příloze P III: Soubor „convert.js“.

6.5 queries.js

Soubor „queries.js“ obsahuje výhradně funkce, které pomocí SQL dotazů generují statistiky o aplikaci používané databázi záznamů, a pro svou činnost vyžaduje pouze import třídy **Client** z modulu **pg**, která slouží pro připojení a interakci s PostgreSQL databází. Všechny definované funkce přijímají vždy dva parametry – instanci aplikace (objekt **app**) a konfigurační objekt **dbConfig**, jenž obsahuje údaje potřebné k navázání spojení s databází – a každý z nich má přiřazenou vlastní HTTP GET cestu. Výsledky všech funkcí jsou vráceny ve formátu JSON, a v případě výskytu jakékoli chyby při práci s databází je výjimka zachycena a odeslána klientovi ve formě chybového hlášení. Spojení s databází je po zpracování každého požadavku vždy korektně ukončeno.

První definovanou funkcí je **top5Locations**, která vrátí seznam pěti lokalit, kde bylo nalezeno nejvíce čtyřlístků. Funkce nejprve zjistí celkový počet záznamů v tabulce „records“, aby mohla v následném dotazu vypočítat relativní procentuální zastoupení jednotlivých lokalit (dle počtu nalezených čtyřlístků). Hlavní SQL dotaz využívá spojení (**JOIN**) mezi tabulkami „locations“ a „records“ (viz kapitolu 6.1 Schéma databáze), přičemž z výsledků vyřazuje anonymizované lokace. Výsledky se totiž zobrazují na veřejné části webu a zveřejnění by jen názvů anonymizovaných lokalit by potenciálně mohlo narušit soukromí majitele stránek. Dotaz vrátí název lokality, počet v ní nalezených čtyřlístků a jejich procentuální podíl na celkovém počtu záznamů. Výsledky jsou seřazeny sestupně podle četnosti výskytu a omezeny na pět nejfrekventovanějších.

Další funkcí je **firstLastYear**, která vrátí záznam prvního a posledního čtyřlístku nalezeného v každém jednotlivém roce. Funkce provádí vícestupňový SQL dotaz využívající

tzv. společné tabulkové výrazy (CTE). První z nich je `yearly_records`, který extrahuje rok z časového razítka každého záznamu v tabulce „records“. Následují dva výrazy `first_records` a `last_records`, jež pomocí konstrukce `DISTINCT ON (year)` vybírají z každého roku první, respektive poslední záznam podle časového razítka. Výsledkem je tabulka, která pro každý rok obsahuje identifikátor a časové razítko prvního i posledního záznamu.

Následuje funkce `earliestLatestHour`, která vrací záznamy čtyřlístků nalezených v nejčasnější a nejpozdější denní dobu. Jinými slovy, cílem funkce je zjistit, který čtyřlístek byl nalezen nejdříve ráno a který nejpozději večer. Funkce provádí dva oddělené SQL dotazy, každý pro jeden z časových extrémů. Dotaz `earliestQuery` vyhledává čtyřlístek s nejdřívějším časem nálezů, zatímco dotaz `latestQuery` hledá záznam s nejpozdějším časem. V obou případech je časové razítko s celým datumem převedeno na čistý čas (bez dní, měsíců a roků), díky čemuž lze nezávisle na datu identifikovat minimální a maximální hodnotu časového údaje. Funkce vrací odpověď ve formátu JSON obsahující identifikátor a časové razítko nejčasnějšího a nejpozdějšího záznamu.

Další funkcí definovanou v souboru „queries.js“ je `mostInDay`, která hledá den, během něhož bylo nalezeno nejvíce čtyřlístků. Jedná se o jednoduchý SQL dotaz, jenž seskupí všechny záznamy v tabulce „records“ podle dne bez ohledu na čas, pro každý den sečte počet nálezů a dny poté sestupně seřadí. Navraceno je datum dne s nejvyšším počtem nálezů a tento počet.

Dále je definována čtveřice funkcí, které poskytují přehled o počtu nálezů v různých časových úsecích. Všechny tyto funkce pracují pouze s časovým razítkem záznamů v tabulce „records“, avšak každá s ním nakládá odlišným způsobem. První je definována funkce `byYear`, která z časového razítka každého nálezů nejdříve pomocí funkce `EXTRACT (YEAR FROM timestamp)` vyextrahuje rok a následně pro každý jednotlivý rok sečte počet nálezů. Výsledkem je přehled celkového počtu nálezů za jednotlivé roky.

Následuje funkce `byMonth`, která obdobně z časového razítka nálezů získá pomocí funkce `EXTRACT (MONTH FROM timestamp)` údaj o měsíci a záznamy pak pro každý měsíc sečte nezávisle na roce či dni. Výsledkem je tedy souhrn, kolik čtyřlístků bylo nalezeno v jednotlivých měsících bez ohledu na rok, a funkce tedy slouží např. k určení, které měsíce bylo nalezeno nejvíce čtyřlístků.

Funkce `byDay` seskupuje nálezy nikoliv podle pořadového čísla dne v měsíci, ale podle dne v týdnu. Je použit složitější výraz $(\text{EXTRACT}(\text{DOW FROM timestamp}) + 6) \% 7 + 1$, který převádí dny týdne na číselné hodnoty v rozsahu 1-7 a zároveň zajišťuje převod z amerického systému číslování dnů, kde je za první den týdne považována neděle, na evropský systém, kde je za první den považováno pondělí. To je důležité pro správné vykreslení statistik na veřejné části webu (viz kapitolu 6.7.4 `statistics.html`). Dotaz záznamy seskupí podle těchto čísel a vrátí počet nálezů pro každý den v týdnu.

Poslední ze čtveřice funkcí je funkce `byHour`, která pomocí výrazu `EXTRACT (HOUR FROM timestamp)` seskupuje záznamy podle hodiny nálezu a vrátí počet záznamů pro každou celou hodinu dne, nezávisle na konkrétním dni, měsíci či roce; může například vrátit údaj, že mezi 14.00 a 15.00 bylo za celou dobu evidence nalezeno 99 čtyřlístků.

Poslední funkcí definovanou v souboru „`queries.js`“ je `mostInHour`, jejímž cílem je určit, v kterou konkrétní hodinu konkrétního dne bylo nasbíráno nejvíce čtyřlístků. Funkce tedy nekonkuruje předchozí funkci `byHour`, která agreguje počty nálezů podle hodin napříč všemi dny, ale snaží se najít absolutně nejproduktivnější hodinový úsek v celé historii evidence. Podobně jako v předchozích funkcích je i zde z časového razítka extrahována hodina nálezu, tentokrát však v kombinaci s datem. Záznamy jsou následně seskupeny podle celých hodin jednotlivých dnů a je vybrán ten časový úsek, ve kterém bylo nalezeno nejvíce čtyřlístků.

Zcela na konci souboru jsou všechny funkce – `top5Locations`, `firstLastYear`, `earliestLatestHour`, `mostInDay`, `byYear`, `byMonth`, `byDay`, `byHour` a `mostInHour` – prostřednictvím exportu modulu zpřístupněny pro použití v jiných částech serverové aplikace. Soubor „`queries.js`“ má celkem (včetně komentářů) 311 řádků a je přiložen jako příloha P IV: Soubor „`queries.js`“.

6.6 pdfGenerator.py

Majitel webové aplikace požadoval možnost generovat z databáze aplikace PDF soubor s vybranými záznamy. Pro implementaci této funkcionality byl vytvořen skript v jazyce Python, jehož použití se jevilo jako jednodušší než realizace stejné funkcionality v jazyce JavaScript.

Soubor „`pdfGenerator.py`“ začíná stejně jako soubory psané v jazyce JavaScript importováním potřebných modulů, jichž je v tomto případě použito poměrně velké množství.

Většina z nich pochází z knihovny `reportlab`, která slouží k tvorbě PDF dokumentů. Jedná se o moduly – `pdfbase.pdfmetrics` a `pdfbase.ttfonts`, které umožňují registraci a používání vlastních TrueType fontů, `pdfgen.canvas` umožňující vykreslování textu, tvarů a obrázků, `lib.pagesizes`, jenž poskytuje přednastavené rozměry stránek, v našem případě formát klasického dopisního papíru (`letter`), a `lib.utils` usnadňující načítání a vkládání obrázků. Modul `psycpg2` je importován pro připojení k PostgreSQL databázi. Kromě toho jsou použity tři standardní knihovny jazyka Python: `sys` a `os` pro práci s cestami a parametry příkazové řádky a `datetime` pro práci s datem a časem.

V další části kódu je definována proměnná zajišťující konfiguraci pro připojení k aplikací používané PostgreSQL databázi. V této proměnné jsou uloženy přihlašovací údaje k databázovému serveru, jako je jeho adresa, název databáze, uživatelské jméno, heslo a číslo portu, na kterém server naslouchá. Dále jsou v různých proměnných specifikovány tři cesty – k adresáři obsahujícímu mapy s místy, kde byly nalezeny čtyřlístky (viz kapitulu 6.7.6 `upload.html`), a k adresáři obsahujícímu dva TrueType fonty: běžnou a tučnou variantu písma *Arial Unicode*, které jsou nezbytné pro správné vykreslení textu v českém jazyce. Tyto fonty jsou následně registrovány pomocí modulu `pdfmetrics` a zpřístupněny pod názvy `Arial` a `Arial-Bold`.

Dále je definována první funkce `fetch_records_in_range`, která přijímá dva parametry – `min_id` a `max_id`, jež dohromady určují rozsah záznamů zahrnutých do generovaného PDF souboru. Společně s identifikátory záznamů jsou pomocí SQL dotazu získány také časové razítko, zeměpisné souřadnice a cesta k miniatuře fotografie z tabulky „records“ a název lokace z tabulky „locations“ pro každý záznam. V případě chyby je vypsána chybová hláška a funkce vrátí `None`.

Následně jsou v souboru definovány tři pomocné funkce pro formátování údajů získaných z databáze. Funkce `decimal_to_dms` převádí souřadnice uložené v desetinném formátu na formát DMS obsahující stupně, minuty a vteřiny. Funkce nejprve oddělí celočíselnou část jako počet stupňů, následně přepočítá desetinnou část na minuty a zbývajících zlomek minut převede na vteřiny. Výstup této funkce je využit v rámci funkce druhé, `format_coordinates`, která doplňuje převedené souřadnice o písmeno značící příslušnou světovou stranu. Výstupem funkce je textový řetězec ve formátu: „49° 13‘ 51.0“ S“. Poslední z trojice funkcí je `format_timestamp`, která modifikuje v databázi uložené časové razítko, obohacuje ho o název dne v týdnu a nahrazuje v něm číselné označení měsíce jeho slovním

vyjádřením. Výsledný formátovaný řetězec má podobu: „středa 23. dubna 2025 16:26:40“. Obdobné funkce byly vytvořeny také v souboru „uploadData.js“ a jsou popsány v kapitole 6.3 uploadData.js.

Následuje de facto předposlední funkce `draw_image_preserve_aspect`, která do PDF souboru přidává ke každému z generovaných záznamů mapu místa nálezu, jež je v levém horním rohu překryta miniaturou fotografie nalezeného čtyřlístku. Funkce za parametry přijímá plátno PDF dokumentu (`c`), cestu k obrázku mapy (`mapPath`), souřadnice `x` a `y` určující umístění obrázku na stránce, požadovanou výšku a šířku obrázku, a volitelně také cestu k miniatuře fotografie. Pomocí nástroje `ImageReader` obsaženého v modulu `reportlab.lib.utils` jsou nejprve zjištěny původní rozměry obrázku s mapou a vypočítán poměr jeho stran, který je zachován i při následném zmenšení, aby nedošlo k jeho deformaci. Následně jsou vypočítány výsledné rozměry obrázku tak, aby co nejlépe vyplnil dostupný prostor, obrázek je zmenšen a umístěn na správné místo v PDF souboru. V případě, že byla nalezena cesta k miniatuře fotografie čtyřlístku, je zmenšena i ta na velikost odpovídající 23 % šířky a 36 % výšky mapy a umístěna do jejího levého horního rohu. Pokud není miniatura fotografie čtyřlístku nalezena, je do konzole vypsána chybová hláška, dokument je však i přesto vygenerován bez ní.

Poslední funkcí v souboru je `generate_pdf`, která je zodpovědná za samotné generování výstupního PDF souboru. Za své vstupní parametry přijímá seznam záznamů a cestu k vygenerovanému souboru. Na začátku funkce jsou definovány rozměry jednotlivých prvků stránky, jako je výchozí pozice textu, rozestupy mezi jednotlivými záznamy nebo velikost obrázku mapy. Každý záznam je následně vykreslen do jednoho ze šesti dostupných bloků na stránce – každá stránka souboru je rozdělena na dva sloupce po třech řádcích. U každého záznamu jsou uvedeny následující informace: jeho pořadové číslo (identifikátor), mapa s miniaturou fotografie (pomocí funkce `draw_image_preserve_aspect`), datum a čas nálezu ve zformátované podobě, zformátované zeměpisné souřadnice a název lokace. Soubor je nakonec uložen a do konzole je vypsán výsledek procesu. Ukázku výsledného PDF dokumentu můžeme nalézt na konci kapitoly 6.7.6 upload.html.

Soubor končí definicí funkce `main()`, která představuje výchozí bod pro spuštění celého skriptu. V ní je nejprve zkontrolován uživatelem zadaný rozsah záznamů pro generování a určen název výsledného souboru – „records_report.pdf“. Následně je zavolána funkce `fetch_records_in_range` a po jejím úspěšném dokončení je výsledek předán

jako parametr funkci `generate_pdf`. Při neúspěšném dokončení funkce `fetch_records_in_range` je na konzoli vypsána chybová hláška. Zcela na konci souboru je pomocí konstrukce `if __name__ == "__main__": main()` zajištěno, že se funkce `main()` spustí pouze při přímém spuštění skriptu.

Soubor „pdfGenerator.py“ má včetně komentářů 208 řádků a je k dispozici v příloze P V: Soubor „pdfGenerator.py“.

6.7 Adresář public

V adresáři „public“ se nachází celkem dvanáct souborů – šest HTML souborů, pět souborů s příponou .svg a jeden soubor ve formátu PNG –, které dohromady tvoří uživatelské webové rozhraní.

Všechny soubory ve formátu SVG – „comment_icon.svg“, „map_icon.svg“, „menu_icon.svg“, „photo_icon.svg“ a „secret_icon.svg“ – slouží jako ikony a jejich účelem je působit na návštěvníka stránek moderním, profesionálním a estetickým dojmem. Všechny ikony jsou volně dostupné a byly staženy ze stránek SVG Repo LLC [17]. Soubor ve formátu PNG – „logo.png“ – obsahuje obrázek loga stránek, který poskytl jejich majitel.

HTML soubory tvoří samotné uživatelské rozhraní stránek. Čtyři soubory – „main.html“, „records.html“, „list.html“ a „statistics.html“ – představují veřejnou část webu; zbývající dva soubory – „login.html“ a „upload.html“ – tvoří administrátorskou část webu. Všechny HTML soubory začínají jednotnou deklarací typu dokumentu HTML5 a nastavením kódování znaků na UTF-8. Jednotlivé HTML soubory jsou podrobně popsány v samostatných subkapitolách.

Soubory, které tvoří veřejnou část webu, jsou většinou popisovány „odspodu“: nejprve je komentována část s podpůrným vnořeným JavaScriptovým kódem, po ní následuje popis grafického zpracování stránky, které je primárně zajištěno kaskádovými styly v hlavičce dokumentu. Na všech veřejných stránkách je dále v hlavičce načten externí bezpatkový font *Raleway*, importovaný ze služby Google Fonts pro sjednocení typografie.

Barevné ladění veřejných stránek je zeleno-žluté. Gradient různých odstínů zelené barvy je většinou použit na pozadí stránek, zatímco žlutá barva je vyhrazena pro pozadí funkčních bloků. Všechny veřejně dostupné stránky obsahují navigační menu reprezentované malým žlutým čtvercem v pravém horním rohu stránky s ikonou menu.

Tento blok se po kliknutí rozbálí do většího čtverce a zobrazí odkazy na tři zbývající stránky. Po kliknutí na kterýkoliv odkaz je uživatel automaticky přesměrován na příslušnou stránku.

6.7.1 main.html

Soubor „main.html“ poskytuje domovskou stránku veřejné části webové aplikace.

Část s vnořeným skriptem obsahuje pouze funkci `toggleMenu()`, která zajišťuje rozbalení a sbalení navigačního menu.

Navigační menu, včetně ikony a odkazů na ostatní stránky, je definováno v těle HTML dokumentu společně s úvodním textem. Ten vítá návštěvníka na stránkách Čtyřlístkotéky a předkládá informaci o jejich účelu, majiteli a autorovi. Text je doplněn logem stránek.

Stránku nelze „scrollovat“ – vertikálně ani horizontálně posouvat – neboť je navržena tak, aby se snažila vždy automaticky přizpůsobit výšce obrazovky zařízení. Výchozí pozadí stránky je tmavě zelené, ale díky animovanému přechodu (vertikálnímu gradientu) se postupně mění na světlejší odstíny a pak zpět. Tento efekt imituje plynulý pohyb a přidává tak jinak statické stránce mírnou dynamičnost a vizuální zajímavost. Efektu je dosaženo pomocí CSS animace `@keyframes gradientAnimation`. Ve spodní polovině stránky je prezentováno logo aplikace, nad kterým se nachází zmiňovaný uvítací text s mírným stínováním a v pravém horním rohu se nachází malý žlutý čtverec s navigačním menu.

Soubor „main.html“ má celkem 141 řádků a je k dispozici v příloze P VII: Soubor „main.html“. Podobu domovské stránky můžeme vidět na obrázku 2 Domovská stránka.



Obrázek 2 Domovská stránka

(zdroj: vlastní)

6.7.2 records.html

Soubor „records.html“ vytváří stránku Záznamy, na které si může návštěvník zobrazit údaje o konkrétním čtyřlístku.

Vnořený skript začíná importováním externí knihovny Leaflet vyvinuté Vladimírem Agafonkinem [18], která slouží k vykreslování interaktivních map (s využitím podkladových dat z projektu OpenStreetMap [19]), a pokračuje definováním funkce `toggleMenu()` pro ovládání navigačního menu.

Dále je definována funkce `initializeMap`, jež zajišťuje inicializaci mapy v místě nálezu vyhledaného čtyřlístku. Jsou-li do funkce jako parametry předány anonymizované souřadnice s hodnotami rovnajícími se 0.0000 (decimální tvar), jsou tyto souřadnice automaticky nahrazeny souřadnicemi 49.2309 pro zeměpisnou šířku a 17.6571 pro zeměpisnou délku, na kterých se nachází sídlo Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně. V mapě je zobrazena šipka označující přesné místo nálezu, doplněná textem „Nalezen zde“. V případě anonymizovaného záznamu se popisěk změní na „Nález anonymizován“. Funkce rovněž zajišťuje odstranění zobrazené mapy při načítání nového záznamu.

Nejdůležitější část skriptu tvoří funkce `retrieveData`, která slouží k asynchronnímu načtení dat z databáze na základě zadaného ID záznamu. Jedná se o poměrně složitou funkci, neboť zajišťuje nejen správné formátování a dynamické vykreslování údajů, ale také nahrazování výchozího loga a textu za každou novou fotografii čtyřlístku, která je, je-li na ni namířeno kurzorem, automaticky nahrazena její miniaturou. Funkce také pochopitelně ošetřuje vyhledávání záznamů, které byly anonymizovány, a v takových případech zobrazí pouze fotografie, status a čas nálezu. Ostatní údaje jsou nahrazeny oznámením o jejich anonymizaci a mapou ukazující Fakultu aplikované informatiky Univerzity Tomáše Bati ve Zlíně. Funkce rovněž ošetřuje případné chyby při získávání dat z databáze. Na konci skriptu je implementován posluchač událostí, který kontroluje platnost vstupního ID a v případě úspěšné validace spouští funkci `retrieveData`.

Stejně jako domovská stránka je i stránka Záznamy ve své výchozí podobě navržena tak, aby automaticky vyplnila celou výšku obrazovky zařízení a není na ní tedy možné vertikální ani horizontální posouvání („scrollování“). Také zde je zachován gradient pozadí, tentokrát však ve své statické podobě – tmavě zelená barva v horní části přechází směrem dolů ve světle zelenou.

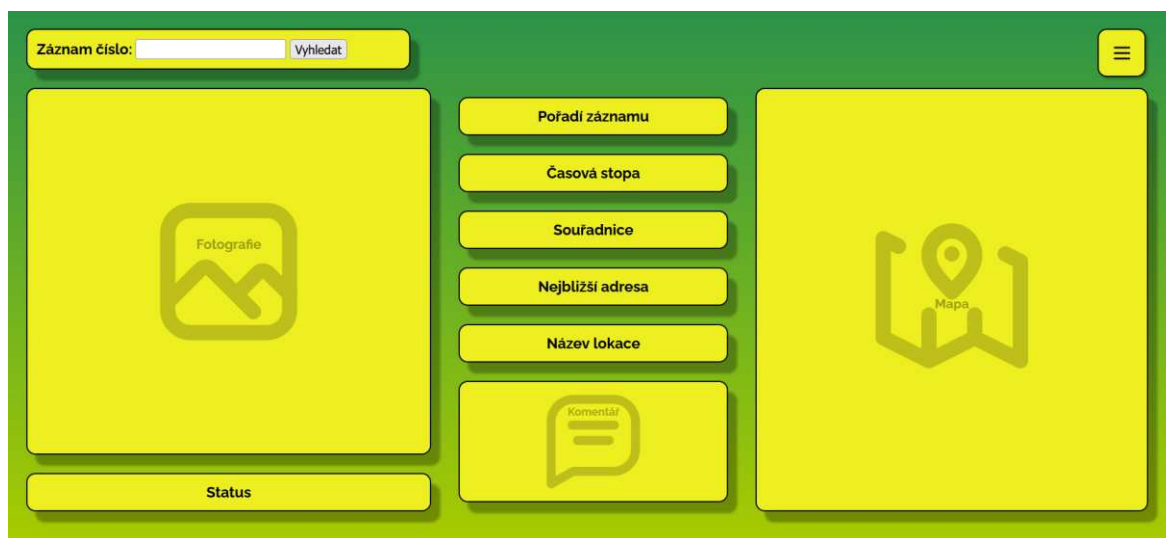
Po vyčlenění úzkého pásu v horní části stránky je zbývající prostor rozdělen do tří pomyslných vertikálních sloupců. Levý a pravý sloupec zabírají přibližně stejný prostor, zatímco střední je proporčně o něco užší. Pravý sloupec je určen výhradně pro blok, do kterého je po vyhledání konkrétního záznamu načtena interaktivní mapa místa nálezu. Ve výchozím stavu je uprostřed bloku zobrazeno stylizované logo s textem „Mapa“.

Levý sloupec je z většiny vyhrazen pro blok s fotografií čtyřlístku, případně její miniaturu. Ve výchozí podobě je uprostřed bloku zobrazeno logo s textem „Fotografie“. Pod tímto blokem se nachází menší obdélníková sekce pro výpis statusu záznamu.

Střední sloupec je rozdělen na pět stejně velkých obdélníkových bloků a jeden větší blok v dolní části pro zobrazení komentáře k místu nálezu. Ve výchozí podobě obsahuje tento spodní blok uprostřed logo s textem „Komentář“. Zbývající bloky slouží postupně (shora dolů) k zobrazení identifikačního čísla (ID) záznamu, zobrazení formátovaného časového razítka, zobrazení formátovaných zeměpisných souřadnic, nejbližší adresy a názvu lokace.

Horní část stránky obsahuje dvě ovládací sekce – vpravo se nachází čtvereček s navigačním menu, zatímco vlevo je umístěn obdélníkový blok s formulářem pro vyhledání záznamu dle jeho identifikačního čísla. Všechny bloky mají žlutou barvu pozadí, jsou orámovány tmavou linkou, mají zaoblené rohy a stínování, které navozuje dojem prostorové hloubky.

Soubor má celkem 392 řádků a je přiložen v příloze P VIII: Soubor „records.html“. Výchozí podoba stránky je zobrazena na obrázku 3 Výchozí podoba stránky „Záznamy“. Podoba stránky po načtení konkrétního záznamu je zachycena na obrázku 4 Stránka „Záznamy“ s načteným záznamem. Na obrázku 5 Stránka „Záznamy“ s anonymizovaným záznamem je pak vidět podoba stránky po vyhledání anonymizovaného záznamu.



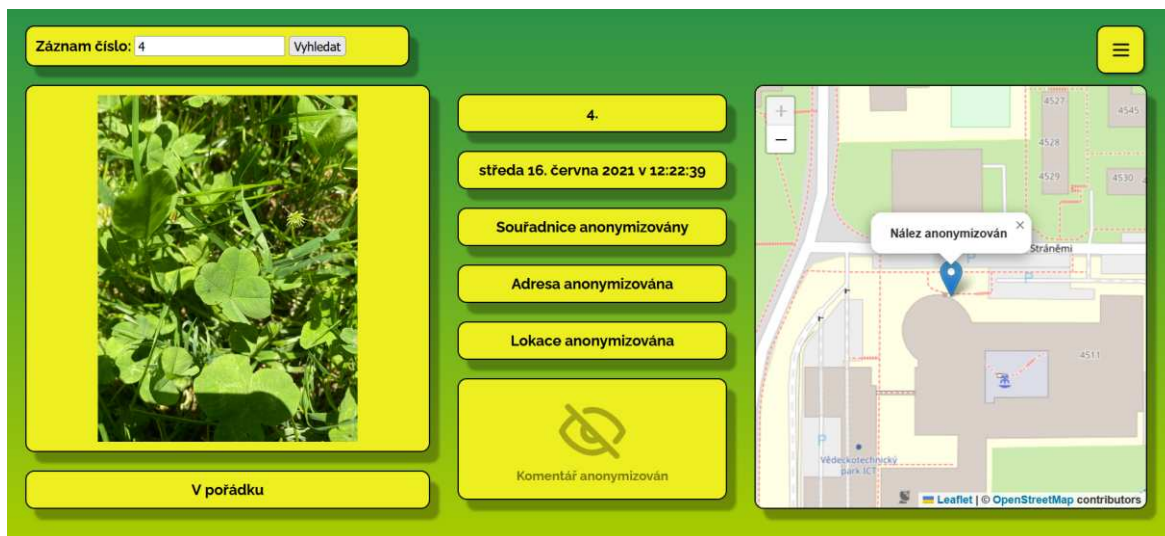
Obrázek 3 Výchozí podoba stránky „Záznamy“

(zdroj: vlastní)



Obrázek 4 Stránka „Záznamy“ s načteným záznamem

(zdroj: vlastní)



Obrázek 5 Stránka „Záznamy“ s anonymizovaným záznamem

(zdroj: vlastní)

6.7.3 list.html

Soubor „list.html“ slouží ke generování stránky Seznam záznamů, na níž je ve formě tabulky zobrazen přehled všech do databáze uložených čtyřlístků. Vnořený skript začíná definicí globálních proměnných, a to `currentPage` pro uchování čísla aktuálně zobrazené stránky seznamu (s počáteční hodnotou 1), konstanty `recordsPerPage`, která omezuje maximální počet záznamů zobrazovaných na jedné stránce na 50, a proměnné `records` pro pole obsahující všechny načtené záznamy. Dále je definována funkce `toggleMenu()`, sloužící k ovládání navigačního menu.

Následuje sada funkcí zajišťujících přepínání mezi stránkami seznamu. Funkce `goToFirstPage()` uživatele vždy jednoduše navrací na první stránku seznamu. Funkce `goToLastPage()` dělí celkový počet záznamů hodnotou `recordsPerPage`, čímž zjistí číslo stránky, na které se nachází poslední nahraný záznam, a uživatele na ni přesměruje. Funkce `goToPreviousPage()` a `goToNextPage()` umožňují přepínání na předchozí, respektive následující stránku. Obě funkce zároveň kontrolují, zda uživatel nechce přejít na neexistující stránku – spodní hranice je omezena hodnotou 1, horní je určena stejným výpočtem jako ve funkci `goToLastPage()` – a tento přechod případně znemožňují.

Následuje dvojice funkcí, které dohromady umožňují přechod na libovolnou existující stránku seznamu. Funkce `validatePageInput()` kontroluje, zda je uživatelem zadané číslo stránky v rozsahu existujících stránek, a pakliže není, deaktivuje spuštění funkce `goToPage()`, která uživatele na požadovanou stránku přesměruje.

Všechny výše popsané navigační funkce spouštějí při validním vstupu funkce `renderTable()` a `updatePaginationControls()`. Funkce `renderTable()` dynamicky vykresluje požadovanou stránku seznamu včetně příslušných záznamů. Pro každý záznam je vytvořen jeden HTML řádek obsahující identifikační číslo (ID), fotografii miniaturny, status, formátovanou časovou známku, zeměpisnou šířku, zeměpisnou délku, nejbližší adresu, název lokace a komentář k lokaci. Funkce `updatePaginationControls()` aktualizuje číslo aktuální stránky zobrazené v uživatelském rozhraní.

Na konci skriptu je implementován posluchač událostí, který odesílá požadavek na serverové API pomocí metody `fetch()`. Vrácená data jsou převedena do formátu JSON a následně uložena do proměnné `records`, přičemž jsou seřazena vzestupně podle identifikačního čísla (ID). Tabulka je poté vykreslena. V případě chyby během načítání dat je v konzoli prohlížeče vypsáno chybové hlášení.

Stejně jako na domovské stránce a stránce Záznamy, je i na stránce Seznam záznamů zachován barevný přechod pozadí (gradient). Jelikož je ale nutné stránku vertikálně posouvat („scrollovat“), je přechod mezi jednotlivými odstíny zelené ve vertikálním směru jemnější; gradient je ale na této stránce aplikován i ve směru horizontálním.

Stránka obsahuje tabulku se záznamy, přičemž je v její hlavičce popsán každý sloupec. Šířka jednotlivých sloupců je v kódu explicitně nastavena pomocí kaskádových stylů. Tabulka má žluté pozadí a její hlavička je zvýrazněna světle zelenou barvou. Tabulka je orámována černou linkou a nemá zaoblené rohy. Tělo tabulky je v HTML kódu ponecháno prázdné, neboť je jeho obsah dynamicky doplňován pomocí skriptu.

V horní části stránky se nachází již zaoblený a stínovaný blok se žlutým pozadím sloužící jako hlavní nadpis stránky (s jejím názvem tj. „Seznam záznamů“). Pod ním se nachází šest horizontálně seřazených prvků pro ovládání stránkování. Konkrétně se jedná o následující prvky jdoucí zleva doprava: tlačítko pro přechod na první stránku, tlačítko odkazující na předchozí stránku, formulář pro zadání čísla stránky, tlačítko pro potvrzení přechodu na konkrétní stránku, tlačítko pro přechod na další stránku a tlačítko odkazující na poslední stránku seznamu. Tlačítka vizuálně korespondují se stylem nadpisu a při najetí kurzorem mění barvu, čímž uživateli poskytují vizuální zpětnou vazbu. V pravém horním rohu se již tradičně nachází blok s navigačním menu.

Soubor „list.html“ má 343 řádků a je uveden v příloze P IX: Soubor „list.html“. Ukázka jedné ze stránek testovací databáze je zachycena na obrázku 6 Stránka „Seznam záznamů“.

Záznam	Miniatura	Status	Časová stopa	Zeměpisná šířka	Zeměpisná délka	Nejbližší adresa	Název lokace	Komentář
651.		V pořádku	čtvrtek 25. května 2023 v 07:18:05	49° 13' 51.4" S	17° 39' 24.9" V	Nad Stráněmi, 760 05 Zlín	10. lokace	Poslední testovací lokace.
652.		V pořádku	čtvrtek 25. května 2023 v 07:19:10	49° 13' 52.3" S	17° 39' 24.4" V	5656 Nad Stráněmi, 760 05 Zlín	1. lokace	Komentář k první testovací lokaci
653.		V pořádku	čtvrtek 25. května 2023 v 07:19:17	0° 0' 0.0" S	0° 0' 0.0" V	Adresa anonymizována	Lokace anonymizována	Komentář anonymizován
654.		V pořádku	čtvrtek 25. května 2023 v 07:19:59	49° 13' 51.4" S	17° 39' 24.7" V	Družstevní, 760 05 Zlín	10. lokace	Poslední testovací lokace.
655.		V pořádku	čtvrtek 25. května 2023 v 07:20:28	0° 0' 0.0" S	0° 0' 0.0" V	Adresa anonymizována	Lokace anonymizována	Komentář anonymizován
656.		V pořádku	čtvrtek 25. května 2023 v 07:21:02	49° 13' 51.3" S	17° 39' 24.8" V	Nad Stráněmi, 760 05 Zlín	Lokace 6.	V komentáři k šesté lokaci testuji zobrazení delšího komentáře

Obrázek 6 Stránka „Seznam záznamů“

(zdroj: vlastní)

6.7.4 statistics.html

Soubor „statistics.html“ generuje stránku, která návštěvníkům zobrazuje statistiky o databázi čtyřlístků. Vnořený skript začíná importem externí knihovny Chart.js, která je vydaná pod licencí MIT a slouží pro generování grafů. Následuje definice funkce `toggleMenu()`, která slouží k ovládání navigačního menu. Zbytek skriptu je tvořen asynchronními funkcemi, jež prostřednictvím API požadavků volají funkce popsané v kapitole 6.5 `queries.js`.

První z těchto funkcí je `fetchTopLocations()`, která vrací data o pěti lokacích, kde bylo nalezeno nejvíce čtyřlístků. Sestupně seřazené výsledky, jež obsahují název lokace, počet nalezených čtyřlístků a jejich procentuální podíl na celé sbírce, jsou dynamicky vykresleny do tabulky ve žlutém bloku. Blok obsahuje také nadpis „Pět nejčastějších lokací“, který je z obou stran ohraničen ikonou mapy. Ukázku bloku s testovacími daty může čtenář vidět na obrázku níže.



Lokace	Nalezeno čtyřlístků	Procentuálně
Lokace 6.	121	10.23 %
3. lokace	118	9.97 %
10. lokace	118	9.97 %
Lokace číslo 8	115	9.72 %
1. lokace	113	9.55 %

Obrázek 7 Blok „Pět nejčastějších lokací“

(zdroj: vlastní)

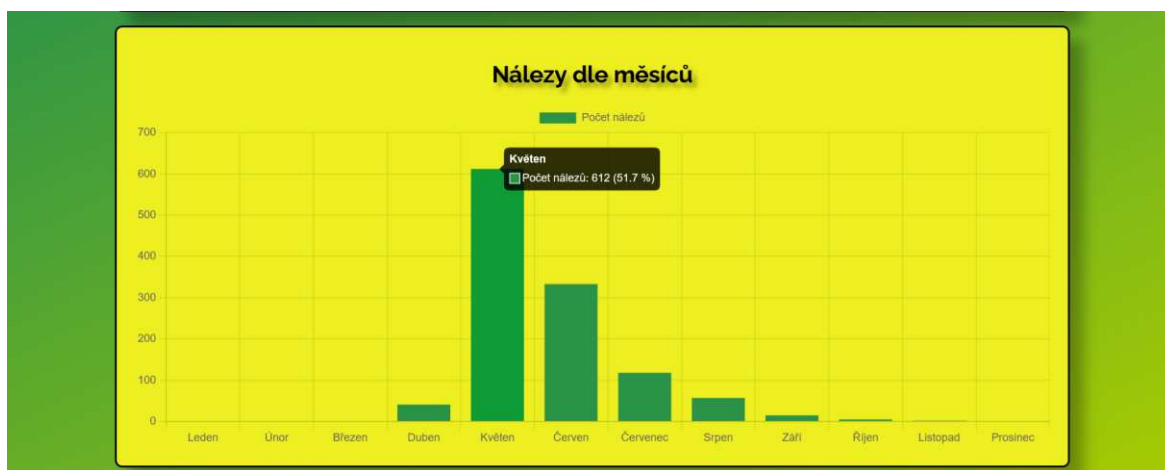
Druhý blok je vyplněn vizualizací dat získaných pomocí funkcí `fetchFindingsByYear()` a `fetchYearlyRecords()`. Větší část tohoto bloku, umístěná na levé straně, je věnována výstupu funkce `fetchFindingsByYear()`, která pod nadpisem „Nálezů za rok“ vykresluje sloupcový graf. Tento graf vizualizuje počet nalezených čtyřlístků v jednotlivých letech, přičemž každý zelený sloupec reprezentuje jeden konkrétní rok a jeho výška (osa y) odpovídá počtu zaznamenaných nálezů. Po najetí kurzorem na konkrétní sloupec se objeví přesný počet nálezů v příslušném roce a jejich procentuální podíl na celkovém počtu čtyřlístků. Menší pravá část bloku, která zabírá 30 % jeho celkové šířky, obsahuje tabulku s výstupem funkce `fetchYearlyRecords()`. Pod příslušným nadpisem je v tabulce zobrazen vždy první a poslední nález daný rok. Každý z těchto nálezů je reprezentován identifikačním číslem (ID) a přesným datem jeho nálezu. Pro ilustraci je blok s načtenými testovacími daty zachycen na obrázku uvedeném níže.



Obrázek 8 Blok s počtem nálezů podle let a s prvním a posledním nálezem v každém roce

(zdroj: vlastní)

Třetí blok je nadepsán „Nálezy dle měsíců“ a jeho obsah je generován pomocí funkce `fetchFindingsByMonth()`. V rámci této funkce musela být inicializována dvě pole o délce dvanácti prvků. Do jednoho z nich jsou ukládány číselné hodnoty reprezentující měsíce převedené do indexového formátu (0-11), a to v souladu s konvencí používanou v jazyce JavaScript. Druhé pole uchovává počty nálezu v jednotlivých měsících. Výsledkem je sloupcový graf, ve kterém jsou jednotlivé měsíce reprezentovány zelenými sloupci, jejichž výška odpovídá počtu nalezených čtyřlístků. Stejně jako u předchozích grafů se při najetí kurzorem na konkrétní sloupec zobrazí přesný počet nálezu za daný měsíc a jejich procentuální podíl na celkovém počtu čtyřlístků ve sbírce. Blok s testovacími daty je pro ukázkou uveden na obrázku níže.



Obrázek 9 Blok „Nálezy dle měsíců“

(zdroj: vlastní)

Předposlední, čtvrtý blok je stejně jako blok druhý rozdělen na dvě části v poměru sedm ku třem. Jeho větší část zabírá graf, ve kterém jsou zelenými sloupci znázorněny počty nalezených čtyřlístků podle jednotlivých dnů v týdnu. Také tento graf je opatřen vlastním nadpisem a při najetí kurzorem na konkrétní sloupec se i zde zobrazí přesný počet nálezu daný den spolu s jejich procentuální podílem na celkovém počtu. Data jsou pro graf zpracována pomocí funkce `fetchFindingsByDay()`, která převede číselné značení dnů v týdnu z rozsahu 1–7 na indexový formát 0–6, jenž odpovídá konvenci v jazyce JavaScript. Výsledky jsou následně ukládány do pole o sedmi prvcích, kde každý prvek reprezentuje jeden den v týdnu a uchovává počet nálezu pro daný den. Funkce zároveň počítá procentuální podíl nálezu pro každý den vůči celkovému počtu. Menší pravá část bloku je pak vyplněna jednoduchou jednořádkovou tabulkou s nadpisem, která prezentuje den, kdy bylo nalezeno nejvíce čtyřlístků. Uvedeno je datum a celkový počet nálezu tento den. Tyto informace jsou

získány pomocí funkce `fetchDayWithMost()`, která ze získaných dat extrahuje datum a převádí jej pomocí metody `toLocaleDateString()` do lépe čitelného formátu. Celý čtvrtý blok je znázorněn na obrázku níže za využití testovacích dat.



Obrázek 10 Blok s počtem nálezů dle dnů v týdnu a se dnem s nejvíce nálezů

(zdroj: vlastní)

Poslední blok je rozdělen na tři části. Horizontálně je rozdělen v poměru 65:35 a větší levou část vyplňuje nadepsaný sloupkový graf, který vizualizuje četnost nálezů dle jednotlivých hodin. Graf je generován funkcí `fetchFindingsByHour()`, jež seskupuje časové údaje dle hodin a zároveň počítá procentuální podíl nálezů v každém hodinovém intervalu vůči jejich celkovému počtu. Každý sloupec reprezentuje jeden hodinový interval, má i zde zelenou barvu a jeho výška odpovídá počtu nálezů, jehož přesný počet i procentuální podíl získáme po najetí kurzorem na konkrétní sloupec. V menší pravé straně bloku se nachází dvě nadepsané tabulky. Data do vrchní z nich jsou vybrána na základě funkce `fetchHourWithMost()`, která vrací absolutně nejproduktivnější hodinový interval ze všech zaznamenaných dnů. V tabulce je uvedeno konkrétní datum, časový úsek a počet nálezů. Spodní tabulka, naplněná prostřednictvím funkce `fetchEarliestLatest()`, zobrazuje identifikátor (ID) a přesný čas nejdříve a nejpozději zaznamenaného nálezu.



Obrázek 11 Blok s počty nálezů dle hodin, se dnem s nejvíce nálezy za hodinu a s nejdřívějším a nejpozdějším nálezem

(zdroj: vlastní)

Zcela na konci skriptu je implementován naslouchač události `DOMContentLoaded`, který po načtení stránky automaticky spustí všechny popsané funkce.

Jelikož byl design stránky Statistiky částečně popsán při funkčním popisu jednotlivých bloků, zbývá už jen shrnout její celkové rozvržení. Na stránce je zachován barevný přechod pozadí (gradient) obdobným způsobem jako na stránce Seznam záznamů, tedy jak ve vertikálním, tak i horizontálním směru. Všechny bloky na stránce mají žluté pozadí, jsou ohraničeny černou linkou, mají zaoblené rohy a jsou doplněny jemným stínem. I na této stránce se v pravém horním rohu nachází minimalizované navigační menu.

Celý soubor „statistics.html“ má celkem 630 řádků a je uveden v příloze P X: Soubor „statistics.html“.

6.7.5 login.html

Přihlašovací stránka do administrátorské části webu je spravována souborem „login.html“ a neobsahuje žádné styly. Na zcela bílé stránce se nachází pouze nadepsaný přihlašovací formulář s jedním vstupním polem typu `password`, které zajišťuje nahrazení psaného textu tečkami, a jedním tlačítkem. Pomocí atributu `required` je zajištěno, že formulář nemůže být odeslán prázdný.

Vnořený skript tvoří jeden posluchač událostí, který zajišťuje autentizaci a autorizaci uživatele. Po zadání hesla a kliknutí na tlačítko je formulář odeslán. Z jeho vstupního pole je načtena hodnota hesla a odeslána ve formátu JSON na server. Pokud se odeslané heslo shoduje s administrátorským heslem uloženým na serveru, je ze serveru navracena pozitivní odpověď společně s autentizačním tokenem a uživatel je automaticky přesměrován na

administrátorskou stránku. Pokud se zadaný text s heslem neshoduje, je uživatel informován o chybě. Autentizace je tedy prováděna na straně klienta prostřednictvím asynchronního HTTP požadavku. Bezpečnost je posílněna umístěním přihlašovací stránky na „nepředvídatelný“ koncový bod s URL adresou typu AAbbl1cc, což ztěžuje automatizované útoky, které by jinak cílily na běžně používané koncové body jako je například /login.

Soubor má pouhých 41 řádků a je k dispozici v příloze P XI: Soubor „login.html“. Na obrázku 12 Přihlašovací stránka je pro ilustraci zobrazen snímek přihlašovací stránky.

Login

max. 5 pokusů za 15 minut

The image shows a simple login interface. At the top, the word "Login" is displayed in a bold font. Below it, a message states "max. 5 pokusů za 15 minut". Underneath this message, there is a text input field labeled "Password" and a button labeled "Login".

Obrázek 12 Přihlašovací stránka

(zdroj: vlastní)

6.7.6 upload.html

Posledním souborem v adresáři „public“, který nebyl doposud popsán, je „upload.html“. Tento soubor vytváří administrátorské uživatelské rozhraní a stránka, kterou generuje, neobsahuje, stejně jako stránka generovaná souborem „login.html“, mnoho vizuálních prvků. Stránka má bílé pozadí a je rozdělena do tří sloupců. Než ale budou jednotlivé sloupce podrobně popsány, popíšeme začátek vnořeného skriptu.

Skript začíná importem dvou externích JavaScriptových knihoven a to knihovnou Leaflet.js, která je podepsána Vladimírem Agafonkinem a slouží pro práci s interaktivními mapami, a jejím rozšířením, knihovnou leaflet-image.js, jež umožňuje převod vykreslené mapy na obrazový formát vhodný k uložení [18]. Obě knihovny jsou importovány ze serveru unpkg.com.

Bezprostředně po načtení knihoven je definována a spuštěna asynchronní funkce `fetchUploadPage()`, která načítá obsah stránky, ovšem pouze uživateli, jenž disponuje platným JSON Web Tokenem. Neověřeného uživatele nebo uživatele s expirovaným tokenem funkce automaticky přesměruje na přihlašovací stránku.

Levý ze tří sloupců je tvořen dvěma vertikálně umístěnými bloky. Vrchní blok, který je ve výchozím stavu sbalen a lze jej rozbalit kliknutím, obsahuje seznam zkratk statusů s jejich významy. Blok slouží uživateli k správnému pojmenovávání nahrávaných souborů, neboť status je do databáze ukládán na základě názvu nahrávaného souboru (viz kapitolu 6.3 `uploadData.js`). Rozbalování a sbalování bloku zajišťuje funkce `toggleStatusList()`, která je spouštěna posluchačem událostí reagujícím na kliknutí. Tato funkce zároveň spouští asynchronní funkci `fetchAndDisplayStatuses()`, jež získává a formátuje seznam statusů prostřednictvím HTTP požadavku na server. Funkce `fetchAndDisplayStatuses()` je ovšem spuštěna také ihned po načtení stránky, společně s níže popsanou funkcí `fetchAndDisplayLocations()`, prostřednictvím jiného posluchače událostí. Spodní blok je zcela statický a obsahuje pouze instrukce pro správné zpětné nahrávání (doplňování) miniatur k již nahraným záznamům. Tyto instrukce jsou formátovány jako očíslovaný seznam. Levý sloupec je, společně s několika dalšími prvky administrátorského rozhraní, zachycen níže na obrázku 13 Vrchní část administrátorské stránky.

Pravý sloupec tvoří blok se seznamem všech v databázi existujících lokací, neboť i ta musí být uživatelem specifikována v názvu nahrávaného souboru (viz kapitolu 6.3 `uploadData.js`). Seznam uživateli tuto práci usnadňuje, neboť zobrazuje všechny informace z databázové tabulky „locations“ (viz kapitolu 6.1 Schéma databáze). Blok funguje na stejném principu jako blok zobrazující statusy – je tedy rozbalován a sbalován funkcí `toggleLocationList()`, kterou spouští posluchač událostí reagující na kliknutí. Tato funkce zároveň spouští asynchronní funkci `fetchAndDisplayLocations()`, jež načítá a formátuje data z databáze prostřednictvím HTTP požadavku na server. Část pravého sloupce zobrazujícího data z testovací databáze je společně s několika dalšími prvky administrátorského prostředí zachycena níže na obrázku 13 Vrchní část administrátorské stránky.

Největší prostřední sloupec se skládá ze šesti bloků, které zajišťují klíčové funkční prvky stránky. První z nich obsahuje formulář pro nahrávání fotografií na server. Formulář je ovládán dvěma tlačítky. Prvním je tlačítko označené anglickým výrazem „Browse...“,

kteřé slouží k výběru fotografií z lokálního adresáře. Vybrané fotografie jsou v cyklu vkládány do objektu typu `FormData` a akceptovány jsou pouze soubory s příponou `.heic`. Jakmile jsou fotografie do objektu vloženy, je jejich celkový počet uveden v krátkém anglickém textu vedle tlačítka. Fotografie se nahrají stiskem druhého tlačítka „Upload“, které vytvoří asynchronní HTTP požadavek na server. Správné kódování pro odesílání binárních dat zajišťuje atribut `enctype="multipart/form-data"`. Celá funkcionality je realizována naslouchačem událostí. Blok „Upload Files“ pro nahrávání fotografií je společně s několika dalšími prvky administrátorského rozhraní zachycen níže na obrázku 13. Vrchní část administrátorské stránky.

Druhý blok slouží k ukládání nových lokací do databáze. Uživatel do formuláře zadává název nové lokace, komentář k ní a pomocí zaškrtnutí políčka určuje, zda má být lokace anonymizována. Poté kliknutím na tlačítko „Nahrát novou lokaci“ lokaci uloží prostřednictvím asynchronního HTTP požadavku na server. O funkcionality se, stejně jako v předchozím případě, stará naslouchač událostí. Společně s několika dalšími prvky administrátorského rozhraní je blok „Upload Location“ zachycen na obrázku 13. Vrchní část administrátorské stránky.

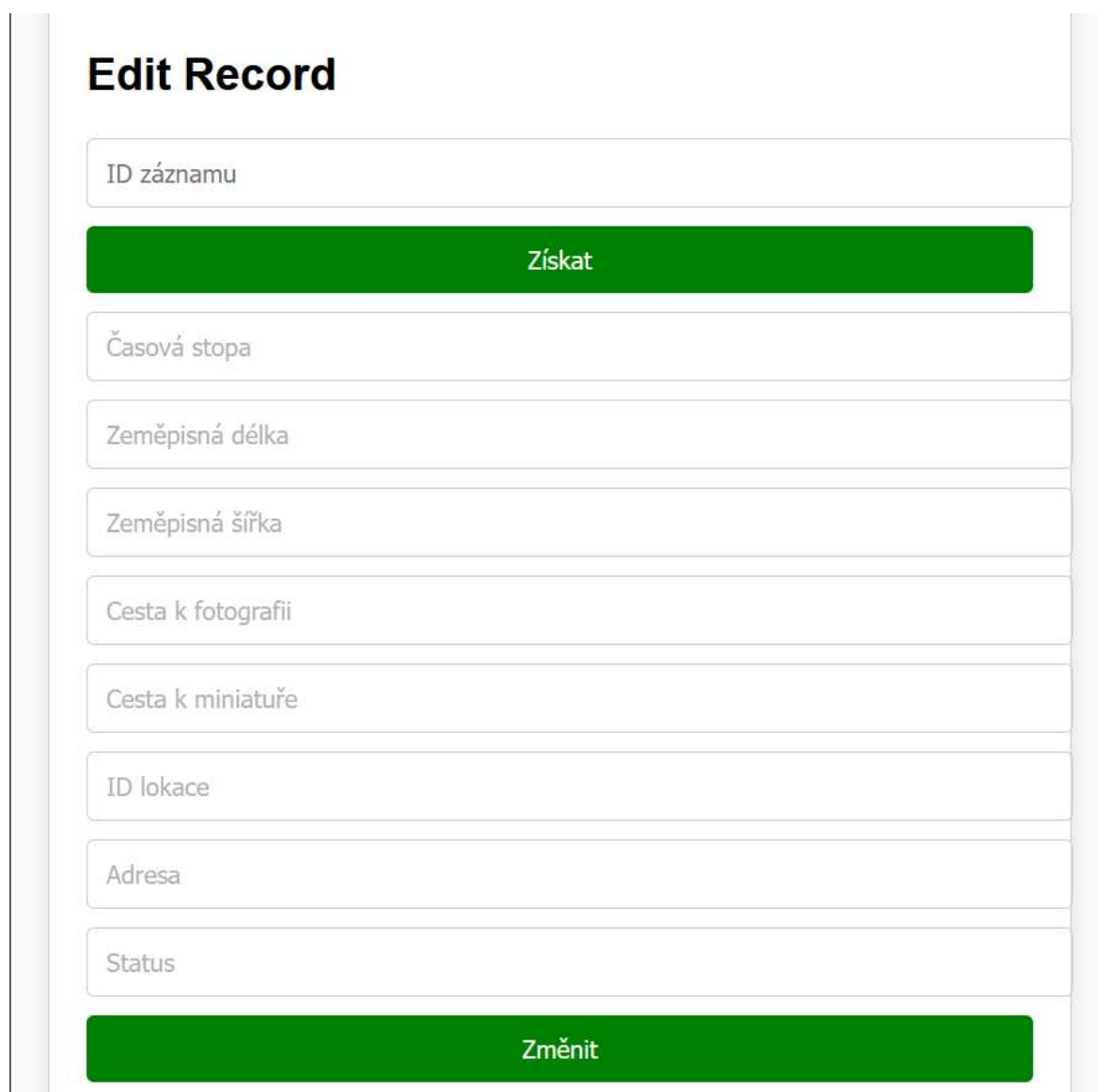
The screenshot displays the top portion of an administrative web interface, organized into three main vertical panels. The leftmost panel contains a list of filter buttons: 'V - V pořádku', 'D - Darován', 'Z - Ztracen', 'N - Nevyfocen', 'L - Bez zadané lokace', 'G - Byl mi darován', 'S - Status není určitelný', and 'J - S jinou fotografií'. Below these is a text block explaining the HEIC upload process and a numbered list of steps. The middle panel features two sections: 'Upload files' with a 'Browse...' button, a '24 files selected.' status, and an 'Upload' button; and 'Upload Location' with input fields for 'Název nové lokace' and 'Přidat komentář k lokaci', an 'Anonymizovat lokaci:' checkbox, and a prominent green 'Nahrát novou lokaci' button. The rightmost panel shows a list of existing locations, with three entries visible: 'ID lokace: 1' (Name: 1. lokace, Comment: První lokace - neanonymní, Anonymized: Ne), 'ID lokace: 2' (Name: 2. lokace, Comment: Údaje nepůjdou mimo administrátorské prostředí vidět, Anonymized: Ano), and 'ID lokace: 3' (Name: 3. lokace).

Obrázek 13 Vrchní část administrátorské stránky

(zdroj: vlastní)

Třetí blok „Edit Record“ slouží k editaci již nahraných záznamů z webového rozhraní. Pod nadpisem se nachází vstupní pole, do kterého uživatel zadá identifikační číslo (ID) záznamu, který chce upravit, a po stisknutí tlačítka „Získat“ dojde k odeslání asynchronního požadavku `GET` na server, který do kolonek pod ním načte všechny ostatní údaje z databáze.

Po provedení požadovaných změn může uživatel aktualizované údaje nahrát do databáze stiskem tlačítka „Změnit“, které spouští asynchronní požadavek PUT na server. Funkcionalitu zajišťuje robustní naslouchač událostí, který mimo jiné zpřístupňuje kolonky (kromě vstupního pole pro identifikační číslo) až po úspěšném načtení konkrétního záznamu. Naslouchač také kontroluje, zda uživatel nezadá do formuláře prázdné hodnoty a pokud ano, nahrání změn znemožní. Blok „Edit Record“ je pro ilustraci zobrazen na obrázku 14 Blok „Edit Record“.



Edit Record

ID záznamu

Získat

Časová stopa

Zeměpisná délka

Zeměpisná šířka

Cesta k fotografii

Cesta k miniatuře

ID lokace

Adresa

Status

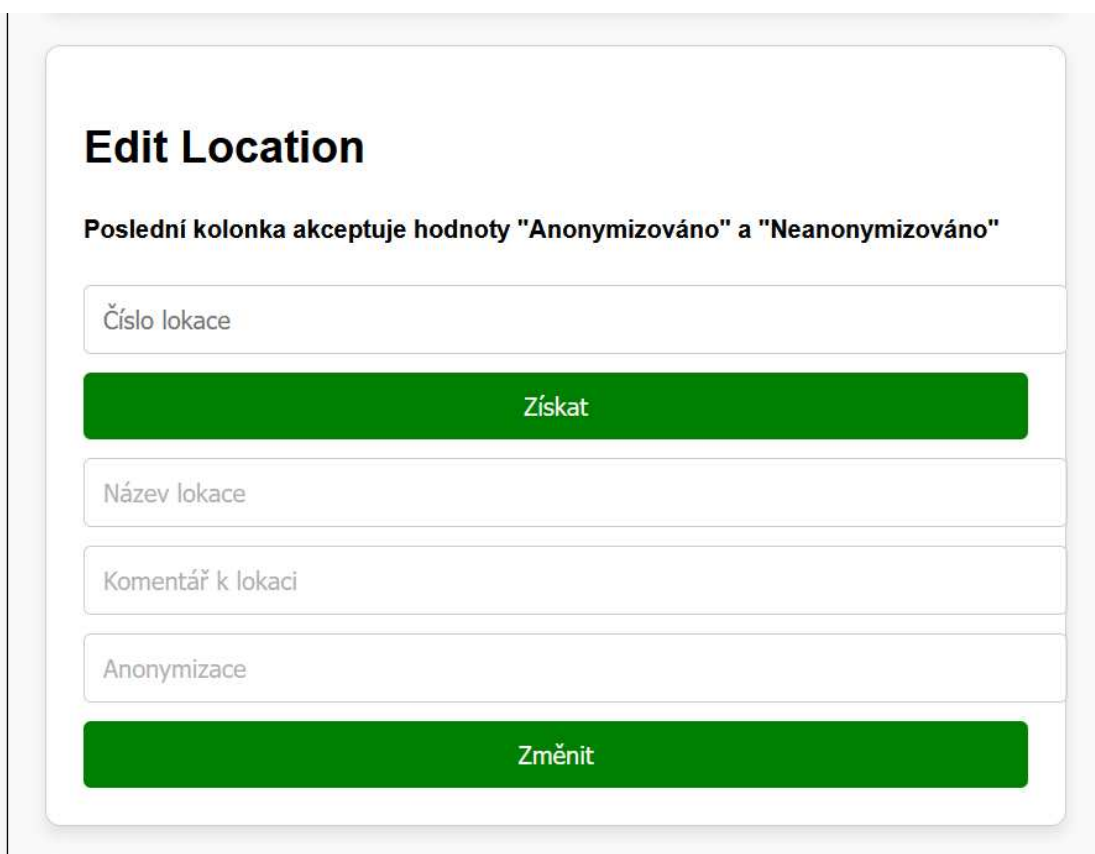
Změnit

Obrázek 14 Blok „Edit Record“

(zdroj: vlastní)

Jako další je uveden blok pro editaci lokací. Blok „Edit Location“ funguje na stejném principu jako blok „Edit Record“. Uživatel nejprve zadá identifikační číslo (ID) lokace,

podle kterého je po kliknutí na tlačítko „Získat“ načten požadovaný záznam z tabulky „locations“ v databázi. Hodnota pro anonymizaci je převedena na jednu z textových hodnot „Anonymizováno“ nebo „Neanonymizováno“, které je nutné použít i při případné editaci této kolonky. Po načtení záznamu se uživateli zpřístupní vstupní pole pro editaci záznamu, přičemž změněné, ne však prázdné, hodnoty může uložit do databáze stisknutím tlačítka „Změnit“. Funkcionalitu znovu zajišťuje robustní naslouchač událostí. Blok je zobrazen níže na obrázku 15 Blok „Edit Location“.



Edit Location

Poslední kolonka akceptuje hodnoty "Anonymizováno" a "Neanonymizováno"

Číslo lokace

Získat

Název lokace

Komentář k lokaci

Anonymizace

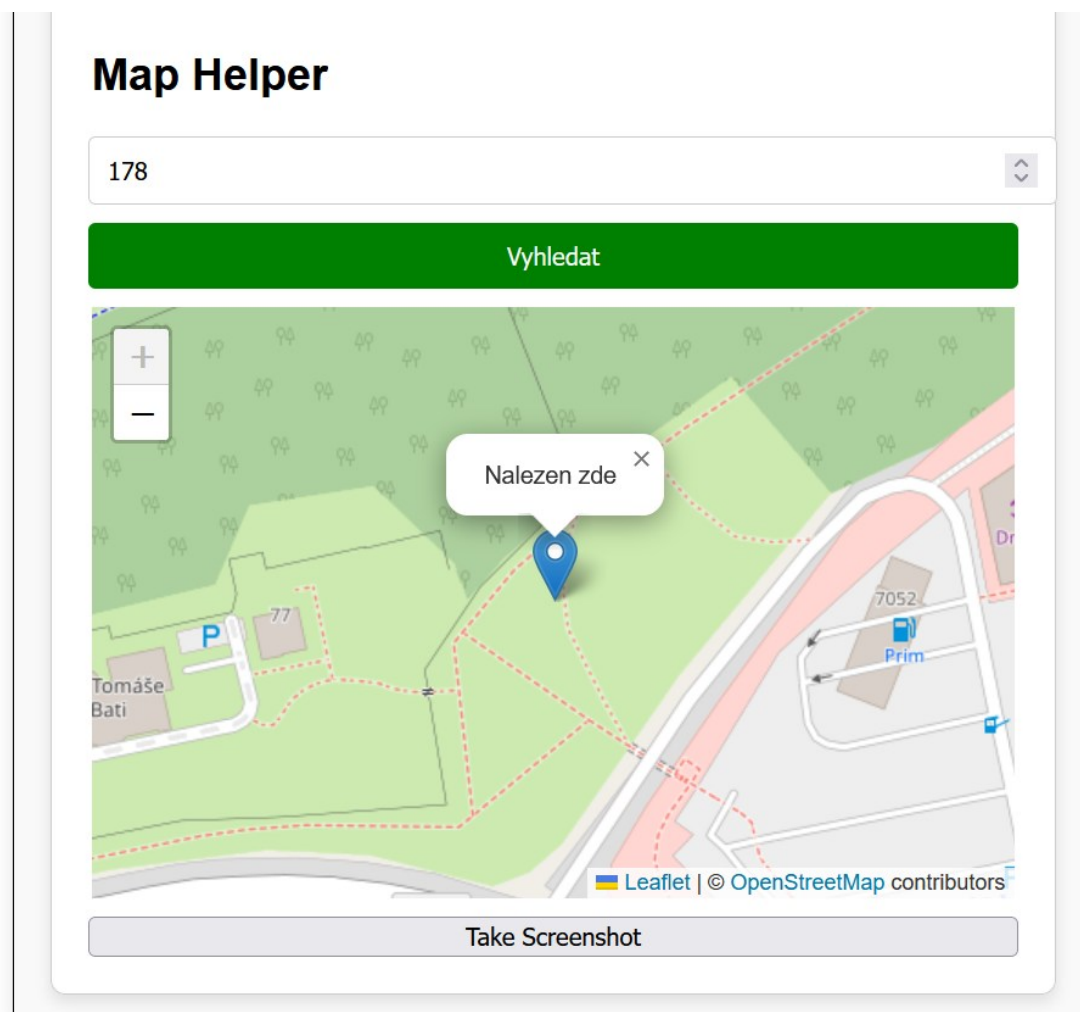
Změnit

Obrázek 15 Blok „Edit Location“

(zdroj: vlastní)

Předposlední blok prostředního sloupce má nadpis „Map Helper“ a uživatel do něj zadává identifikační číslo záznamu, pro který chce vyhledat a případně uložit mapu místa nálezu. Tato mapa je následně použita při generování PDF souboru s vybranými záznamy. V bloku se nacházejí dvě tlačítka oddělená prostorem pro zobrazení mapy. Nad horním tlačítkem je umístěno vstupní pole pro zadání identifikačního čísla záznamu, jehož místo nálezu bude po stisknutí tlačítka „Vyhledat“ zobrazeno. Spodní tlačítko „Take screenshot“ slouží k uložení obrázku s mapou ve formátu PNG do adresáře „mapy“. Každé tlačítko je obsluhováno samostatným naslouchačem událostí. Naslouchač pro tlačítko „Vyhledat“

spouští funkci `retrieveCoordinates`, která ze záznamu získaného asynchronním požadavkem na server extrahuje souřadnice a předává je funkci `initializeMap`. Ta podle těchto souřadnic vyhledá místo na stránkách OpenStreetMap a zobrazuje mapu s šipkou ukazující na přesné místo [19]. Blok „Map Helper“ je zobrazen na obrázku 16 Blok „Map Helper“.

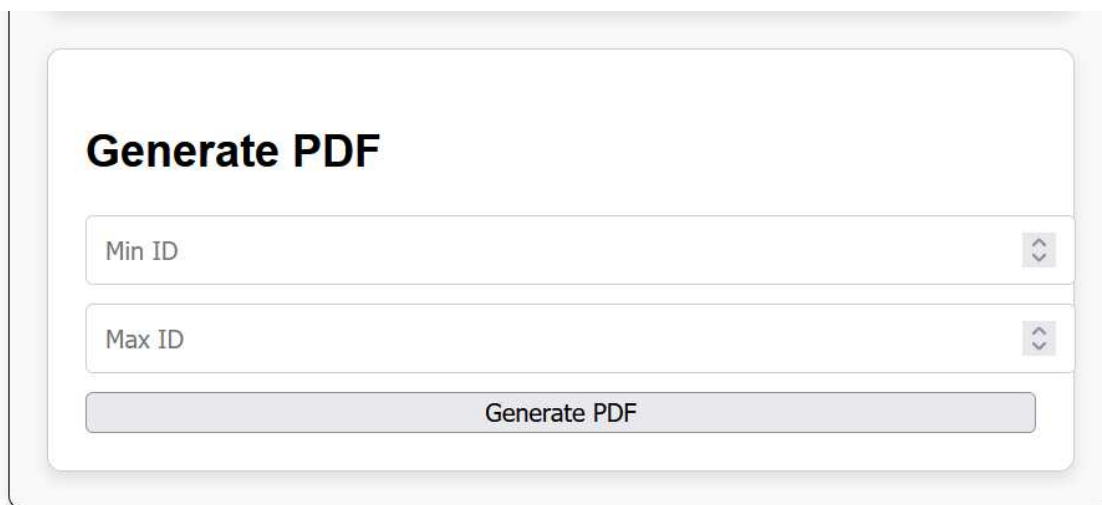


Obrázek 16 Blok „Map Helper“

(zdroj: vlastní)

Poslední blok prostředního sloupce slouží ke generování PDF souboru na základě specifikovaného rozsahu identifikačních čísel záznamů. Uživatel do vstupních polí zadá identifikační čísla (ID) dvou záznamů, která určují spodní a vrchní hranici seznamu, dle nějž bude PDF souboru vygenerován. Po stisknutí tlačítka „Generate PDF“ je spuštěn nasloucháč událostí, který pomocí asynchronního HTTP požadavku typu POST odesílá na server JSON soubor se těmito hodnotami. Server požadavek zpracuje a zajistí vytvoření požadovaného dokumentu. Grafickou podobu bloku „Generate PDF“ ukazuje obrázek 17 Blok „Generate

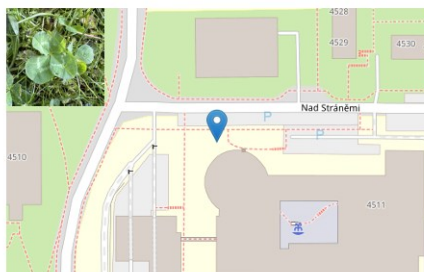
PDF“, zatímco podobu výsledného PDF dokumentu s testovacími daty nalezneme na obrázku 18 Příklad vygenerovaného PDF dokumentu.



Obrázek 17 Blok „Generate PDF“

(zdroj: vlastní)

525.



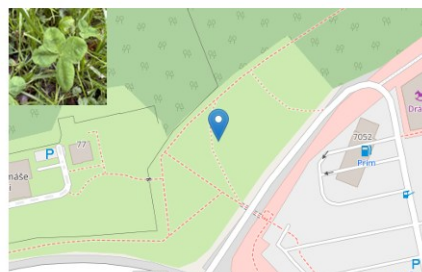
neděle 22. května 2023 v 07:56:46

49° 13' 51.4" S

17° 39' 24.8" V

Lokace - 5. lokace

526.



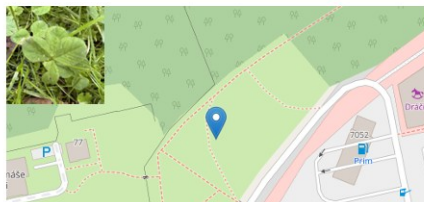
neděle 22. května 2023 v 10:38:41

49° 13' 46.1" S

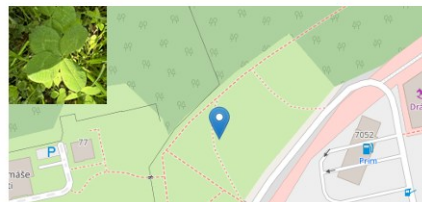
17° 39' 33.4" V

Lokace - Lokace 6.

527.



528.



Obrázek 18 Příklad vygenerovaného PDF dokumentu

(zdroj: vlastní)

Soubor „upload.html“, který je k dispozici jako příloha P XII: Soubor „upload.html“, obsahuje celkem 629 řádků.

7 NASAZENÍ APLIKACE

Aplikace byla nasazena na virtuální server, jehož pronájem byl zajištěn majitelem aplikace, doktorem Petrem Žáčkem, prostřednictvím společnosti SecurityNet.cz s.r.o. na stránkách hukot.net. Pro počáteční nasazení byl zakoupen virtuální server s 20 GB SSD úložištěm, 2 GB RAM a jedním procesorovým jádrem. Jako operační systém byla zvolena otevřená linuxová distribuce Ubuntu.

Server byl nakonfigurován vzdáleným přístupem z osobního počítače pomocí protokolu SSH. Nejprve byl na server nainstalován potřebný software, zejména Node.js, PostgreSQL, Python a ImageMagick. Dále byl nainstalován PM2, což je software s otevřeným zdrojovým kódem pro správu procesů Node.js, který zajišťuje nepřetržitý běh aplikace.

Zdrojový kód aplikace byl stažen z platformy GitHub, kde byl během vývoje zálohován. Poté byly pomocí příkazu `npm install` nainstalovány všechny závislosti uvedené v souboru „package.json“. Následně byly nainstalovány také knihovny jazyka Python, které aplikace využívá v podpůrném skriptu. Za pomoci souboru „schema.sql“, jehož obsah je podrobně popsán v kapitole 6.1 Schéma databáze, byla vytvořena PostgreSQL databáze. Konfigurační soubor „.env“ byl na serveru vytvořen ručně, neboť obsahuje citlivé údaje.

Po úspěšném spuštění aplikace byl nainstalován a nakonfigurován webový server s reverzní proxy Nginx, který má otevřený zdrojový kód. Bezpečnost serveru byla zajištěna nejen SSL (Secure Sockets Layer) certifikátem vystaveným pomocí nástroje Certbot, ale také nastavbovými bezpečnostními HTTP hlavičkami. Nástroj Security Headers společnosti Snyk ohodnotil zabezpečení stránek známkou A [20] a služba SSL Server Test společnosti Qualys dokonce ohodnotila bezpečnost aplikace známkou A+ [21].

Server byl nakonec propojen se zakoupenou doménou ctyrlstkoteka.cz. Funkčnost aplikace byla ověřena v produkčním prostředí a poté předána majiteli stránek.

ZÁVĚR

Tato bakalářská práce představila webovou aplikaci Čtyřlístkotéka, vytvořenou za účelem evidence, správy a veřejné prezentace sbírky čtyřlístků doktora Petra Žáčka. Aplikace je dostupná na adrese www.ctyrlistkoteca.cz.

Hlavní důraz byl kladen na popis zdrojového kódu aplikace, většinu práce proto tvoří praktická část. Teoretická část je zaměřena na akademicko-teoretickou reflexi klíčových technologií použitých při vývoji aplikace. Konkrétně jsou v samostatných kapitolách teoretické části stručně popsány principy architektury mikroslužeb, programovací jazyk JavaScript, v němž je napsána většina kódu aplikace, programovací jazyk Python, v němž je napsán jeden ze souborů zdrojového kódu, databázové systémy, systém řízení báze dat PostgreSQL, který aplikace používá, jazyk SQL a základy bezpečnosti webových aplikací.

V úvodní kapitole praktické části byly nejprve stručně představeny jednotlivé soubory aplikace, které byly následně podrobně popsány v samostatných podkapitolách. Popsána byla struktura databáze a jako další jednotlivé JavaScriptové soubory. Soubor „server.js“ představuje vstupní bod aplikace, zatímco soubor „uploadData.js“ zajišťuje nahrávání fotografií, extrakci dat, jejich formátování a ukládání do databáze. Soubor „convert.js“ slouží ke konverzi fotografií z formátu HEIC do formátu JPEG a soubor „queries.js“ obsahuje funkce, které pomocí SQL dotazů zjišťují statistické údaje o aktuálním stavu databáze aplikace. Dále byl popsán jediný soubor napsaný v jazyce Python „pdfGenerator.py“, který umožňuje generování PDF souboru s vybranými údaji z databáze.

V poslední podkapitole první kapitoly praktické části byly v samostatných oddílech popsány soubory napsané v jazyce HTML, které tvoří uživatelské rozhraní aplikace. Veřejnou část webu vytváří soubory „main.html“, který tvoří úvodní domovskou stránku aplikace, „records.html“, který vytváří stránku Záznamy, jež slouží k prezentaci jednotlivých čtyřlístků, „list.html“, jenž generuje stránku Seznam záznamů s několikastránkovou tabulkou obsahující všechny doposud nalezené čtyřlístky, a „statistics.html“, který prezentuje statistické údaje o sbírce na stránce Statistiky. Neveřejnou část webu tvoří soubory „login.html“, který slouží jako přihlašovací stránka do administrátorského rozhraní, jež je vytvořeno na základě souboru „upload.html“ a které umožňuje samotnou správu aplikace.

Druhá a zároveň poslední kapitola praktické části se stručně věnovala nasazení aplikace na internet.

Aplikace jistě není dokonalá a nabízí prostor nejen pro zdokonalování stávajících funkcionalit, ale i pro přidání funkcionalit nových. Konkrétně by dalším krokem například mohlo být zajištění responzivního designu, aby byla aplikace optimalizována i pro mobilní zařízení. Jedná se nicméně o plně funkční a zabezpečenou webovou aplikaci, která zcela naplňuje svůj účel a snad i zcela vyhovuje požadavkům jejího majitele, doktora Petra Žáčka.

SEZNAM POUŽITÉ LITERATURY

- [1] NORTHWOOD, Chris. *The Full Stack Developer: Your Essential Guide to the Everyday Skills of a Modern Full Stack Web Developer*. 1. vydání. Berkeley: Apress, 2018. ISBN 978-1-4842-4152-3.
- [2] HUNTER, Thomas II. *Advanced Microservices: A Hands-on Approach to Microservice Infrastructure and Tooling*. 1. vydání. Berkeley: Apress, 2017. ISBN 978-1-4842-2887-6.
- [3] MENDEZ, Michael. *The Missing Link: An Introduction to Web Development and Programming*. Geneseo: Open SUNY Textbooks, 2014. ISBN 978-0-9897226-5-0.
- [4] MARDAN, Azat. *Full Stack JavaScript: Learn Backbone.js, Node.js and MongoDB*. 1. vydání. Berkeley: Apress, 2015. ISBN 978-1-4842-1751-1.
- [5] FLANAGAN, David. *JavaScript: The Definitive Guide*. 6. vydání. Sebastopol: O'Reilley Media, Inc., 2011. ISBN 978-0-596-80552-4.
- [6] BROWN, Ethan. *Web Development with Node and Express*. 1. vydání. Sebastopol: O'Reilley Media, Inc., 2014. ISBN 978-1-491-94930-6.
- [7] ELLIOTT, Eric. *Programming JavaScript Applications*. 1. vydání. Sebastopol: O'Reilley Media, Inc., 2014. ISBN 978-1-491-95029-6.
- [8] LUTZ, Mark. *Learnig Python*. 4. vydání. Sebastopol: O'Reilley Media, Inc., 2009. ISBN 978-0-596-15806-4.
- [9] CORONEL, Carlos; MORRIS, Steven; ROB Peter. *Database Systems: Design, Implementation, and Management*. 9. vydání. Boston: Cengage Learning, 2011. ISBN 978-0-538-74884-1.
- [10] KROENKE, David M.; AUER, David J.; VANDENBERG, Scott L.; YODER, Robert C. *Database Processing: Fundamentals, Design, and Implementation*. 15. vydání. New Jersey: Pearson Education Inc., 2018. ISBN 978-0-13-480274-9.
- [11] JUBA, Salahaldin.; VANNAHME, Achim.; VOLKOV, Andrey. *Learning PostgreSQL*. 1. vydání. Birmingham: Packt Publishing Ltd., 2015. ISBN 978-1-78398-918-8.
- [12] MATTHEW, Neil a STONES, Richard. *Beginning Databases with PostgreSQL: From Novice to Professional*. 2. vydání. Berkeley: Apress, 2005. ISBN 978-1590594780.

- [13] BEAULIEU, Alan. *Learnig SQL*. 2. vydání. Sebastopol: O'Reilley Media, Inc., 2009. ISBN 978-0-596-52083-0.
- [14] MUELLER, John Paul. *Security for Web Developers*. 1. vydání. Sebastopol: O'Reilley Media, Inc., 2016. ISBN 978-1-491-92864-6.
- [15] HOFFMAN, Andrew. *Web Application Security*. 3. vydání. Sebastopol: O'Reilley Media, Inc., 2020. ISBN 978-1-492-08796-0.
- [16] OPENSTREETMAP CONTRIBUTORS. *Nominatim*. Online. Dostupné z: <https://nominatim.openstreetmap.org/>. [cit. 2025-05-26].
- [17] SVG REPO LLC. *SVG Repo: Free SVG Vectors and Icons*. Online. 2025. Dostupné z: <https://www.svgrepo.com/>. [cit. 2025-05-26].
- [18] AGAFONKIN, Volodymyr. *Leaflet: a JavaScript library for interactive maps*. Online. 2010, 2025. Dostupné z: <https://leafletjs.com/>. [cit. 2025-05-26].
- [19] OPENSTREETMAP CONTRIBUTORS. *OpenStreetMap*. Online. Dostupné z: <https://www.openstreetmap.org>. [cit. 2025-05-26].
- [20] SNYK. *Security Headers*. Online. Dostupné z: <https://securityheaders.com/>. [cit. 2025-05-27].
- [21] QUALYS, INC. *SSL Server Test: (Powered by Qualys SSL Labs)*. Online. 2009, 2025. Dostupné z: <https://www.ssllabs.com/ssltest/>. [cit. 2025-05-27].

SEZNAM OBRÁZKŮ

Obrázek 1 Model databáze	27
Obrázek 2 Domovská stránka	38
Obrázek 3 Výchozí podoba stránky „Záznamy“	41
Obrázek 4 Stránka „Záznamy“ s načteným záznamem	41
Obrázek 5 Stránka „Záznamy“ s anonymizovaným záznamem	42
Obrázek 6 Stránka „Seznam záznamů“	44
Obrázek 7 Blok „Pět nejčastějších lokací“	45
Obrázek 8 Blok s počtem nálezů podle let a s prvním a posledním nálezem v každém roce	45
Obrázek 9 Blok „Nálezy dle měsíců“	46
Obrázek 10 Blok s počtem nálezů dle dnů v týdnu a se dnem s nejvíce nálezy	47
Obrázek 11 Blok s počty nálezů dle hodin, se dnem s nejvíce nálezy za hodinu a s nejdřívějším a nejpozdějším nálezem	48
Obrázek 12 Přihlašovací stránka	49
Obrázek 13 Vrchní část administrátorské stránky	51
Obrázek 14 Blok „Edit Record“	52
Obrázek 15 Blok „Edit Location“	53
Obrázek 16 Blok „Map Helper“	54
Obrázek 17 Blok „Generate PDF“	55
Obrázek 18 Příklad vygenerovaného PDF dokumentu	55

SEZNAM PŘÍLOH

Příloha P I: Soubor „server.js“

Příloha P II: Soubor „uploadData.js“

Příloha P III: Soubor „convert.js“

Příloha P IV: Soubor „queries.js“

Příloha P V: Soubor „pdfGenerator.py“

Příloha P VI: Soubor „schema.sql“

Příloha P VII: Soubor „main.html“

Příloha P VIII: Soubor „records.html“

Příloha P IX: Soubor „list.html“

Příloha P X: Soubor „statistics.html“

Příloha P XI: Soubor „login.html“

Příloha P XII: Soubor „upload.html“