VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

PODPORA VYHLEDÁVÁNÍ MATRIČNÍCH UDÁLOSTÍ

SUPPORT FOR SEARCHING IN PARISH BOOKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **DOMINIK POP**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

146877

Ústav:	Ústav inteligentních systémů (UITS)
Student:	**Pop Dominik**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Podpora vyhledávání matričních událostí**
Kategorie:	Webové aplikace
Akademický rok:	2022/23

Zadání:

1. Nastudujte problematiku genealogie a proces tvorby rodokmenů z matričních záznamů.
2. Na základě informací o tom, jak si genealogové vedou záznamy o hledání, navrhněte webovou aplikaci, která přihlášeným uživatelům bude automaticky z jejich GEDCOM souboru vytvářet záznamy k jednotlivým chybějícím událostem, tyto události přiřazovat do jednotlivých matrik podle obcí a řadit podle data. Tyto záznamy bude možné ručně editovat a přidávat k nim další údaje jako např. kandidátní záznamy o dalších osobách. Případně umožněte také zobrazení hranic mezi jednotlivými farnostmi, případně obcemi.
3. Webovou aplikaci implementujte a naplňte daty podle pokynů vedoucího.
4. Aplikaci otestujte a navrhněte případná vylepšení.

Literatura:

- Prostredníková Hana: Pokročilé zobrazování genealogických dat, bakalářská práce, Brno, FIT VUT v Brně, 2016.
- Valecký Dušan: Zobrazování genealogických dat, bakalářská práce, Brno, FIT VUT v Brně, 2017.
- Konetzný, Jakub. Podpora vyhledávání matričních událostí. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu:	Hanáček Petr, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	3.11.2022

Abstrakt

Cílem této práce je návrh a vytvoření webové aplikace, jež je schopná navrhnout uživateli, genealogovi, vhodné matriky pro dohledání informací k chybějícím událostem konkrétní osoby. Podstatou aplikace je tedy zpracování vstupního GEDCOM souboru, zvolení vhodných matrik pro konkrétní události a nabídnutí těchto matrik uživateli, přes grafické uživatelské prostředí.

Abstract

The goal of this thesis is to design and implement application which is capable of advising user, genealogist, suitable parish books for searching information about missing events of practicular person. The base of this application is processing input GEDCOOM file, choosing of suitable parish books and offering of these books to user through a graphical user interface.

Klíčová slova

genealogie, genealog, matrika, webová aplikace, GEDCOM, PHP, Python, MySQL

Keywords

genealogy, genealogist, parish book, web application, GEDCOM, PHP, Python, MySQL

Citace

POP, Dominik. *Podpora vyhledávání matričních událostí*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Podpora vyhledávání matričních událostí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Dominik Pop
8. května 2023

Poděkování

Rád bych tímto poděkoval Ing. Jaroslavu Rozmanovi, Ph.D. za ochotu, konzultace a nápomocné připomínky týkající se vypracování práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Za posledních několik let roste popularita genealogie směrem nahoru. Lidé chtějí znát kam až vedou kořeny jejich rodokmenu, zjistit různé informace o svých předcích a nebo genealogii považují za svůj koníček. Ovšem postup vyhledávání takových informací není vůbec jednoduchou záležitostí. Aby badatel dohledal všechny potřebné informace musí si projít dlouhým procesem vybírání vhodných matrik, ve kterých by dané informace získal. Musí určit všechny oblasti, ve kterých se osoba vyskytovala, časové rozmezí a nahlížet při tom i na jeho příbuzné.

Tato práce se snaží zjednodušit tento zdlouhavý postup. Jejím cílem je navrhnout a zrealizovat webovou aplikaci, která by uživatelům zjednodušila proces vyhledávání chybějících informací o konkrétních osobách. Aplikace bude jako vstup zpracovávat soubor o formátu GEDCOM, vytvoří záznamy o chybějících informacích k daným osobám a k těmto záznamům navrhne vhodné matriky na základě oblastí a časových rozmezí zjištěných ze známých informací dané osoby, či jejich příbuzných. Zvolené matriky pak budou uživateli nabídnuty prostřednictvím uživatelského prostředí a bude možné se přes ně navigovat na samotnou digitalizovanou podobu matriční knihy. Uživatel bude mít dále možnost zapisovat si k záznamům poznámky, nebo záznamy ignorovat.

Práce je členěna do 4 částí. První část dokumentu se zaobírá teoretickými znalostmi, potřebnými ke správnému návrhu a implementaci aplikace. Je zde vysvětlen pojem genealogie a s ním pojmy svázané. Jsou zde popsány matriční knihy a jejich digitální vyobrazení. A jako poslední je zde stručný popis GEDCOM formátu, jenž působí jako vstup pro naši aplikaci.

V druhé části dokumentu jsou popsány technologie použité při implementaci aplikace, včetně důležitých knihoven, modulů, či frameworků.

Třetí část se věnuje analýze a návrhu aplikace. Tato část opět apeluje na cíle práce. Dále provádí analýzu uživatelů, pro vyobrazení funkčností, kterých by měla aplikace nabývat. Důležitý je také popis návrhu databázové struktury vhodné pro ukládání genealogických dat, rovněž obsažený v této části. Závěr kapitoly je věnován jednoduchému návrhu samotného grafického uživatelského rozhraní.

Čtvrtá část obsahuje popis jednotlivých fází implementace aplikace. Jsou zde popsané fáze od přípravy potřebných dat, zpracování jednotlivých skriptů až po implementaci samotné aplikace.

Kapitola 2

Studium problematiky

Tato kapitola se zabývá stručným popisem problematiky. Obsahuje informace potřebné k pochopení tématu, správnému návrhu a implementaci aplikace. Je zde vysvětlen pojem genealogie, způsob jakým jsou ukládány informace v matričních knihách a popsán formát vstupních souborů aplikace.

2.1 Genealogie

Genealogie[1] je odvětví zabývající se rodovými vztahy mezi jedinci určitého druhu. Zkoumá tedy rodové souvislosti. Zabývá se buď studiem jednotlivých osobností, nebo sledováním proměn jednotlivých druhů vztahů.

Genealogii považujeme za jednu z metod genetického výzkumu. Základem genealogické metody je grafické zaznamenání rodových vztahů mezi jedinci v jednotlivých generacích. Tuto grafickou podobu nazýváme rodokmen [20].

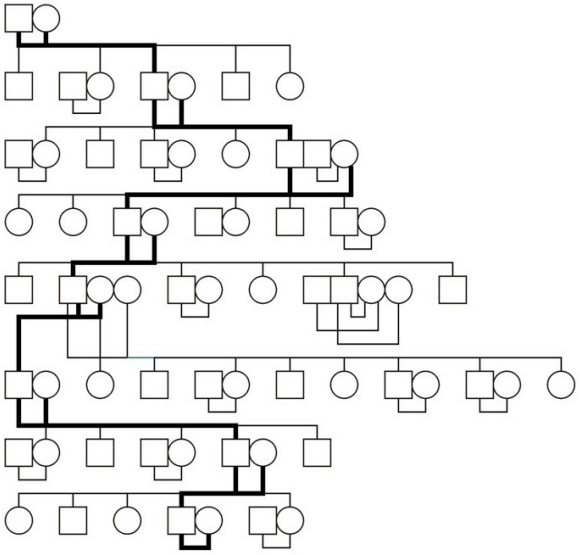
2.1.1 Genealogické diagramy

Jedná se o schéma reprezentující vztahy v rámci rodiny pomocí stromové struktury. V následujících podkapitolách budou popsány nejznámější schémata. K popisu diagramů bylo v této podkapitole čerpáno z webové stránky Václava Příborského [17].

Rodokmen

Jedná se o základní a nejjednodušší schéma. Začíná u zakladatele rodu, tedy nejstaršího známého předka. Zahrnuje jeho děti a potomky všech jejich synů, kteří si založili vlastní rodinu. Zkoumáme pouze potomky synů, jelikož syn je obvykle nositelem příjmení, kdežto dcera nikoliv. Příjmení je totiž typickým znakem rodokmenu. Pomocí rodokmenu lze zkoumat mohutnost celého rodu, počty manželství v jednotlivých generacích a mnoho dalších zajímavých informací. Schéma rodokmenu se nachází na obrázku 2.1.

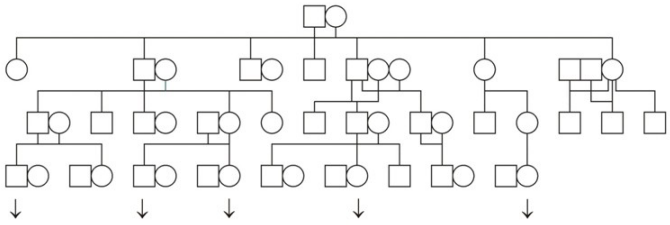
[1] Genealogie - pochází z řeckých slov genea = původ a logos = znalost



Obrázek 2.1: Schéma rodokmenu [převzato ze stránky [14]]

Rozrod

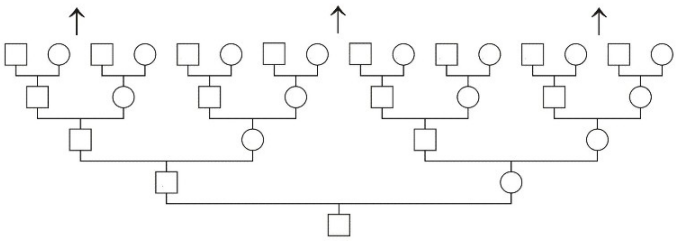
Rozrod oproti rodokmenu zahrnuje všechny potomky zakladatele rodu. Je tedy rozšířen i o potomky dcer. Dají se z něj vysledovat osudy celé široké rodiny. V genealogii se jedná o nejrozsáhlejší a nejsložitější schéma. Obsahuje větší počet osob než rodokmen, je tak i náročnější na vytvoření, ale jsme z něj schopni získat více informací a sledovat veškeré potomstvo. Schéma rozrodu se nachází na obrázku 2.2.



Obrázek 2.2: Schéma rozrodu [převzato ze stránky [14]]

Vývod

Vývod vychází od zkoumané osoby směrem nahoru. Zahrnuje tedy mužské i ženské předky dané osoby směrem do minulosti. Vypovídá o generacích rodičů a prarodičů výchozí osoby. Lze z něj získat větší znalost o rodinných kořenech osoby a zkoumat předchozí generace. Vývod můžeme rozdělit na agnátní a kognátní. Agnátní sleduje pouze mužskou linii, kognátní ženskou. Schéma vývodu se nachází na obrázku 2.3.



Obrázek 2.3: Schéma vývodu [převzato ze stránky [14]]

2.2 Matriky

Matriky jsou základním pramenem dat při provádění genealogického výzkumu. Jedná se o knihy vedené ve stylu úředního seznamu jmen. Obsahují různé informace související s danou událostí a osobou. Matriky jsou tedy seznamem uchovávajícím záznamy o narození, sňatcích a úmrtí dané osoby. Tyto seznamy jsou vedeny matrikáři na patřičných matričních úřadech. Takovým úřadem může být například obecní úřad, úřad městské části nebo městského obvodu.

Matriky rozdělujeme na dva druhy podle jejich dostupnosti, na živé a mrtvé. Živé matriky jsou takové, které v sobě obsahují záznamy u nichž neuplynulo od posledního narození 100 let, a nebo 75 let od posledního sňatku či úmrtí. Takové matriky nejsou veřejně přístupné a smí se do nich nahlížet pouze za přítomnosti matrikáře a při splnění alespoň jedné z určitých podmínek, které jsou popsány na webu ministerstva vnitra České republiky[2]. Ostatní matriky prohlašujeme za mrtvé a jsou veřejně dostupné na úřadech či v digitální podobě.

Z počátku se všechny druhy záznamů zapisovaly do jedné knihy společně. Později, zhruba od druhé poloviny 18. století, se začaly záznamy pro jednotlivé události zapisovat do pro ně konkrétních matrik. Typy těchto matrik jsou popsány v následujících podkapitolách [15].

2.2.1 Matriky narození

V této knize nalezneme údaje týkající se narození dané osoby. Můžeme zde nalézt jméno i příjmení, datum narození ale i křtu, pohlaví nebo třeba náboženství. V těchto knihách ovšem často nalezneme také údaje nejen o této osobě, ale také jméno, stav a rod jeho matky, otce, křtitele nebo i kmotra. Jak vypadá záznam v takové matrice je ukázáno na obrázku 2.4.

2.2.2 Matriky úmrtí

Úmrtní matriky obsahují kromě základních informací jako je jméno, datum a číslo domu, také informace specifické k této události. Tedy stáří zemřelého a příčinu úmrtí.

2.2.3 Matriky oddání

Tyto matriky v sobě uchovávají záznamy o oddání dvou osob. Obsahují základní informace o události, tedy datum sňatku, číslo popisné domu, a dále pak informace o osobách účast-

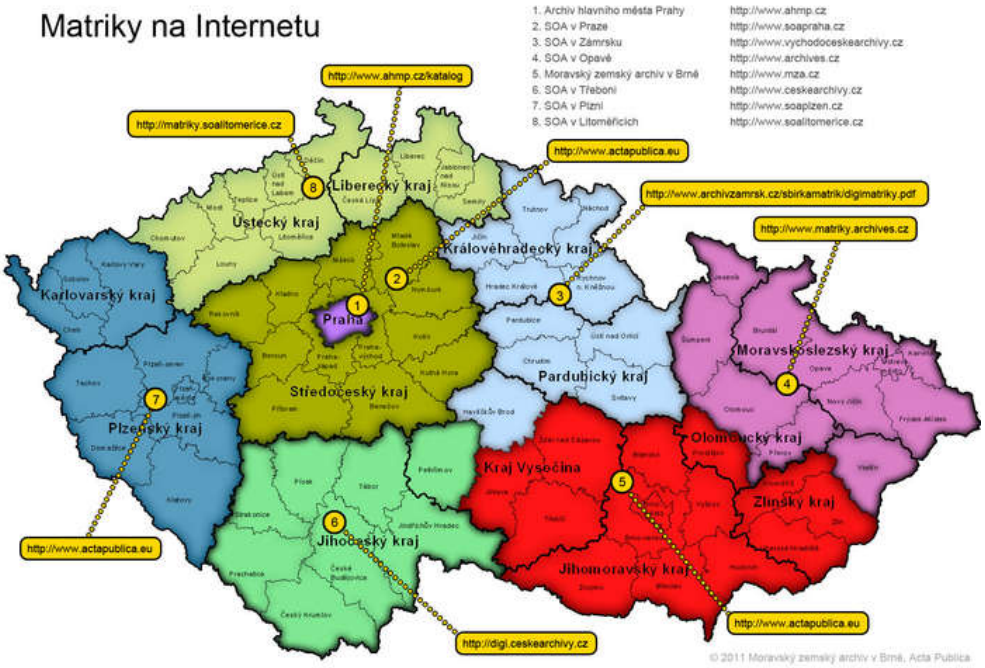
[2] `https://www.mvcr.cz/migrace/docDetail.aspx?docid=21814471`

nících se sňatku. Zde se opět řadí základní informace jako jsou jména, náboženství, stáří, ale také třeba stav osoby (svobodný/vdovec). Můžeme zde nalézt i údaje o svědcích.

Obrázek 2.4: Část snímku č. 111 z matriky narození [převzato z archivu [12]]

2.2.4 Digitalizace

V roce 2007 započala digitalizace matrik v rámci České republiky. Byla iniciována americkou genealogickou společností Family Search[3]. Digitalizace matrik umožňuje komukoliv nahlédnout do knihy bez nutnosti cestovat do archivu, kde jsou matriky uloženy. Digitalizované knihy se nacházejí na celkově osmi online archivech[4], rozdělení je znázorněno na obrázku 2.5. Tyto weby umožňují vyhledávat knihy na základě parametrů. Nejedná se o přepis matrik do digitální podoby, ale o ofocení jednotlivých stránek matriční knihy a zveřejnění těchto fotek [2].



Obrázek 2.5: Rozdělení archivů ČR [převzato ze stránky [3]]

[3]Family Search - `https://www.familysearch.org/en/`
[4]Rozcestník - `https://digi.ceskearchivy.cz/Matriky-Matricni-rozcestnik-Ceske-republiky`

2.3 GEDCOM

GEDCOM, Genealogical Data Communication, je čistě textový souborový formát určený pro ukládání a distribuci genealogických dat. Jedná se o nejpoužívanější datový formát v oboru genealogie. Specifikace byla vytvořena Církví Ježíše Krista Svatých posledních dnů, jež jej zpravuje dodnes. Aktuálně používaný a nejrozšířenější standart je 5.5.1, ale posledním vydaným je 7.0, jenž se snaží o vylepšení. Implementovaná aplikace pracuje výhradně se standardem 5.5.1.

2.3.1 Popis formátu

Specifikace GEDCOM má velmi rozsáhlou dokumentaci [5]. V následující části z ní budou vybrány pravidla důležité pro tuto práci.

Soubory GEDCOM končí příponou .ged. Soubor se skládá z hlavičky, která popisuje obecné informace o souboru (verze, kódování, aj.), množiny záznamů a speciálního ukončovacího záznamu TRLR, jenž oznamuje konec souboru.

Záznamy jsou v souboru organizovány do hierarchické struktury po řádcích. Každý řádek záznamu je označen číselnou úrovní, indikující hloubku zanoření, kdy první řádek má hodnotu 0. Další řádky mají hodnotu vyšší a platí, že řádek s hodnotou n+1 spadá pod řádek s hodnotou n. Řádek s hodnotou 0 tedy znamená začátek nového záznamu. Za úrovní může následovat křížový odkaz, který jednoznačně identifikuje danou osobu/rodinu. Odkaz je následován značkou, neboli tagem. Tag oznamuje jaká informace bude následovat. Tagů je velké množství, význam jednotlivých tagů je vysvětlen v dokumentaci specifikace. V tagu může být také obsažen identifikátor rodiny, do které daná osoba patří.

Záznam

Jak už bylo v předchozím odstavci naznačeno, kromě speciálního záznamu TRLR existují dva typy záznamů.

Prvním je záznam o individuální osobě (INDI), jenž uchovává základní genealogické informace o dané osobě. Nás zajímá jméno osoby a události narození (BIRT) a úmrtí (DEAT). U každé takové události je uvedeno datum (DATE) a lokace (PLAC), kdy a kde událost proběhla. Dále tento záznam obsahuje odkazy na rodiny, kterých je osoba součástí.

Druhým typem je záznam o rodině (FAM). V tomto záznamu může být popsána událost oddání (MARR). Ta je opět popsána datem a lokací. Dále tento záznam obsahuje odkazy na členy rodiny. V rodině může mít osoba určitou roli, manžel (HUSB), manželka (WIFE), dítě (CHIL). Tento záznam může obsahovat také různé statistické hodnoty, jako například počet dětí.

Datum

Den, měsíc a rok mají přesně daný formát zápisu. Den a rok se zapisují pomocí číselné hodnoty ve tvaru DD pro den, a YYYY pro rok. Měsíc se skládá ze tří prvních písmen z jeho anglického jména ve tvaru MMM. Datum je pak možné zapsat jako celek, tedy DD MMM YYYY, nebo postupně odebírat prvky zleva až po rok (MMM YYYY, YYYY).

Datum může také obsahovat různé fráze, jako např.: BEF, AFT, FROM. Tyto fráze pak mohou specifikovat rozsah nebo období.

Lokace

Jediné pravidlo popisující lokaci je pravidlo hierarchie územních celků. To říká, že při zápisu lokace se mají řadit od nejmenšího po největší zleva doprava a mají být odděleny čárkami. Jinak je možné do lokace zapsat prakticky cokoliv, což všeobecně vytváří problémy při implementaci aplikací pracujících s GEDCOM formátem.

Kapitola 3

Analýza požadavků a návrh řešení

V této kapitole je popsána analýza uživatele a jeho potřeb. Na základě uživatelem potřebných funkčností je zde proveden návrh databázové struktury, která v sobě bude uchovávat informace potřebné pro vykonávání daných funkcí. Jako poslední obsahuje tato kapitola jednoduchý návrh grafického uživatelského prostředí na základě provedené analýzy uživatele.

3.1 Cíl práce

Cílem je dosáhnout funkční webové aplikace schopné pojmout jako vstup soubor formátu GEDCOM, tento soubor správně zpracovat, uložit potřebné informace a vytvořit záznamy o chybějících informacích jednotlivých událostí. Tyto vytvořené záznamy je pak potřeba uživateli graficky zobrazit a navrhnout mu matriky, ve kterých by se informace mohly vyskytovat.

Hlavní myšlenkou je vytvořit aplikaci, která pomůže svým uživatelům s dohledáním chybějících informací, tím že jim nabídne vhodné matriky, ve kterých by informace mohli najít. Tedy urychlit a usnadnit proces dohledávání informací.

3.2 Uživatel a jeho potřeby

Uživatelem aplikace nemusí být pouze genealog, může se jednat o jakoukoliv osobu provádějící genealogický výzkum. Aplikace je využívána k ulehčení určování vhodných matrik pro vyhledávání chybějících informací v již rozpracovaném GEDCOM souboru. Je tedy nutné aby uživatel byl schopen poskytnout tento soubor naplněný genealogickými daty. Aplikace nemá za úkol tyto soubory vytvářet, nýbrž pomáhat s nalezením chybějících informací a musí tedy pracovat s rozpracovanými soubory.

V následujících kapitolách jsou vysvětleny funkce, které bude aplikace poskytovat svým uživatelům. Tyto funkce jsou také vyobrazeny v diagramu případů užití (obrázek 3.1).

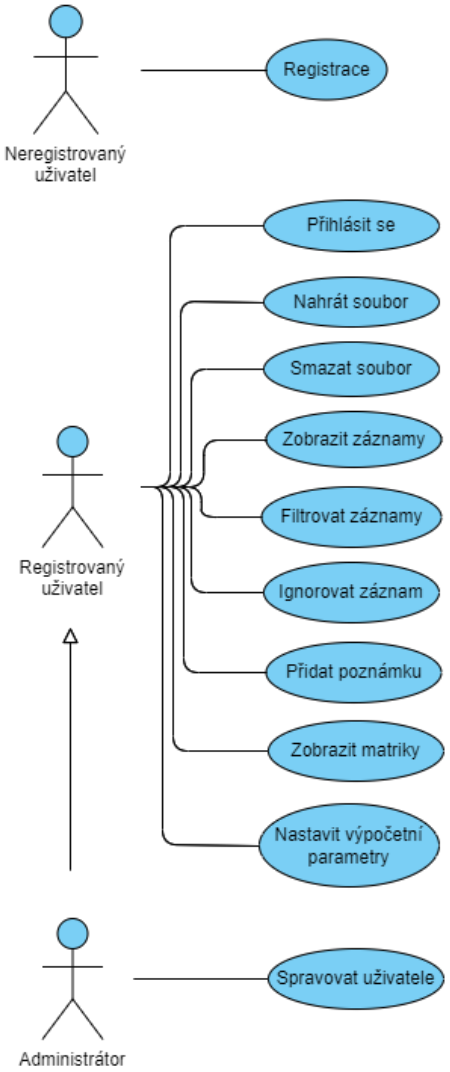
Práce se soubory

Uživatel musí být schopen nahrát vstupní GEDCOM soubor do aplikace. U nahrání souboru bude mít možnost jej pojmenovat a nastavit výpočetní parametry, které ovlivňují výpočty časových rozsahů. Aplikace mu umožní zobrazit si přehled nahraných souborů. U těchto souborů bude moci editovat jméno, psát poznámky nebo je ze systému mazat. Soubory bude také schopen filtrovat nebo vyhledávat podle jména. Zároveň mu tyto soubory proklikem

umožní dostat se na přehled záznamů vázaných k tomuto souboru. V rámci zpracování souboru bude uživatel povinen upřesnit území, které se v souboru nedají jasně identifikovat. Při identifikaci mu bude aplikace nabízet území z databáze.

Záznamy

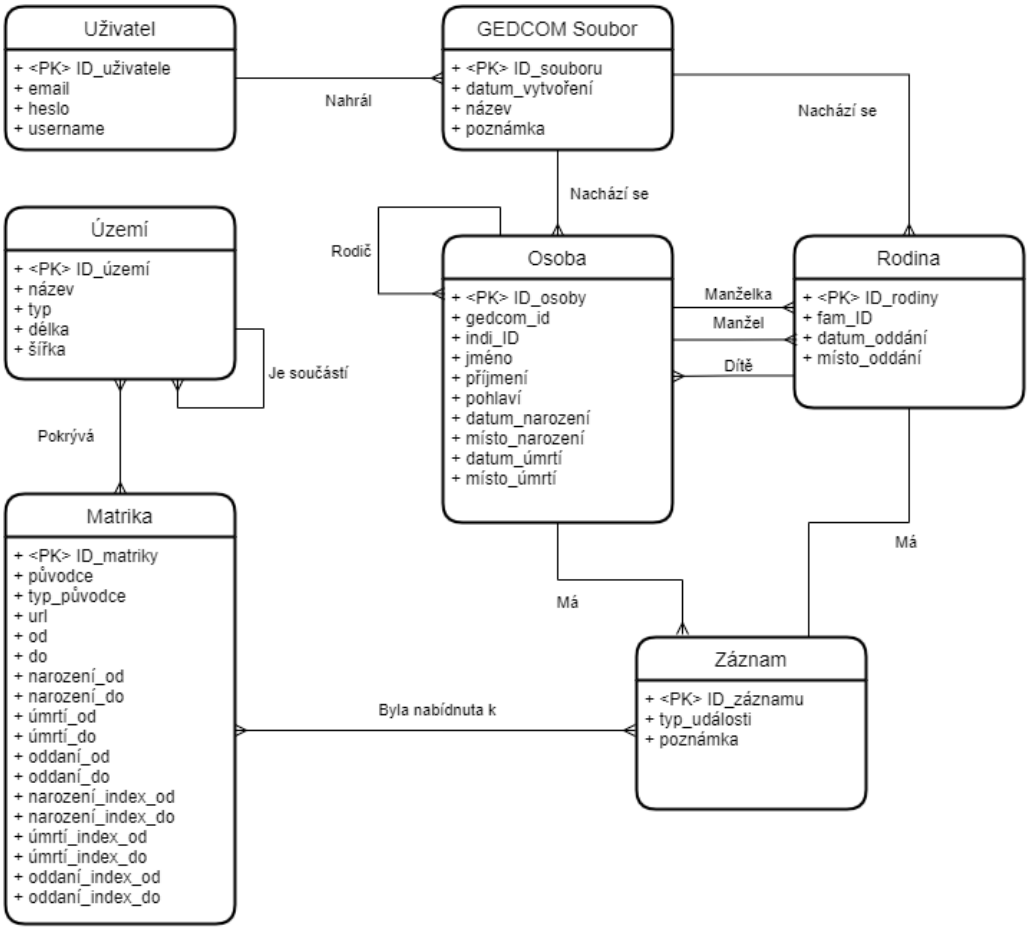
Na přehledu zobrazující záznamy pro daný GEDCOM soubor si bude uživatel moci prohlížet vytvořené záznamy. Záznamy si bude moci filtrovat podle typu události, ke které jsou vázané. Se záznamem bude uživateli umožněno provádět různé operace. Mezi ty spadá možnost zobrazit si nabídnuté matriky, přes které se bude schopen dostat přímo na jejich detail v digitálním archivu. Matriky bude také schopen odkliknout. Uživatel bude mít možnost záznam ignorovat nebo schválit. Schválení vybídne uživatele k doplnění chybějících informací, při ignoraci aplikace přestane se záznamem pracovat. K jednotlivým záznamům si bude uživatel také moci napsat poznámku.



Obrázek 3.1: Diagram případů užití

3.3 Návrh databáze

Návrh databáze je zde reprezentován pomocí diagramu vztahů mezi entitami[1] (ER[2]). V rámci databáze MySQL jsou poté tyto jednotlivé entity a vztahy převedeny na tabulky na základě pravidel převodu. Entity nám zde představují jednotlivé objekty, se kterými pracujeme v rámci aplikace. Jednotlivé entity a vztahy jsou pak popsány v podkapitolách nacházejících se pod ER diagramem (obrázek 3.2).



Obrázek 3.2: Diagram vztahů mezi entitami

Uživatel

Entita uživatel reprezentuje jednotlivé uživatele v rámci aplikace. Slouží pro uchování základních informací o uživateli jako jeho email, heslo a uživatelské jméno. Tato entita je důležitá, protože se k ní váže entita GEDCOM souboru, kdy tento soubor musí vždy spadat pod nějakého uživatele.

[1] entita - objekt reálného světa zachycený v datovém modelu
[2] ER - Entity relationship

GEDCOM soubor

Entita reprezentuje GEDCOM soubor v rámci systému. Jejími atributy jsou jméno souboru a datum vytvoření. Tato entita je ve vztahu N:1 s uživatelem, který ji vlastní. Dále je entita ve vztahu s entitami Osoba a Rodina. Tato vazba reprezentuje osoby/rodiny nacházející se v tomto souboru. Jsou ve vztahu 1:N.

Osoba

Jedná se o entitu, která obsahuje záznamy o konkrétních osobách nacházejících se v nějakém GEDCOM souboru. Její atributy obsahují základní informace vytažené ze souboru, jako celé jméno, ID v rámci souboru, pohlaví, ale hlavně informace týkající se sledovaných událostí. Tedy datum a místo narození/úmrtí. Osoba má vazbu vůči sama sobě, která určuje otce/matku dané osoby. Dále je osoba ve vztahu 1:N s rodinou. Tato vazba znamená, že osoba v rodině působí jako manžel/manželka, tedy i jako rodič pro ostatní členy rodiny. Osoba může být s rodinou také ve vztahu N:1, kdy tato vazba znamená, že osoba působí v rodině jako dítě.

Rodina

Rodina je v aplikaci reprezentována pomocí stejnojmenné entity. Uchovává v sobě informace o události oddání. Je propojena několika vazbami s entitou Osoba, které jsou popsány výše. Umožňuje reprezentovat vztahy mezi osobami v rámci rodiny.

Záznam

V případě, že u události chybí nějaká informace je o tom vytvořen záznam. Ten je v aplikaci reprezentován entitou Záznam. Jeho atributem je typ popisující o jakou událost se jedná. Dále pak umožňuje uživateli zapisovat k sobě poznámky uložené do stejnojmenného atributu. Je ve vztahu s entitami Osoba a Rodina, kdy pro osobu mohou existovat maximálně 2 záznamy a pro rodinu jeden. Dále je ve vztahu M:N s entitou Matrika, kdy tento vztah vyjadřuje matriky nabídnuté k danému záznamu.

Matrika

Tato entita slouží k popsání jednotlivých matrik používaných v aplikaci. Obsahuje mnoho atributů, hlavními jsou rozsahy od/do pro jednotlivé události, které nám říkají jaké události v sobě matrika uchovává a mezi jakými roky. Dále pak obsahuje atributy určené k odlišení matrik při zobrazení v grafickém prostředí, jako původce a jeho typ. Atribut url pak umožňuje aplikaci přesměrovat uživatele na detail konkrétní matriky. Entita se nachází ve vztahu s entitou Území, kdy tento vztah popisuje jaké území daná matrika pokrývá. Dále je pak ve vztahu se záznamem, jenž je popsán v předchozí podkapitole.

Území

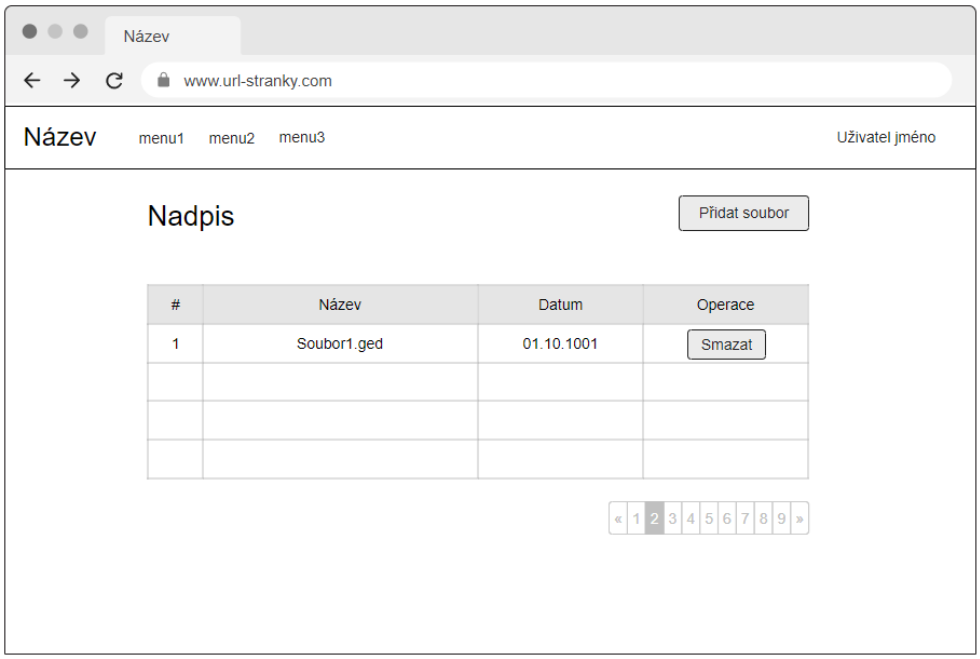
Jedná se o entitu, jež představuje jednotlivé územní celky v rámci České republiky. Obsahuje název celku, typ, o který se jedná, a dále pak zeměpisnou šířku a délku popsanou pomocí souřadnic. Území je v unárním vztahu samo se sebou, který říká, že nějaké území může být součástí jiného, většího území. Dále je pak ve vztahu s matrikou, pod kterou spadá.

3.4 Návrh uživatelského prostředí

Jak bylo zmíněno v podkapitole 3.2 aplikace musí poskytovat uživateli nějaké funkcionality. Ty poskytuje prostřednictvím grafického rozhraní. V této podkapitole budou zobrazeny jednoduché návrhy takového rozhraní pro jednotlivé funkcionality. V rámci návrhu chceme zachovat minimalistický a jednoduchý vzhled, který uživatele dovede přímo ke chtěným operacím za použití co nejméně přechodů mezi stránkami.

Práce se soubory

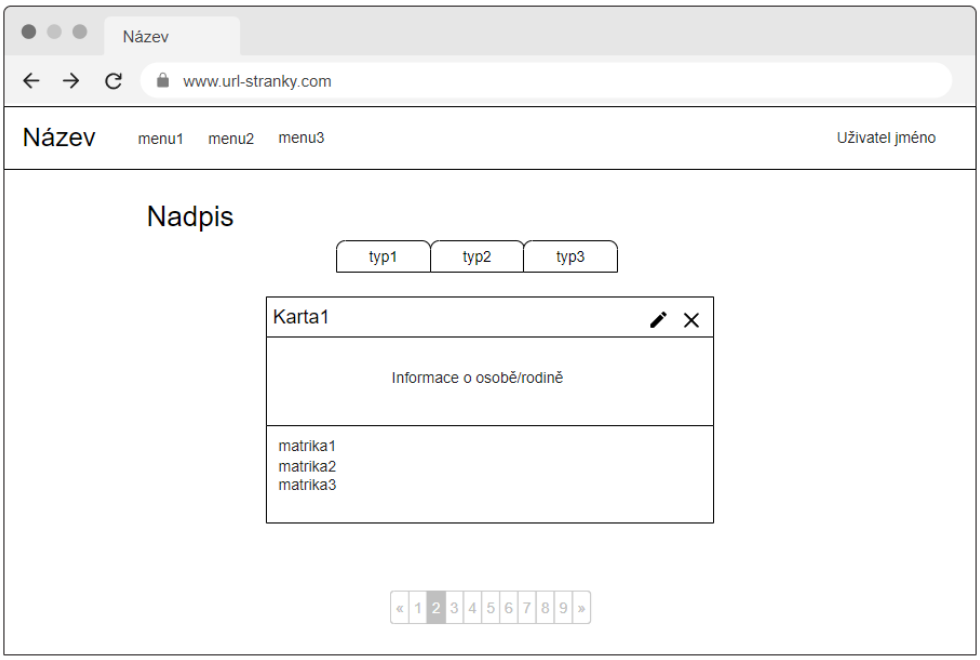
Aplikace by měla obsahovat nějakou stránku s přehledem souborů. V naší aplikaci bude touto stránkou naše hlavní strana. Zde se bude nacházet jednoduchá tabulka obsahující nahrané soubory, které budou řazeny podle data nahrání. V rámci této tabulky bude uživatel schopen provádět dříve zmíněné operace. Zároveň bude tuto tabulku používat k přístupům na stránku, která bude zobrazovat jednotlivé záznamy vytvořené pro daný soubor. Na této stránce bude uživatel schopen také nahrát svůj soubor za pomocí tlačítka a vyskakovacího okna.



Obrázek 3.3: Stránka souborů - tabulka souborů

Záznamy

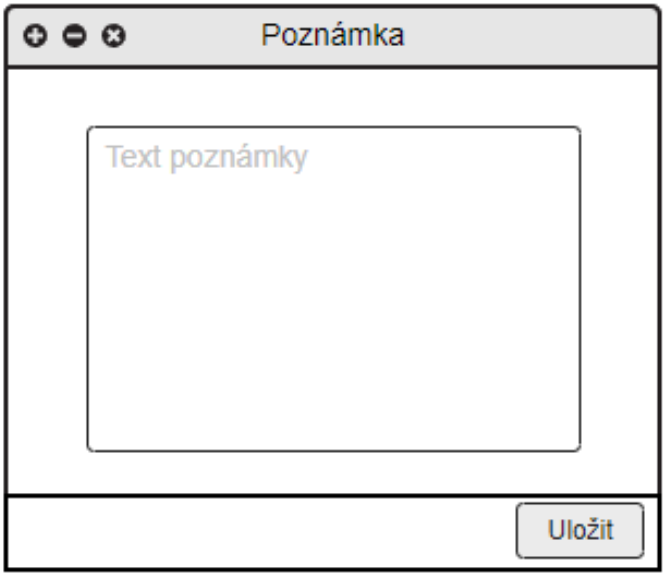
Nejdůležitější stránkou aplikace bude ta, která zobrazí jednotlivé vytvořené záznamy souborů. Stránka by měla indikovat, který soubor právě zkoumáme nějakým nadpisem. Záznamy zde budou rozděleny podle jejich typu a možnost přecházet mezi typy bude uživateli dostupná bez nutnosti prokliku. Záznamy budou zobrazeny jinou formou nežli tabulkovou pro zpříjemnění práce s aplikací. K zobrazení záznamů budou tedy využity kartičky. Matriky navržené k jednotlivým záznamům by měli být přidruženy ke kartičkám, aby opět nebylo nutné přecházet na jinou stránku.



Obrázek 3.4: Stránka záznamů - zobrazení kartiček

Poznámky

Vytváření poznámek by mělo být uživateli přístupné u jednotlivých objektů, kterých se budou týkat. Jejich vytváření by opět mělo proběhnout na stejné stránce. Využijeme k tomu tedy zase vyskakovací okénko. Uživatel by měl mít možnost si někde zobrazit všechny poznámky pohromadě. K tomu bude sloužit samostatná stránka tvořená tabulkou zobrazující jednotlivé poznámky tvořené pro konkrétní typy objektů. Návrh poznámkového okna je vyobrazen na obrázku 3.5



Obrázek 3.5: Vyskakovací okno - editace poznámky

Kapitola 4

Použité technologie

Tato kapitola se zabývá popisem technologií, které byly využity při implementaci aplikace. U každé technologie je její stručný popis a způsob jakým byla v aplikaci využita. Jsou zde popsány i důležité knihovny, moduly nebo balíčky, které byly použity.

4.1 Python

Python je vysokoúrovňový interpretovaný objektově orientovaný skriptovací jazyk. Dynamické typování, vázání a vestavěné vysokoúrovňové datové struktury z něj dělají jazyk často používaný pro rychlý vývoj aplikací. Python má jednoduchou a lehce čitelnou syntaxi, díky čemuž je kód lehce udržovatelný. Python také podporuje moduly a balíčky, za pomocí kterých lze do kódu vkládat různé funkce a datové struktury [16].

Tento jazyk byl v práci využit pro implementaci parseru GEDCOM souborů, matcheru záznamů na matriky a všech skriptů týkajících se extrakce dat.

Python-gedcom

Python-gedcom[1] je modul vytvořený Nicklasem Reincke, určený k parsování, analyzování a manipulaci s GEDCOM soubory ve verzi 5.5.

Tento modul byl v práci využit pro parsování vstupních GEDCOM souborů.

Textsearch

Textsearch[2] je knihovna určená k vyhledávání či manipulaci řetězců v textu. Je schopná vyhledat řetězce ze zadané množiny v textu na základě podobnosti celého slova. Výhodou je její rychlost, které nabývá díky použití jazyka C.

Knihovna se používá pro vyhledávání názvů území v řetězcích reprezentující místo kde proběhla zkoumaná událost.

[1] Python-gedcom - https://pypi.org/project/python-gedcom/
[2] Textsearch - https://github.com/kootenpv/textsearch

Orator

Orator[3] ORM[4] je nástroj poskytující základní operace nad relačním databázovým modelem. Syntaxí byl inspirován Laravelem, což je ORM framework pro jazyk PHP (viz níže). Kromě základních ORM operací sebou Orator přináší i možnost vytvářet dotazy skrze query builder.

Tento nástroj byl použit pro lepší práci s databází v jazyce Python. Využívají ho parser i matcher.

Beautiful Soup

Beautiful Soup[5] je knihovna používaná k získávání informací z webových stránek. Využívá HTML nebo XML parser, pomocí kterého vytváří strom složený z elementů stránky obsahující informace vyskytující se na dané stránce.

V práci byla knihovna použita při zpracování archivů obsahující matriční knihy.

Selenium

Tento balíček umožňuje využívat Selenium[6] v rámci jazyka Python. Selenium slouží k automatizaci interakcí webového prohlížeče.

Balíček byl použit v případech, kdy zpracovávaný archiv využíval technologií podporujících asynchronní načítání dat.

4.2 PHP

PHP, hypertextový preprocesor, je imperativní objektově orientovaný skriptovací jazyk. Je primárně využívaný k programování dynamických webových stránek a aplikací. Pracuje převážně na straně serveru, kde generuje HMTL kód, který pak posílá klientovi jakožto výsledek akcí. Je to jeden z nejrozšířenějších jazyků a právě i díky tomu na něj existuje řada fragmentů, funkcí či frameworků [19].

V práci byl použit k implementaci webové aplikace.

4.2.1 Laravel

Laravel[7] je webový aplikační framework s výstižnou a elegantní syntaxí, jehož cílem je lehčí a rychlejší vývoj webových aplikací. Definuje rozložení MVC a ulehčuje implementaci běžných funkčností jako autentizace, routování, práce s sezením. Také vylepšuje práci s databází obalováním SQL dotazů do kódové syntaxe a brání tak útokům typu SQL injection.

Laravel byl využit pro definici základní webové struktury aplikace.

4.3 HTML

HTML, HyperText Markup Language, je značkovací jazyk, který se používá k vytváření základní obsahové kostry webových stránek. Obsah webové stránky mohou tvořit texty,

[3]Orator - `https://orator-orm.com/`

[4]ORM - Objektově relační mapování, technika zajišťující konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem

[5]Beautiful Soup - `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`

[6]Selenium - `https://www.selenium.dev/`

[7]Laravel - `https://laravel.com/`

obrázky, tabulky, multimédia a další prvky. Dříve jazyk HTML sloužil i k formátování vzhledu, dnes už se k tomu používají kaskádové styly, které umožňují vytvářet vzhled jako druhou, na obsahu nezávislou vrstvu a různě ho měnit podle aktuálního kontextu [1].

V práci byl použit k implementaci pohledů.

4.4 JavaScript

JavaScript je skriptovací jazyk učený pro tvorbu moderních dynamických webů. Pracuje hlavně na straně klienta společně s HTML a kaskádovými styly. V dnešní době je tento jazyk velice populární a existuje pro něj řada knihoven a frameworků [8].

V aplikaci se JavaScript používá k implementaci funkcí spuštěných na klientské straně.

4.4.1 jQuery

Jedná se o knihovnu napsanou pro jazyk Javascript, která usnadňuje manipulaci s elementy HTML dokumentu, ovládání JavaScriptových událostí, práci s AJAX a také sebou přináší různé funkce, animace a efekty. Výhodou jQuery[8] je, že díky její velké oblíbenosti pro ni existuje mnoho pluginů, které se dají jednoduše vložit do kódu a používat.

4.4.2 Ajax

Ajax[9], Asynchronous JavaScript and XML, je technologie používaná k vytváření asynchronních webových aplikací. Umožňuje vytvářet asynchronní požadavky, které běží na pozadí a nenesou sebou nutnost načítat celou stránku.

V aplikaci byl Ajax využit pro jakékoliv operace nevyžadující znovu načtení stránky.

4.5 JSON

JSON, JavaScriptový objektový zápis, je datový formát nezávislý na počítačové platformě. Slouží k přenosu dat, jež mohou být organizována v polích nebo agregována v objektech. Vstupem může být libovolná datová struktura a jeho výstupem je pro člověka jednoduše čitelný řetězec. JSON patří k jedněm z nejpoužívanějším datových formátů [4].

V práci byl využit pro serializaci informací o jednotlivých matričních knihách, pro předávání parametrů požadavků či výsledků funkcí a metod.

4.6 MySQL

Jedná se o relační multiplatformní databázový server komunikující pomocí rozšířeného jazyka SQL. MySQL se zaměřuje na výkon a dobrou škálovatelnost (lze jej instalovat na většinu známých operačních systémů). Díky volné licenci se jedná o jeden z nejpoužívanějších databázových systémů [13].

V aplikaci bylo MySQL využito pro veškerou práci s databází.

[8] jQuery - `https://jquery.com/`

[9] Ajax - `https://www.w3schools.com/whatis/whatis_ajax.asp`

Kapitola 5

Implementace

Obsahem této kapitoly je popis jednotlivých fází implementace. Je zde popsáno, jakým způsobem byly zpracována data potřebná pro funkčnost aplikace. Následně tato kapitola obsahuje způsob řešení jednotlivých částí, které pokrývají hlavní funkce aplikace, jako jsou zpracování vstupního souboru a navazování matrik na záznamy. V jednotlivých fázích jsou také vypíchnuty nejdůležitější problémy, na které bylo naraženo a jejich řešení.

5.1 Příprava dat

5.1.1 Území České republiky

Jelikož má být aplikace schopná pracovat i s územím bylo nutné zpracovat územní celky v rámci České republiky. Tyto celky jsou popsány pomocí RUIAN[1] číselníků ISÚI, které v sobě obsahují následující informace:

- **RUIAN kód** - číselná identifikace v rámci typu území
- **Název** - řetězec označující území
- **Kód nadřazeného území** - číselná identifikace území, nacházejícího se o jeden stupeň v hierarchii výše
- **Od** - datum od kdy jsou tyto informace platné
- **Do** - datum do kdy jsou tyto informace platné
- **Datum vzniku** - datum přidání informací do číselníku
- **Ostatní** - informace specifické ke konkrétnímu typu území

Pro každý typ území bylo tak zpracováno 7 číselníků od státu až po části obcí. K zpracování číselníků byl napsán shell skript, který číselníky zformátoval do podoby SQL skriptu. Ke každému území byla přiřazena číselná hodnota označující typ území od 0 do 5. Části obcí a městské celky byly zařazeny do stejného typu 5 (viz tabulka 5.1). V databázi nebyly RUIAN kódy použity jako primární klíč tabulky území, protože nebyly v rámci celku unikátní (pouze v rámci typu). Z tohoto důvodu byl napsán Python skript, který navázal území na jejich nadřazené celky za pomocí identifikačního čísla uděleného v rámci databáze a také upravil SQL skript aby obsahoval tyto hodnoty.

[1] RUIAN - `https://www.cuzk.cz/ruian/Poskytovani-udaju-ISUI-RUIAN-VDP/Ciselniky-ISUI.aspx`

Číselný typ území	Název typu území
0	Stát
1	Regiony
2	Kraje
3	Okresy
4	Obce
5	Části obcí
5	Městské celky

Tabulka 5.1: Zjednodušené rozdělení území ČR na typy

Zeměpisné souřadnice

Důležitou součástí zpracování území bylo přiřadit k jednotlivým celkům souřadnice. Získání souřadnic bylo docíleno pomocí web scrapingu[2] v jazyce Python. Skript souřadnice získával z webu Wikipedie, kde každá obec a její části mají vytvořenou stránku, na které se nachází odkaz na web GeoHack[3]. Zde jsou pro každé území uložené souřadnice v podobě zeměpisné šířky a délky. Tyto souřadnice v decimální podobě skript přidává k vytvořenému SQL skriptu, tak aby mohli být uloženy v databázi. Vyhledávání na webu Wikipedie probíhalo pomocí zadání názvu a RUIAN kódu území. Jestliže se jednalo o část obce nesoucí stejné jméno jako její nadřazená obec, pak byly i jejich souřadnice stejné. Souřadnice museli být otestovány na výskyty duplicitních hodnot pro území, které spolu neměli nic společné a v těchto případech manuálně opraveny.

Souřadnice pro území jiných typů než 4 a 5 nebylo potřeba zpracovat, protože v aplikaci pracujeme hlavně s dříve zmíněnými typy území.

Díky zpracování souřadnic všech potřebných celků bylo možné namapovat sousedy jednotlivých území do okruhu 5 km a tuto mapu uložit do databáze pro příští využití, tak aby se vzdálenost nemusela pokaždé vypočítávat. Vzdálenost mezi územími byla počítána pomocí Haversinova vzorce [6]:

$$d = 2 * r * arcisn(\sqrt{sin^2(\frac{\varphi_2 - \varphi_1}{2}) + cos\varphi_1 * cos\varphi_2 * sin^2(\frac{\lambda_2 - \lambda_1}{2})}) \tag{5.1}$$

kde d je vzdálenost mezi dvěma body, r je poloměr země v kilometrech (6371 km), φ_1, φ_2 jsou souřadnice zeměpisných šířek a λ_1, λ_2 zeměpisných délek bodů 1 a 2 v desetinných hodnotách. Funkce, která pracuje s daným vzorcem byla převzata z webu GeeksforGeeks [18]. Při testování je možné narazit na pár odchylek, které jsou však v našem případě naprosto zanedbatelné.

5.1.2 Matriky

Extrakce dat

Bylo zapotřebí zpracovat archivy uchovávající digitalizované matriční knihy. Jejich zpracování bylo docíleno opět pomocí web scrapingu v jazyce Python. Muselo být napsáno osm skriptů, "scraper_NAZEVMESTA.py", jelikož každý archiv měl jinou strukturu.

Nejdříve proběhla analýza struktury webu, zkoumání elementů, které uchovávají důležité informace. Každý web měl seznamy matrik zobrazené pomocí tabulkových elementů,

[2]web scraping - sběr dat z webových stránek

[3]Geohack - `https://geohack.toolforge.org/`

mezi kterými se přecházelo přes stránkování. Bylo zapotřebí tedy nejdříve získat odkazy na jednotlivé detaily matrik z těchto tabulkových elementů přes všechny jejich stránky. Některé archivy neměli možnost zobrazit matriky všech typů zároveň a bylo tedy zapotřebí procházet přes všechny tabulky všech typů. Zpracování odkazů skripty provádí po spuštění se vstupním argumentem "links"a jejich výstupem je soubor obsahující jednotlivé odkazy po řádcích. Z tohoto souboru se pak čerpá při zpracovávání jednotlivých matrik.

Při zpracování konkrétní matriky jsou pro účely aplikace důležité jen určité informace. Matriky byly s těmito informacemi uloženy do souborů formátu JSON pro každý archiv zvlášť.

Nejdůležitější informací je uzemní rozsah matriky. Tedy jaké obce a části obcí tato matrika pokrývá. Pokud je zde uveden, pak je pro nás důležitý i okres, pod kterým působí, a to pro jasnou identifikaci obcí kvůli duplicitě jmen. Do souboru JSON jsou území ukládány do pole, kde klíčem je název území a hodnotou slovník obsahující dodatečné informace o území, které archiv poskytuje. Další důležitou informací je obsah matriky a jeho časový rozsah. Tedy jaký typ události je v matrice zapsán a v jakém časovém rozmezí. Typy událostí jsou v souboru JSON označovány písmeny N (narození), Z (zemřelí), O (oddáni) a u každého typu je uveden časový rozsah. V případě indexů je postup stejný jen se pro označení používají zkratky I-N, I-Z, I-O. Pokud archiv neuvádí časové rozsahy pro jednotlivé typy, pak je každému typu přiřazen obecný rozsah matriky. Mezi ostatní důležité informace pak patří původce, signatura a inventární číslo, které pak aplikace používá jako identifikaci matrik pro uživatele.

Přiřazení území

Soubory JSON, které vznikly při extrakci dat z archivů, bylo zapotřebí ještě upravit. Konkrétně bylo potřeba namapovat území uvedené u jednotlivých matrik na konkrétní územní prvky obsažené v databázi aplikace.

Způsob jakým archivy zapisují území k matrikám je popsán metodikou popisu matrik, kterou zveřejnilo Ministerstvo vnitra České republiky [11]. Podle této metodiky mohou být lokality vyplněny dvěma hlavními způsoby. V případě, že se matrika vztahuje ke všem částem nějaké obce mohou být vypsány buďto všechny části této obce nebo jen název dané obce. Druhý způsob zápisu vytváří problém, protože je potřeba aby při využití daného zápisu archivy uváděli u jednotlivých území i jejich typ. Pokud tomu tak není je velmi obtížné rozeznat jestli je myšlena obec jako celá nebo pouze její stejnojmenná část. Dalším problémem s touto metodikou je, že ne všechny archivy ji zcela dodržují a při zkoumání jednotlivých archivů se narazilo na případy kdy byla uvedena jak obec, tak i některá její část, a to i přestože uvedení dané obce automaticky zahrnuje i tu zmíněnou část. Z těchto důvodů se území mapují 1:1 k jejich uvedenému typu a neprovádí se zahrnutí částí v případě uvedení území typu obec. Dané zahrnutí se řeší až v kapitole popisující navazování matrik k záznamům 5.3. Postup přiřazování území se lišil pro jednotlivé archivy a je popsán v následujících odstavcích.

Jestliže byly územní prvky v archivech uváděny pomocí hierarchie (část obce, obec, okres) pak se dalo území jednoznačně určit i v případě duplicitních jmen. Výjimkou byly pouze obce Březina, nacházející se v okrese Brno-venkov, jelikož se jedná o speciální případ, kdy se nachází dvě stejnojmenné obce ve stejném okrese. Zde bylo zapotřebí přiřadit matriky manuálně. Výhodou tohoto zápisu také bylo, že u každého území byl známý jeho typ. Mezi takové archivy patří Opavský, Litoměřický, Brněnský a archiv Hradce Králové.

Pouze jediný archiv uváděl u území jejich identifikační RUIAN kód a tím byl Třeboňský archiv. Zde bylo tedy přiřazení nejjednodušší, jelikož se díky tomuto kódu hned vědělo, o který územní prvek se jedná i jaký je jeho typ.

Jestliže ovšem archivy neuvedly ani RUIAN kód a ani žádnou územní hierarchii musel být zvolen jiný postup přiřazení území. Mezi takové archivy patřily Středočeský a Plzeňský. Pro tyto archivy byla vytvořena mapa okresů, které pokrývají, a okresů, které s těmito okresy sousedí. Sousední okresy byly uvedeny z důvodu lokalit vyskytujících se na hranicích okresů. Tímto způsobem byl zúžen územní rozsah a odpadly tak některé duplicitní názvy. Postup se pak nadále u těchto dvou archivů lišil.

U Středočeského archivu byly všechny uvedené území typu část obce a vždy zde byl uveden původce a okres, ve kterém se původce nachází. Díky tomu se dalo území jednoznačně určit za pomocí souřadnic na základě vzdálenosti mezi odhadovaným územím a původcem. V případě duplicitních názvů bylo tedy zvoleno to území, které se nacházelo nejblíže k původci.

Určování u Plzeňského archivu bylo nejtěžší jelikož na rozdíl od Středočeského zde nebyl uveden okres původce. Jestliže se tedy dal původce jednoznačně určit, pak se dali opět na základě souřadnic určit i ostatní území. Pokud ovšem nebylo možné původce určit, určili se ostatní jednoznačně vypsané území, z jejichž souřadnic se vypočítal centroid, který nám nahradil původce, a na základě kterého jsme pak určili ostatní nejednoznačné území. Plzeňský archiv také neuváděl jakého typu je dané uvedené území a tak v případě, že nešlo určit jestli je myšleno území jako obec nebo jako jeho stejnojmenná část byla zvolena část. Obec byla zvolena pouze v případě, že k sobě uvedená obec neměla stejnojmennou část, tak jako je tomu například u obce Plzeň. Tento postup byl odůvodněn zkoumáním archivu a zjištěním, že v mnoha případech kde se nedalo rozhodnout o typu zde byla uvedena i jiná část spadající do stejné obce a dle výše zmíněné metodiky by to tedy znamenalo, že i nejasné území by pak mělo být myšleno jako typ část obce.

Pokud bylo území určeno, tak k němu v JSONu byl připsán jeho RUIAN kód, typ označený číselnou hodnotou, souřadnice pomocí zeměpisné šířky a délky, název okresu a případně i název obce (pro území typu 5), pod kterou území spadá.

Ne všechny území se dalo přiřadit na území v rámci naší databáze. U matrik mohli být totiž uvedeny i zaniklé území, příliš malé územní prvky (ulice, ZSJ[4], kostely, ...) a nebo i území z ciziny. V případech kdy jsme tedy nebyli schopni určit území mu byla přiřazena hodnota RUIAN jako -1.

Finální verze souborů JSON po přiřazení území tedy pak vypadá takto:


```
{
    "url": "https://digi.archives.cz/da/permalink?xid=29e3f91ac933f2f4",
    "puvodce": "Pavlovice u Přerova, římskokat. f. ú.",
    "signatura": "Př XI 64",
    "invCislo": null,
    "typ": "katolická",
    "jazyk": null,
    "rozsah": "1947 - 1949",
    "obsah": [
    "O•1947-1949"
    ],
    "uzemi": {
        "Pavlovice u Přerova": {
```


[4]ZSJ - základní sídelní jednotka

```
            "typ": 4,
            "ruian": 516694,
            "latitude": 49.469425,
            "longitude": 17.547772,
            "okres": "Přerov",
            "varianty": []
        }
    },
    "okresy": [
    "Přerov"
    ]
}
```


Výpis 5.1: Ukázka zpracované matriky ve formátu JSON z Opavského archivu. Jedná se o matriku oddaných v rozmezí let 1947 až 1949. Tato matrika pokrývá jediné území, Pavlovice u Přerova, obec v okrese Přerov.

Zpracování dat

Připravené soubory JSON pak zpracovával implementovaný Python skript, který pro jednotlivé matriky vytvořil příslušné záznamy do tabulky matrik v databázi. Zároveň tento skript vytvořil záznamy do propojovací tabulky reflektující vazbu mezi územím a matrikou pro přiřazené území. Nepřiřazené území ignoroval.

5.2 GEDCOM parser

K zpracování souborů, které uživatel nahraje prostřednictvím aplikace slouží skript "gedcom_parser.py" napsaný v jazyce Python. Tento skript se stará o zpracování všech osob a rodin zapsaných v nahraném souboru, kontrolu validity informací a vytváření záznamů v případě nevalidní informace. Využívá k tomu knihovnu "python-gedcom". Skript prostřednictvím knihovny prochází přes jednotlivé osoby/rodiny v souboru a získává o nich informace. Při zpracování osoby se kontroluje celistvost událostí narození a úmrtí, u rodin událost oddání. Nejdřív je zkontrolován formát data, tudíž jestli je datum zapsáno a zdali je ve správném tvaru. Následně je kontrolováno místo, ke kterému je událost připsána. U míst mohou nastat 3 typy situací. Místo nemusí být uvedeno vůbec a tím pádem je událost nevalidní, nebo je místo uvedeno a jednoznačně přiřazeno, a v tomto případě je událost validní. Poslední situace, jež může nastat je, že řetězec u místa není prázdný, ale buďto jsme k němu nenašli v databázi území nebo jsme jich našli více a nemůžeme ho jednoznačně určit. Pokud nastane poslední situace, tak může být informace obsažená v řetězci jak validní tak nevalidní. Jestliže by se jednalo o místo z ciziny, zaniklé místo nebo příliš malý územní prvek, pak by byla informace validní a místo by se pouze nenacházelo v databázi. Zároveň se ale může jednat o poznámku nebo nesmyslný řetězec kdy by se jednalo o nevalidní informaci. V takové situaci skript nerozhoduje validitu informací. Ta je rozhodnuta až prostřednictvím aplikace samotným uživatelem. Osoby a rodiny, ke kterým se řetězec obsažený v tagu místa vztahuje, jsou tak podle tohoto řetězce sdružovány za pomocí pythonovského slovníku. To z toho důvodu, aby se pak uživatelem zvolené místo mohlo přiřadit ke všem osobám/rodinám, kterých se místo týká.

U události úmrtí je vhodné ještě zmínit, že pokud známe věk osoby, tedy známe její rok narození, a víme, že je osoba mladší 100 let, pak v případě, že zde není událost úmrtí

uvedena, ji neprohlašujeme za chybějící. Pokud ale nejsme schopni zkontrolovat věk osoby, pak automaticky říkáme, že událost chybí.

U události oddání může nastat, že uživatel vyplnil do kolonky místa dvě území. V tomto případě je myšleno, že svatba proběhla v jednom ze dvou míst a tato místa reprezentují místa narození obou svatebčanů. I s tímto případem aplikace počítá a uloží obě tyto místa k patřičnému záznamu rodiny. V případě, že by však uživatel uvedl tři možná území, je tím pravděpodobně myšleno, že první území představuje místo kde proběhla svatba a zbylé dvě jsou opět místa narození svatebčanů. V tomto případě je uloženo pouze první zmíněné místo.

Po kontrole validity informací je rozhodnuto zda se bude pro osobu/rodinu vytvářet záznam a jakého typu bude. Máme tři typy záznamů. Ty kde chybí pouze datum (1), ty kde chybí místo (2) a ty kde chybí obojí (3). Rozdělování na typy je důležité pro navazování matrik k záznamům při určování časových rozsahů a lokalit (viz. kapitola 5.3).

Po té co skript provedl kontrolu všech elementů jsou osoby a rodiny zapsány do databáze a následně provázány prostřednictvím cizích klíčů ukazujících na otce/matku v případě osoby a manžela/manželku v případě rodiny. Díky těmto vazbám jsme později schopni získat veškeré potřebné příbuzné osob při navazování matrik. Jako další skript nahraje do databáze záznamy vytvořené k osobám/rodinám.

Úplně poslední činností parseru je, že se pokusí automaticky určit území s více návrhy míst na základě okolí, ve kterém se pohybovali osoby z celého souboru. Vybere sousedy do 5km ze všech míst uvedených v souboru, které byly jednoznačně určeny. Jestliže se některý z návrhů území nachází mezi těmito sousedy, pak se pravděpodobně jedná o dané území. Pokud se ale nachází více stejnojmenných území mezi těmito sousedy, pak opět nelze určit, o které se jedná.

Výstupem parseru je slovník obsahující jak určené území, jenž jsou uživatelem potřeba zkontrolovat, tak území, které se nepodařilo identifikovat a uživatel je musí doplnit.

5.3 Matcher

Jedná se o Python skript sloužící k navazování vhodných matrik k záznamům, které vytvořil parser. Prochází přes všechny záznamy a postupně sbírá informace od blízkých příbuzných osoby. Tedy z rodiny, kde osoba působí jako dítě, a z rodiny, kde osoba působí jako rodič. U události je potřeba určit dva typy informací. Skript určuje v jakém časovém rozsahu se událost mohla odehrát a v jakých lokalitách se osoba mohla pohybovat. Po určení časového rozsahu a lokalit jsou vybrány matriky, které vyhovují stanoveným podmínkám. Tedy matriky, které jsou navázány na území vybrané v lokalitách, a které alespoň jedním rokem spadají do časového rozsahu. Jestliže událost má jednu z potřebných informací vyplněnou, jedná se tedy o záznamy typu 1 nebo 2, pak není zapotřebí tuto informaci odhadovat a je použita vyplněná hodnota.

Časový rozsah

Časový rozsah se určuje na základě roků, ve kterých se odehráli události jak zkoumaných osob tak jejich příbuzných. Pro určení rozsahů se používají stanovené hodnoty, které určují minima či maxima pro specifické události osob. Tyto hodnoty si může uživatel přizpůsobit v rámci prostředí webové aplikace při nahrávání souboru. Hodnoty byly přejaty a následně upraveny z bakalářské práce Jakuba Konetzného [7].

Výchozí hodnoty používané pro výpočet časových rozsahů:

- **BIRTH_MIN = 15** - minimální věk, kterého musela osoba dosáhnout aby mohla mít děti
- **BIRTH_MAX = 70** - maximální věk muže, kdy ještě mohl počít dítě
- **BIRTH_MAX_WOMAN** = 50 - maximální věk ženy, kdy ještě mohla počít dítě
- **DEATH_MAX = 100** - maximální věk, kterého mohla osoba dosáhnout
- **MARRIAGE_MIN = 15** - minimální věk potřebný k možnosti vstoupit do manželství
- **MARRIAGE_MAX = 100** - maximální věk kdy osoba mohla vstoupit do manželství

Pokud je určena pouze jedna z hodnot od/do, je ke druhé přičtena či odečtena hodnota 100 let. Pokud není určena ani jedna strana rozsahu, pak neprobíhá navazování matrik z důvodu nedostatečných informací. Při výběru hodnot od/do se pro hodnotu:

- **od** vybírá nejvyšší hodnota ze všech navržených
- **do** nejnižší hodnota ze všech navržených

Tím je dosaženo nejmenšího možného rozsahu.

Lokalita

Možné území, ve kterých se událost mohla odehrát, se vybírají ze známých událostí příbuzných či dané osoby. Každé zvolené místo je zařazeno do prioritní skupiny, která označuje relevanci daného místa k události. Tato relevance ukazuje jak moc je dané území, vybrané z určitého typu události, relevantní k typu zkoumané události. Zároveň tato priorita představuje pravděpodobnost nalezení hledané informace v navržené matrice. Do lokalit se zahrnují i území nacházející se v okruhu 5 km okolo navrženého území. Dalším důležitým dodatkem je, že pokud je lokalita typu část obce, pak se do možných území musí přidat i obec, ve které tato část leží. To kvůli metodice MVČR[5], která je popsána v kapitole 5.1.2 v části o přiřazení území.

Prioritní skupiny:

- 1-3 - území vybrané z vysoce relevantních událostí
- 4-6 - území vybrané z méně relevantních událostí
- 7-9 - území vybrané z událostí s velice nízkou relevancí

5.3.1 Událost narození

U události narození jsou pro nás nejvíce relevantní osoby z rodiny, kde zkoumaná osoba vystupovala jako dítě. Nejvyšší prioritu zde mají sourozenci. Jestliže víme kde se narodili sourozenci, pak se zde s vysokou pravděpodobností narodila i naše osoba. Další významnou událostí je pak úmrtí rodičů osoby. Lokalita, ve které zemřeli rodiče zkoumané osoby nám s vysokou pravděpodobností představuje místo, kde se pohybovala rodina dané osoby a tedy

[5]MVČR - Ministerstvo vnitra České republiky

také místo, kde se osoba mohla narodit. Tyto události jsou proto označeny prioritami 1 a 2. Poslední událostí z nejvyšší prioritní skupiny je svatba osoby. V minulosti totiž bývalo zvykem, že se svatba konala v místě narození jednoho ze svatebčanů. Priority ostatních událostí jsou uvedeny v tabulce relevance 5.2.

Priorita	Vybraná událost
1	Narození sourozenců
2	Úmrtí rodičů
3	Svatby osoby
4	Svatba rodičů
5	Narození rodičů
6	Úmrtí osoby
7	Narození dětí
8	-
9	-

Tabulka 5.2: Tabulka relevance událostí pro událost narození.

U časových rozsahů je tomu podobně jako u lokalit. Některé události nám zúží rozsah více, některé méně. U narození nám při určení rozsahu hodně sdělí například roky úmrtí rodičů zkoumané osoby. Víme totiž, že osoba se nemohla narodit po úmrtí rodiče (u otce +1 rok). Dále nám pak například pomohou jiné události ze života zkoumané osoby. Pokud víme v jakém roce proběhla svatba osoby, pak víme že se musela narodit nejpozději do takového roku aby měla dostatečný věk na vstup do manželství. Také narození prvního dítěte nám může pomoci upřesnit rozsah. Celkově všechny události z okolí nám řeknou hrubý časový rozsah, ale jen některé nám pomohou ho dostatečně zúžit.

Příklad vzorce pro výpočet časového rozsahu narození osoby pomocí roku úmrtí rodiče:

$$rok_{OD} = umrti\_rodice - DEATH\_MAX + BIRTH\_MIN \tag{5.2}$$

$$rok_{DO} = umrti\_rodice\ (+1) \tag{5.3}$$

Ve výše uvedeném vzorci 5.3 je hodnota do velmi jasná, osoba se nemohla narodit po té co její rodič zemřel. Závorka obsahující hodnotu +1 je zde pro případ, že rok úmrtí je od otce osoby. Pak je zde ještě rezerva jednoho roku. Hodnota od ve vzorci 5.2 počítá s maximálním možným věkem, kterého se rodič osoby mohl dožít. Hodnota je ovšem ještě posunuta o minimální věk, který rodič osoby musel mít aby mohl počít dítě. Podobným způsobem lze odhadovat rozsah i z ostatních událostí.

5.3.2 Událost úmrtí

U události úmrtí jsou pro nás nejvíce relevantní osoby z rodin, kde zkoumaná osoba působí jakožto rodič. Nejvyšší prioritu mají děti osoby. Místo narození, či místo svatby dětí zkoumané osoby nám mohou reprezentovat lokalitu, ve které osoba prožila svůj život a tím pádem tedy i místo kde osoba pravděpodobně zemřela. Tyto události řadíme do nejvyšší prioritní skupiny a mají tak priority 1 a 2. Úmrtí partnera je pak poslední událost spadající do nejvyšší skupiny, jelikož je velice pravděpodobné, že osoba zemřela ve stejném místě kde její partner. Další událostí, jež nám může něco povědět o místě úmrtí osoby, může být místo kde se konala svatba osoby. To ovšem jen v případě, že se svatba konala v místě kde osoba žila a tak tedy tato událost spadá do střední prioritní skupiny, hodnota priority 4.

Určit lokalitu, ve které osoba zemřela, je náročné, protože málo událostí z jejího okolí je relevantní k jejímu úmrtí.

Priorita	Vybraná událost
1	Narození dětí
2	Svatba dětí
3	Úmrtí partnera
4	Svatby osoby
5	-
6	-
7	Narození osoby
8	-
9	-

Tabulka 5.3: Tabulka relevance událostí pro událost úmrtí

U určení časového rozsahu úmrtí nám hodně poví jiné události osoby, nebo osob z rodiny, kde je zkoumaná osoba rodičem. Víme, že osoba nemohla zemřít před rokem kdy se narodila, ani před rokem kdy měla svatbu. Dále víme, že osoba nemohla zemřít před narozením dítěte (u muže -1 rok). Opět určit alespoň hrubý rozsah není nijak složité, protože víme, že maximální možný věk osoby je omezený.

5.3.3 Událost oddání

U události oddání je opět důležité nahlížet na všechny lokality, ve kterých se pohybovali svatebčané a jejich příbuzní. Svatba mohla totiž proběhnout jak v místě narození manžela, tak manželky neboli místě úmrtí jejich rodičů a místě narození sourozenců. Svatba také mohla proběhnout v místě jejich pobytu, tedy tam kde se narodili či kde měli svatbu děti svatebčanů, nebo kde svatebčané zemřeli.

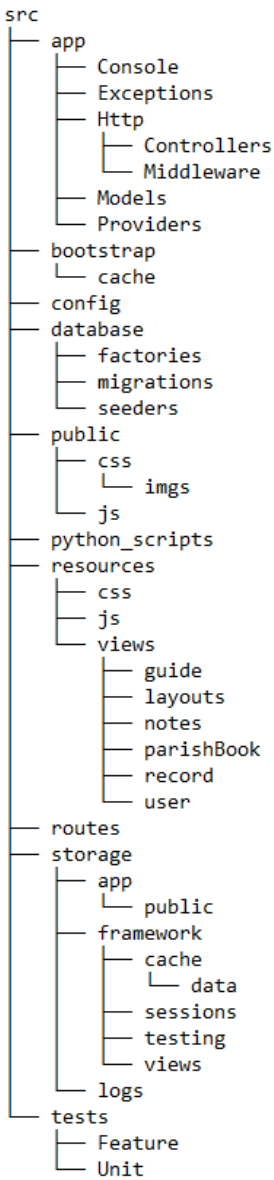
Priorita	Vybraná událost
1	Narození svatebčanů
2	Úmrtí rodičů
3	Narození sourozenců
4	Narození dětí
5	Úmrtí svatebčanů
6	-
7	Svatby dětí
8	Svatby rodičů
9	Narození rodičů

Tabulka 5.4: Tabulka relevance událostí pro událost oddání

U časových rozsahů je opět důležité nahlížet na obě strany. Víme, že svatba musela proběhnout v době, kdy na ni oba svatebčané měli dostatečný věk. Také musela proběhnout před jejich úmrtím. Vzhledem k tomu, že v minulosti nebylo velmi časté, aby se děti narodili po svatbě, tak s tím aplikace počítá a tedy víme, že děti se narodili až po svatbě. Všechny tyto hodnoty nám postupně zužují náš rozsah.

5.4 Webová aplikace

Poslední částí implementace bylo vytvořit samotnou aplikaci, která poběží na serveru a bude spouštět dříve zmíněné Python skripty. Samotná aplikace byla vytvořena pomocí frameworku Laravel, který využívá návhrový vzor MVC, Model View Controller. Laravel sebou přináší pevně danou adresářovou strukturu zobrazenou na obrázku 5.1. Celá aplikace je uložena v adresáři src, ten je tedy kořenovým adresářem. Modely jsou uloženy v adresářích `/app/Models`, kontrolery v adresářích `/app/http/Controllers` a pohledy v `/resources/views`. Za zmínku také stojí soubor starající se o routování[6] v rámci naší aplikace web.php, nacházející se v adresáři `/routes`. Struktura je doplněna o adresář `/python_scripts`, ve které se nachází parser a matcher.



Obrázek 5.1: Adresářová struktura Laravel projetku

[6]routování - směrování požadavků

V následujících kapitolách budou popsány implementace jednotlivých vrstev návrhového vzoru MVC v rámci frameworku Laravel.

5.4.1 Model

Model v Laravel frameworku z většiny případů zastupuje databázovou tabulku, proto modely už v základu dědí z třídy `Model`, která obsahuje proměnné pro definování různých atributů a hodnot, a zároveň nám poskytuje objektově relační mapování pro práci s databází. Navíc můžeme vytvářet testovací data pro pomocí tzv. továrny. To je také důvod, proč model již ze základu používá trait třídu HasFactory [10].

V rámci naší aplikace byly vytvořeny modely pro všechny tabulky reprezentující entity. Jedná se tedy osm modelů. Tyto modely obsahují funkce pro získání vhodných/vyfiltrovaných záznamů z databáze či generování obsahu v kódu HTML. Základní metody například pro vyhledání záznamu pomocí primárního klíče nebylo potřeba implementovat, protože jak bylo zmíněno v odstavci výše, každý z našich modelů dědí od Laravel třídy `Model` a ta tyto metody poskytuje (například metoda `find`). Metody těchto modelů jsou pak volané v kontrolerech (viz kapitola 5.4.3). Asi nejdůležitějším modelem je model `Record`. Obsahuje totiž metody pro paginaci záznamů, generování kartiček a nebo pro získání matrik k jednotlivým záznamům.

Generování HTML kódu

Jak bylo zmíněno výše modely obsahují metody pro generování kódu ve značkovacím jazyce HTML. Tyto metody jsou potřeba kvůli asynchronním požadavkům, kdy je zapotřebí dynamicky zobrazit uživateli elementy naplněné daty z databáze. Ke každé operaci týkající se nějakého objektu databáze tak existuje metoda, která je schopná vygenerovat elementy a naplnit je daty objektu, které je potřeba zobrazit. Většinou se jedná o statické metody náležící modelům příslušným k danému objektu. Tyto metody zapisují kód formou řetězce, který je pak výstupem samotných metod. Řetězec je poté vložen do html prostřednictvím funkce `.html()` z knihovny jQuery. Tato funkce vloží do elementu, nad kterým je volaná, kód HTML, který je funkci předán jako řetězec.

5.4.2 View

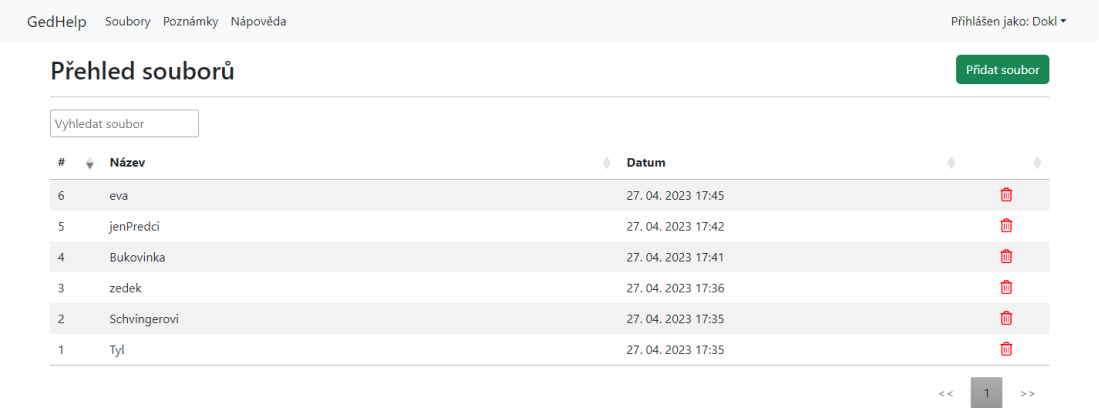
Jedná se o vrstvu, která slouží k zobrazování dat uživateli. Obsahuje Blade šablony (s HTML kódem). Blade je šablonovací systém, který na rozdíl od ostatních systémů nezakazuje používání PHP kódu v pohledu, tudíž neslouží pouze pro vkládání proměnných do HTML kódu. Blade používá speciální syntaxi, jež je poté převedena na normální PHP kód, a zároveň tento vygenerovaný pohled je následně uložen do mezipaměti pro příští použití kvůli rychlejšímu načítání (pokud není daný pohled modifikován) [9].

Naše aplikace kromě základních uvítacích a autentizačních pohledů nabízí tři hlavní pohledy. V následujících podkapitolách budou tyto pohledy podrobně popsány.

Hlavní pohled

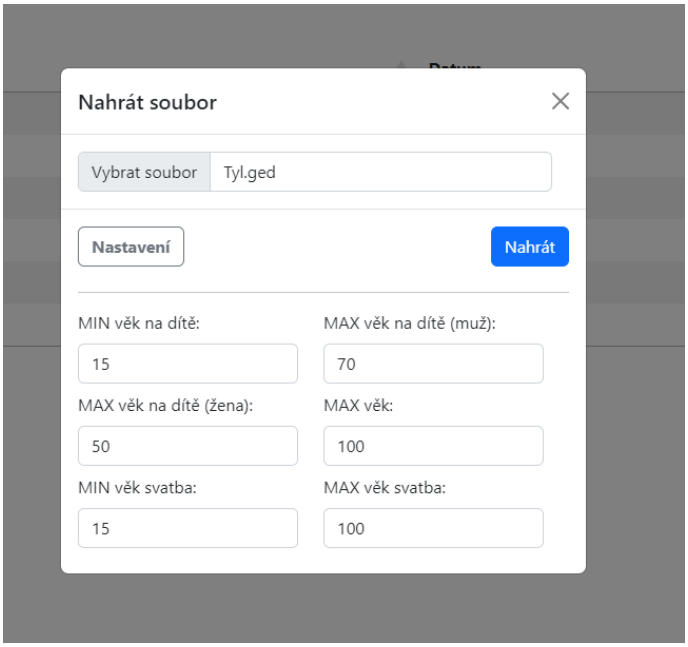
Prvním pohledem, který uživatel uvidí po přihlášení je hlavní stránka (viz obrázek 5.2). Na této stránce uživatel vidí všechny své doposud nahrané soubory prostřednictvím datové tabulky, kterou poskytuje knihovna jQuery. V rámci této tabulky může uživatel vyhledávat soubory na základě jména či je řadit podle data nahrání. Další možností nabízenou uživateli

v rámci tabulky je mazání těchto souborů za pomocí malé červené ikonky. K následkům této operace je uživatel upozorněn vyskakovacím oknem.



Obrázek 5.2: Hlavní stránka aplikace

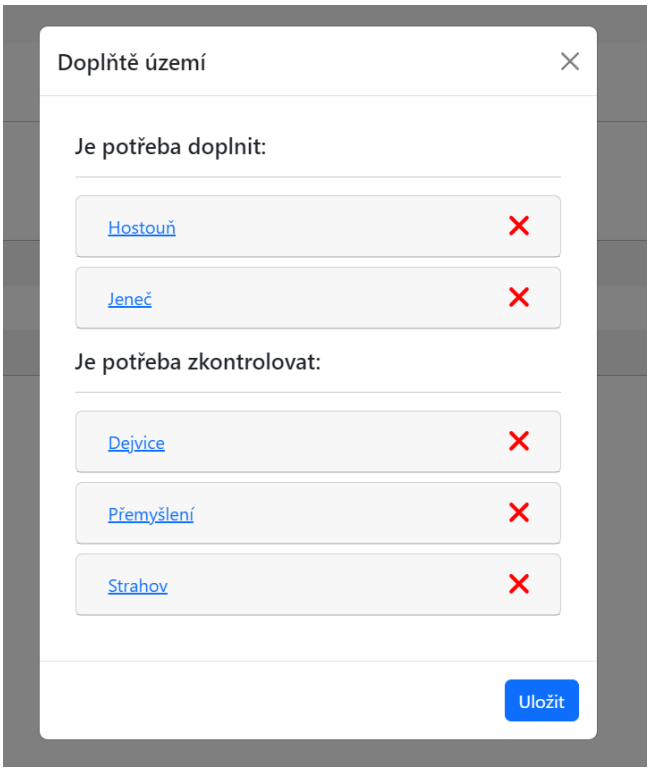
Další akcí, kterou uživatel může provést na hlavní stránce, je přidání souboru. Učiní tak pomocí zeleného tlačítka "Přidat soubor" vyskytujícího se v úrovni nadpisu stránky. Po stisknutí tlačítka se zobrazí vyskakovací okno, v rámci kterého může uživatel vybrat soubor, který chce nahrát (viz obrázek 5.3). Prostřednictvím toho okna si může pro daný soubor uživatel také nastavit parametry výpočtu časových rozsahů (viz časové rozsahy v kapitole 5.3).



Obrázek 5.3: Modalové okno pro nahrání souboru

Po nahrání souboru musí uživatel vyčkat až parser dokončí svou práci. Na aktivitu jiné komponenty je uživatel upozorněn pomocí načítacího kolečka. Během celého procesu nahrá-

vání i zpracování souboru se uživatel pohybuje v dříve zmíněném vyskakovacím okně. Po dokončení parseru je uživatel vybídnut ke zkontrolování přiřazených území a k případnému doplnění území nepřiřazených (viz obrázek 5.4).



Obrázek 5.4: Modalové okno pro kontrolu a doplnění území

Pro každé území je za pomocí bootstrapových komponent vytvořena rozbalovací kartička, ve které se nachází pole pro zadání území. Uživatel má buďto možnost si vybrat z navržených území, nebo se pokusit území najít sám pomocí textového pole. Textové pole využívá komponenty jQueryUI autocomplete[7], která uživateli na základě podobnosti k psanému textu navrhuje území z databáze. Jestliže uživatel, žádné území nevybere, nebo nějakou operací zadané území znevalidní, pak je pole označeno oranžovou barvou. Pro případy, že by uživatel chtěl území nastavit pro jednotlivé osoby/rodiny pak tak může provést za pomocí ozubeného kolečka kdy se mu zjeví stejné dříve zmíněné vstupní pole, které ovšem upravuje pouze individuální osobu/rodinu.

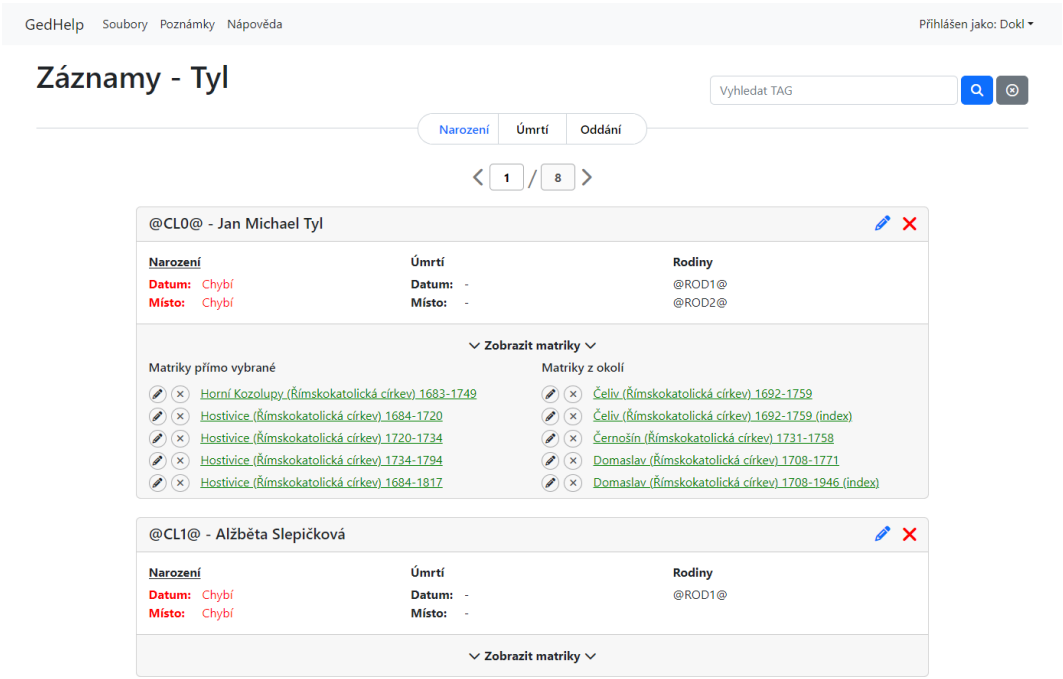
Po dokončení specifikace území je pak spuštěn matcher, který naváže matriky na záznamy. Během této operace uživatel opět vyčkává v prostředí vyskakovacího okna. Po dokončení je pak přesměrován na pohled daného souboru.

Stránka souboru

Tato stránka zobrazuje uživateli jednotlivé záznamy vytvořené v rámci rozkliknutého souboru (obrázek 5.5). Záznamy jsou zobrazeny podle jejich typů. Uživatel si typ může změnit pomocí menu dostupného na vrcholu stránky. Zde se také vyskytuje vstupní pole, které slouží k vyhledávání konkrétních záznamů pomocí ID osoby/rodiny. Vyhledávat lze také

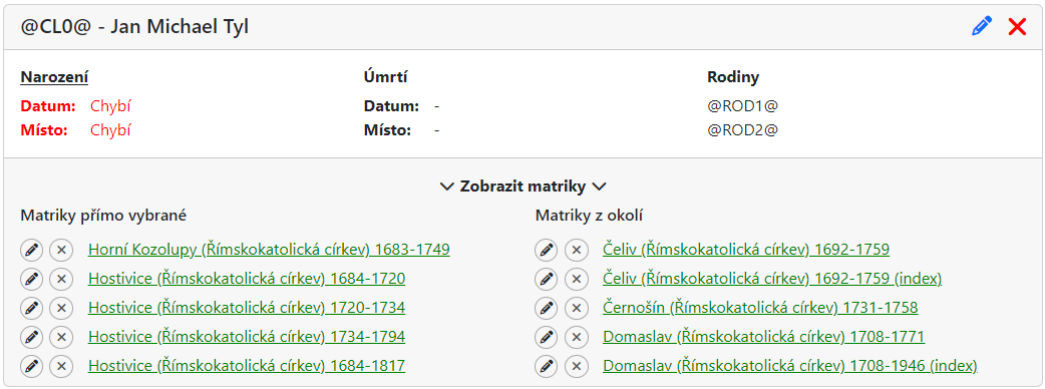
[7] autocomplete - automatické doplnění

pomocí jména, a to i v případě záznamu oddání, kdy se zobrazí záznamy, kde alespoň jeden z svatebčanů má jméno podobné tomu zadanému.



Obrázek 5.5: Pohled zobrazující záznamy vytvořené k souboru Tyl.ged

Pro každý záznam je vytvořena boostrapová kartička, která obsahuje základní informace o osobě/rodině, které se záznam týká (obrázek 5.6). Informace, jenž chybí jsou pak v této kartičce zvýrazněny červeně. V případě, že uživatel chce zahodit vytvořený záznam, může tak učinit pomocí červeného křížku. V rámci této kartičky má uživatel možnost si zobrazit navržené matriky pomocí rozbalovacího tlačítka "Zobrazit matriky".



Obrázek 5.6: Záznam pro událost narození reprezentovaný kartičkou pro osobu Jan Michael Tyl. V obrázku jsou zobrazeny i matriky navržené pro tento záznam.

Matriky jsou zobrazeny ve dvou sloupcích. Levý sloupec představuje matriky vybrané přímo z míst nalezených u dané osoby, rodiny či příbuzných. V pravém sloupci jsou pak ma-

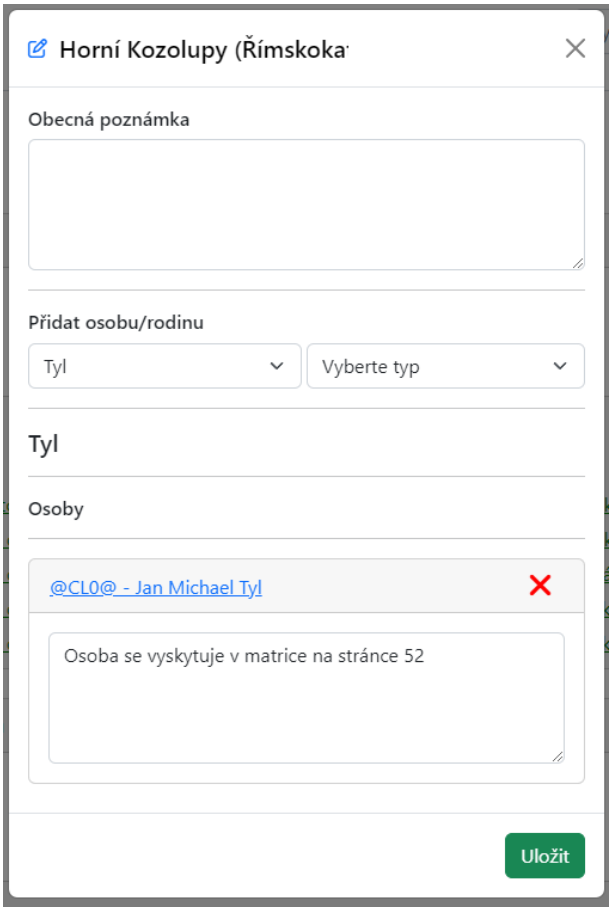
triky vybrané ze sousedních míst v okolí 5 km. Matriky jsou řazeny podle priority relevance (viz sekce lokalit v kapitole 5.3) a barevně odlišeny na základě této priority. Zobrazuje se vždy pouze pět matrik s nejvyšší prioritou. Matriku je možné zahodit za pomocí křížku, a tato zahozená matrika pak bude nahrazena další, za předpokladu, že nějaká další je.

Kartičky se zobrazují na stránce v omezeném množství. Zobrazit si další sadu kartiček uživatel může pomocí šipek vyskytujících se jak na vrchní tak spodní části stránky.

Poznámky

Další důležitou funkcionalitou aplikace je možnost zapisovat si poznámky k osobám, rodinám a matrikám. Tato možnost je uživateli nabízena buďto v záhlaví kartičky záznamu v případě osob/rodin, anebo v případě matrik hned vedle jejich výpisů. Poznámky jsou označeny pomocí ikonky tužky. Po kliknutí na tuto ikonku se vytvoří poznámka a zobrazí se modální okno (viz obrázek 5.7). Prostřednictvím tohoto okna může uživatel upravit a uložit poznámku.

U poznámek pro osobu/rodinu se nabízí možnost přidat matriku a psát poznámku k této matrice. Pokud je pak zobrazena poznámka pro tuto matriku zobrazuje se v ní i osoba/rodina, ve které je tato matrika zmíněna a také jejich sdílený text. Stejně tomu tak je s přidáváním osob/rodin k poznámce matriky.



Obrázek 5.7: Modalové okno zobrazující poznámku k matrice pocházející z Horních Kozolup. Tato poznámka je propojená s poznámkou k osobě Jan Michael Tyl ze souboru Tyl.ged.

Stránka poznámek

Jedná se o poslední pohled, který slouží k zobrazení poznámek na jednom místě. Poznámky jsou zde zobrazeny pomocí tabulek a jsou rozděleny do tří tabulek podle jejich typů (osoba, rodina, matrika). Výběr typu se dá provést pomocí stejného menu použitého u záznamů. V tabulkách se dá poznámky, stejně jako na hlavní stránce u souborů, vyhledávat za pomocí různých parametrů a nebo je řadit. Po kliknutí na řádek tabulky se poznámka rozevře a je možné ji tedy editovat i na této stránce.

5.4.3 Controller

Kontroler je část, se kterou komunikuje uživatel. Předá jí parametry a ona mu vrátí data (např. HTML stránku). Kontroler typicky parametry předá modelům, od kterých získá data. Tato data předá pohledům (šablonám), které data začlení do nějakého HTML kódu. Tento HTML kód pošle kontroler uživateli do prohlížeče. Funguje tedy jako takový prostředník [9].

Pro aplikaci byl ke každému modelu vytvořen jeho příslušný kontroler. Metody těchto kontrolerů jsou volány prostřednictvím routovacího souboru. Pro každou uživatelskou akci existuje metoda nějakého kontroleru, který tuto akci zpracovává, získává data z příslušných modelů a předává je zpátky do pohledu na zobrazení. Metody jsou volány v pohledech přes jejich adekvátní routy, kdy každá routa má své pojmenování. Volány jsou buďto synchronně v případě přechodů mezi stránkami nebo asynchronně pomocí AJAXu, pokud se jedná o nějakou operaci na stránce. Jestliže kontroler přijímá synchronní požadavek, pak vrací načtení celých pohledů. Pokud je požadavek asynchronní, pak vrací pouze potřebné proměnné, či modelem vygenerované elementy v HTML kódu, a to prostřednictvím formátu JSON. V následujících podkapitolách budou popsány způsoby, jakými kontrolery řeší některé důležité operace.

Nahrání souboru

O nahrání, ale i smazání souboru se stará `GFileController`. Operace nahrání souboru se vykonává pomocí asynchronních požadavků a skládá se z několika kroků.

Nejdříve je pomocí metody `store` zkontrolován uživatelem nahraný soubor. Pokud soubor není v pořádku, pak metoda vrací chybovou hlášku, která je uživateli zobrazena. V opačném případě je soubor dočasně nahrán na server a uložen do databáze společně s hodnotami pro výpočty rozsahů, které mohl uživatel upravit.

Dalším krokem je spuštění parseru. O tento krok se stará metoda `executeParser`, která spouští skript a předává mu cestu k nahranému souboru. Metoda pak vyčká na skončení parseru a přijímá od něj výsledek ve formátu JSON. Následně volá metodu modelu `GFile`, která vygeneruje pro získaný výsledek kód v HTML tak, aby jej bylo možné uživateli zobrazit. Výsledkem je v tomto případě modalové okno pro kontrolu a doplnění území.

Posledním krokem, který `GFileController` provádí je spuštění matcheru. Ten se spouští po zkontrolování/doplnění území pomocí metody `executeMatcher`. Metoda vrací zprávu o úspěšném ukončení skriptu. Přesměrování na detail souboru se už pak provádí prostřednictvím jiného kontroleru, tím je `RecordController`.

Při nahrávání souboru se používá ještě `TerritoryController`. Ten obsahuje hlavně metody sloužící k načítání území při vyhledávání v autocomplete poli, a nebo pak metodu `assignTerritories`, která přiřadí doplněné území k příslušným osobám/rodinám.

Zobrazení záznamů

O veškeré operace ohledně záznamů se stará `RecordController`. Synchronní načtení záznamů k nějakému souboru provádí metoda `index`. Tato metoda načte pohled starající se o zobrazení záznamů a předá mu veškeré potřebné parametry. Asynchronní načtení se provádí pomocí metody `get_records`. V obou případech je využíváno metod modelu `Record` pro získání záznamů či generování HTML obsahu. Asynchronní metoda nenačítá celou stránku, ale pouze získává nové kartičky záznamů na zobrazení. Asynchronní načítání záznamů se provádí při paginaci, vyhledávání pomocí tagu/jména, nebo mazání záznamu.

`RecordController` se také stará o procházení skrz navržené matriky. Tedy zahození návrhu pomocí křížku a zobrazení dalšího v pořadí. Tento proces je obstaráván metodou `delete_book_suggestion`, která vymaže návrh matriky pro daný záznam z propojovací tabulky a načte nové knížky. K tomu kontroler opět využívá metod modelu `Record`. Tato metoda vrací nově vygenerovaný HTML kód pro zobrazení navržených knih.

Vytvoření poznámky

U poznámek se ke zpracování požadavků používají celkově 4 kontrolery. `NoteController` je ten hlavní, který se stará o základní operace týkající se poznámek. Tedy zobrazení, vytvoření a aktualizace poznámek. Poznámka k záznamu nebo matrice se vytváří až při kliknutí na ikonu tužky. Nejdříve `NoteController` zkontroluje, zdali poznámka existuje pomocí metody `check_if_exists`. Kontrola na existenci je důležitá, protože ikona tužky neslouží pouze k vytvoření poznámky, ale také k zobrazení již existující.

Jestliže poznámka neexistuje, pak je vytvořena pomocí metody `store`. Tato metoda na základě vstupních parametrů vyhodnotí ke komu má být poznámka napsána (osoba, rodina, matrika), tedy na koho bude v databázi odkazovat cizím klíčem. Jméno poznámky se při vytváření generuje automaticky, a to podle toho, komu patří. Poté je poznámka přidána do databáze a může být zobrazena.

O zobrazení poznámky se stará metoda `fetch`. Kromě získání patřičného záznamu z databáze získává tato metoda také všechny přidružené objekty k této poznámce. Tedy osoby, rodiny nebo knížky, které si uživatel mohl k poznámce přidat. U těchto objektů je zapotřebí dynamicky generovat HTML kód a k tomu se využívá funkcí modelu `Note`. Tato funkce tedy vrací jak proměnné, které budou vsazeny do statického HTML kódu, tak dynamicky vygenerovaný HTML kód.

Poslední operací, o kterou se `NoteController` stará je aktualizace poznámky. Metoda `update` aktualizuje jméno a textový obsah poznámky, ale také textový obsah všech přidaným objektů. Tato operace je vyvolána po použití tlačítka "Uložit".

Jak bylo zmíněno výše, k poznámkám je možné přidávat objekty, ke kterým si uživatel může připisovat další text. Jestliže je poznámka vedena pro matriku, pak uživatel může k poznámce přidat osoby a rodiny. V opačném případě je možné přidat k poznámce matriku. Pro jednotlivý typ objektu, který je možné přidat, jsou implementované metody kontrolerů `PersonController`, `FamilyController`, `ParishBookController`. Pro zpřístupnění hledané osoby/rodiny/matriky existuje metoda `get_person/family/parish_book`[8], která vrací vyhovující záznamy na zobrazení. Pro přidání je pak metoda `add_object`, jež připojí objekt k poznámce skrze propojovací tabulku. Poslední metodou je `delete_object`, která odpojí objekt od poznámky vymazáním záznamu z propojovací tabulky. Dvě poslední uve-

[8] Dále referencované jako "název_object"

dené metody vrací nově vygenerovaný HTML kód obsahující nové rozpoložení připojených objektů. Tento kód opět generují metody modelu `Note`.

Uživatelské operace

O veškeré uživatelské operace jako jsou registrace, přihlášení, editace profilu a odhlášení se stará `UserController`. Tento kontroler pracuje hlavně s komponenty Laravelu pro autentizaci, třída `Auth`, a udržení sezení, metoda `session` patřící pod třídu `Request`.

Kapitola 6

Testování a optimalizace

Testování probíhalo průběžně. Vždy když byla implementována nějaká část aplikace byla odzkoušena její funkcionalita a provedena případná oprava či optimalizace. K testování bylo použito celkem sedm různých GEDCOM souborů. Každý měl různé velikosti a počty záznamů a bylo tedy možné otestovat chování aplikace na menších i větších souborech. Skoro všechny soubory měli také jiného autora, a tak tedy i způsoby zápisů se lišili. Díky tomu se mnohokrát měnil způsob řešení určitých problému, tak aby byla aplikace funkční pro co nejvíce možných způsobů zápisu.

Python skripty

U Python skriptů se hlavně testovalo správné zpracování dat a zápis do databáze. U parseru se nejvíce testovalo zpracování územních řetězců, kde se kvůli povolným pravidlům pro zápis lokace mohlo vyskytnout mnoho problémů. U matcheru se pak testovala správnost výběru matrik.

Kromě funkcionality se u skriptů nahlíželo na rychlost zpracování. Čas průběhu skriptu se měřil pomocí Python knihovny datetime. Na tyto časy se při implementaci vedl veliký důraz jelikož bylo potřeba, aby se soubory zpracovávali v rozumném čase. Zde také přišla na řadu optimalizace, která se týkala hlavně co největšího zmenšení přístupů do databáze. Každý takový přístup je totiž časově náročná operace. Přístupy byly omezeny využitím slovníků a vkládáním několika záznamů naráz. Čas, který si skript vyžadoval na zpracování souboru, byl tak několikanásobně krát zmenšen.

Webová aplikace

U samotné aplikace se pak testovalo hlavně správné zobrazení dat a reakce na provedené operace. Jestliže nastala nějaká chyba, pak byl její původ zobrazen na výstupu. Tato možnost je nastavena v konfiguračním souboru aplikace v sekci `APP_DEBUG`. O zobrazení chyby se v tomto případě stará Laravel třída `Handler`. Na výstupu se zobrazuje sled chyb vedoucí k funkci, kde nastal problém a obsah hlaviček požadavku, který problém vyvolal. Debugovat chybu pak bylo možné za pomocí funkcí `dd` a `dump`, které poskytuje Laravel v rámci pomocných funkcí. Tyto funkce zobrazí na výstup hodnoty proměnných, které jsou jim předány a dá se pomocí nich tak zjistit kde chyba nastala.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit aplikaci, která umožní uživateli nahrát soubor ve formátu GEDCOM, zobrazí mu vytvořené záznamy k chybějícím informacím událostí narození, úmrtí i oddání a navrhne matriky kde by tyto informace mohl nalézt.

V rámci jejího vypracování bylo zapotřebí nastudovat problematiku provádění genealogického výzkumu, strukturu matrik a formátu GEDCOM. Bylo potřeba analyzovat požadavky uživatelů, navrhnout vzhled aplikace, strukturu databáze a zvolit vhodné technologie k vypracování aplikace. Pomocí těchto technologií pak implementovat její jednotlivé části.

Součástí této práce nebyla jen implementace aplikace, ale také příprava a zpracování všech potřebných dat. Tedy pochopení a zpracování územního rozsahu České republiky, a dále extrakce dat ze všech osmi archivů, uchovávajících matriční knihy, do formátu JSON.

Výsledkem práce je tedy aplikace, která přijme soubor ve formátu GEDCOM, zpracuje ho, uloží si veškeré potřebné informace do databáze a zobrazí vytvořené záznamy s navrženými matrikami, které jsou řazené podle priorit. Díky zpracování archivů je možné se přes tyto navržené matriky prokliknout na jejich detail a rovnou v nich vyhledávat. Aplikace disponuje i možností jednoduchého vyhledávání skrz záznamy na základě tagu nebo jména osob/rodin. Aplikace uživateli také nabízí poměrně sofistikovaný poznámkový systém, ve kterém si uživatel může vést poznámky jak k matrikám, tak jednotlivým osobám či rodinám a propojit tyto poznámky mezi sebou.

V budoucnu by se aplikace mohla rozšířit o několik nových funkcionalit. Jednou z nich by byla možnost uživatele dopisovat nalezené informace přímo k záznamům a připisovat je tak do databáze. Při tom by se na základě nových informací mohl provést přepočet a mohli by tak být navrženy k určitým záznamům nové matriky. Dalším rozšířením, souvisejícím s předchozím, by mohlo být, že by aplikace mohla vyrobit nový GEDCOM soubor, který by obsahoval nově vyplněné informace a uživatel by si jej mohl stáhnout pro příští využití. Aplikace by se také dala rozšířit o aliasy území, tedy názvy týkající se nějakých území v cizím jazyce. Existuje mnoho dalších způsobů, kterými by se aplikace mohla rozšířit.

Vypracování této práce mi přineslo mnoho nových zkušeností. Především část přípravy dat, kdy jsem se musel naučit pracovat s daty poskytovanými státem. Při zpracování archivů jsem se naučil, jak extrahovat data z webových stránek pomocí jazyka Python, což pro mě do této doby bylo neprobádané území. Zlepšila se mi schopnost pracovat se všemi použitými jazyky a také schopnost sám rozebrat a řešit problémy, které nastávali při implementaci. Přínosem pro mě také bylo rozšíření obzorů v rámci genealogie.

Literatura

[1] Chris, K. *What is HTML – Definition and Meaning of Hypertext Markup Language* [online]. 2021 [cit. 2023-01-13]. Dostupné z: https://www.freecodecamp.org/news/what-is-html-definition-and-meaning/.

[2] ČGHSP. *Digitalizace* [online]. ČGHSP, 2015 [cit. 2023-01-13]. Dostupné z: http://www.genealogie.cz/aktivity/digitalizace/.

[3] ČGHSP. *Matriky na internetu* [online]. 2015 [cit. 2023-01-13]. Dostupné z: https://rodokmeny.cz/genealogie-a-rodokmeny/matriky-na-internetu/.

[4] ECMA 404. *Úvod do JSON* [online]. 2017 [cit. 2023-01-13]. Dostupné z: https://www.json.org/json-cz.html.

[5] Jones, T. *The FamilySearch GEDCOM 5.5.1 Specification Annotated Edition* [online]. 2019 [cit. 2023-01-13]. Dostupné z: https://www.gedcom.org/gedcom.html.

[6] Kettle, S. *Distance on a sphere: The Haversine Formula* [online]. 2017 [cit. 2023-04-27]. Dostupné z: https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128.

[7] Konetzný, J. *Podpora vyhledávání matričních událostí.* Brno, CZ, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/25278/.cs?year=0&stud=Konetzn%C3%BD.

[8] Koďousková, B. *JAVASCRIPT PRO ZAČÁTEČNÍKY: CO TO JE A JAK FUNGUJE* [online]. 2022 [cit. 2023-01-13]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky.

[9] Lupčík, J. *Lekce 1 - Úvod do Laravel frameworku pro PHP* [online]. ITnetwork, 2019 [cit. 2023-04-27]. Dostupné z: https://www.itnetwork.cz/php/laravel/uvod-do-laravel-frameworku-pro-phpl.

[10] Lupčík, J. *Lekce 3 - První aplikace v Laravel* [online]. ITnetwork, 2019 [cit. 2023-04-27]. Dostupné z: https://www.itnetwork.cz/php/laravel/prvni-aplikace-v-laravel.

[11] Ministerstvo vnitra České republiky. *Popis záznamu matriky a příklady vyplnění jednotlivých údajů* [online]. Ministerstvo vnitra České republiky, 2023 [cit. 2023-04-27]. Dostupné z: https://www.mvcr.cz/clanek/metodika-na-useku-matrik.aspx.

[12] Moravský zemský archiv v Brně. *ACTA PUBLICA: Matriky uložené v Moravském zemském archivu v Brně* [online]. 2020 [cit. 2023-01-13]. Dostupné z: https://www.mza.cz/actapublica/matrika/detail/1568.

[13] Oracle Corporation. *What is MySQL?* [online]. 8.0. Oracle Corporation, 2023 [cit. 2023-01-13]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.

[14] Pecháček, J. *Genealogické diagramy aneb co je to vlastně rodokmen? Část II.* [online]. 2015 [cit. 2023-01-13]. Dostupné z: https://www.odkudjsme.cz/blog/genealogicke-diagramy-aneb-co-je-to-vlastne-rodokmen/.

[15] Podhola, J. *Matriky* [online]. OS Genea, 2012 [cit. 2023-01-13]. Dostupné z: http://www.genea.cz/informace/badani-v-archivu/matriky/.

[16] Python Software Foundation. *What is Python? Executive Summary* [online]. Python Software Foundation, 2001, 2023 [cit. 2023-01-13]. Dostupné z: https://www.python.org/doc/essays/blurb/.

[17] Příborský, V. *Genealogický výzkum v matrikách* [online]. Václav Příborský, 2015 [cit. 2023-01-13]. Dostupné z: https://rodokmeny-priborsky.cz/genealogicky-vyzkum-v-matrikach/.

[18] Rathi, A. *Program for distance between two points on earth* [online]. GeeksforGeeks, 2023 [cit. 2023-04-27]. Dostupné z: https://www.geeksforgeeks.org/program-distance-two-points-earth/.

[19] The PHP Group. *What is PHP?* [online]. Dokumentace. The PHP Group, 2001, 2023 [cit. 2023-01-13]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php.

[20] WikiSkripta. *Genealogie* [online]. 2016 [cit. 2023-01-13]. Dostupné z: https://www.wikiskripta.eu/index.php?title=Genealogie&oldid=340229.

Příloha A

Obsah přiloženého paměťového média

- *src/*
  - Adresářová struktura Laravel aplikace.
- *src/python_scripts/*
  - Python skripty pro parser a matcher.
- *territories/*
  - Skripty pro zpracování území.
- *scrapers/*
  - Python skripty pro extrakci dat z online archivů.
- *scrapers/Jsons*
  - Zpracované JSON soubory jednotlivých archivů.
- *doc/*
  - Text technické zprávy.

Příloha B

Lokální instalace

Tato příloha obsahuje návod pro zprovoznění aplikace na lokálním systému. Způsob nasazení aplikace na server je tomu lokálnímu velice podobný, jen je potřeba nastavit pár věcí navíc.

Prerekvizity

- **Python** v3.10 + pip
- **PHP** v8.1
- **Composer**
- **MySQL**
- **Bash**

Nastavení konfiguračního souboru

Je zapotřebí nastavit konfigurační soubor Laravel aplikace s názvem .env. Na Linuxových OS může být tento soubor skrytý před zobrazením. V rámci tohoto souboru je zapotřebí nastavit v sekci týkající se databáze parametry `DB_USERNAME` a `DB_PASSWORD`. Tedy uživatelské jméno a heslo k účtu na MySQL serveru.

K nachystání potřebných částí slouží shell skript `setup.sh`. V tomto skriptu je potřeba vyplnit databázové uživatelské jméno a heslo. Skript se poté spouští v kořenovém adresáři příkazem:

- `./setup.sh`

V následujících sekcích je vysvětlen postup tohoto skriptu.

Instalace modulů pro jazyk Python

Skript nejdříve provede nainstalování všech potřebných modulů pro jazyk Python za pomocí pip instalátoru.

- `pip install "nazev\_modulu"`

Databáze

Dále `setup.sh` spouští příkaz:

- `mysql -uUSERNAME -pPASSWORD < SQL.sql`

Tento příkaz spustí na databázovém serveru queries ze souboru SQL.sql. Tyto queries vytvoří databázové schéma s názvem `xpopdo00`. Do tohoto schématu vytvoří tabulky a vloží do tabulek počáteční data.

Instalace závislostí

Zaváděcí skript dál přejde do složky `src`, kde spouští příkaz:

- `composer install`

který nainstaluje závislosti nutné ke správnému fungování frameworku Laravel.

Spuštění aplikace

Jako poslední skript provede celkem 3 příkazy:

- `php artisan route:clear`
- `php artisan route:cache`
- `php artisan serve`

První dva příkazy vyčistí a vytvoří nové routy. Poslední příkaz spustí aplikaci, která poběží na lokální instanci na adrese:

- `http://127.0.0.1:8000`