VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PODPORA VYHLEDÁVÁNÍ MATRIČNÍCH UDÁLOSTÍ

SUPPORT FOR SEARCHING IN PARISH BOOKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAKUB KONETZNÝ**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Konetzný Jakub**
Program: Informační technologie
Název: **Podpora vyhledávání matričních událostí**
Support for Searching in Parish Books
Kategorie: Web

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.

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. listopadu 2021
Datum odevzdání: 29. července 2022
Datum schválení: 23. června 2022

Abstrakt

Cílem této práce je navrhnout a implementovat webovou aplikaci na podporu vyhledávání matričních událostí. Vytvořená aplikace umožňuje uživateli nahrát svůj GEDCOM soubor a z chybějících událostí v tomto souboru automaticky vytvoří záznamy, ke kterým přiřadí matriky, kde by se daná událost mohla nacházet. Vytvořené řešení umožní uživateli vést si k jednotlivým záznamům a matrikám poznámky.

Abstract

The aim of this work is to design and implement a web application for support for searching in parish book. The created application allows the user to upload his GEDCOM file from the missing events in this file to automatically create records to which he assigns parish book where the event could be located. The created solution allows you to write notes for individual records and parish book.

Klíčová slova

genealogie, GEDCOM, PHP, Nette Framework, MySQL, webová aplikace

Keywords

genealogy, GEDCOM, PHP, Nette Framework, MySQL, web application

Citace

KONETZNÝ, Jakub. *Podpora vyhledávání matričních událostí*. Brno, 2022. 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.

.....
Jakub Konetzný
27. července 2022

Poděkování

Rád bych tímto poděkoval vedoucímu práce Ing. Jaroslavu Rozmanovi, Ph.D za vedení práce, konzultace a věcné připomínky. Rovněž bych rád poděkoval Ing. Petru Veigandovi za poskytnutí dat o matrikách.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Genealogie je v dnešní době velice populární a hodně lidí se zajímá o původ svých předků. Proto bádají v matričních knihách, aby se o nich něco dozvěděli. Což je dnes díky digitalizaci v archívech mnohem jednodušší než kdy dřív.

Cílem této práce je navrhnout a implementovat webovou aplikaci, která by uživatelům pomohla nalézt chybějící události v jejich rodokmenech.

V úvodní části této práce je stručně popsána genealogie, rodokmeny a pojmy použité v práci. Dále je zde popsána specifikace souboru GEDCOM, se kterým výsledná práce pracuje.

V další části práce jsou popsány použité technologie a principy. U některých z nich je ukázán i příklad použití, schéma či odůvodněna jejich volba.

V následující části je zobrazeno schéma databáze a jsou zde popsány jednotlivé tabulky a relace.

V poslední kapitole je popsána praktická implementace práce. Jsou zde popsány části výsledné aplikace, jak se aplikace chová, ovládá a jak jsou matriky k událostem přiřazovány. Taktéž jsou zde popsána data, se kterýma aplikace pracuje.

Kapitola 2

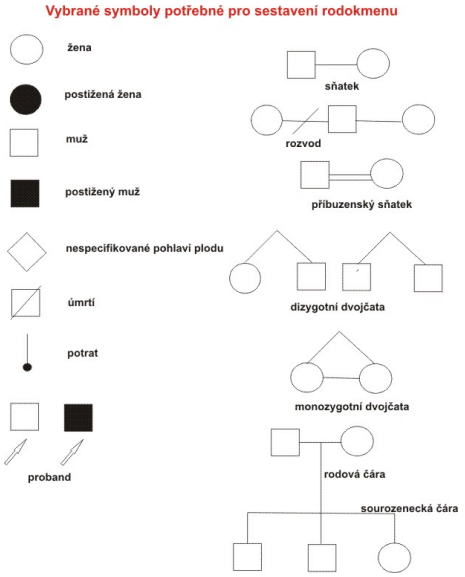
Genealogie

Je pomocná věda historická. Zabývá se studiem rodinných původů a historie. Slovo genealogie pochází ze dvou řeckých slov - genea znamená „rasa“ nebo „rodina“ a logos „teorie“ nebo „věda“. Genealogové sestavují seznamy předků, které uspořádávají do rodokmenových tabulek nebo jiných písemných forem.

2.1 Rodokmen

Rodokmeny jsou základem genealogického výzkumu. Jsou výchozím bodem pro pochopení rodinného příběhu. Můžete je použít k ukládání, organizaci a sdílení toho, co najdete, a vytvořit tak podrobný obrázek o své rodinné historii.

Rodokmen je typ grafu nebo diagramu představující generace rodin a způsob jejich propojení v průběhu let. Rodokmen může obsahovat jména, data narození, data manželství a obrázky. Rodokmeny mohou být jednoduché a mohou zahrnovat pouze vaše blízké členy rodiny, nebo se mohou vrátit o mnoho generací zpět, aby vám umožnily zjistit, odkud jste přišli, kdo byli vaši předkové a jaký s nimi máte vztah.

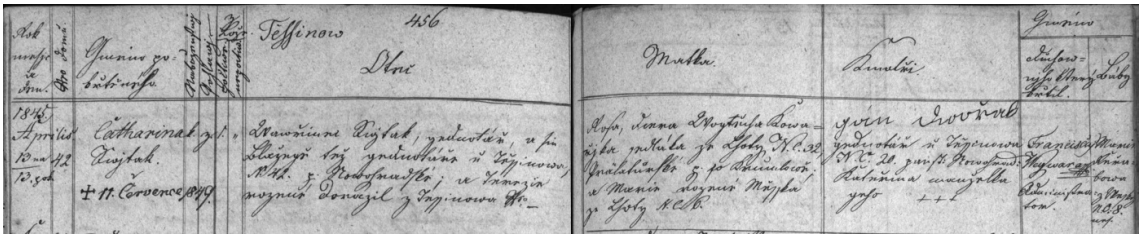


Obrázek 2.1: Symboly používané v rodokmenech[1]

2.2 Matrika

Je veřejný úřední seznam, který uchovává záznamy o narození, sňatcích a úmrtí. Jedná se o písemný genealogický pramen určený k evidenci občanů. Živé matriky se nacházejí na matričních úřadech. Mrtvé matriky se poté nacházejí v jednom městském (Archiv hlavního města Prahy), pěti státních oblastních (v Praze, Plzni, Třeboni, Hradci Králové a Litoměřicích), dvou zemských (v Brně a Opavě) archívech.

Nejstarší dochovanou matrikou na českém území je německy psaná matrika oddaných v Jáchymově z roku 1531. Tato matrika nebyla psána strukturovaně, ale pouze ve větách. Strukturovaně se matriky začaly psát až po reformách Josefa II. z let 1781 a 1784. Před touto reformou byl formát matrik značně roztříštěn. [14]



Obrázek 2.2: Ukázka záznamu v matrice - Římskokatolický farní úřad Jílovice, matrika narozených 1804-1865, strana 456 (SOA Třeboň)[3]

2.2.1 Živé a mrtvé matriky

Matriky se dělí na živé a mrtvé. Matrika je živá, pokud ještě neuplynulo 100 let od posledního narození v knize narození, anebo ještě neuplynulo 75 let od sňatku či úmrtí. Do těchto matrik se smí nahlížet pouze za přítomnosti matrikáře a je-li splněna alespoň jedna z těchto podmínek:

- fyzické osobě, které se zápis týká, nebo členům její rodiny, jejím sourozencům a dále zmocněncům těchto osob
- pro úřední potřebu státních orgánů, nebo výkon přenesené působnosti orgánů územních samosprávných celků
- statutárním orgánům církví, nebo duchovním jimi zmocněným, jde-li o matriční knihy vedené těmito církvemi do 31. prosince 1949
- fyzické osobě, která prokáže, že je to nezbytné pro uplatnění jejich práv před orgány státu nebo před orgány územních samosprávných celků

[7] Ostatní matriky jsou matriky mrtvé a nahlížet do nich může každý.

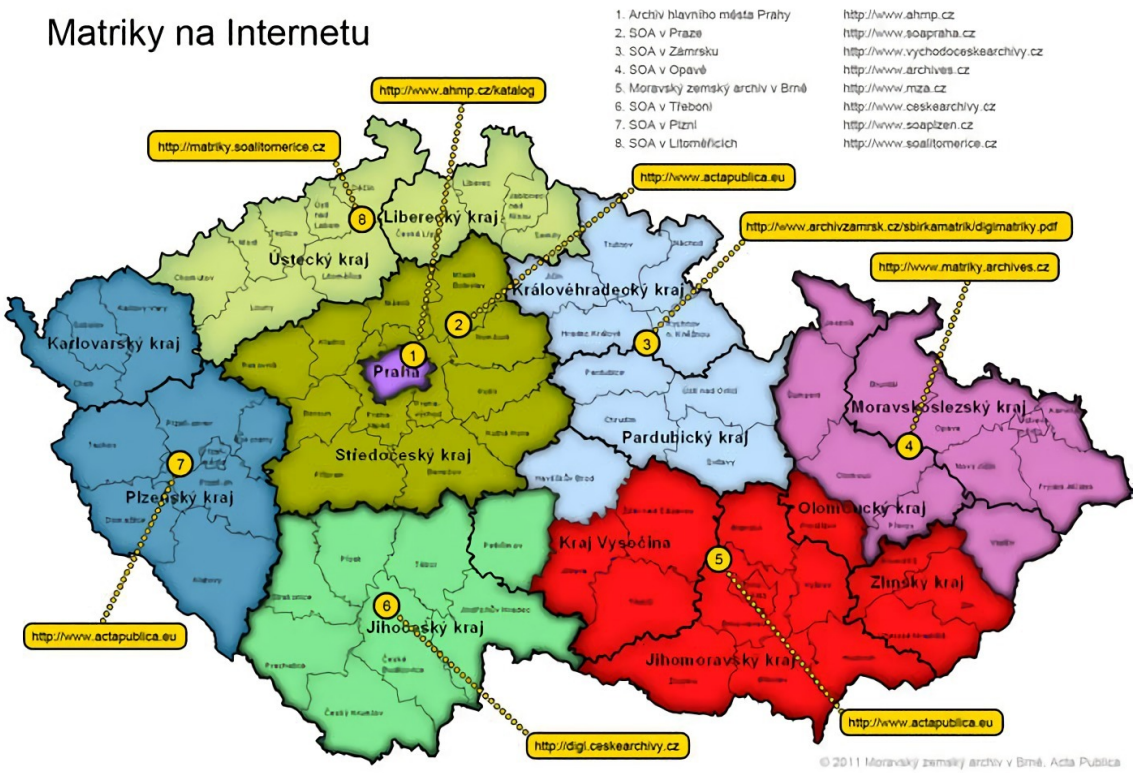
[1] https://www.wikiskripta.eu/w/Genealogie
[3] http://www.genealogie.cz/aktivity/digitalizace/

Druhy matričních knih

- N - kniha narození
- O - kniha sňatků
- Z - kniha úmrtí
- I-N - index knihy narození
- I-O - index knihy sňatků
- I-Z - index knihy úmrtí

Digitalizace matrik

Digitalizace matrik byla zahájena Státním oblastním archívem v Třeboni roku 2007. Ostatní archívy se přidávaly později. Prvních reálných výsledků bylo dosaženo již v roce 2008. Digitalizací je myšleno zpřístupnění archiválií online. Díky tomuto je dnes možné nalézt skeny archivních materiálů online. Skeny jsou na internetu rozděleny podle archívu, ve kterém se nachází, jak je vidět na obrázku 2.3. [15]

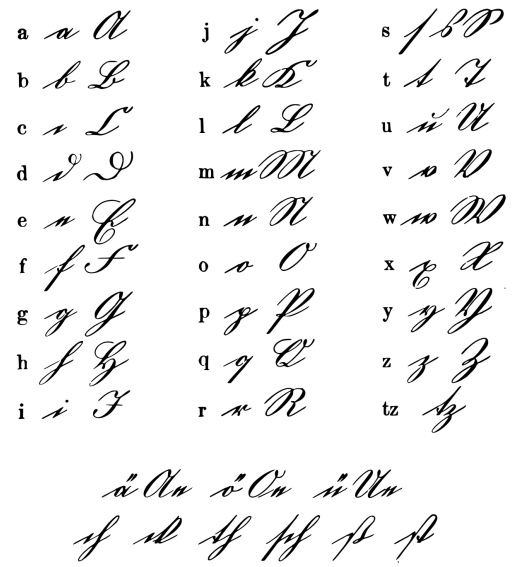


Obrázek 2.3: Mapa s odkazy na digitální archivy[4]

[4] http://rodokmen.nase-koreny.cz/matriky/obsah_matrik.htm

2.2.2 Kurent

Kurent je stará forma německého rukopisu založená na pozdně středověkém kurzivním psaní. Kurent je také znám jako Kurrentschrift.



Obrázek 2.4: Kurent[5]

2.3 GEDCOM

GEDCOM (GEnealogical Data COMmunication) je specifikace pro výměnu genealogických dat mezi různými genealogickými programy.

Soubor GEDCOM je nejrozšířenější standard formátu souborů pro vytváření rodokmenu. Většina genealogických programů tento formát rozpozná, což umožňuje snadno přesunout strom do genealogického programu podle vašeho výběru.

Specifikace GEDCOM je pouze dohodnutý formát pro vytváření databáze, což je v podstatě rodokmen. Bude například obsahovat pole pro křestní jméno a další pole pro příjmení, které budou rozpoznávat všechny programy pro tvorbu rodokmenů, které podporují soubory GEDCOM. Rozpozná také rodinné struktury a seskupí osoby a zdroje dohromady.

Aktuální verze je 5.5.5 vydaná 15. listopadu 2019. GEDCOM 5.5.5 nepřináší velké novinky, ale řeší problémy se samotným standardem. Celkové technické rozdíl mezi GEDCOM 5.5.5 a GEDCOM 5.5.1 spočívá v tom, že GEDCOM 5.5.5 je podstatně jednodušší a přísnější než GEDCOM 5.5.1. Tam, kde má GEDCOM 5.5.1 více způsobů, jak věci dělat, má GEDCOM 5.5.5 jeden způsob. V roce 2001 byl vydán koncept 6.0 XML Draft, který nebyl oficiálně uznán kvůli nekompletní specifikaci.

Soubor GEDCOM se skládá ze záznamů. Záznam je skupina textových řádků, z nichž první začíná nulou „0“. Záznam definuje něco konkrétního, což závisí na typu záznamu. Každý záznam je prezentován ve stromové struktuře. Každá značka může obsahovat libovolný počet dílčích značek. Dílčí značky jsou hierarchicky závislé na předchozí značce vyšší

[5] https://cs.wikipedia.org/wiki/Kurent

[5] https://cs.wikipedia.org/wiki/Kurent

úrovně a mohou zase obsahovat jednu nebo více dílčích značek atd. První a poslední záznam souboru GEDCOM jsou konkrétních typů. První záznam se nazývá záhlaví (značka HEAD) a definuje některé obecné informace o souboru. Poslední záznam se nazývá koncový záznam (značka TRLR). Tento TRLR záznam definuje konec souboru. Každý z ostatních záznamů definuje genealogickou entitu s vlastní sadou značek. Soubor GEDCOM používá 7 kategorií entit. Záznamy, které lze najít v souboru GEDCOM, jsou tedy následující:

- INDI - Záznamy definující jednotlivce
- FAM - Záznamy definující rodiny
- SOUR - Záznamy definující zdroje
- NOTE - Záznamy definující poznámky
- REPO - Záznamy definující úložiště
- SUBM - Záznamy definující zadavatele informací
- OBJE - Záznamy definující multimediální soubory

[3]

```
0 HEAD
1 CHAR ASCII
1 SOUR ID_OF_CREATING_FILE
1 GEDC
2 VERS 5.5
2 FORM Lineage-Linked
1 SUBM @SUBMITTER@
0 @SUBMITTER@ SUBM
1 NAME /Submitter/
1 ADDR Submitters address
2 CONT address continued here
0 @FATHER@ INDI
1 NAME /Father/
1 SEX M
1 BIRT
2 PLAC birth place
2 DATE 1 JAN 1899
1 DEAT
2 PLAC death place
2 DATE 31 DEC 1990
0 TRLR
```

Výpis 2.1: Příklad jednoduchého souboru GEDCOM
Zdroj: http://heiner-eichmann.de/gedcom/simple.ged

Kapitola 3

Použité technologie

V této kapitole jsou stručně popsány technologie a principy, které byly použity.

3.1 HTML

HTML[1] neboli Hypertext Markup Language je značkovací jazyk pro web, který definuje strukturu webových stránek. Je to jeden z nejzákladnějších stavebních kamenů každého webu. Navrhl jej britský vědec Sir Tim Berners-Lee v laboratoři jaderné fyziky CERN ve Švýcarsku v 80. letech 20. století. Značky HTML určují prvky dokumentu, jako jsou nadpisy, odstavce a tabulky. Označují dokument pro zobrazení pomocí počítačového programu známého jako webový prohlížeč. Prohlížeč interpretuje značky a zobrazuje nadpisy, odstavce a tabulky v rozložení, které je přizpůsobeno velikosti obrazovky a dostupným fontům.[11]

3.2 CSS

CSS (Cascading Style Sheets)[2] je jazyk pro popis prezentace webových stránek včetně barev, rozložení a písem. Umožňuje přizpůsobit prezentaci různým typům zařízení, jako jsou velké obrazovky, malé obrazovky nebo tiskárny. CSS je nezávislé na HTML a lze jej použít s jakýmkoliv značkovacím jazykem založeným na XML. Oddělení HTML od CSS usnadňuje údržbu webů, sdílení šablon stylů napříč stránkami a přizpůsobení stránek různým prostředím. Toto je označováno jako oddělení struktury od prezentace.[11]

3.3 Bootstrap

Bootstrap[3]je bezplatná a open-source sada nástrojů pro vytváření responzivních webů a webových aplikací. Jedná se o nejpopulárnější HTML, CSS a JavaScript framework pro vývoj responzivních webových stránek zaměřených na mobilní zařízení. Bootstrap je obrovská sbírka praktických, opakovaně použitelných kousků kódu napsaných v jazycích HTML, CSS a JavaScript. Je to také frontendový vývojový framework, který umožňuje vývojářům a designérům rychle vytvářet plně responzivní webové stránky.

[1] HTML: https://www.w3.org/html/
[2] CSS: https://www.w3.org/Style/CSS/
[3] Bootstrap: https://getbootstrap.com/

3.4 PHP

PHP[4] je jednoduchý, přesto velice výkonný skriptovací programovací jazyk na straně serveru používaný k vytváření statických a dynamických webových stránek, nebo webových aplikací. PHP byl původně zkratkou Personal Home Page, ale nyní je to rekurzivní zkratka pro Hypertext Preprocessor (česky Hypertextový preprocesor). Jedná se o nejrozšířenější programovací jazyk pro weby na straně serveru, využívá jej přibližně 78% [13] webů. Je vhodný pro webové aplikace a databázové aplikace. PHP lze využít spolu s HTML a to tak, že PHP vložíme přímo do HTML.[11]

PHP se používá k těmto účelům:

Skriptování na straně serveru

PHP bylo původně navrženo pro vytváření dynamického webového obsahu a stále se pro tento úkol hodí. Ke generování HTML je potřeba PHP parser a web server k přijímání požadavků a odeslání dokumentů.

Skriptování z příkazového řádku

PHP umí spouštět skripty z příkazového řádku podobně jako Perl, awk nebo Unix shell.

GUI aplikace na straně klienta

Pomocí PHP-GTK[5] je možné psát plnohodnotné multiplatformní GUI aplikace v PHP.

3.5 Composer

Composer[6] lze definovat jako správce závislostí nebo nástroj pro správu závislostí vytvořený speciálně pro PHP. Pomocí nástroje Composer je možné snadno stáhnout a začlenit užitečné a nezbytné balíčky (knihovny vytvořené jinými vývojáři) do svého projektu prostřednictvím terminálu. Jedná se tedy o správce závislostí, který spravuje balíčky pro každý projekt zvlášť. Pro vytvoření konfiguračního souboru Composeru se používá soubor JSON v kořenovém adresáři projektu.

3.6 Nette

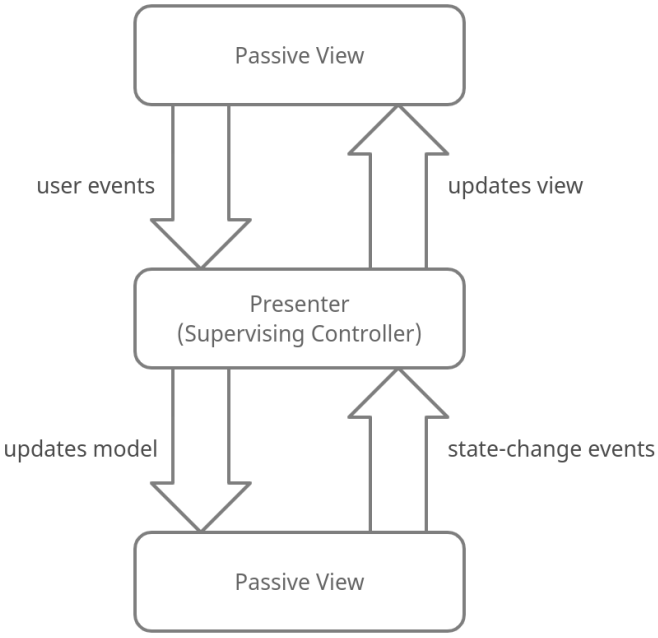
Nette[7] je open source Framework[8] pro tvorbu webových aplikací v PHP. Byl vyvinut Davidem Grudlem, ale nyní se o vývoj stará organizace Nette Foundation. Pro vývoj webových aplikací v nette se používá návrhový vzor MVP(Model–view–presenter).

3.6.1 Model–view–presenter

Je návrhový vzor, který rozděluje aplikaci do tří hlavních logických komponent: model, view a controller. Každá z těchto komponent je vytvořena pro zpracování specifických aspektů

[4] PHP: https://www.php.net/
[5] php-gtk: https://gtk.php.net/
[6] Composer: https://getcomposer.org/
[7] Nette: https://nette.org/
[8] Framework: platforma která poskytuje základ pro vývoj softwarových aplikací

vývoje aplikace. MVP je jedním z nejčastěji používaných návrhových vzorů pro vývoj webových aplikací.



Obrázek 3.1: MVP[9]

3.6.2 Latte

Latte[10] je šablonovací systém určený pro PHP a je přímo součástí Nette Frameworku. Hlavní výhody:

- Latte je rychlé: kompiluje šablony do prostého optimalizovaného kódu PHP.
- Latte je bezpečné: je to první PHP engine s kontextově sensitivním escapováním.
- Latte mluví vaším jazykem: má intuitivní syntax a pomáhá snadno vytvářet lepší webové stránky. [10]

3.6.3 Tracy

Tracy[11] je velice užitečná PHP knihovna, jež programátorovi pomůže s odhalováním chyb a optimalizací. Umožňuje například:

- rychlé detekování a opravu chyb
- zaznamenávání chyb
- vypisování hodnot proměnných a objektů

[9] https://en.wikipedia.org/wiki/Model
[10] Latte: https://latte.nette.org/
[11] Tracy: https://tracy.nette.org/

- měření času skriptů a SQL/DQL dotazů
- měření paměťových nároků

[1]

3.7 Relační databáze

Relační databáze ukládá data do řady tabulek, takže data modelují matematickou teorii vztahů. Model umožňuje dotazy založené mimo jiné na projekci, výběru a spojení a spojování dat v tabulkách pomocí klíčů. Dotazy jsou vyjádřeny ve standardní syntaxi zvané SQL, což je standardní dotazovací jazyk, který je společný pro všechny typy relačních databází.

Teorie vztahů říká, že data jsou uspořádána jako různé sady n-tic, nazývaných vztahy, kde n-tice je soubor hodnot pro atributy. Vztah uvádí, které atributy shromažďuje. Konkrétně řečeno atributy jsou sloupce tabulky a n-tice jsou řádky v tabulce. Omezení mezi atributy umožní, aby byly platnými členy vztahu pouze určité n-tice, a databáze by neměla umožňovat vkládání řádků do tabulky, pokud by porušovaly omezení.[4]

3.7.1 SQL

SQL (Structured Query Language) je standardní jazyk pro práci s relačními databázemi. SQL lze použít pro vkládání, vyhledávání, aktualizaci a mazání záznamů databáze. SQL umí spoustu dalších operací včetně optimalizace a údržby databází.

Tabulka je nejzákladnější jednotkou databáze a skládá se z řádků a sloupců dat. Jedna tabulka obsahuje záznamy a každý záznam je uložen v řádku tabulky. Tabulky jsou nejpoužívanějším typem databázových objektů nebo struktur, které uchovávají nebo odkazují na data v relační databázi. Mezi další typy databázových objektů patří následující[4]:

- **Pohledy** jsou logické reprezentace dat sestavených z jedné nebo více databázových tabulek.
- **Indexy** jsou vyhledávací tabulky, které pomáhají urychlit funkce vyhledávání v databázi.
- **Sestavy** se skládají z dat získaných z jedné nebo více tabulek, obvykle podmnožiny těchto dat, která je vybrána na základě vyhledávacích kritérií.

3.8 JavaScript

JavaScript[12] je dynamický programovací jazyk, který se používá pro vývoj webu ve webových aplikacích, pro vývoj her a mnoho dalšího. Umožňuje implementovat dynamické funkce na webových stránkách, které nelze provést pouze pomocí HTML a CSS. Jedná se o nejpoužívanější programovací jazyk vůbec [12]. Ve webových aplikacích se tento jazyk převážně používá na frontendu, avšak jeho využití může být i na backandu například pomocí Node.js. Node.js je open source, multiplatformní runtime prostředí a knihovna, která se používá pro spouštění webových aplikací mimo prohlížeč klienta. Používá se pro programování na straně serveru.

[12] JavaScript: https://www.javascript.com/

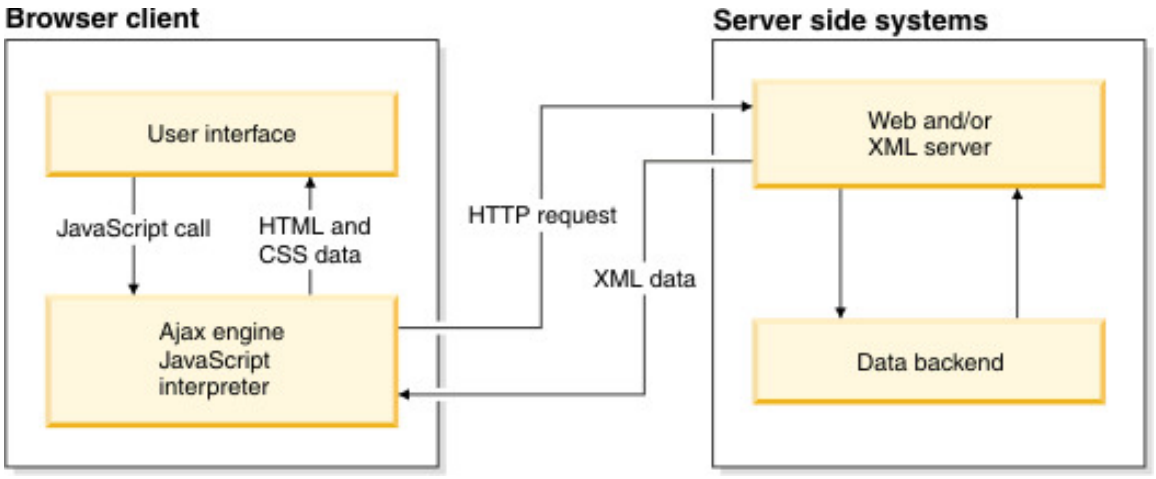
3.8.1 jQuery

jQuery[13] je malá JavaScriptová open-source knihovna, která nám pomáhá vytvářet interaktivní webové stránky s animacemi, vizuálními efekty a pokročilými funkcemi. Je to nejpopulárnější JavaScriptová knihovna, kterou používá asi 70 milionů webových stránek po celém světě. Motto jQuery je „write less, do more“ (překlad: pište méně, udělejte více), protože díky jednoduchému rozhraní redukuje mnoho řádků surového kódu JavaScript na jeden řádek. Mezi hlavní funkce jQuery patří:

- Zpracování událostí
- Manipulace s DOM
- Animace a efekty
- AJAX framework

3.8.2 AJAX

AJAX (Asynchronous Javascript and XML) je technologie (nikoli programovací jazyk), která se používá pro webové aplikace k asynchronnímu přenosu a příjmu dat ze serveru, aniž by to ovlivnilo zbytek obsahu stránky nebo vyžadovalo její opětovné načtení. XML (Extensible Markup Language) se používá k šifrování zpráv tak, aby je mohli číst lidé i stroje. XML je podobné HTML, ale umožňuje vytvářet a upravovat vlastní značky.



Obrázek 3.2: AJAX [14]

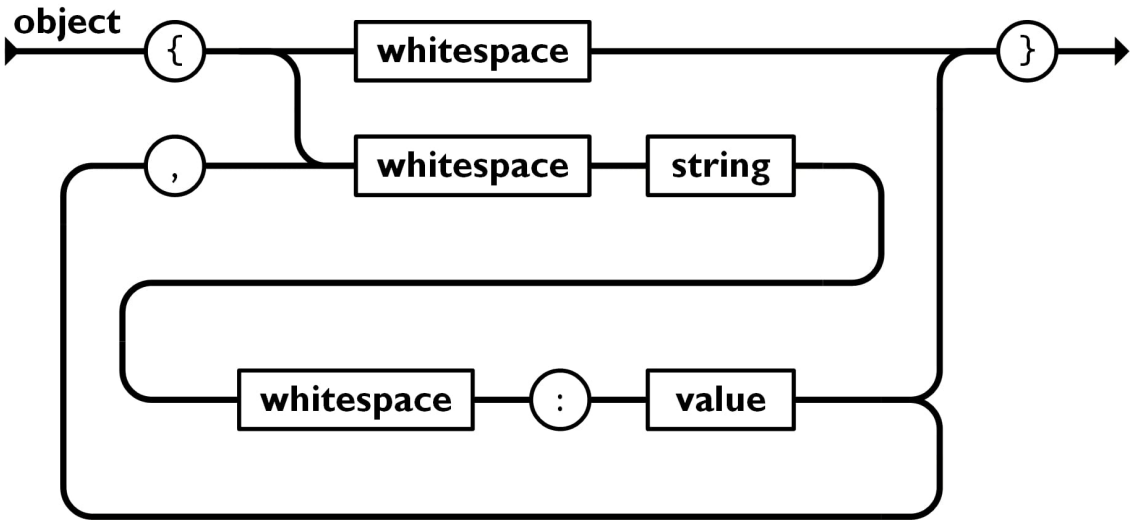
3.8.3 JSON

JavaScript Object Notation (JSON)[15] je standardizovaný formát běžně používaný k přenosu dat jako textu, který lze posílat přes síť. Používá ho spousta API rozhraní a databází a je snadno čitelný jak pro lidi, tak pro stroje.

[13] jQuery: https://jquery.com/

[14] https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=page-asynchronous-javascript-xml-ajax-overview

[15] JSON: https://www.json.org/json-en.html



Obrázek 3.3: JSON[16]

3.9 Python

Python[17] je vysokoúrovňový, interpretovaný a objektově orientovaný skriptovací jazyk. Python je navržen tak, aby byl vysoce čitelný. Používá anglická klíčová slova často tam, kde ostatní jazyky používají interpunkci, a má méně syntaktických konstrukcí než jiné jazyky.

- Python je interpretovaný: Python je zpracováván za běhu interpretem. Před spuštěním programu nemusíte kompilovat. Je to podobné jako u PERL a PHP.
- Python je interaktivní: ve skutečnosti můžete sedět u příkazového řádku Pythonu a komunikovat s interpretem přímo při psaní svých programů.
- Python je objektově orientovaný: Python podporuje objektově orientovaný styl nebo techniku programování, která zapouzdřuje kód do objektů.
- Python je jazyk pro začátečníky: Python je vhodný jazyk pro začínající programátory a podporuje vývoj široké škály aplikací od jednoduchého zpracování textu, přes WWW prohlížeče, až po hry.

3.10 Shell

Shell je software, který poskytuje rozhraní pro uživatele operačního systému pro poskytování přístupu ke službám kernelu[18]. V operačních systémech založených na Unixu nebo Linuxu lze *shell* vyvolat pomocí příkazu shell v rozhraní příkazového řádku (CLI), což uživatelům umožňuje řídit operace prostřednictvím počítačových příkazů, textu nebo skriptu. Pro programovací jazyky existují také shelly, které jim poskytují autonomii od operačního systému a umožňují kompatibilitu napříč platformami.

[16] https://www.json.org/img/object.png
[17] Python: https://www.python.org/
[18] kernel: jádro operačního systému

3.11 Docker

Docker[19] je softwarová platforma, která umožňuje rychle vytvářet, testovat a nasazovat aplikace. Docker balí software do standardizovaných jednotek nazývaných *kontejnery*, které mají vše, co software potřebuje ke spuštění, včetně knihoven, systémových nástrojů, kódu a runtime. Pomocí Dockeru můžete rychle nasadit a škálovat aplikace do jakéhokoli prostředí a vědět, že tento kód poběží všude stejně bez nutnosti další konfigurace a instalování závislostí. Alternativou k Dockeru může být ne tak známý Podman[20]. Docker je v dnešní době velice populární, a to nejenom díky zjednodušení nasazování aplikací, ale i díky orchestraci kontejnerů například pomocí Kubernetes[21].

Dockerfile

Dockerfile je soubor, který obsahuje sadu instrukcí a argumentů, které popisují konfiguraci prostředí. Tyto instrukce jsou vykonávány postupně a automaticky provádějí akce na základním obrazu za účelem vytvoření nového obrazu. Používá se k vytvoření nebo spuštění obrazu Docker zadáním pokynů k úpravě existujícího obrazu Docker na základě našich požadavků, a to automatizovaným způsobem se spuštěním kontejneru Docker. [2]

```
FROM php:8.0.0-apache

RUN a2enmod rewrite
RUN docker-php-ext-install mysqli pdo pdo_mysql

RUN apt-get update && apt-get install -y python3 python3-pip
RUN pip3 install python-gedcom==1.0.0
RUN pip3 install mysql-connector-python-rf
```

Výpis 3.1: Dockerfile použitý v projektu

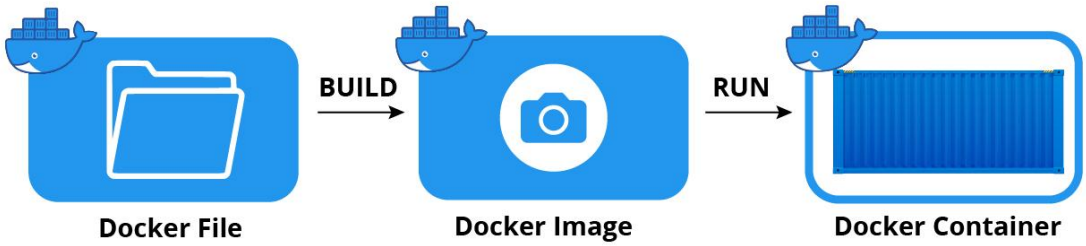
Docker image

Image je inertní neměnný soubor, který je v podstatě obraz (snapshot) kontejneru. Obsahuje instrukce pro vytvoření kontejneru Docker. Docker image je často založen na jiném Docker image. Image se vytvářejí pomocí příkazu *build* a po spuštění příkazem *run* vytvoří kontejner. Image jsou uloženy v registru Dockeru.

Docker kontejner

Docker kontejner je instancí Docker image. Kontejnery lze spustit, restartovat a zastavit. Z jednoho image jsme schopni vytvořit tolik kontejnerů, kolik potřebujeme. Tento koncept usnadňuje škálování služby.

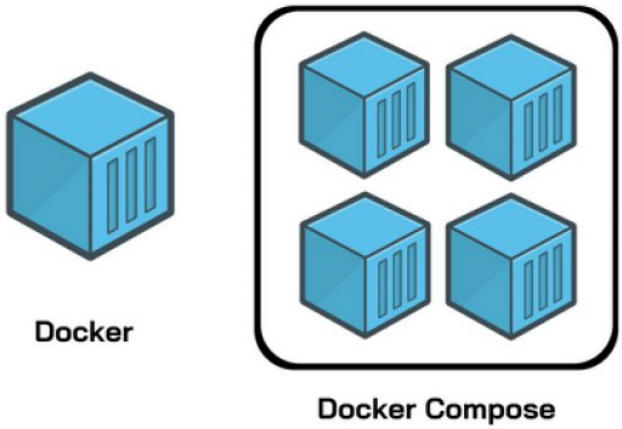
[19] Docker: https://www.docker.com/
[20] Podman: https://podman.io/
[21] Kubernetes: https://kubernetes.io/



Obrázek 3.4: Docker[22]

3.11.1 Docker compose

Docker compose[23] je jednoduchý, ale výkonný nástroj, který se používá ke spouštění více kontejnerů jako jediné služby. Docker Compose funguje na základě použití pravidel definovaných v YAML (YAML Ain't Markup Language) souboru *docker-compose.yaml*. Soubor YAML konfiguruje služby aplikace a obsahuje pravidla určující, jak mají být spuštěny. Po zavedení souboru můžete všechny služby spustit, zastavit nebo obnovit pomocí jediného příkazu. Kromě toho můžete kontrolovat stav služby, zobrazovat výstupy protokolu a spouštět jednorázové příkazy. [2]



Obrázek 3.5: Docker-compose[24]

[22] https://jfrog.com/knowledge-base/a-beginners-guide-to-understanding-and-building-docker-images/

[23] Docker compose: https://docs.docker.com/compose/

[24] https://www.freecodecamp.org/news/a-beginners-guide-to-docker-how-to-create-a-client-server-side-with-docker-compose-12c8cf0ae0aa/

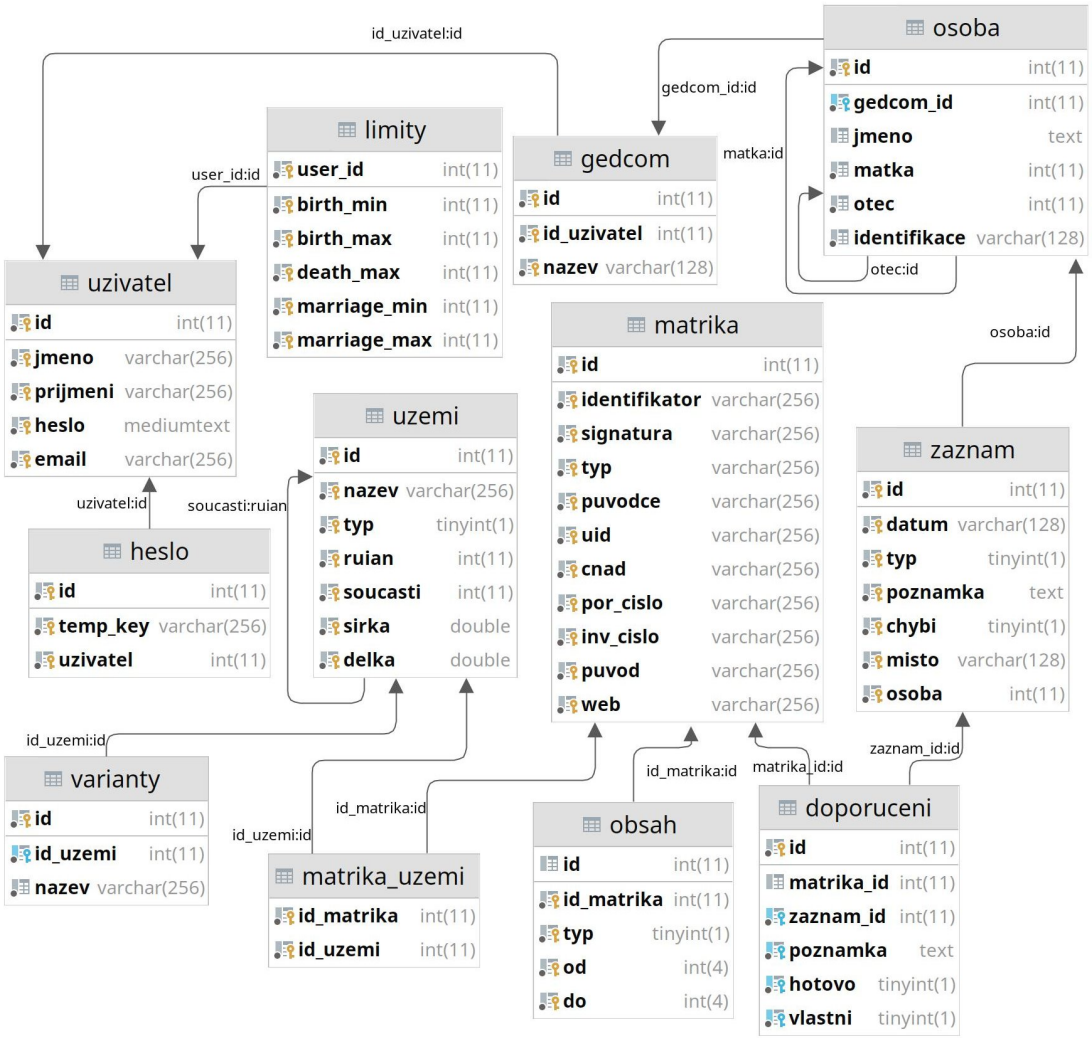
3.12 GitHub Actions

GitHub Actions je platforma, plně integrovaná s GitHubem pro kontinuální integrace a kontinuálního nasazení kódu (CI/CD), která umožňuje automatizovat sestavování, testování a nasazování. Pro konfiguraci se využívá YAML soubor, ve kterém jsou definovány jednotlivé akce a příkazy. YAML soubor je uložen v adresáři *.github/workflows* v kořenovém adresáři úložiště.

Kapitola 4

Návrh databáze

Bylo potřeba vytvořit databázi, ve které budou uložena všechna data. Byla použita relační databáze mysql. V této kapitole budou popsány její tabulky a relace.



Obrázek 4.1: Schéma databáze

Uživatel

V tabulce *uživatel* jsou uloženy veškeré informace o uživateli. Jejím primárním klíčem je *ID*. Dále tabulka uživatel obsahuje jméno, příjmení, zahashované heslo a email. Ten je nastaven jako unikátní klíč, slouží totiž k přihlášení uživatele.

Heslo

Tabulka *heslo* slouží k ukládání dočasných tokenů použitých k restartování hesla. Pokud tedy uživatel zapomene svoje heslo, může si jej restartovat tak, že na úvodní straně klikne na odkaz "Zapomněli jste heslo? Obnovte si jej" a zadá email, pod kterým se zaregistroval. Následně mu přijde email s URL adresou pro obnovení hesla. V této URL adrese bude jako parametr token. Po použití se token z databáze odstraní. Uživatel je cizí klíč do tabulky *uzivatel.id*.

Gedcom

Tato tabulka reprezentuje nahraný GEDCOM od uživatele. Jako její primární klíč slouží sloupec *id*. Dále tabulka Gedcom obsahuje sloupec *název*, díky němuž si uživatel daný GEDCOM může pojmenovat. Pomocí cizího klíče *id_uzivatel* je tabulka v relaci s tabulkou uživatel.

Osoba

Tabulka *osoba* slouží k uložení dat o konkrétní osobě z GEDCOMu. Jejím primárním klíčem je pole *ID*. Cizí klíč do tabulky *gedcom* je gedcom_id. Pole *jmeno* obsahuje celé jméno dané osoby čili křestní jméno, příjmení, popřípadě i prostřední jméno. Tento sloupec je možno rozdělit na více sloupců, ale v aplikaci to nebylo potřeba. Atribut *identifikace* je použit k uložení ID konkrétní osoby z GEDCOMu. Tato tabulka obsahuje 2 unární vztahy[1], a to konkrétně v atributech *matka* a *otec*. Ty slouží k uložení *ID* z tabulky osoba matky nebo otce, pokud tuto informaci máme. Díky tomuto je možné najít děti, vnoučata, rodiče či prarodiče.

Záznam

V této tabulce jsou uložena data o jednotlivých událostech, tedy konkrétně narození, smrt a svatba. Primárním klíčem této tabulky je atribut *ID*. Atribut *datum* slouží k uložení data záznamu řádku DATE z GEDCOMu a je datového typu varchar. A to z toho důvodu, že v GEDCOMu může být zapsán různými způsoby a ve webové aplikaci slouží pouze k zobrazení kompletních záznamů. Dále tabulka záznam obsahuje atribut *místo*, který slouží k uložení místa záznamu řádku PLAC z GEDCOMu. Pole *poznámka* slouží k uchování poznámky pro konkrétní záznam. Sloupec *chybí* zaznamenává, zda je záznam kompletní, nebo chybí alespoň jedno z dvojice místo nebo datum. Pole *typ* určuje, o jaký typ se jedná. Číslo jedna označuje záznam narození, číslo dva značí, že se jedná o záznam úmrtí, a číslo tři značí, že se jedná o záznam oddání. Atribut *Osoba* je cizím klíčem do tabulky osoba.

[1] Unarní vztah: relace je spojena sama se sebou

Matrika

Tabulka *matrika* je určená k uložení dat o entitě matriky. Primárním klíčem této tabulky je sloupec *ID*. Atribut *web* slouží k určení, na kterém webu je matrika dostupná, ne konkrétní URL pro danou matriku. Dále obsahuje atributy *identifikator, typ, signatura, puvodce, uid, cnad, por_cislo, inv_cislo a puvod*, které slouží k identifikaci matriky.

Území

V této tabulce jsou uložena veškerá data o územích (okresy, obce, části obcí ...). Jejím primárním klíčem je pole *ID*. Sloupec *název* slouží k uložení názvu konkrétního území. Atribut *typ* slouží k uložení typu území (0 - stát, 1 - region, 2 - kraj, 3 - okres, 4 - obec, 5 - část obce/města). Pole *ruian* určuje RÚIAN[2] kód daného území. Obsahuje také sloupce *sirka* a *delka* pro uložení dat zeměpisné šířky a zeměpisné délky. Tato tabulka obsahuje unární vztah, a to *soucasti:ruian*.

Doporučení

Tabulka slouží k uložení doporučení matrik k chybějícím záznamům. Sloupec *ID* je primárním klíčem této tabulky. Tato tabulka obsahuje 2 cizí klíče, a to *matrika_id* a *zaznam_id*. Tabulky záznam a matrika jsou tedy ve vztahu M:N a tato tabulka slouží jako spojovací tabulka. Cizí klíč *matrika_id* se odkazuje na sloupec *matrika.id*. Cizí klíč *zaznam_id* se odkazuje na sloupec *zaznam.id*. Pole *poznámka* slouží k uchování poznámky pro konkrétní doporučení. Atribut *hotovo* uchovává data o tom, zda byla daná matrika uživatelem již prohledána.

Obsah

Tabulka *obsah* je určená k tomu, aby uchovávala informace o tom, jaké typy záznamů a z jakého období matriky obsahuje. Je ve vztahu 1:N s tabulkou matrika. *id_matrika* je cizí klíč do tabulky matriky. Dále tabulka obsahuje sloupce *od* a *do*, jež slouží k uložení informací, od kterého roku, po který rok, daný typ záznamů matrika obsahuje. Atribut *typ* určuje jakého typu záznamy matrika obsahuje (1 - INDEX Narozených, 2 - INDEX Oddaných, 3 - INDEX Zemřelých, 4 - Narození, 5 - Oddaní, 6 - Zemřelí).

Matrika - území

Jedná se o propojovací tabulku mezi tabulkami *matrika* a *uzemi*. Tabulka má složený primární klíč, a to ze dvou atributů *id_matrika* a *id_uzemi*. Cizí klíč *id_matrika* se odkazuje na sloupec *matrika.id*. Cizí klíč *id_uzemi* se odkazuje na sloupec *uzemi.id*.

Limity

Tabulka *limity* slouží k uložení uživatelem zadaných limitů. Je využita pouze v případě, že uživatel nepoužije výchozí hodnoty, ale rozhodne se použít svoje vlastní. Jejím primárním klíčem je *user_id*. Dále obsahuje sloupce *birth_min, birth_max, death_max, marriage_min, marriage_max* které slouží k uložení těchto limitů.

[2]RÚIAN: Registr územní identifikace, adres a nemovitostí url: https://www.cuzk.cz/ruian

Varianty

Jedná se o tabulku, ve které jsou uloženy varianty názvů území. Jejím primárním klíčem je sloupec *id*. Dále je tabulka ve vztahu 1:N s tabulkou území, a to přes cizí klíč *id_uzemi*. Pomocí atributu *nazev* je uložena konkrétní varianta názvu daného území. Díky tomuto je možné dohledat území i pokud bude v GEDCOMu pod jiným názvem. Je tedy možné najít Bohuslavice pokud bude v GEDCOMu uvedeno Bohuslawicz, Bohuslawitium, Boslawicz, Bouslawicze, Buohoslavice, Buslavice nebo Buslawitz.

Kapitola 5

Implementace

5.1 Území

Bylo potřeba naplnit databázi daty o územích. Jako zdroj těchto dat byla zvolena volně dostupná data z Českého úřadu zeměměřického a katastrálního dostupná v *.csv formátu z webu https://www.cuzk.cz/ruian/Poskytovani-udaju-ISUI-RUIAN-VDP/Ciselniky-ISUI.aspx. Konkrétně RÚIAN[1] Číselníky ISÚI[2]. Tato data se ještě dále dělí na vyšší prvky (stát až obec s pověřeným obecním úřadem) a nižší prvky (obec až ulice).

Pomocí shell a awk[3] bylo vytvořeno 6 skriptů, které z těchto *.csv souborů vytváří SQL skripty. Tímto způsobem byla zpracována data o regionech, krajích, okresech, obcích, částech měst a částech obcí. Data o státu Česká republika byla do vytvořeného SQL skriptu vložena ručně z toho důvodu, že se jedná pouze o jeden řádek a bylo by zbytečné na toto vytvářet skript. Každý typ území (kraj, obec, okres ...) má svůj vlastní *.csv soubor a kromě názvu a kódu (který je jedinečný pro daný typ) obsahuje i kód nadřazeného území. Například pokud vezmeme obec Alojzov s kódem 506761, můžeme zjistit, že se tato obec nachází v okrese Prostějov (OKRES_KOD = 3709). Kdyby bylo v tomto procesu pokračováno, ve výsledku by bylo dosaženo až České republiky. Díky tomuto je u každého území možné zjistit, jakého území je součástí. Toho je později využito u parsování dat matrik.

```
cat UI_OBEC.csv | awk -F ";" '{printf("insert into uzemi (nazev,typ,ruian,
    soucasti) values (\"%s\",4, %d, %d); \n", $2, $1, $5)}' | head -n -1 |
    tail -n+2
```

Dále byl vytvořen skript, který spustí vložení SQL dat na SQL server.

```
docker exec -i $(docker-compose ps -q db) mysql -f --reconnect -uroot -
    ppassword bp < uzemi.sql
```

[1] RÚIAN: Registr územní identifikace, adres a nemovitostí https://www.cuzk.cz/ruian/
[2] ISÚI: Informační systém územní identifikace
[3] awk: programovací jazyk pro práci s textem

```
KOD;NAZEV;STATUS_KOD;POU_KOD;OKRES_KOD;CLENENI_SM_ROZSAH_KOD;
CLENENI_SM_TYP_KOD;PLATI_OD;PLATI_DO;DATUM_VZNIKU
554979;Abertamy;3;1121;3403;;;25.03.2020 00:00:00;;
531367;Adamov;2;221;3205;;;30.11.2016 00:00:00;;24.11.1990 00:00:00
535826;Adamov;2;647;3301;;;05.06.2015 00:00:00;;24.11.1990 00:00:00
581291;Adamov;3;2691;3701;;;02.07.2011 00:00:00;;
547786;Adršpach;2;2283;3605;;;30.05.2016 00:00:00;;01.09.1990 00:00:00
547981;Albrechtice;2;2631;3611;;;26.03.2020 00:00:00;;24.11.1990 00:00:00
598925;Albrechtice;2;3565;3803;;;23.03.2020 00:00:00;;01.01.1869 00:00:00
549258;Albrechtice nad Vltavou;2;868;3305;;;17.04.2020 00:00:00;;
563528;Albrechtice v Jizerských horách;2;1694;3504;;;17.04.2020 00:00:00;;
568741;Albrechtičky;2;3671;3804;;;05.06.2015 00:00:00;;24.11.1990 00:00:00
506761;Alojzov;2;3140;3709;;;01.04.2020 00:00:00;;28.02.1990 00:00:00
551929;Andělská Hora;3;3425;3801;;;05.06.2015 00:00:00;;24.11.1990 00:00:00
538001;Andělská Hora;2;1104;3403;;;07.01.2021 00:00:00;;24.11.1990 00:00:00
```

Výpis 5.1: Ukázka číselníků ISÚI obcí
Zdroj: https://www.cuzk.cz/CUZK/media/CiselnikyISUI/UI_OBEC/UI_OBEC.zip

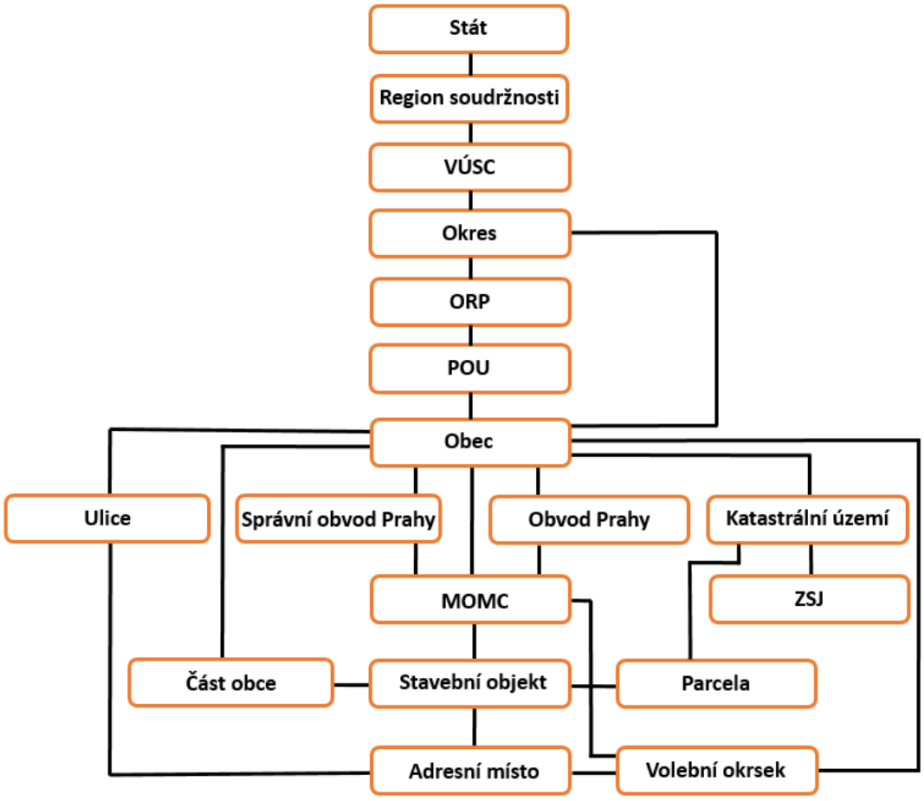
5.1.1 RÚIAN

Jedná se o jeden ze čtyř základních veřejně dostupných registrů dle zákona č. 111/2009 Sb. Tyto registry jsou základním stavebním kamenem digitalizace veřejné správy. Tento veřejný registr neobsahuje žádné osobní informace a je unikátním zdrojem adres a jiných dat týkajících se katastru. Správcem registru je Český úřad zeměměřický a katastrální. Základními registry jsou:

- Registr obyvatel (ROB) - gestor MVČR
- Registr osob (ROS) - gestor Český statistický úřad
- Registr územní identifikace, adres a nemovitostí (RÚIAN) - gestor Český úřad zeměměřický a katastrální
- Registr práv a povinností (RPP) - gestor MVČR

RÚIAN obsahuje:

- údaje o územních prvcích
- údaje o územně evidenčních jednotkách
- adresy (údaje pro doručování prostřednictvím poštovních služeb) [6]



Obrázek 5.1: Schéma prvků RÚIAN[4]

5.1.2 Zeměpisné souřadnice

Bylo potřeba k daným územím přidat i zeměpisné souřadnice, které se využívají při výpočtu vzdálenosti obcí při přiřazení matriky k chybějící události. Vzhledem k tomu, že data použitá pro vytvoření záznamů o územích zeměpisné souřadnice neobsahovaly, bylo je potřeba doplnit jinak. Pro tento účel byl zvolen projekt pod MIT licencí od Česko.Digital[5] obce. Tento projekt nabízí již zpracovaná data o obcích včetně zeměpisných souřadnic ve formátu JSON[6]. Tato předzpracovaná data byla pomocí Python skriptu uložena do databáze k příslušným záznamům.

[4] https://www.cuzk.cz/ruian/Poskytovani-udaju-ISUI-RUIAN-VDP/Vymenny-format-RUIAN-(VFR)/Struktura-a-popis-VFR-1_8_0.aspx

[5] Česko.Digital: https://cesko.digital/

[6] JSON: JavaScript Object Notation https://www.json.org/json-en.html

5.2 Zpracování dat o matrikách

Data o matrikách byla poskytnuta Ing. Petrem Veigendem ve formátu JSON. Tato data bylo potřeba zpracovat a přiřadit ke správným územím.


```
{
    "zdroj": "actapublica",
    "vytvoreno": "2021-03-10 13:08:13.825752",
    "stazeno": "2020-09-27 13:28:30.306616",
    "pocet": 11641,
    "snimky_zaklad": "http://actapublica.eu/brno/",
    "matriky": [
        {
            "id": "1",
            "typ": "římskokatolická církev",
            "okres": {
                "id": "3701",
                "nazev": "Blansko",
                "kraj": {
                    "id": "116",
                    "nazev": "Jihomoravský kraj"
                }
            },
            "jazyky": [
                "němčina"
            ],
            "puvodce": "Adamov",
            "obce": {
                "Adamov": {
                    "ruian": {
                        "id": "581291",
                        "nazev": "Adamov",
                        "okres": {
                            "id": "3701",
                            "nazev": "Blansko",
                            "kraj": {
                                "id": "116",
                                "nazev": "Jihomoravský kraj"
                            }
                        }
                    }
                }
            },
            "obsah": {
                "typ": "Narození",
                "rozsah": {
                    "od": "1857",
                    "do": "1884"
                }
            },
            "pocet_snimku": "122"
        },
```


Výpis 5.2: ukázka JSON souboru poskytnutého Ing. Petrem Veigendem z actapublica

Pro tato data bylo potřeba napsat několik různých parserů, a to z toho důvodu, že každý archív měl svůj vlastní JSON soubor. Tyto soubory se od sebe lišily, protože každý archív zveřejňuje na webu různá data o matrikách. Všechny tyto parsery byly napsány v jazyce Python a data přímo vkládají do databáze pomocí *mysql.connector*. Pomocí těchto parserů byly naplněny tabulky *matrika*, *obsah*, *varianty* a *matrika_uzemi*. Pokud objekt území v matrice v JSON souboru obsahoval RÚIAN kód, tak byl záznam přímo vložen do databáze. Pokud RÚIAN kód neobsahoval, bylo potřeba zjistit, o které území se jedná.

Toho bylo docíleno tak, že pokud se jednalo o unikátní název území (žádný stejný se v databázi nevyskytoval), bylo použito toto území. Pokud ne, hledalo se území, které bylo součástí stejného území, jako jiné území obsažené v matrice. Předpokládáme, že v jedné matrice budou data o územích nacházejících se poblíž sebe. Je-li takovýchto území více, je použito území, jež se vyskytuje v jedné matrice nejvícekrát.

5.3 Doporučení Matriky

Jednou z nejdůležitějších částí aplikace je způsob jakým, bude přiřazovat chybějící události k matrikám. Za chybějící událost je považována taková, u které chybí alespoň jeden z dvojice datum a místo. V této části bude popsáno, jakým způsobem jsou matriky doporučovány.

5.3.1 Parsování GEDCOMu

Pro parsování GEDCOMu byl zvolen modul pro jazyk Python python-gedcom[7]. Jedná se o jednoduchý, přesto velice užitečný modul, který kromě parsování GEDCOMu umožňuje i jeho analýzu a manipulaci s jednotlivými záznamy. Tento modul podporuje pouze GEDCOM >= 5.5. Z tohoto důvodu aplikace podporuje pouze GEDCOM vyšší než je tato verze. Pokud se uživatel pokusí použít starší verzi, zobrazí se mu hláška, že tato verze není podporována. Většině uživatel toto problémy způsobovat nebude vzhledem k tomu, že standard 5.5 pochází již z roku 1995. Například jeden z nejpoužívanějších genealogických webů MyHeritage[8] používá verzi 5.5.1.

5.3.2 Přiřazování matrik k událostem

Byl vytvořen Python skript, který pomocí python-gedcom rozparsuje GEDCOM na jednotlivé objekty (Rodina, Osoba ...). Objekty osoby z jednotlivých rodin jsou následně procházeny cyklem a je zjišťováno, zda některé z událostí chybí. Pokud je osoba mladší 100 let, tak událost úmrtní není považována za chybějící. Rovněž pokud osoba nemá děti, není událost sňatek považována za chybějící. Pokud je zjištěno, že událost chybí je postupováno následovně.

Výchozí hodnoty použité pro přiřazování matrik k chybějícím událostem (tyto hodnoty mohou být uživatelem změněny):

- BIRTH_MIN = 15 (minimální věk početí dítěte)
- BIRTH_MAX = 45 (maximální věk početí dítěte)
- DEATH_MAX = 100 (maximální věk v době úmrtí)
- MARRIAGE_MIN = 18 (minimální věk v době svatby)

[7] python-gedcom: https://github.com/nickreynke/python-gedcom
[8] MyHeritage: https://www.myheritage.cz/

- MARRIAGE_MAX = 50 (maximální věk v době svatby)

Pokud se jedná o událost narození, je vytvořen seznam míst, kde by se mohla daná osoba narodit. Tento seznam se skládá z míst narození dětí, svatby rodičů a úmrtí rodičů. Z tohoto seznamu jsou odstraněny duplicity a jsou přidány obce v okolí přibližně 5 kilometrů. Tato vzdálenost byla aproximována pomocí tohoto vzorce:

$$dist = \sqrt{(lat1 - lat2)^2 + (lng1 - lng2)^2} * 0.012$$

kde lat1, lat2, lng1, lng2 jsou souřadnice bodů, jejichž vzdálenost máme určit, a 0.012 je magická konstanta, která v našich zeměpisných šířkách upravuje výsledek tak, aby udával vzdálenost v kilometrech.[5]

Nejedná se sice o přesný výsledek, ale pro naše účely je dostatečný. Následně je třeba určit období, ve kterém se tato chybějící událost mohla stát. Období **od** je určeno tak, že je zvoleno nejmenší číslo z dvojice:

- narození mladšího rodiče + BIRTH_MIN
- narození nejmladšího dítěte - BIRTH_MAX

Období **do** je určeno tak, že je zvoleno větší číslo z dvojice:

- narození mladšího rodiče + BIRTH_MAX
- narození nejmladšího dítěte - BIRTH_MIN

Pokud máme informaci o svatbě rodičů, je parametr *od* posunut na toto datum. Následně si skript v databázi nalezne matriky, které jsou typu narození, nebo index narození a obsahují alespoň jedno z míst, kde se osoba mohla narodit, a obsahují informace o rozmezí, které bylo určeno. Pokud ovšem datum narození známe, použije se přímo tento rok. Výsledné matriky jsou pak přiřazeny k chybějící události. Pokud žádné informace o místech, kde se osoba mohla narodit, nemáme (neznáme informace o narození dětí, svatbě rodičů a úmrtí rodičů) nebo pokud chybí data o narození/úmrtí rodičů i dětí, žádné matriky přiřazeny nejsou.

Pokud se jedná o událost oddání, je vytvořen seznam území, kde se mohla svatba uskutečnit. Tento seznam se skládá z místa narození nevěsty, místa narození ženicha, míst narození rodičů a míst narození dětí. Tato místa jsou doplněna o obce v rozsahu 5 kilometrů stejným způsobem, jako je popsáno u narození. Následně je třeba určit období, ve kterém byl tento sňatek uzavřen. Pokud neznáme narození ani jednoho z partnerů, nedojde k doporučení žádné matriky.

Období **od** je zvoleno dále popsaným způsobem. Pokud známe narození obou partnerů, je zvoleno narození mladšího z partnerů + MARRIAGE_MIN. Pokud známe narození jednoho z nich, je zvoleno toto datum + MARRIAGE_MIN.

Období **do** je zvoleno tímto způsobem. Pokud známe narození obou partnerů, je zvoleno narození staršího z partnerů + MARRIAGE_MAX. Pokud známe narození jednoho z nich, je zvoleno toto datum + MARRIAGE_MAX. Pokud ovšem známe data narození dětí, je tento parametr změněn na narození nejstaršího dítěte - 1 rok.

Pokud osoba nemá děti, tak není událost oddání považována za chybějící. Ovšem pouze v tom případě, pokud není v souboru GEDCOM uvedeno datum nebo místo sňatku. Výsledné matriky jsou pak přiřazeny k chybějící události. Pokud žádné informace o místech, kde se osoba mohla narodit nemáme, žádné matriky přiřazeny nejsou.

Pokud se jedná o událost úmrtí, je vytvořen seznam území, kde mohla daná osoba zemřít. Tento seznam se skládá z místa svatby, místa úmrtí partnera, míst narození dětí a místa narození této osoby. Tato místa jsou doplněna o obce v rozsahu 5 kilometrů rovněž stejným způsobem, jako je popsáno u narození. Následně je třeba určit období, ve kterém se mohla smrt odehrát.

Období **od** je zvoleno tímto způsobem, a to tak, že nižší číslo má vyšší prioritu:

1. pokud známe narození dětí, tak je zvoleno datum narození nejmladšího dítěte - 1
2. pokud známe datum narození dané osoby, je zvoleno toto datum

Pokud neznáme ani jedno z výše uvedených, žádná matrika doporučena není.
Období **do** je zvoleno tímto způsobem, a to tak, že nižší číslo má vyšší prioritu:

1. pokud známe datum narození dané osoby, je zvoleno toto datum + DEATH_MAX
2. pokud známe datum narození dětí, tak je zvoleno datum narození nejstaršího dítěte + DEATH_MAX - BIRTH_MIN

Pokud neznáme ani jedno z výše uvedených, žádná matrika doporučena není. Výsledné matriky jsou pak přiřazeny k chybějící události. Pokud žádné informace o místech, kde se osoba mohla narodit nemáme, žádné matriky přiřazeny nejsou.

Skript výsledná data vypisuje ve formátu JSON do standardního výstupu. Tato data jsou následně zpracována webovou aplikací pomocí PHP. Tento skript tedy není závislý na webové aplikaci, ale pouze na databázi.


```
{
  "info": {
    "return_code": 0,
    "text": "OK"
  },
  "PEOPLE": {
    "@I500017@": {
      "ID": "@I500017@",
      "NAME": "Hedvika Kone\u010dn\u00fd",
      "MOTHER": "@I500025@",
      "FATHER": "@I500026@",
      "BIRT": {
        "DATE": "1921",
        "PLACE": "",
        "MISSING": true,
        "VR": [
          28498,
          28793,
          33172,
          33216
        ]
      },
      "DEATH": {
        "DATE": "",
        "PLACE": "",
```

```
        "MISSING": true,
        "VR": [
          28449,
          28450,
          28451,
          28452
        ]
      },
      "MARRIAGE": {
        "DATE": "",
        "PLACE": "",
        "MISSING": true,
        "VR": [
          28528,
          28529,
          28530
        ]
      }
    }
  }
}
```


Výpis 5.3: Příklad výstupu skriptu pro přiřazování matrik

Problém duplicity jmen

U některých míst se při vyhledávání naskytl problém, že nešlo jednoznačně určit, o které místo se jedná, a to v případě, že se jedná o duplicitní jméno místa například Nová Ves, Petrovice, Němčice, Vrbice nebo Lhotka. Obcí s duplicitními názvy je v republice více. Při doporučení matriky je potřeba určit správné místo a nelze doporučit všechny. Pokud doporučujeme matriku a název daného území, kde by se chybějící událost mohla odehrát, není jednoznačný (území s tímto místem se v databázi nachází více), je potřeba odhadnout, které území konkrétně to může být. Toho je v aplikaci docíleno následovně. V první řadě zjistíme, zda lze jednoznačně určit místa narození, úmrtí a svateb nejbližších příbuzných. Tedy rodičů, sourozenců a dětí. Pokud toto možné není, nemáme o nich žádné informace nebo veškeré události blízkých příbuzných proběhly v oblastech, které pouze z názvu nelze jednoznačně určit, je toto území odebráno z možností. Pokud tomu tak není, lze tedy jednoznačně určit z jakých oblastí rodina pochází, tak je pomocí vzorce uvedeného výše spočítána průměrná vzdálenost mezi všemi těmito místy a všemi variantami hledaného místa a je vybráno takové, které je v průměru nejblíže. Předpokládáme tedy, že rodina pochází z jedné oblasti.

Pokud tedy budeme například hledat událost narození nějaké osoby a budeme se domnívat, že by se tato událost mohla stát ve Lhotce, kterých se v republice nachází několik, nebude jasné, které matriky doporučit. Pokud však budeme vědět, že rodina této osoby pocházela z Hlučínska, můžeme předpokládat, že se daná událost mohla odehrát ve Lhotce u Hlučína viz, obrázek níže.

Obrázek 5.2: Příklad doporučení

V tomto případě by byla zvolena Lhotka B.

5.4 Implementace webu

Další velice důležitou částí bylo vytvořit webovou aplikaci, která bude pracovat s implementovanou databází a skriptem pro doporučování matriky. Jako kostra pro implementaci webové aplikace byl zvolen projekt *nette/web-project nette-blog*, který je doporučen na webu Nette pro první aplikaci. Jedná se minimální o projekt, který definuje základní doporučenou strukturu projektu viz níže.

```
nette-blog/
├── app/              ← adresář s aplikací
│   ├── Presenters/   ← třídy presenterů
│   │   └── templates/← šablony
│   ├── Router/       ← konfigurace URL adres
│   └── Bootstrap.php ← zaváděcí třída Bootstrap
├── bin/              ← skripty spouštěné z příkazové řádky
├── config/           ← konfigurační soubory
├── log/              ← logování chyb
├── temp/             ← dočasné soubory, cache, …
├── vendor/           ← knihovny instalované Composerem
│   └── autoload.php  ← autoloading všech nainstalovaných balíčků
└── www/              ← veřejný adresář - jediný přístupný z prohlížeče
    └── index.php     ← prvotní soubor, kterým se aplikace spouští
```

Obrázek 5.3: Struktura aplikace[9]

Adresář www/ je určen pro ukládání obrázků, JavaScript souborů, CSS stylů a dalších veřejně přístupných souborů. Pouze tento adresář je přístupný z internetu, takže je potřeba nastavit kořenový adresář vaší aplikace tak, aby směřoval právě sem (to lze nastavit v konfiguraci Apache nebo nginx). Nejdůležitější složka je pro naše účely app. Zde nalezneme soubor Bootstrap.php, ve kterém je třída, která slouží k načtení celého frameworku a nastavení aplikace. Aktivuje se zde autoloading, nastaví se zde debugger a routy.[8]

Webová aplikace se skládá ze 3 základních celků. Jsou to model, view a presenter.

5.4.1 Model

Model je datový a zejména funkční základ celé aplikace. Je v něm obsažena celá aplikační logika (používá se i termín byznys logika). Je to ono M z MVC nebo MVP. Jakákoliv akce uživatele (přihlášení, vložení zboží do košíku, změna hodnoty v databázi) představuje akci modelu. Model si spravuje svůj vnitřní stav a ven nabízí pevně dané rozhraní. Voláním funkcí tohoto rozhraní můžeme zjišťovat či měnit jeho stav. Model neví o existenci view nebo controlleru.[9]

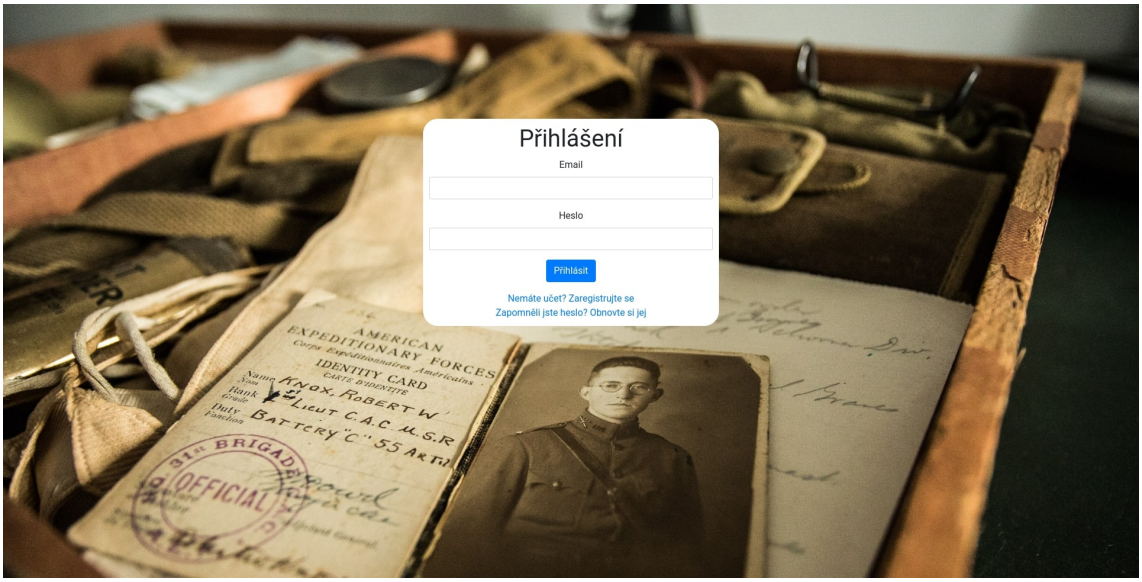
V modelu je tedy veškerá logika a práce s databází. Pro práci s databází byla vytvořena abstraktní třída *Repository*, od které dědí všechny modely. V této obecné třídě jsou definovány obecné metody pro práci s tabulkami jako například *final public function get($key): ActiveRow* pro získání řádku tabulky pomocí primárního klíče nebo *final public function beginTransaction()* pro zahájení transakce. Dále jsou v projektu použity tyto modely.

DoporuceniRepository, který se stará o práci s daty doporučení. *GedcomRepository*, který zpracovává veškerá data o GEDCOMu, jako například vytváření nového, aktualizace stávajícího a odebírání. Model *HesloRepository* slouží pro změnu hesla a obnovení hesla. Model *OsobaRepository* slouží pro manipulaci s daty osob z nahraného GEDCOMu. *UzivatelRepository*, který slouží pro práci s tabulkou uživatele *ZaznamRepository*, pracuje s daty událostí jednotlivých osob z GEDCOMu.

[9] https://doc.nette.org/cs/quickstart/getting-started

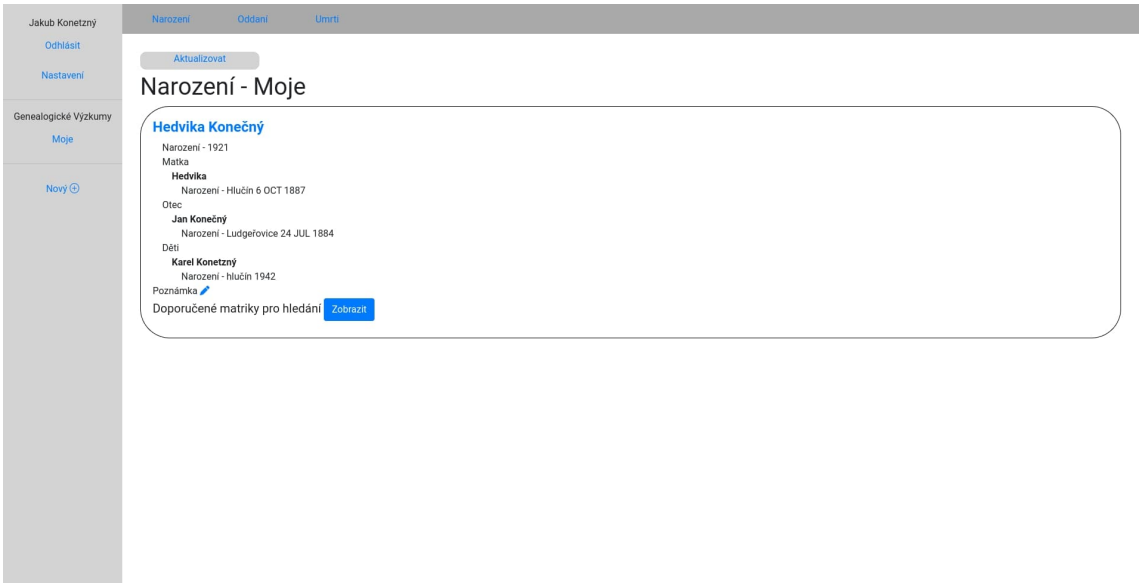
5.4.2 View

View je zobrazovací vrstva, která je zodpovědná za zobrazení dat, která jí předá presenter nebo komponenta. V projektu jsou definovány dvě šablony. První *@layoutLogin.latte* určená pro zobrazení před přihlášením, tedy při přihlášení, při registraci a při obnovení hesla. Tato šablona slouží pouze k zobrazení bloku uprostřed stránky s nadpisem a zobrazení formuláře.



Obrázek 5.4: Webová aplikace - ukázka zobrazení @layoutLogin.latte

Druhá *@layout.latte* určená k zobrazení všech ostatních stránek. Ve výchozím stavu je v ní importované menu, díky čemuž jej není nutné importovat jinde, ale stačí použít šablonu. V této šabloně jsou použita dvě makra, *content* sloužící pro vložení obsahu vedle menu a *skript* určené k importování javascriptu.

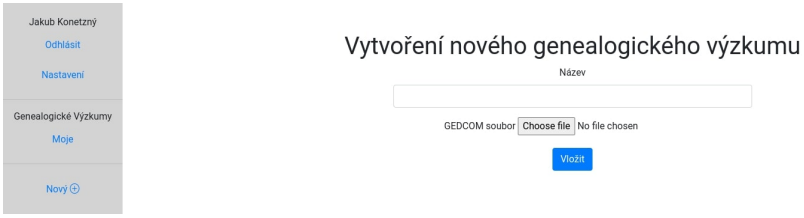


Obrázek 5.5: Webová aplikace - ukázka zobrazení @layout.latte

5.4.3 Presenter

Jedná se o potomka třídy *Nette/Application/UI/Presenter*. Presenter je třída, která zpracuje požadavek uživatele a provede požadovanou akci. To, jakým způsobem má být prováděno routování, je popsáno v třídě *RouterFactory*. Byl implementován presenter *SignPresenter*, který má na starost přihlašování, registraci a obnovení zapomenutého hesla.

Dále byl vytvořen *BasePresenter*, který slouží jako bázová třída pro všechny ostatní presentery. Tento presenter se pomocí metody *startup*, která se spouští vždy po konstruktoru, stará o to, aby byl nepřihlášený uživatel vždy přesměrován na presenter *Sign:in* (Presenter SignPresenter, akce in), čili znemožní nepřihlášenému uživateli aplikaci používat. Dále implementuje metodu pro odhlášení a metodu pro naplnění menu položkami. Z tohoto presenteru dědí všechny ostatní presentery. *SettingsPresenter* je presenter, který má na starosti veškeré akce ohledně změny nastavení. Presenter *HomepagePresenter* má jediný úkol, a to zobrazení domovské stránky po úspěšném přihlášení. *GedcomPresenter* je Presenter, který se stará o vytváření nového genealogického výzkumu a aktualizaci stávajícího.



Obrázek 5.6: Webová aplikace - vytvoření nového výzkumu

Rovněž se stará o zobrazení všech chybějících záznamů, a to tak, že pomocí *Nette/Application/UI/Multiplier* několikrát vykreslí komponentu[10] *Osoba*, která slouží pro zobrazení údajů osoby. Tato komponenta dále používá komponentu *Poznámka*, která slouží jak k vedení poznámek pro konkrétní záznam, tak k vedení poznámek k jednotlivým matrikám. Toto je znovu řešeno pomocí *Multiplier*. Tento presenter se dále stará o předání GEDCOMu skriptu k doporučování matrik, výsledek následně zpracuje a uloží do databáze. Pokud skript vrátí chybu, je zobrazena uživateli. *RecordPresenter* slouží k zobrazení dat o jedné osobě, využívá k tomu komponentu *Osoba*.

Restartování hesla

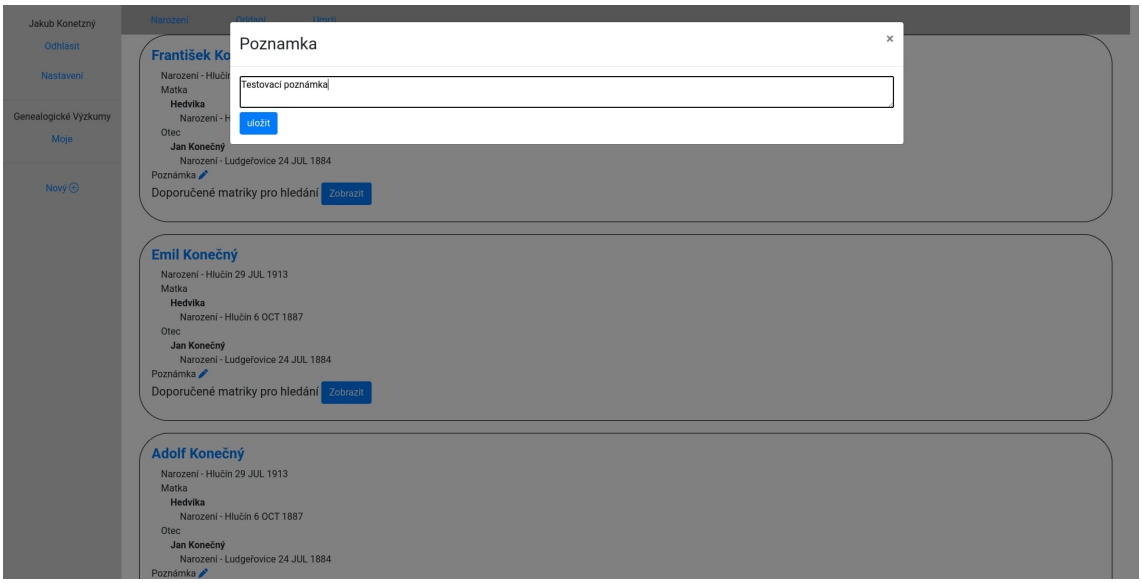
Pokud uživatel zapomene heslo, je možné jej obnovit. Slouží k tomu odkaz na úvodní stránce. Po zadání platného emailu, který je veden v databázi, je aplikací vygenerován token, který se uloží do databáze. Následně je uživateli na zadaný email odeslán link, ve kterém je daný token jako parametr. Pokud uživatel přejde na zaslaný link, aplikace mu umožní změnit heslo a token je z databáze odstraněn.

Vytváření poznámek

Webová aplikace umožňuje vést si dva typy poznámek. První je určen pro konkrétní chybějící záznam (Jan Novák - Oddání). Tato poznámka se zobrazí vždy a je možné ji editovat pomocí ikony tužky. Druhý typ poznámky je určen pro konkrétní matriku v daném záznamu. Tato poznámka se zobrazí až při rozkliknutí doporučených matrik. Rovněž se dá

[10] Komponenta: samostatně znovupoužitelný objekt

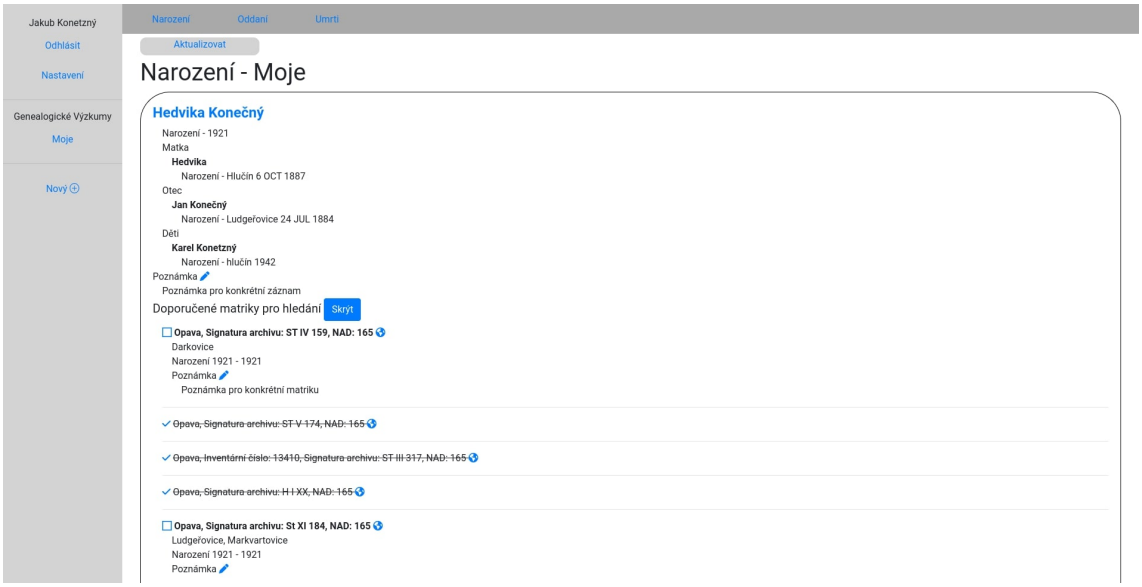
editovat pomocí ikony tužky. Vzhledem k tomu, že se v obou případech jedná o stejnou komponentu, u které se parametrem určuje, zda se jedná o poznámku pro osobu či matriku, chovají se a vypadají stejně.



Obrázek 5.7: Webová aplikace - vytváření poznámky

Zobrazení doporučení

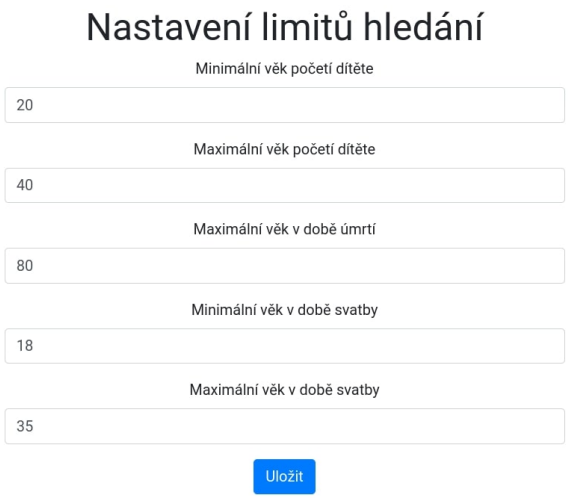
Při zobrazení osoby je zobrazeno její jméno a veškeré informace o všech událostech, které se k dané osobě v souboru GEDCOM nachází. Data o událostech jsou rovněž zobrazena o rodičích a dětech dané osoby, pokud je máme k dispozici. Vzhledem k tomu, že doporučených matrik může být opravdu hodně, jsou ve výchozím stavu skryté a pro zobrazení je nutné kliknout na tlačítko *Zobrazit*. V horním menu můžeme přepínat mezi chybějícími událostmi narození, oddání a úmrtí. Pokud dojde ke změně GEDCOM souboru, je možné aktualizovat data pomocí tlačítka *Aktualizovat* v levém horním rohu. Pokud data aktualizujete, znovu se spustí skript na doporučování matrik. Pokud již událost nebude považována za chybějící, tak osoba nebude zobrazena. Můžou přibýt nebo být odebrány doporučení matrik. Poznámky a ručně provedené úpravy budou zachovány.



Obrázek 5.8: Webová aplikace - zobrazení doporučených matrik

Nastavení limitů

Pokud chce uživatel změnit limity, které se využívají pro doporučování matrik, tak může. Slouží k tomu odkaz *Nastavení limitů hledání* v *nastavení*. Pokud byla změna provedena až po nahrání souboru GEDCOM, je potřeba tento soubor aktualizovat.



Obrázek 5.9: Webová aplikace - nastavení limitů

5.5 Nasazení webové aplikace

Díky využití GitHub Actions bylo možné zautomatizovat nasazování webové aplikace. To znamená, že kdykoliv byl nějaký commit nahrán do main větve, spustil se na serveru skript, který stáhl novou verzi aplikace, vytvořil nový docker image a pokud nenastala žádná chyba, tak automaticky nový image spustil.

5.6 Testování webové aplikace

Testování aplikace bylo prováděno postupně. Vždy, když byla implementována nová funkce, tak byla následovně otestována. Díky tomu, že se veškeré změny v main větvi pomocí GitHub Actions během několika desítek sekund objevily i na serveru, bylo možné změny testovat i tam. K testování byly využity převážně nástroje přímo integrované v Nette. Konkrétně nástroj Tracy, který pokud dojde k chybě, zobrazí o chybě informace, které pomůžou s jejím odstraněním. Pokud se jedná o nesprávný název metody, dokáže poměrně dobře odhadnout, jakou metodu jsme chtěli použít. Dále poskytuje velice užitečnou metodu *bdump*, která nám za běhu aplikace dokáže zobrazovat data o proměnných a objektech.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat webovou aplikaci, která by uživatelům pomohla při hledání chybějících genealogických událostí. Díky této webové aplikaci se uživatelé nemusí snažit odhadnout, ve které matrice by chybějící záznamy mohly být, ale mohou využít doporučení a vést si k události poznámky.

Byla zpracována data o matrikách a územích, které mezi sebou byly propojeny pomocí relační databáze.

Byla vytvořena databáze, která uchovává data o matrikách, jejich obsazích, jednotlivých GEDCOM souborech uživatelů a územích.

Webová aplikace byla implementována za pomocí PHP 8.0 a Nette frameworku. Ten se ukázal jako velice užitečný a velmi zjednodušil implementaci webu. Dále bylo využito JavaScriptu a AJAXU, díky čemuž celá aplikace působí svižněji.

Pomocí docker-compose lze jak z databáze, tak u webové aplikace vytvořit docker image a je tak možné celou aplikaci na jakémkoliv zařízení velice jednoduše zprovoznit.

Díky této práci jsem se naučil používat Nette framework a kontejnerizaci aplikací pomocí Dockeru. Získal jsem hodně zkušeností a zlepšil se v návrhu a implementaci jak webových aplikací, tak relačních databází.

Literatura

[1] David Grudl. *Tracy*. Vid. 2022-02-14. Dostupné z: https://latte.nette.org/cs/guide#.

[2] Docker. *Docker Documentation*. Vid. 2022-07-04. Dostupné z: https://docs.docker.com/.

[3] gedcom.org. *GEDCOM Specifications*. Vid. 2022-01-20. Dostupné z: https://www.gedcom.org/gedcom.html.

[4] Hernandez, M. J. *Návrh databází*. Grada, 2014.

[5] id-sign. *Výpočet vzdálenosti z GPS souřadnic*. Vid. 2022-03-03. Dostupné z: https://www.id-sign.com/poradna/vypocet-vzdalenosti-z-gps-souradnic.

[6] Institut pro veřejnou správu Praha. *Registr územní identifikace, adres a nemovitostí*. Vid. 2022-02-23. Dostupné z: https://www.institutpraha.cz/obj/obsah_fck/egon/pdf_programy/ruian.pdf.

[7] Ministerstvo vnitra České republiky. *Vydávání matričních dokladů a doslovných výpisů z matričních knih*. Vid. 2021-01-10. Dostupné z: https://www.mvcr.cz/migrace/docDetail.aspx?docid=21814471.

[8] Nette Foundation. *Píšeme první aplikaci!* Vid. 2022-02-28. Dostupné z: https://doc.nette.org/cs/quickstart/getting-started.

[9] Nette Foundation. *Slovníček pojmů*. Vid. 2022-03-08. Dostupné z: https://doc.nette.org/cs/glossary#toc-model.

[10] Nette Foundation. *Začínáme s Latte*. Vid. 2022-02-13. Dostupné z: https://latte.nette.org/cs/guide#.

[11] Nixon, R. *Learning PHP, mysql, JavaScript, CSS ET HTML5: A step-by-step guide to creating dynamic websites*. O'Reilly, 2014.

[12] Statista. *Most used programming languages among developers worldwide, as of 2021*. Vid. 2022-02-13. Dostupné z: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/.

[13] w3techs. *Historical trends in the usage statistics of server-side programming languages for websites*. Vid. 2022-02-10. Dostupné z: https://w3techs.com/technologies/history_overview/programming_language.

[14] Wikipedia, The Free Encyclopedia. *Matrika*. Vid. 2022-07-07. Dostupné z: https://cs.wikipedia.org/wiki/Matrika.

[15] Česká genealogická a heraldická společnost v Praze. *Digitalizace v archívech*. Vid. 2022-07-12. Dostupné z: http://www.genealogie.cz/aktivity/digitalizace/.

Seznam příloh

Příloha A

Obsah přiloženého paměťového média

src/web/	veškeré zdrojové kódy k webové aplikaci
src/db/	sql skript pro vložení veškerých potřebných dat do databáze
doc/	veškeré soubory pro technickou zprávu
src/matriky/	všechna potřebná data a skripty pro zpracování dat matrik
src/uzemi/	všechna potřebná data a skripty pro zpracování dat území

Příloha B

Instalace

Pokud se rozhodnete spustit aplikaci lokálně, v této příloze najdete stručný návod, jak na to. Veškeré zde uvedené příkazy je potřeba spouštět ve složce. *src/web/*

Prerekvizity

- composer >= 2.3
- Docker >= 20.10
- docker-compose >= 3.0

Instalace závislostí

Veškeré potřebné závislosti se nainstalují pomocí Composeru příkazem

```
composer install
```

Veškeré potřebné závislosti se nainstalují do složky *vendor/*

Vytvoření a spuštění Docker image

Vytvoření docker-compose image:

```
docker build . -t web
docker-compose build
```

Spuštění:

```
docker-compose up -d
```

V tuto chvíli běží aplikace na http://localhost:8000/ a phpMyAdmin na http://localhost:8080/. Aplikace ještě ale není plně funkční, je potřeba importovat data do databáze.

Vypnutí:

```
docker-compose down
```

Oprávnění

Dále je potřeba spustit tento příkaz

```
sudo chmod 777 -R ../web
```

Databáze

Importovat veškerá potřebná data lze pomocí tohoto příkazu:

```
docker exec -i $(docker-compose ps -q db) mysql -f --reconnect -uroot
-ppassword bp < db/bp.sql
```

Přihlašovací údaje

Přihlašovací údaje do databáze jsou:

```
Jmeno: root
Heslo: password
```