VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

MODELOVÁNÍ NA ZÁKLADNĚ DAT Z ARCHIVÁLIÍ

MODELLING ON HISTORICAL DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DANIEL PÁTEK**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. ZBOŘIL FRANTIŠEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Pátek Daniel**
Program: Informační technologie
Název: **Modelování na základně dat z archiválií**
Modelling on Historical Data
Kategorie: Modelování a simulace
Zadání:

1. Seznamte se se současnými systémy, které mají vztah ke zpracování historických dat z archiválií. Také prostudujte současný stav implementace projektu MoragenGUI.
2. Navrhněte systém, nebo vylepšení stávajícího systému, který by umožňoval na základě takto opsaných dat vytvářet sociální modely. Vymezte role osob, které se pro jednotlivé typy záznamů objevují a navrhněte, jak lze uživatelsky přívětivě tyto role přisuzovat objektům vytvářeného modelu.
3. Realizujte takový systém a pro poskytnutá data (v rozsahu 1 farnosti a 200 let) vytvořte několik modelů, například rodokmenů některého z rodů.
4. Začleňte tento systém do systému DEMoS, který je vyvíjen na FIT. Diskutujte dosažené výsledky a navrhněte další možná rozšíření takového systému.

Literatura:

- Fellegi, I.,P., Sunter, A.,B.: A theory for record linkage, 1969

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Zbořil František, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Cílem této práce je navrhnout a implementovat webovou aplikaci, která umožní zobrazení a úpravu genealogických modelů. Tyto modely jsou vytvořené pomocí programu umožňujícího jejich automatické sestavení z dostupných matričních záznamů. Velký důraz je v rámci řešení věnován propojení webové aplikace s tímto programem. Aplikace je vytvořena pomocí systému Node.js, psána v jazyce JavaScript s využitím jeho knihovny React.

Webová aplikace zprostředkovává uživateli přístup ke zmíněnému programu a umožňuje mu zobrazení a editaci výstupů tohoto programu. Zaměřuje se na uživatelskou přívětivost a jednoduchost při ovládání aplikace. Součástí řešení je také integrace do existujícího systému, který uživateli poskytuje možnost přepisu údajů z archiválií a spravuje komunitní databázi.

Abstract

The aim of this thesis is to design and implement a web application that allows the display and editing of genealogical models. These models are created using a program that permits their automatic assembly from available registry records. A great focus of the solution is the connection of the web application with this program. The application is developed using Node.js, written in JavaScript using its React library.

The web application provides the user with access to the mentioned program and allows him to view and edit the output of this program. It is especially concerned with the user-friendliness and simplicity of the application. The solution also includes integration into an existing system which provides the user with the possibility of transcribing data from archives and manages a community database.

Klíčová slova

genealogie, archiválie, rodokmen, matrika, webová aplikace, React, NodeJS

Keywords

genealogy, archival material, family tree, register, web application, React, NodeJS

Citace

PÁTEK, Daniel. *Modelování na základně dat z archiválií*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Zbořil František, Ph.D.

Modelování na základně dat z archiválií

Prohlášení

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

.....
Daniel Pátek
10. května 2021

Poděkování

Mé poděkování patří Doc. Ing. Františku Zbořilovi, Ph.D. za odborné vedení, cenné rady a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval.

Obsah

Kapitola 1

Úvod

Úplný počátek vzniku matričních záznamů můžeme spojit s nezastupitelnou rolí křesťanské církve ve středověku, jak píše Josef Peterka. V té době vznikaly první církevní seznamy pokřtěných. Jejich vedení nebylo jednotné a záviselo především na ochotě a schopnostech daného faráře. Nicméně již v průběhu 16. století byla farám stanovena povinnost tyto záznamy vést. Konkrétně se jednalo o záznamy sezdání a křtu. Matriky o úmrtích se povinnosti vedení dočkaly o století později. Za vlády Josefa II. byla dále stanovena jednotná forma matrik a záznamů v nich. S krátkou přestávkou v druhé polovině 20. století, kdy komunistický režim v Československu vedení matrik předal národním výborům, se tak matriky staly veřejnými listinami. [17]

V současnosti se lidem otevřela možnost vyhledat si své předky a pátrat po informacích o svém původu online. To zejména díky internetu a rozsáhlému snažení nejenom zemských archivů o digitalizaci všech dostupných matrik. Zájemci potřebují jen počítač a schopnost přečíst staré matriční texty. I z tohoto důvodu je genealogické pátrání mezi lidmi stále oblíbenější koníček. Toho si začaly všímat i různé společnosti, které se těmto lidem snaží zprostředkovat co možná nejlepší systémy pro správu jejich mnohdy velmi dlouho sestavovaných rodokmenů. Tyto systémy fungují samostatně a nijak nereflektují využití matrik nebo jednotlivých matričních záznamů pro práci s rodokmeny.

Z tohoto důvodu v rámci Fakulty informatiky na Vysokém učení technickém v Brně vzniká systém, který svou vizí posouvá genealogické pátrání ještě o úroveň výš. Dostal jméno DEMoS a jeho cílem je vznik komunitní genealogické databáze matričních záznamů a tvorba genealogických modelů. Pro člověka se zájmem o genealogii to znamená, že bude moct přispívat do sdílené databáze přepisem matričních záznamů, následně v nich vyhledávat nebo z těchto záznamů jedním kliknutím vygenerovat rodokmen.

Vývoj výše zmíněného systému stále pokračuje a jeho součástí bude i tato práce. Ta si klade za cíl vytvořit webovou aplikaci, která bude umožňovat zobrazení a úpravu sestavených rodokmenů. Uživatel si tak bude moct pouze na základě matričních záznamů nechat sestavit genealogický model. Tento model může obsahovat chyby, a tudíž je uživateli umožněno model měnit a nechat si jej vygenerovat s reflektovanými změnami znovu. Součástí řešení bude i co možná nejlepší integrace této webové aplikace do systému DEMoS.

První kapitola se věnuje analýze již existujících systému pro práci s rodokmeny nebo matričními záznamy. V druhé kapitole se konkrétně představí systém DEMoS a související programy, z kterých tato práce bude vycházet. Další kapitoly se postupně zaměřují na návrh, implementaci a zhodnocení webové aplikace. Na závěr jsou také uvedeny možné směry, jak postupovat při dalším vývoji.

Kapitola 2

Analýza existujících řešení

V současné době existuje několik programů, které se svými funkcemi blíží vyvíjenému systému. Nicméně je důležité, že žádný funkční systém doposud nepodporuje takovou funkcionalitu, kterou bude disponovat kompletní systém DEMoS.

Uživatel je v současnosti nucen pro přepis údajů z matrik a následného vytvoření rodokmenu využít nejméně dva různé systémy. Co se týče samotného přepisu údajů a tvorby genealogické databáze, nejblíže se tomuto problému dostává *FamilySearch Indexing*. Tvorbou samotného rodokmenu se zabývá některý ze systémů, které budou konkrétně rozebrány v následujících odstavcích. Většina takových systémů funguje na velmi jednoduché bázi, kdy uživatelé klikáním postupně přidávají osoby do rodokmenu, který poté mohou upravovat. Taková aplikace jistě nedisponuje funkcemi, jež jsou vyvíjeny v rámci projektu DEMoS a pro něž bude také sloužit vznikající webová aplikace.

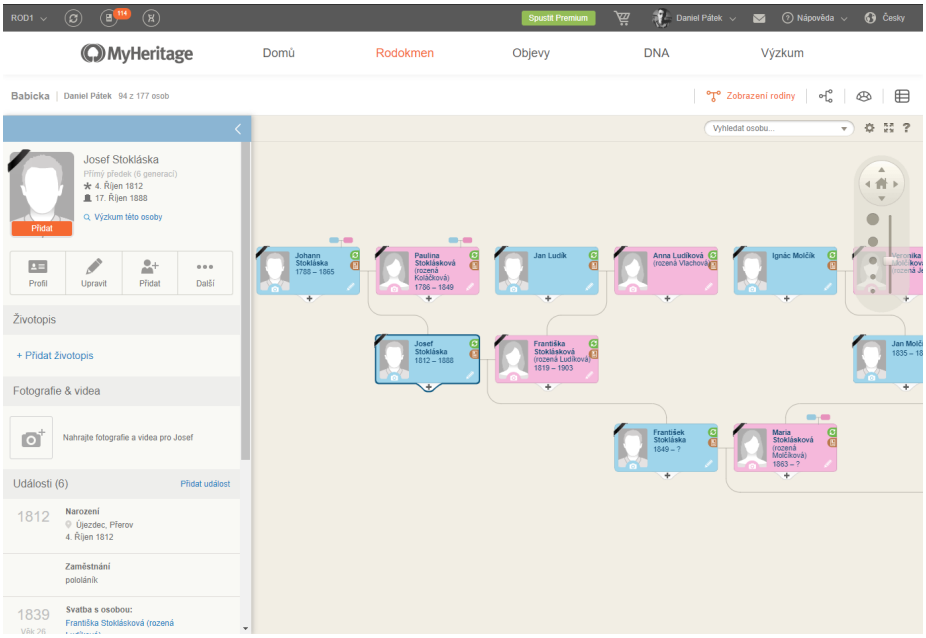
Nutno podotknout, že zde uvedu pouze ty nejpoužívanější systémy v rámci české genealogické komunity.

2.1 MyHeritage

Tato společnost působí ve světě již od roku 2003. V době psaní této práce mají uživatelé k dispozici téměř 13 miliard historických záznamů a svůj rodokmen si zde spravuje více než 60 milionů lidí. [13] Společnost se zaměřuje především na propojování osob v rodokmenu s historickými záznamy, přičemž uživatelům neumožňuje historické záznamy editovat či přidávat. Důraz webové aplikace je kladen nejvíce na rodinné stránky. Členové těchto stránek spravují společný rodokmen, sledují rodinné statistiky a mohou i plánovat budoucí události. Odzkoušený systém této společnosti umožňuje efektivně a přehledně vytvářet rodokmeny nebo je upravovat. Obrázek 2.1 obsahuje ukázku, jak prostředí pro správu rodokmenu vypadá. K dispozici jsou také rodinná alba přidaných fotografií nebo třeba možnost nechat si vygenerovat různé diagramy.

Je nutné zmínit, že služba je placená. Propojovat historické záznamy může jen uživatel, který zaplatil roční příspěvek. Je k dispozici omezená verze bez poplatku, která umožňuje pouze správu rodokmenu. Takto vytvořený rodokmen je omezený do velikosti 500 MB dat nebo 250-ti osob.

Lidé mohou za poplatek také využít DNA test, jehož účelem je odhalit jejich etnický původ a najít nové příbuzné. *MyHeritage* není jedinou společností, která tento test nabízí. DNA test si uživatelé mohou objednat také na webu společnosti *Ancestry*.



Obrázek 2.1: Ukázka z prostředí webové aplikace *MyHeritage*

2.2 Ancestry

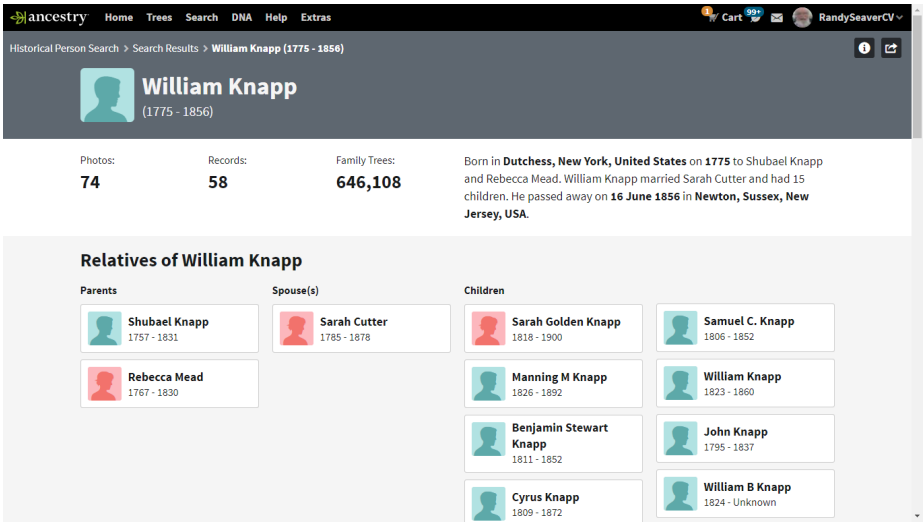
Další známou genealogickou webovou aplikací je *Ancestry*. Společnost se chlubí 60 miliardami historických záznamů, ve kterých mohou její uživatelé vyhledávat. Podobně jako u *MyHeritage* je i zde kladen důraz na propojování historických záznamů se záznamy v rodokmenu. Samozřejmě je i zde dostupná správa rodokmenů a jejich úprav. Snímek z webového prostředí je možné najít pod číslem 2.2. Webová aplikace se zaměřuje především na občany USA. Z tohoto důvodu nepočítá například s českou variantou příjmení u vdané ženy. Co se týče přístupu, celá webová služba je placená a není k dispozici zdarma ani v omezeném režimu. [2]

Součástí aplikace je také její desktopová verze *Family Tree Maker*. V minulosti byla tato aplikace málo využívaná a společnost se ji rozhodla dále nepodporovat. Nyní ale spolupracuje s firmou *Software MacKiev*. Společně dál provozují aplikaci se stejným jménem pro systémy Windows i Mac.

Pro přehlednost je dobré uvést, že existuje také desktopová aplikace pro tvorbu rodokmenu se shodným jménem *Ancestry*, která ale nemá s výše zmíněnou webovou aplikací nic společného. Vytvořil ji český autor Martin Doležal. Bohužel je tato aplikace již delší dobu neaktualizovaná. [1]

2.3 FamilySearch

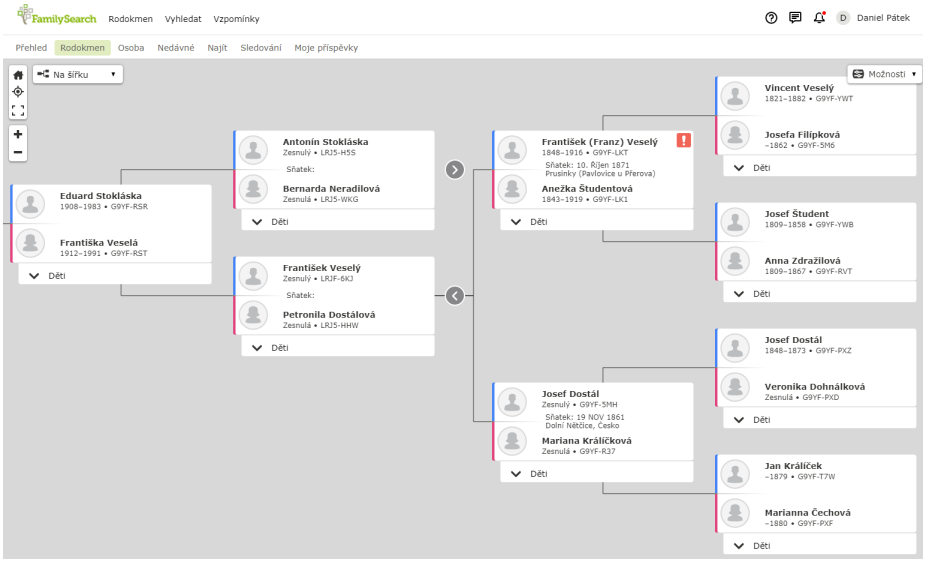
Tato webová aplikace funguje díky sponzorství od Církve Ježíše Krista Svatých posledních dní, v českém prostředí známou jako Mormoni. Jedná se o celosvětovou databázi rodokmenů, jmen i historických záznamů. Dlouhodobým cílem této společnosti je vytvořit jeden ucelený rodokmen lidstva obsahující všechny osoby. [8] Pro takový ambiciózní cíl společnost sází na komunitu lidí, kteří tento rodokmen mohou doplnit svým, upravit již přidaný nebo hledat a spojovat duplicitně přidané rodokmeny či osoby.



Obrázek 2.2: Ukázka z prostředí webové aplikace *Ancestry* [1]

Velkou výhodou této aplikace je její přístupnost. Veškeré záznamy i rodokmeny jsou vloženy zdarma pro všechny, stejně jako možnost vytvoření vlastního rodokmenu a jeho editace. Na druhou stranu zde uživatel může spravovat pouze jeden rodokmen. Ukázka běžného rodokmenu v prostředí *FamilySearch* se nachází na obrázku 2.3. Zajímavostí je, že v důsledku již zmíněného sponzorství církví nemůže do rodokmenu uživatel přidat dvě osoby stejného pohlaví jako partnery.

FamilySearch umožňuje uživateli také dobrovolně přidávat historické záznamy. Tento projekt nese název *FamilySearch Indexing*. Vzhledem k tomu, že se jedná mezi podobnými webovými aplikacemi o unikátní produkt, bude mu v této práci věnováno více prostoru.



Obrázek 2.3: Ukázka rodokmenu v prostředí webové aplikace *FamilySearch*

[1] Obrázek převzat z geneamusings.com.

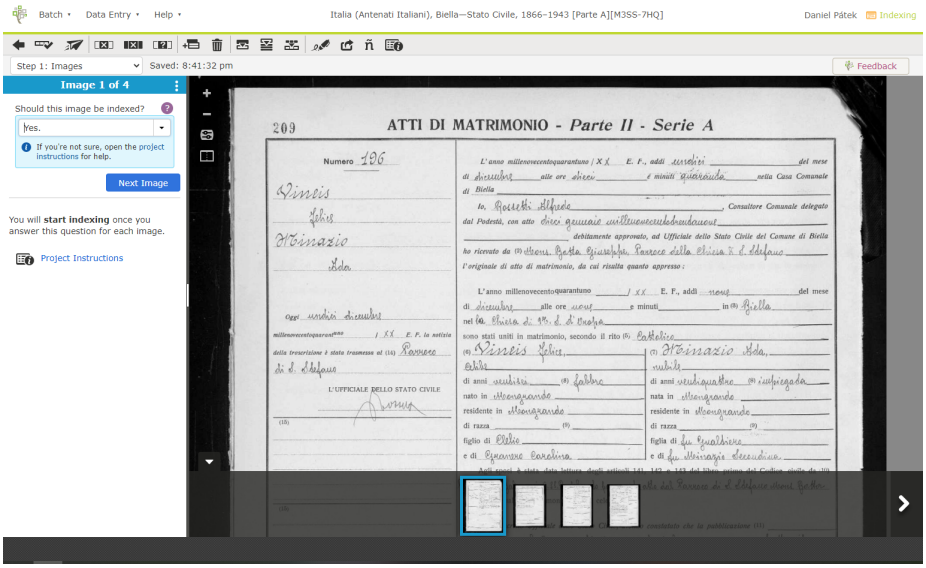
2.3.1 FamilySearch Indexing

Jedná se o projekt, který patří do správy webové aplikace *FamilySearch* a poskytuje uživatelům možnost přepsat historický záznam do indexovatelné podoby, aby mohl být vyhledán. Společnost tak dává uživatelům jedinečnou možnost podílet se na digitalizaci historických dokumentů.

Nejčastěji se takto přepisují záznamy z matrik, ale může se jednat také o sčítací operáty, vojenské dokumenty nebo vládní listiny. Tyto záznamy jsou poté komukoliv dostupné a uživatelé v nich mohou pátrat po více informacích o svým předcích. Navzájem si tak pomůžou a zajisté jim takovéto záznamy ulehčí práci. Vzhledem k tomu, že tento projekt funguje již od roku 2006 a je v genealogické komunitě oblíbený, podařilo se takto vytvořit již více než miliardu vyhledatelných záznamů.

Každý dobrovolník si nejprve vybere projekt, do kterého chce přispět, a následně přepisuje zobrazené digitalizované dokumenty. Jeho přepis podléhá další kontrole jedním ze zkušenějších uživatelů. Ukázku přepisu historického záznamu vystihuje obrázek 2.4. Je možné si zvolit projekty z celého světa, v době psaní tohoto textu jich mají dobrovolníci k dispozici více než sto. V minulosti probíhaly projekty také z území České republiky a Slovenka, kdy se takto přepisovaly některé matriky. [9]

Poměrně zásadní fakt pro tuto práci je, že *FamilySearch Indexing* z přepsaných záznamů žádným způsobem nevytváří genealogické modely. Tuto možnost ponechává na jednotlivcích, kteří je mohou využít při tvorbě rodokmenu.



Obrázek 2.4: Ukázka přepisu historického záznamu v prostředí *FamilySearch Indexing*

Kapitola 3

Současný stav řešení

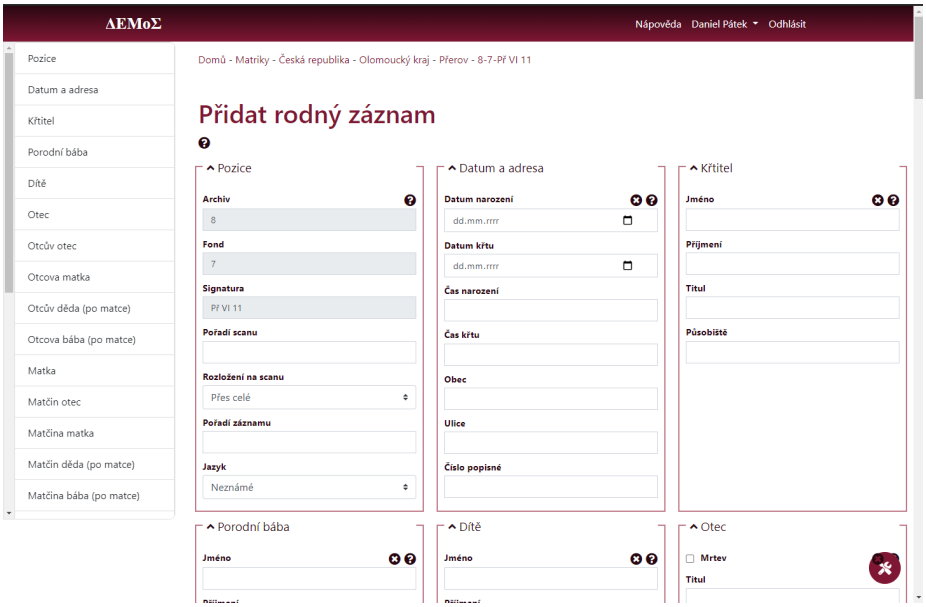
V rámci projektu DEMoS v současnosti existuje několik programů. Dva z nich pro tuto bakalářskou práci představují důležitý odrazový můstek. Jedná se o programy *Moragen* a jeho nadstavbu *MoragenGUI*. Tato bakalářská práce bude vycházet ze znalostí a informací získaných z těchto projektů. Následující odstavce se věnují jejich podrobné analýze.

3.1 DEMoS

Na tomto projektu spolupracuje Fakulta informačních technologií VUT s brněnskou Masarykovou univerzitou, konktrétně s Ústavem pomocných věd historických a archivnictví Filozofické fakulty. Jedná se o komunitní **genealogickou databázi**, její zkratka „*DEMoS*“ znamená *Database of Early Modern Sources*. Cílem tohoto projektu je zpřístupnění prostředí pro přespis historických dokumentů do digitalizované podoby. Těmito dokumenty mohou být matriky, ale také pozemkové knihy, sčítací operáty nebo urbáře. Dalo by se říci, že záznamy v této databázi budou mít dvojí podobu. Jednak budou uloženy ve formátu věrného přepisu (transliterace) z archivního záznamu a jednak budou uživatelsky nebo počítačově převedeny do normalizované podoby pro potřeby vyhledávání.

Slovo *komunitní* z předchozího odstavce znamená, že se do přepisu záznamů může zapojit každý uživatel podobně jako u zmíněného *FamilySearch Indexing*. Algoritmus ale funguje s menším rozdílem, kdy přepsané záznamy nečekají na kontrolu od uživatele vyšší úrovně, ale jsou ihned zveřejněny. Případnou revizi poté může provést důvěryhodnější uživatel. Ukázka přepisu rodného záznamu se nachází na obrázku 3.1

Databáze dále počítá s tím, že bude uchovávat také rodokmen (přesněji *sociální model*) všech osob. Tento sociální model nebude uchovávat pouze příbuzné osoby tak, jak je tomu u běžných rodokmenů, ale jeho součástí budou i svědci, kmotři nebo porobní báby - tedy osoby, které se v archivních záznamech také vyskytují. Systém ve výsledku nabídne mnohem podrobnější model s mnoha možnostmi vyhledávání. [4]



Obrázek 3.1: Ukázka přepisu rodného záznamu v prostředí databáze *DEMoS*

3.2 Moragen

Tento program vytváří genealogický model z matričních záznamů. Je napsaný v jazyce *Prolog* a konkrétně se mu věnuje publikace *Algorithmic creation of genealogical models.* [24] Program předpokládá existenci tří typů matričních záznamů:

1. Záznam narození dítěte případně jeho křtu
2. Záznam svatby
3. Záznam smrti případně pohřbu

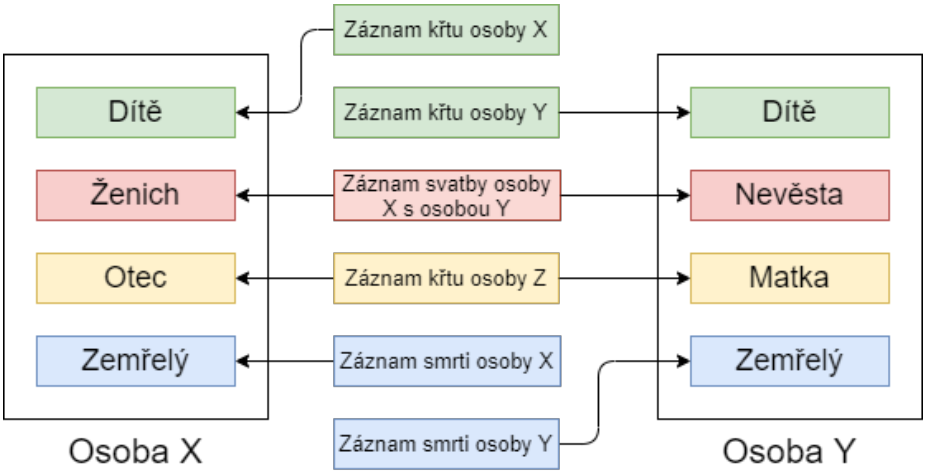
Každý záznam obsahuje data, se kterými program následně pracuje. Nejdůležitějším typem záznamu je první z jmenovaných, jelikož v něm program objeví informace o třech lidech najednou, tedy o narozeném dítěti, jeho matce a jeho otci. Zbylé dva typy záznamů samozřejmě zpřesňují výpočet genealogického modelu a také díky nim program dokáže odhalit větší množství chyb.

3.2.1 Výpočet genealogického modelu

Výpočet genealogického modelu probíhá v několika krocích. Nejprve se z dodaných záznamů vytvoří množina všech **identit**, které se v záznamech nacházejí. Identitou se rozumí role osoby v některém ze záznamů. Takovou rolí může být dítě, otec, matka, ženich, nevěsta nebo zemřelý.

V dalším kroku se program snaží tyto identity co nejlépe propojit. Vzniká díky tomu množina všech **osob**. Jedna takto vzniklá osoba se dá charakterizovat jako množina identit, z nichž každá představuje jednoho a toho samého člověka. Každá osoba může mít několik různých identit. Vzniku identit a osob ze záznamů se podrobně věnuje obrázek 3.2. *Moragen* samozřejmě propojuje osoby i mezi sebou a tvoří tím rodokmen, přesněji genealogický model.

Posledním krokem výpočtu je vzniklý genealogický model zkontrolovat. Při nalezení **chyby** se program pokusí model opravit a najít jiné řešení, nicméně pokud odpovídající řešení nenajde, vloží tento problém do seznamu chyb.



Obrázek 3.2: Diagram vzniku identit a osob ze záznamů v programu Moragen

3.2.2 Vstupy a výstupy programu

Program *Moragen* pracuje s matričními záznamy. Na vstupu proto očekává soubor právě s těmito daty, který má povinnou příponu *.gpm*. Vzhledem k tomu, že je *Moragen* napsán v jazyce *Prolog*, vstupní soubor má formát **predikátů**, kdy každý predikát je na novém řádku. Tyto predikáty se rozlišují na typy *brecord* pro narození, *mrecord* pro svatbu a *zrecord* pro smrt. Každý predikát tedy představuje jeden záznam a obsahuje kromě důležitého ***ID* záznamu** další potřebné informace. Těmi mohou být datum, jména a příjmení osob i jejich role v záznamu. Zde jsou uvedeny příklady těchto predikátů:

```
brecord(91,13,5,1688,'DOBRÉ POLE','JAN','CAHEL','JAN',",'ANNA',",",",",").
mrecord(2010,6,2,1839,'MARTIN','GREGOR','ONDREJ','MARIE',",23,'MARIE',
'JURDIC','ANTONIN',",",",24,",").
zrecord(399,1,2,1772,",'BARTOLOMEJ','KOSCAK',",",57,'FRL',0,0,52,",").
```

K výstupům tohoto programu řadíme tři soubory. Jedná se o množinu identit, množinu osob a množinu chyb. Každá množina představuje jeden soubor. Všechny tyto soubory mají koncovku *.mrg*. Množinu identit obsahuje soubor *roles.mrg*. Příklad takového predikátu se nachází na následujícím řádku.

```
identity(16,role(9,father,4,3,[1645,1645],'Ondrej','Bezrouk',[]),null,
[identity(16,100)]).
```

Dalším souborem je *coreFamilies.mrg*, který obsahuje seznam osob respektive spojených identit. Znovu je uvedeno, v jakém formátu se tato data vyskytují, podle následujícího příkladu.

```
coreIdt(22,[22,25,26]).
```

Posledním výstupním souborem tohoto programu je soubor s chybami, které vznikly v průběhu generování. Nese název *semanticCheck.mrg* a obsahuje predikáty ve formátu jako následující příklad.

```
mw_errors(45,17,[rec(628992,withMother,3856,3856)]).
```

Aby program umožňoval uživateli měnit vypočtený model, umí pracovat se souborem *updates.mrg*. Tento soubor slouží jako seznam změn, které ve vypočteném modelu uživatel ručně provede. Při dalším výpočtu program s těmito změnami počítá a vytvoří nový model, který tyto změny reflektuje. Informace v souboru *roles.mrg* je tedy možné považovat za vstupně-výstupní. Jeho obsah se také skládá z predikátů.

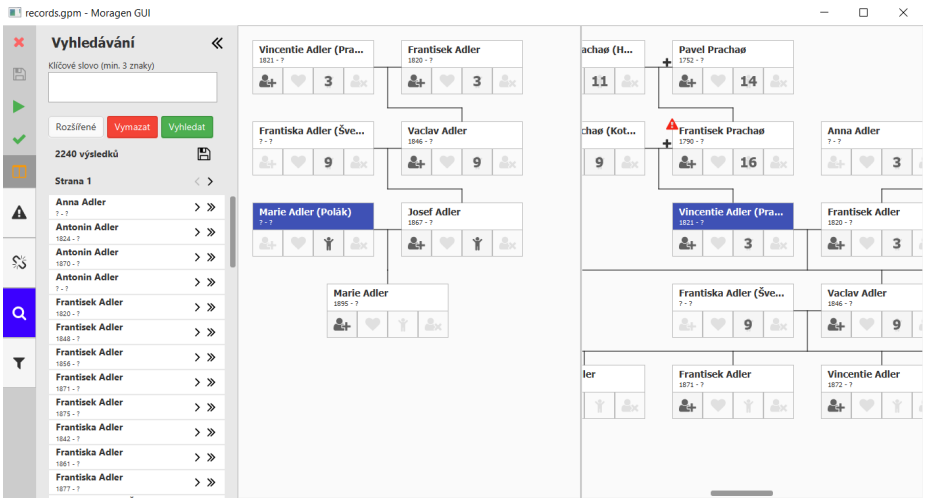
Prvním typem je predikát `mw_force`. Udává, jaká identita byla uzamčena ke které osobě. Další možností správy výpočtu modelu je predikát `mw_exclude`, jenž se využívá pro zákaz vztahu mezi danou identitou a osobou. Jako poslední možný predikát zde figuruje `submodel`. Ten umožňuje tvorbu a uložení sociálního modelu. Pro přehlednost jsou níže uvedeny příklady těchto predikátů.

```
mw_force(68,767).
mw_exclude(19,349).
submodel('rodokmen příjmení Hanák',["","'Hanák',"","",-1,-1,-1,-1,"","]).
```

3.3 MoragenGUI

Jedná se o navazující projekt na zmíněný program *Moragen*. Účelem tohoto projektu bylo vytvořit aplikaci, která zobrazuje genealogické modely vypočtené programem *Moragen*. Modely je možné také upravovat a takto upravené je znovu přepočítat. Dále program umožňuje uživateli uložit rozpracovaný projekt.

Aplikace je napsaná v jazyce *Java* a je určena pro desktopové operační systémy. Pro své uživatelské rozhraní používá nadstavbu *JavaFX*. Aby aplikace mohla zobrazit genealogický model, musí mít k dispozici výsledky programu *Moragen*. Ukázka uživatelského rozhraní této aplikace je k nalezení na obrázku 3.3. [22]



Obrázek 3.3: Ukázka zobrazení genealogického modelu v aplikaci *MoragenGUI*

3.3.1 Funkce programu

Hlavní a nejdůležitější funkcí této aplikace je již zmíněné uživatelsky přívětivé zobrazení vypočteného modelu. Tuto funkcionalitu program vykoná vždy k určité osobě. To znamená, že si uživatel zvolí osobu, jíž rodokmen chce v aplikaci zobrazit. Vygenerovaný rodokmen osob je možné zobrazit i ve dvou oknech vedle sebe.

Aplikace podporuje také ruční spuštění výpočtu a s tím související aktualizaci rodokmenu a kontrolu chyb pomocí jednoho tlačítka. Nutno podotknout, že k tomuto kroku je potřeba mít nainstalovaný překladač jazyka *Prolog* a přirozeně musí být k dispozici samotný program *Moragen*.

Pro úpravu vygenerovaného modelu je možné přesouvat identity jednotlivých osob k jiným osobám. Osoby je možné také uzamknout a označit je takto za správné nebo naopak zakázat vztah mezi osobami. Program dovoluje také tvorbu vlastního modelu osob, které uživatel vybere.

Co se týče další funkcionality, aplikace umožňuje vyhledávat a filtrovat jednotlivé osoby, zobrazit seznam osob s chybami nebo seznam identit, které nebyly přiřazeny k žádné osobě.

3.3.2 Vstupy a výstupy programu

Vstupy programu jsou shodné s výstupy programu *Moragen*, které již byly uvedeny a vysvětleny. Výstupem tohoto programu je soubor *updates.mrg*. Tento soubor můžeme považovat za vstupní i výstupní, jelikož ho program *MoragenGUI* používá jako soubor pro ukládaní změn v projektu. Jeho struktura již také byla popsána výše.

3.4 Zhodnocení současného stavu řešení

Aby mohla tato práce pokračovat, je nutné shrnout nejdůležitější skutečnosti, které uživatel musí splnit, aby mohl využívat generování a zobrazování genealogických modelů z historických záznamů. Následně je potřeba z těchto poznatků vyjít a v rámci dalšího vývoje umožnit uživateli co možná nejjednodušší cestu k využívání zmíněných funkcí. Vývoji takovéto webové aplikace se budou věnovat následující kapitoly.

Každý, kdo chce nějakým způsobem generovat genealogické modely, si musí nejprve určit, z jakých historických záznamů bude čerpat data. Pro vznik rodokmenu je nejlepším řešením použití matričních záznamů. Značným cílem systému *DEMoS* je umožnit každému využití dat z komunitní databáze přepsaných matričních záznamů. Systém takovou možností v současnosti ještě nedisponuje, a proto ji zatím nebude podporovat ani vznikající webová aplikace. Pro potřeby této práce bude využito dostupných vzorkových dat dodaných vedoucím práce.

Za předpokladu, že uživatel disponuje daty přepsaných záznamů, může přejít k použití samotných programů. Musí využít desktopový operační systém a mít nainstalované *JRE* (*Java Runtime Enviroment*) pro spuštění programu *MoragenGUI*. Následně je třeba uložit do počítače program Moragen a pro jeho funkcionalitu nainstalovat překladač jazyka *Prolog* (tím může být například *SWI-Prolog*). Po naplnění všech těchto předpokladů je možné vygenerovat a zobrazit genealogický model.

Je zřejmé, že takovýto postup je náročný a pro běžného uživatele nepohodlný. Zjednodušení uživatelského přístupu patří k hlavním cílům práce. Výsledná aplikace bude fungovat ve webovém prohlížeči v rámci systému *DEMoS* a nebude po uživateli vyžadovat žádné instalace. Mezi výhody takového řešení pak bude patřit zejména uživatelsky jednodušší práce

s programem a snazší aktualizace programů. Velkou výhodou bude také přenositelnost projektů mezi zařízeními, jelikož práci bude možné na webu uložit a později se k ní vrátit, z jakéhokoliv zařízení.

Důležité je zmínit, že jazyk *Java*, ve kterém je napsaná aplikace *MoragenGUI*, disponuje možností fungovat i ve webovém prostředí. V takovém případě by stačilo pouze spustit hotový a existující program v prohlížeči. Bohužel takový postup není vůbec jednoduchý, jelikož aplikace využívá nadstavbu *JavaFX*, která v současnosti nemá téměř žádnou podporu v technologiích, které se zaměřují na vývoj webových aplikacích v jazyce *Java*. Určité návody, jak spustit v prohlížeči aplikaci využívající *JavaFX* dohledatelné jsou, nicméně je na místě zásadní otázka - co se stane v případě, že takový postup již přestane fungovat?

Dalším argumentem proti takovému postupu je fakt, že *Oracle*, společnost vyvíjející jazyk *Java*, oznámila směr jejich budoucího firemního zaměření. Z tohoto dokumentu je patrné, že nadstavba *JavaFX* nebude v příštích verzích jazyka *Java* podporována a jeho podpora bude odebrána i z *Javy* verze 11. Nejlepším řešením se v takovém případě jeví budoucí webovou aplikaci vytvářet v jiném programovacím jazyce, který má vysokou podporu jak ze strany jeho vydavatelů, tak ze strany prohlížečů. [16]

Kapitola 4

Návrh webové aplikace

Při tvorbě webové aplikace je zapotřebí značnou pozornost věnovat požadavkům na její funkčnost a také samotnému návrhu aplikace. Je důležité navrhnout řešení všech částí aplikace a problémů, které s těmito částmi mohou souviset. Následující odstavce se proto tomuto tématu budou věnovat.

Z důvodu přehlednosti je tato kapitola rozdělena do několika podkapitol. V části *Specifikace požadavků* budou uvedeny uživatelské nároky na výslednou aplikaci a její funkčnost. Návrh vzhledu webové aplikace a s ním související grafické prvky budou objasněny v další části *Návrh grafického uživatelského rozhraní*. Poslední podkapitolou bude *Architektura a datový model*. Zde bude specifikován návrh architektury webové aplikace a také objasněno, jakým způsobem bude aplikace pracovat s daty. V poslední kapitole bude projekt zasazen do kontextu systému DEMoS.

Vznikající aplikace dostala jméno *MoragenWeb*. V této práci bude proto tímto jménem označována.

4.1 Specifikace požadavků

Při tvorbě této webové aplikace je nutné si uvědomit, pro koho bude určena. Díky této informaci poté bude analýza požadavků probíhat více cíleně a vývoj se nedostane do situace, kdy bude aplikace funkční pro jinou skupinu uživatelů (například více softwarově vzdělanou) a pro určené uživatele bude velmi náročná nebo irelevantní. Dále je třeba zmínit, že některé ze zmíněných požadavků bude shodných s požadavky pro aplikaci *MoragenGUI*, které její autor Vojtěch Wawreczka uvedl ve své bakalářské práci při jejím návrhu. [24] Některé z nich proto budu v následujících odstavcích parafrázovat.

Aplikace bude sloužit zejména české genealogické komunitě, tedy uživatelům, kteří se zajímají o genealogii a baví je tvorba rodokmenů a dalších genealogických modelů. Tito uživatelé mohou, ale i nemusí být počítačově vzdělaní. Z tohoto důvodu bude nutné vývoj aplikace cílit na intuitivnost a jednoduchost při jejím ovládání. Stejně tak bude aplikace vykazovat efektivní práci při úpravách genealogických modelů nebo při navigaci mezi jednotlivými rodokmeny či osobami.

Každý uživatel musí mít přehled, co dělá. Proto budou v aplikaci uvedeny popisky jednotlivých tlačítek či jiných interaktivních prvků. Tyto popisky budou zobrazeny po najetí a podržení myši na daném prvku, budou proto takzvaně nenásilné a uživatelé, kteří aplikaci budou ovládat déle, se s nimi vůbec nesetkají.

Lze očekávat, že se uživatelé budou chtít k výsledku dostat co nejjednodušší cestou. Proto bude veškerá práce s programem *Moragen* prováděna automaticky na serverové straně. Generování modelu se tak obejde bez potřeby instalace produktů třetích stran, jako je například překladač jazyka *Prolog*. V rámci této práce nebude řešeno uživatelské selektování dat, ze kterých má být model vygenerován. Uživatel si tak zatím bude moci vygenerovat pouze model z ukázkových dat. Očekává se, že v budoucnu dojde k rozšíření tohoto systému a umožnění uživateli zvolit si např. matriku, ze které budou použity přepsaná data pro generování genealogického modelu.

Stejně tak jako aplikace od bc. Wawreczky, i tato webová aplikace má za úkol zobrazit vygenerované genealogické modely a poté umožnit uživateli jejich úpravu a spuštění výpočtu s provedenými změnami. Jedná se tedy o uzamčení identity k osobě když víme, že identita plně odpovídá modelu a je správně umístěná, nebo naopak o restrikci (zákazu vztahu jedné identity a osoby). Důležitou součástí aplikace bude také zobrazování chyb, které generováním vznikly. Uživatel tak bude mít přehled o osobách, které jistě neodpovídají realitě a jsou špatně propojeny nebo vygenerovány. Dále uživatel musí najít seznam nepropojených identit, tedy záznamů, které se programu nepodařilo propojit s žádnými existujícími osobami. Uživatel bude moct tyto identity propojit ručně.

Pro rychlou navigaci mezi jednotlivými osobami bude zajištěno vyhledávání, které bude fungovat na principu *Incremental search*. Jedná se o vyhledávání už v momentě psaní dotazu, dosahuje stejných vyhledávacích rychlostí při méně úhozech do klávesnice. [10] [12] Ve výsledcích budou zobrazeny všechny odpovídající výsledky z genealogického modelu.

V aplikaci *MoragenGUI* bylo přesouvání identit řešeno pomocí *Drag&Drop*, tedy uchopení identity a tažení myší. Tento způsob interakce se ve webových aplikacích používá jen výjimečně, jelikož není pro webové prostředí vhodný a uživatelé na takovou metodu ve webových aplikacích nejsou zvyklí. Rovněž příliš nevyhovuje při použití zařízení s dotykovou obrazovkou. Proto bude možnost propojování osob řešena jiným způsobem, a sice pomocí tlačítka u identity a následného kliknutí na osobu, ke které chci zvolenou identitu připojit. Celý proces bude pro přehlednost doplněn zobrazením doprovodného dialogového okna, které nebude proces propojování identit rušit či do něj vstupovat.

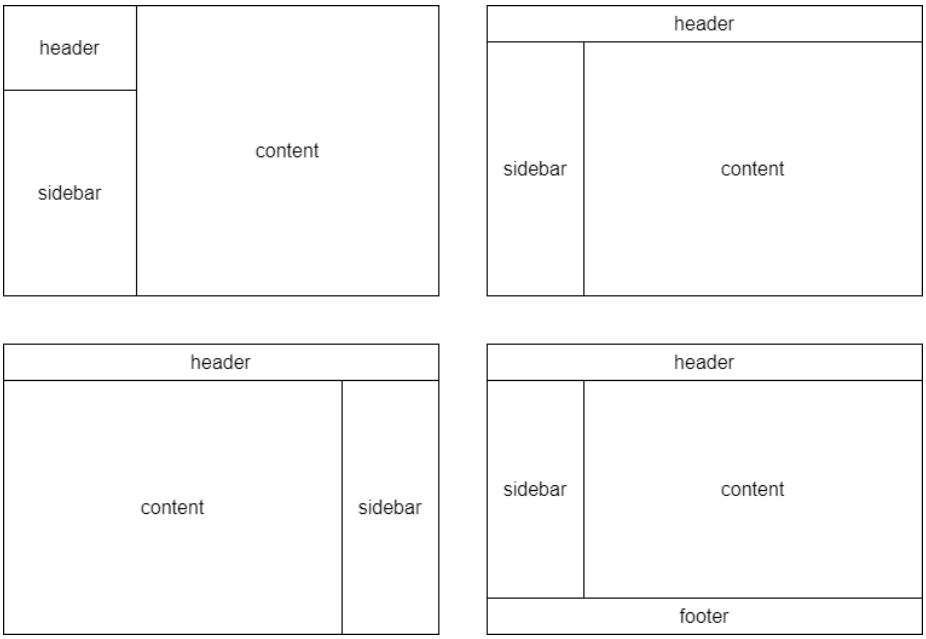
4.2 Návrh grafického uživatelského rozhraní

Jak píše Jakob Nielsen ve své knize o webdesignu - *Bad usability equals no customers*. Webová aplikace musí být především dobře ovladatelná. Stejně tak by měla disponovat svižnou navigací a neobsahovat zbytečné prvky, které by práci s ní jen zpomalily. Zkrátka musí splňovat to, co od ní uživatel čeká. [14]

Od aplikace *MoragenWeb* se očekává návrh funkčního a jednoduchého uživatelského prostředí. Na začátku bylo stanoveno, že aplikace bude jednostránková. To znamená, že nebude podporovat tzv. *routing* (směrování mezi stránkami webu). Stejně tak uživatel aplikace nebude mít k dispozici roletku, nebude moci *scrollovat*. Důvodem je budoucí hlavní element stránky - zobrazený rodokmen. Ten bude moct být uživatelsky transformován a posouván v rámci prostoru na stránce. Společně s rolováním stránky by vznikaly potíže s ovládáním, proto se od takového řešení hned upustilo a roletka nebude na stránce povolena.

4.2.1 Rozložení obsahu

Ze specifikace požadavků bylo zjištěno, jaké funkce bude výsledná aplikace podporovat. Dominantou stránky bude obsahová část, která bude obsahovat vygenerovaný rodokmen. Tato část půjde libovolně transformovat (přibližovat a posouvat), aby si každý uživatel mohl zvolit přijatelnou velikost rodokmenu a mohl si rodokmen v rámci přiblížení prohlížet. Další část bude zobrazovat data rodokmenu, primárně půjde o seznam osob a vyhledávání v tomto seznamu. Důležitou součástí bude ovládací panel, který umožní kontrolu výpočtu, uložení nebo i navigaci v datovém modelu.



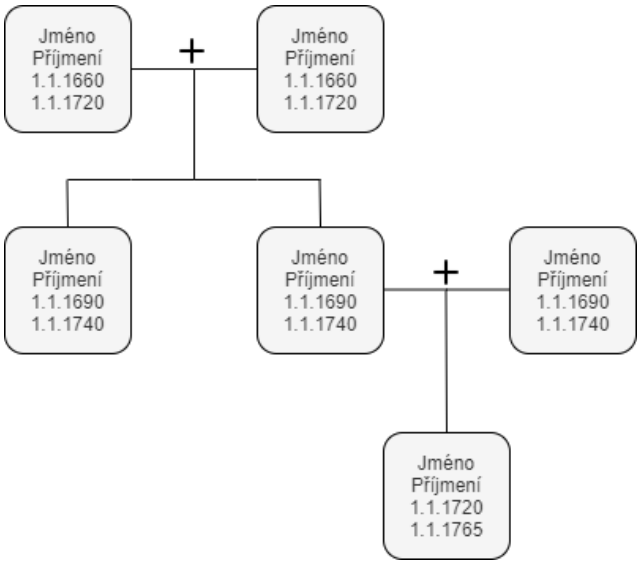
Obrázek 4.1: Navrhované formy rozložení obsahu ve webové aplikaci

V rámci návrhu rozložení webové stránky bylo zkoumáno několik možných řešení. Navrhované a testované formy rozložení se nachází na obrázku 4.1. V rámci testování vyplynulo, že *footer* v aplikaci nemá význam implementovat. Stejně tak se ukázalo, že nejvíce intuitivní je pro uživatele rozložení se *sidebarem* na levé straně. Hlavička (*header*) mohla tedy být umístěna vlevo nebo nahoře po celé délce obrazu. Zvolena byla druhá možnost, jelikož uživatelé mají tendenci ovládací prvky hledat v horní části obrazu po obou stranách. Vítězné rozložení se tedy nachází na obrázku 4.1 vpravo nahoře.

4.2.2 Zobrazení rodokmenu

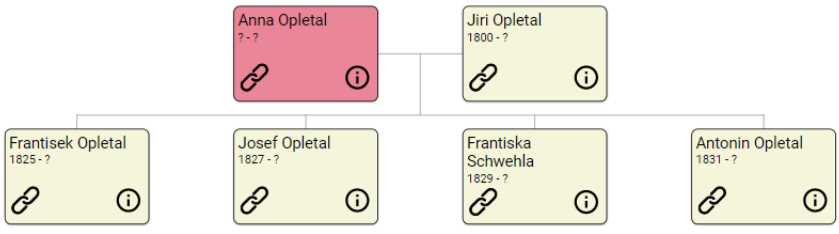
Pro uživatele nejdůležitější prvek na stránce bude samotný rodokmen. Zobrazen bude v hlavním obsahovém okně stránky. Pro zajištění maximální flexibility bude uživatelům umožněna manipulace s tímto rodokmenem, jak již bylo zmíněno výše.

Pro návrh zobrazení rodokmenu byl brán v potaz vztah uživatelů k již existujícím systémům. Všechny tyto systémy zobrazují rodokmen podobně a lidé, kteří jejich služby využívají, jsou na takové zobrazení zvyklí. Proto je vhodné takový způsob prezentace rodokmenu zachovat, aby si uživatelé nemuseli zvykat na jiný. Rodokmen tedy bude zobrazen vertikálně a chronologicky, tedy starší generace nahoře a mladší generace směrem dolů. Jednotlivé osoby poté budou představeny jako vizuální prvky ve tvaru obdélníku nebo čtverce a budou propojeny čárami na základě vztahů mezi nimi. Na tomto základě vytvořený mockup ukazuje obrázek 4.2.



Obrázek 4.2: První mockup zobrazování rodokmenu ve webové aplikaci

Tento návrh byl dále zlepšován na základě testování, které probíhalo zejména formou diskuze o možných problémech a řešeních. Bylo přidáno tlačítko pro zobrazení detailu osoby, jelikož kliknutí na samotný box bude sloužit pro vykreslení rodokmenu dané osoby. V detailu osoby budou uvedeny všechny údaje a také identity dané osoby. Dále bylo přidáno tlačítko pro propojení identit. Zde je nutné zmínit, že tlačítko pravděpodobně bude ještě přesunuto do detailu osoby k jednotlivým identitám, jelikož v tomto zobrazení zatím nemá využití. Vylepšený návrh je zobrazen na obrázku 4.3.



Obrázek 4.3: Vylepšený návrh zobrazování rodokmenu ve webové aplikaci

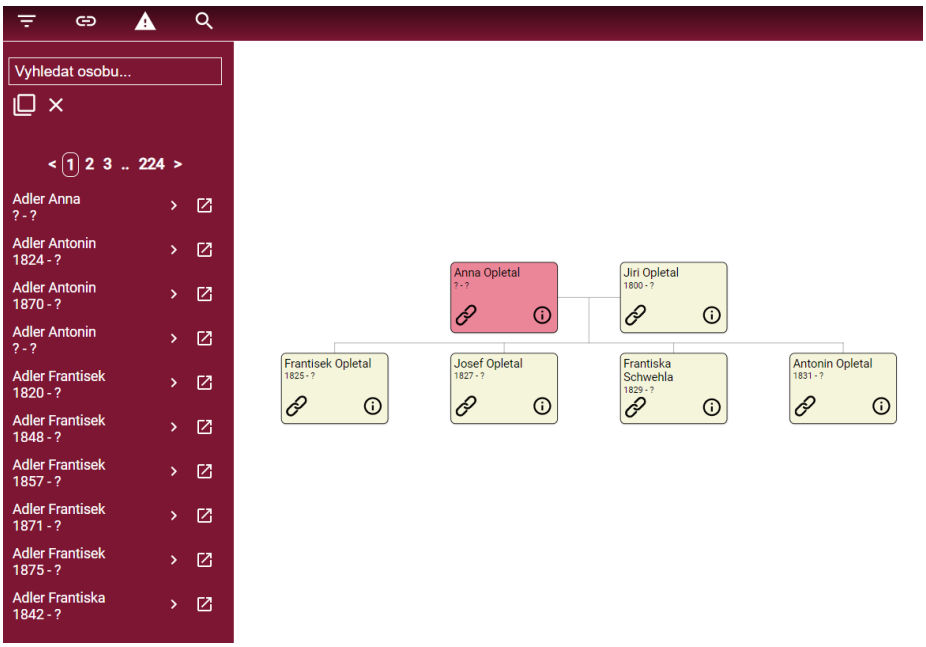
4.2.3 Testování prototypu a další vylepšení

Na základě výše zmíněných údajů se mohl vývoj přesunout ke vzniku první funkční webové aplikace, prototypu. Tento prototyp vznikl pouze pro účely testování, které probíhalo dvakrát ve skupině s pěti členy s různými zkušenostmi s počítačovou technikou. Všichni dostávali postupně úkoly, které měli na stránce splnit. Účastníkům byl vysvětlen účel projektu i samotného testovaní a objasněn i proces generování genealogických modelů programem *Moragen*. Snímek obrazovky při práci s prototypem je přiložen na obrázku 4.4.

Uživatelům se většinou podařilo úkoly splnit bez větších obtíží. Problémy nastaly jen při propojování identit s jinými osobami. To bylo do jisté míry způsobeno tím, že prototyp tuto funkcionalitu podporoval jen částečně. Dále účastníci kladně hodnotili rychlost práce s aplikací a využití adekvátních sad ikon. Dva z uživatelů vyslovili zájem o implementaci tzv. *dark mode*, tedy režimu, kdy aplikace funguje v tmavém barevném schématu.

Na základě tohoto testování vznikl seznam vylepšení, které budou figurovat v rámci vyvíjené aplikace.

- Přidání informačního okénka pro lepší přehlednost při propojování identit s osobami.
- Zlepšení přehlednosti vyhledávání, když je aktivní některý z filtrů.
- Implementace nočního režimu.



Obrázek 4.4: Prototyp aplikace pro účely testování

4.3 Datový model

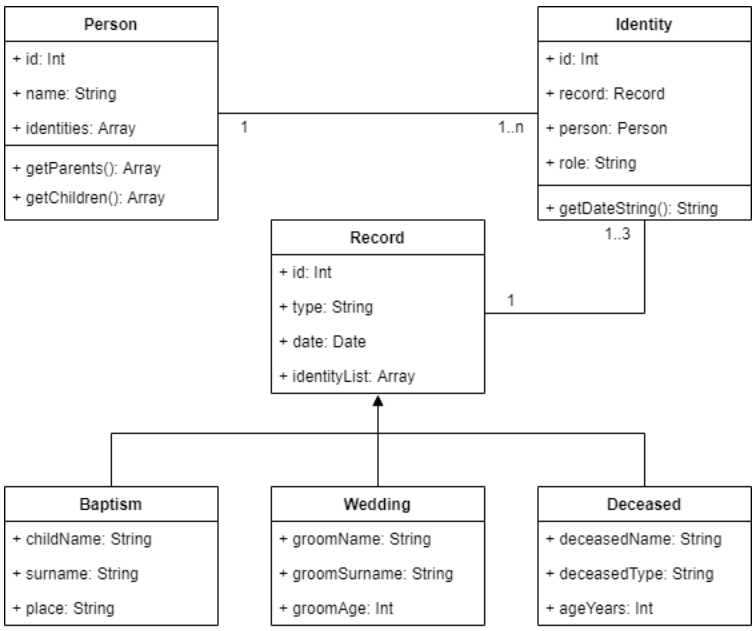
Webová aplikace *MoragenWeb* bude pracovat s velkým množstvím dat. Půjde především o seznamy matričních záznamů i seznamy identit osob. Pro značné množství dat je nutné si předem specifikovat, jak bude vypadat datová vrstva aplikace a jakým způsobem se s ní bude pracovat.

Po dohodě s vedoucím práce jsme rozhodli, že data v aplikaci budou spravována podobným způsobem, jak je tomu u aplikace *MoragenGUI*. To znamená, že bude vytvořen *objektový model*. Tento model je vyjádřen pomocí diagramu tříd na obrázku 4.5. Na obrázku jsou uvedeny pouze základní atributy a metody jednotlivých tříd.

Jak je již z obrázku patrné, data budou uchována v jednotlivých třídách. Třídy `Baptism`, `Wedding` a `Deceased` uchovávají informace o odpovídajících matričních záznamech. Například třída `Baptism` obsahuje jméno a příjmení dítěte, jméno otce i matky, datum i místo narození. Všechny tyto typy tříd dědí ze třídy `Record`, která obsahuje `ID` záznamu.

Dále třída `Identity` obsahuje údaje o jedné identitě patřící k určitému záznamu. Každý záznam naproti tomu obsahuje seznam identit, které definuje. V rámci zjednodušení práce s identitami byl v rámci této třídy uveden i typ identity, například dítě, matka, nevěsta nebo zemřelý.

Třída `Person` představuje jednu osobu ve vygenerovaném modelu. Každá osoba má, stejně jako další třídy, své atributy. V případě třídy `Person` mluvíme o jméně, příjmení, pohlaví, datu narození a atributů, které představují blízké osoby - matka, otec, děti nebo partneři. Může mít několik různých identit, jak je v obrázku znázorněno. Samotná identita ale může patřit jen k jedné osobě.



Obrázek 4.5: Diagram tříd datového modelu.

Kapitola 5

Podstatné části implementace

Tato kapitola popisuje proces implementace webové aplikace *MoragenWeb*. Konkrétně se zabývá zajímavými problémy, které bylo při implementaci nutné vyřešit. První část pojednává o vybraném programovacím jazyce a využitých knihovnách.

5.1 Programovací jazyk a knihovny

Před samotnou implementací webové aplikace je nutné si zvolit, jaké budou využity prostředky pro psaní kódu. Mezi nejdůležitější volby patří výběr vhodného programovacího jazyka. Tento projekt zpracovává webovou aplikaci, a tudíž bylo uvažováno několik technologií a postupů, které se na takové téma hodí.

Obecně mezi nejvíce využívané postupy patří zajisté využití jazyka *PHP* a v českém prostředí velmi rozšířeného frameworku *Nette*. Další možností je *Python* a jeho frameworky *Flask* nebo *Django*. Pro tento projekt byla zvolena třetí možnost, a sice framework jazyka *JavaScript Express* a jeho knihovna *React*.

Pro správné pochopení, jak funguje *JavaScriptový framework* a co přesně se skrývá za tímto slovním spojením, je správné nejprve objasnit samotný pojem *framework* a zaměřit se také na jazyk *JavaScript*.

Mat Marquis ve své knize charakterizuje ***JavaScript*** jako lehký, ale neuvěřitelně výkonný skriptovací jazyk. [11] *JavaScript* nemá kompilační krok jako ostatní jazyky, kód je vykonán počítačem přímo. Tento jazyk je velmi známý především z webového prostředí, kde umožňuje přidat na webovou stránku logiku a dynamicky měnit její obsah nebo styl. Postupem času se ale z *JavaScriptu* stal plnohodnotný programovací jazyk a vývojaři dokázali překladač tohoto jazyka použít v odlišných prostředích. Takovým způsobem vznikly projekty jako je *Node.js* - webový server na bázi *JavaScriptu*, nebo *React*.

Druhý pojem ***framework*** značí především plnou sadu nástrojů, která umožňuje vytvářet a spravovat webovou aplikaci. Pro správné pochopení rozdílu mezi knihovnou a frameworkem se v technologickém světě vžila následující metafora. Knihovny *JavaScriptu* jsou jako kusy nábytku, které můžeme umístit do interiéru domu, abychom mu zlepšily styl nebo funkci. Naproti tomu framework má funkci jakési šablony, kterou využíváme pro stavbu samotného domu. Každý z těchto přístupů má své výhody i nevýhody. Hlavní výhodou použití frameworku je zajisté lepší organizace a struktura projektu. Knihovna zase umožňuje svobodný přístup ke svým funkcím, a tedy možností použití pouze v případě potřeby.

5.1.1 React

Následující text vychází z knihy *ReactJS by Example - Building Modern Web Applications with React*. [21] *React* (někdy označovaný jako *ReactJS*) je knihovna jazyka *JavaScript* určená především pro tvorbu uživatelských rozhraní. Je optimálním řešením pro vývoj jednostránkových webových aplikací, jelikož je založená na jednoduchém toku dat. Kdykoliv se vnitřní stav dat aplikace změní, *React* změnu identifikuje a potřebnou část aplikace obnoví. Tato knihovna je vyvíjena a využívána společností *Facebook*. Dále ji využívají například služby *AirBnB*, *Instagram*, *Netflix*, *Yahoo* nebo *Alibaba*.

Za svůj úspěch *React* vděčí především způsobem, jakým využívá *pohledy*. Velmi známá architektura aplikací *MVC* (tedy *Model, View, Controller*) žádným způsobem blíže nespecifikuje svou část *View*, tedy *pohled*, jež slouží pro zobrazení obsahu. *React* si tuto část rozdělil na jednotlivé komponenty uživatelského rozhraní, které nazývá ***Components***. Komponenta může být z aplikace vyjmuta nebo naopak vložena velmi jednoduše. Každá komponenta v sobě obsahuje postup, jak vykreslit zmíněný *pohled*, a také *pohled* samotný. Stejně tak uchovává data aplikace, na jejichž základě dochází k vykreslení uživatelského prostředí. Tento postup umožňuje využít ve vyvíjených aplikacích komponenty, které vytvořil a dal k dispozici jiný vývojář nebo vznikly komunitním vývojem. Výrazně tak zjednodušuje proces tvorby uživatelských rozhraní. Ukázka vzoru jednoduché komponenty je uvedena níže.

Zdrojový kód 5.1: Vzor komponenty knihovny *React*

```
import React from 'react'

function myComponent() {
    return (
        <div>
            React component
        </div>
    )
}

export default myComponent
```

5.1.2 Express

Jedná se o backendový webový framework pro *Node.js*. Samotný *Node.js* je podle Jeffa Dickeyho *JavaScriptové* prostředí vytvořené pro běh mimo webový prohlížeč. [5] Takové prostředí je možné využít pro více různých činností, například pro monitorování nebo právě pro spuštění webového serveru. Vzhledem k tomu, že *Node.js* disponuje jazykem *JavaScript* a také velmi snadnou manipulací s jeho funkcemi, jeví se jako ideální kandidát pro tvorbu webového serveru, přesněji řečeno backendu a API webové aplikace. Stejně jako výše zmíněný *React* byl využit v mnoha známých projektech, z nichž nejznámější jsou *LinkedIn*, *PayPal*, *Walmart* nebo *Groupon*.

Framework *Express* dodává jednoduchý funkční webový server. Níže se nachází ukázka kódu jednoduchého webového serveru vytvořeného pomocí *Express*. Ukázka je převzata z oficiální dokumentace frameworku *Express*. [6]

Zdrojový kód 5.2: Jednoduchého webového serveru vytvořeného pomocí *Express*

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
})
```

5.1.3 Využité knihovny

Pro vývoj webové aplikace *MoragenWeb* bylo využito několik veřejně dostupných knihoven jazyka *JavaScript*. Nejvýznamější z nich, *React*, byl už objasněn. Aby byla tvorba aplikace usnadněna a také, aby při ní, jak se říká, nebylo znovu vynalezeno kolo, je vhodné využít již dostupné a rozšířené knihovny. Nejinak tomu bylo i při psaní této webové aplikace.

Pro správu knihoven a jejich instalaci byl využit nástroj *NPM*, patřící k frameworku *Node.js*. Podle svých oficiálních webových stránek pracuje *NPM* jako správce instalačních balíčků. [15] Umožňuje snadnou instalaci a správu verzí požadovaných balíčků. Běžná instalace funguje pomocí jednoduchého příkazu, jehož vzor je uveden níže.

```
npm install styled-components -save
```

Velmi důležitým balíčkem pro tuto práci je *react-family-tree*. [18] Tento balíček nabízí práci s komponentou *ReactFamilyTree*, která umožňuje zobrazení rodokmenu osob. Stará se o výpočet rozmístění osob a jejich propojení. Pro svou funkci vyžaduje pole objektů osob, které je nutné vykreslit. Generování tohoto pole a také strukturu a styl samotných buněk rodokmenu je nutné vyřešit po svém. Níže je uvedena jednoduchá ukázka kódu pro vytvoření rodokmenu osob. Důležitým faktem je, že komponenta zatím nepodporuje zobrazení jedné osoby na dvou místech v rodokmenu. Program Moragen zatím není bezchybný a v některém případě přiřadí osobu se stejným jménem na dvě nebo více míst v jednom rodokmenu. V takovém případě je nutné pro účely zobrazení rodokmenu ručně změnit *ID* osoby. *MoragenWeb* tento problém řeší automaticky.

Další knihovnou nainstalovanou prostřednictvím *NPM* je *react-zoom-pan-pinch*. [19] Tato knihovna dodává stejnojmennou komponentu, která umožňuje přesně to, co její název popisuje. Umožňuje svým obsahem manipulovat, a sice posunem nebo přiblížením. Tato komponenta v aplikaci *MoragenWeb* obaluje vygenerovaný rodokmen a uživatel si jej tak může pohodlně přiblížit nebo posunout tak, jak mu to vyhovuje.

Zdrojový kód 5.3: Tvorby rodokmenu ve webové aplikaci *MoragenWeb*

```
<ReactFamilyTree
    nodes={persons}
    rootId={id}
    width={800}
    height={600}
    renderNode={(node) => (
        <FamilyNode
            key={node.id}
            node={node}
            style={{width: WIDTH,
                    height: HEIGHT,
                    transform: `translate(${node.left * (WIDTH / 2)}px,
                        ${node.top * (HEIGHT / 2)}px)`}}
        />
    )}
/>
```

Každá moderní webová aplikace potřebuje dobře vypadat. Z tohoto důvodu a také kvůli přehlednějšímu kódu tento projekt využívá knihovnu ***styled-components***. [20] Díky tomuto rozšíření je možné definovat vlastní stylizované komponenty. Styl se vkládá pomocí kaskádových stylů, známých jako *CSS*. Níže je uvedený příklad vytvořeného elementu `<div>` pomocí *styled-components*. Největší výhoda takového postupu tkví v oddělení kódu pro styl od kódu pro rozmístění prvků a logiky aplikace. Vytvořený element na stránce má vlastní přidělenou třídu, která je automaticky generovaná. O přiřazení správných stylů ke konkrétním prvkům na stránce se tak stará rozšíření samo, čímž samozřejmě zjednodušuje práci a omezuje množství chyb.

Součástí projektu bude i **tmavý režim** pro pohodlnější práci. Zmíněná knihovna *styled-components* má funkci přesně odpovídající tomuto záměru. V souboru *themes.js* stačilo definovat barevná schémata a do aplikace přidat přepínač. Podle interního stavu si poté webová aplikace sama vyhodnotí, jaké barevné schéma má použít, a aplikuje je do stylů prvků na stránce. Knihovna k takovému postupu využívá proměnnou `theme`.

Zdrojový kód 5.4: Prvek `<div>` vytvořený pomocí rozšíření *styled-components*

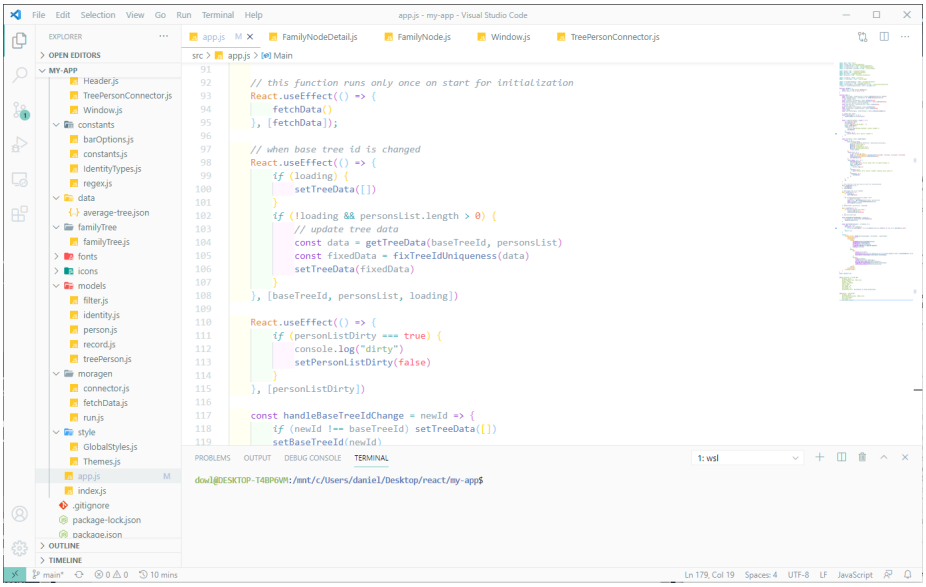
```
const Container = styled.div`
    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: ${({theme}) => theme.bar};
    box-shadow: 0 3px 5px 2px rgba(0, 0, 0, 0.26);
`
```

Za zmínku stojí také knihovny *create-react-app* a *express-generator*, které umožňují rychle vygenerovat základní strukturu aplikace a kostru kódu. Dále byla využita knihovna *react-paginate* pro příjemnější stránkování seznamu osob v aplikaci. Ikony v aplikaci byly převzaty z rozšíření *material-ui*, známé především z aplikací pro operační systém *Android*.

5.2 Průběh implementace

Veškerý vývoj aplikace probíhal v prostředí ***Visual Studio Code***. Se svými funkcemi jako je *IntelliSense* nebo integrací *Gitu* přímo do programu se řadí mezi nejpopulárnější textové editory pro psaní kódu. V rámci řešení této práce bylo *Visual Studio Code* využito také při testování nebo debugování kódu, a to za pomocí komunitních rozšíření. Pro spouštění webových serverů, zejména testovacích v rámci vývoje, bylo použito rozšíření operačního systému *Windows Subsystem for Linux* neboli *WSL*.

Hlavním nástrojem pro vývoj této webové aplikace byl ale **webový prohlížeč**. *MorageWeb* byl testován ve dvou různých prohlížečích. Prvním z nich je *Google Chrome* verze 90, druhým *Mozilla Firefox* ve verzi 87.



Obrázek 5.1: Vývoj webové aplikace v prostředí *Visual Studio Code*

5.2.1 Struktura projektu

Strukturu tohoto projektu by se dalo rozdělit do dvou částí. První je webová aplikace *MoragenWeb* implementovaná pomocí knihovny *React* v jazyce *JavaScript*. Druhá část je samotný webový server a API pro komunikaci s webovou aplikací. Tato část je vytvořená pomocí frameworku *Express*.

Struktura hlavní serverové části zahrnuje startovací skript *app.js* a složku *public*, která obsahuje soubory webové stránky a její obsah je jako jediný přímo přístupný z klientské strany. Adresář *routes* má v tomto webovém serveru důležitou funkci. Nachází se zde směrování a skripty pro jednotlivé volání API. Složka *node_modules* a soubor *package.json* se zde objevují kvůli správě rozšíření a knihoven pro *Node.js*. Stejně tak je tomu i u dále zmíněné webové aplikace.

Aplikace *MoragenWeb* je napsána pomocí knihovny *React*. Od této skutečnosti se odvíjí implementační struktura aplikace. Kořenový adresář je tvořen dvěma složkami - *public* a *src*. Je zřejmé, že složka *src* slouží pro zdrojové kódy aplikace, a tudíž nebude dostupná z klientské strany prohlížeče. Hlavními soubory pro spuštění celé aplikace jsou tentokrát *index.js* společně s *app.js*. Všechny zdrojové soubory projektu jsou rozděleny do složek podle typu, zde v popisu struktury projektu budou řazeny abecedně.

- V adresáři ***api*** se nachází soubory s asynchronními funkcemi pro komunikaci s webovým serverem. Například soubor *moragen.js* obsahuje funkci pro spuštění výpočtu genealogického modelu na serveru.
- V dalším adresáři ***components*** jsou uloženy veškeré vytvořené komponenty aplikace, příkladem může být komponenta `Bar` pro funkci postranního menu nebo `Header` reprezentující horní lištu. Každá komponenta má svůj vlastní soubor.
- Ve složce ***constants*** se nachází několik souborů s proměnnými, které mají fixní hodnotu. Patří zde i vyhledávací řetězce pro *regex*.
- Adresáře s názvem ***fonts*** a ***icons*** obsahují soubory fontů respektive ikon použitých v aplikaci.
- Obsah důležité složky s názvem ***models*** tvoří soubory s třídami podle již specifikovaného datového modelu.
- Další složka ***moragen*** skrývá soubory s funkcemi pro propojení datových modelů při spuštění aplikace.
- Poslední adresář ***style*** slouží k definici globálních stylů na stránce a také pro správu proměnné `theme`, jež obstarává funkci tmavého režimu.

5.2.2 Podstatné části implementace

Implementace webové aplikace *MoragenWeb* probíhala v několika krocích. V první fázi vývoje byl zprovozněn funkční prototyp. Tento prototyp byl následně rozšiřován a zlepšován až do finální verze. V této části práce bude nastíněno, jakým způsobem program funguje, a také zde budou zmíněny některé důležité informace vycházející z napsaného kódu.

Horní část obrazovky je určena funkčnímu panelu. V pravé části uživatel najde ovládání výpočtu na serveru i správu projektu. Samotná správa projektu nebyla součástí této práce, jelikož se strany projektu *DEMoS* v době psaní této práce nebyla podporována. Dále se zde nachází přepínač nočního režimu a na levé straně je umístěno ovládání **postranního panelu**, přesněji řečeno jeho přepínání. Tento panel může fungovat jako vyhledávač osob, ale může také zobrazovat chybné osoby nebo nepropojené identity. O samotné přepínání funkcionality panelu se starají funkce `changeBarOption` a také `renderBarOption`.

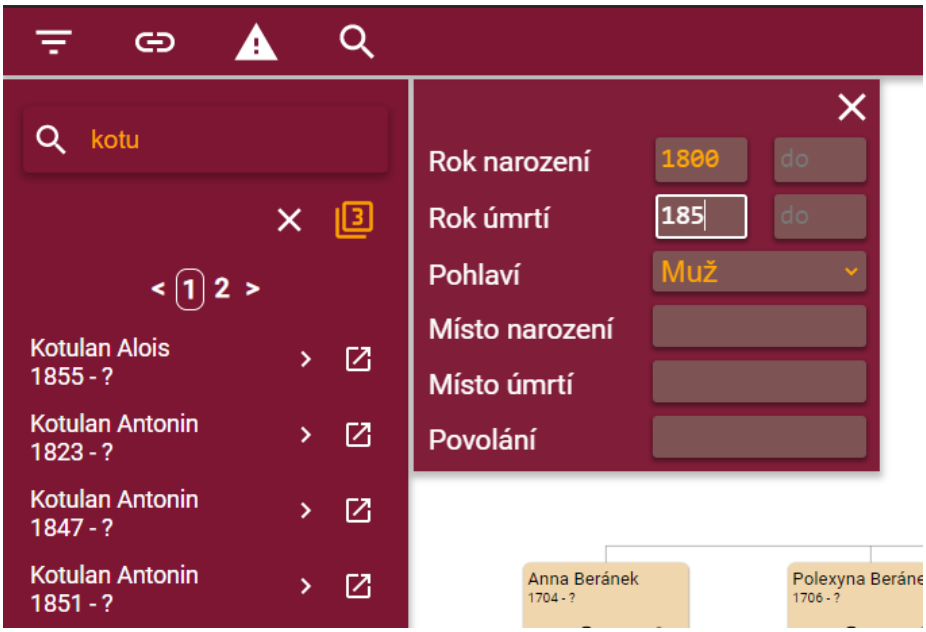
Mezi nejdůležitější funkcionality postranního panelu patří **hledání a filtrování osob**. V případě nutnosti použití filtrování se zobrazí další okno. To je změna oproti návrhu aplikace, kde se měly tyto filtry vyskytovat hned pod vyhledávacím tlačítkem. Testováním se ukázalo, že poté nezbývalo dostatečné místo pro zobrazení výsledků osob. Z tohoto důvodu se filtry přesunuly do nového panelu. Pokud je některý z filtrů aktivní, je zobrazen výraznou barvou a současně počet aktivních filtrů je uveden u hlavního tlačítka pro filtrování. Celý proces je lépe viditelný na obrázku 5.2.

Zobrazovaní panelu s filtry umožňuje funkce `handleFiltersWindow`. O samotné filtrování osob se stará funkce `handleFilterChange`. `Filter` je v této aplikaci objekt, který uchovává informace o zadaných parametrech. Tento objekt je poté využit při zobrazování seznamu osob, kdy každá osoba je porovnána s hodnotami tohoto filtru a vyhodnocena, zda se zobrazí, nebo ne. K tomuto vyhodnocení jsou využity funkce `useEffect`, které jsou jednou z funkcionalit *Reactu*. Každá taková funkce je spojena s nějakým objektem (v tomto případě se jedná o objekt `filter`) a spustí se v okamžiku, kdy v objektu dojde k nějaké změně, například ke změně hodnoty jednoho z atributů.

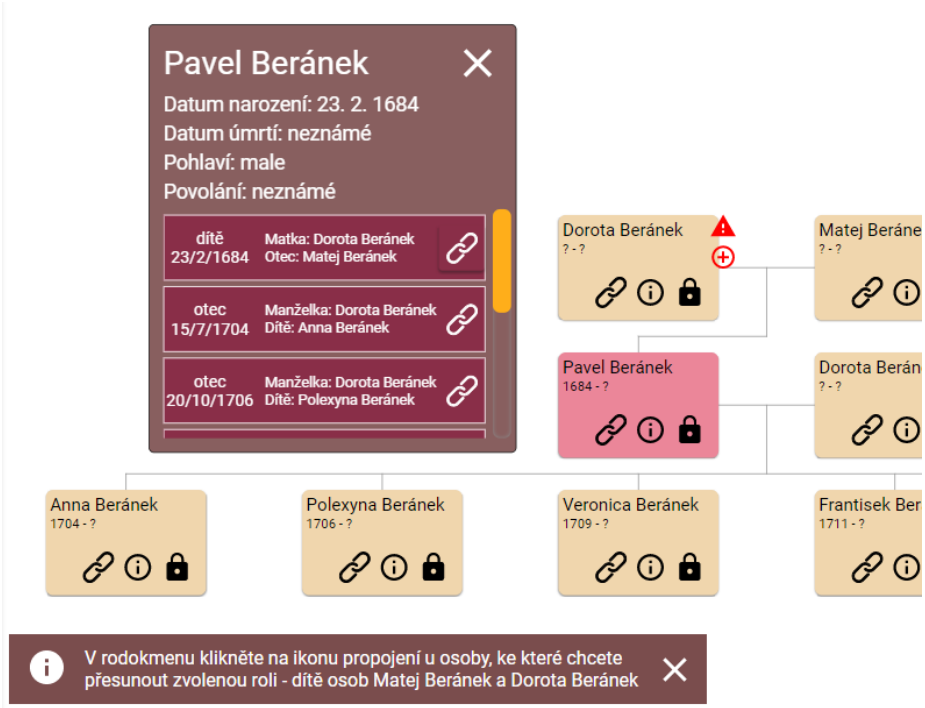
Pro zobrazení **rodokmenu** byla, jak již bylo zmíněno, využita knihovna *react-family-tree*. Každý vykreslený rodokmen se týká jedné osoby, pro kterou je vygenerován. Tuto skutečnost řeší funkce `getTreeData`, která na základě identifikačního čísla hlavní osoby vygeneruje pole objektů osob, které budou zobrazeny v rodokmenu. Při každé změně základní osoby se spustí nové generování dat, samozřejmě s použitím funkce již zmíněné `useEffect`.

Implementované tlačítko pro **detail osoby** umožňuje zobrazení dalšího okna, které obsahuje informace o dané osobě i seznam identit této osoby. Pro okno detailu osoby bylo oproti návrhu umožněno posouvání. Testování totiž ukázalo, že se okno ne vždy zobrazí na dobrém místě, kde uživateli nepřekáží. Byla tak přidána možnost pomocí myši okno přetáhnout na jiné místo, které více vyhovuje práci s rodokmenem. Součástí webové aplikace je také možnost propojování identit, tuto skutečnost ilustruje obrázek 5.3.

Funkcionalitu propojování identit a osob zajišťují stavová proměnná `identityToMove` představující přesouvanou identitu a funkce s názvem `connectIdentityToPerson`. Ta má za úkol identitu propojit se zvolenou osobou.



Obrázek 5.2: Hledání a filtrování osob ve webové aplikaci *MoragenWeb*



Obrázek 5.3: Zobrazení detailu osoby a propojování identit s jinými osobami ve webové aplikaci *MoragenWeb*

5.3 Integrace do systému DEMoS

Součástí řešení této práce bylo také propojení se systémem *DEMoS*. V tomto odstavci bude uvedeno řešení takového propojení. Uživatelé tohoto systému by na základě integrace měli mít přístup k programu *Moragen* a mohli by si také prohlížet a upravovat vygenerované modely. Program *Moragen* by měl v ideálním případě pracovat s daty, které vznikají komunitním přepisem historických záznamů, tedy těmi daty, které jsou uloženy v databázi systému *DEMoS*. Současně by v rámci tohoto systému vznikala grafová databáze obsahující uživatelsky vytvořené a upravené rodokmeny, přičemž každý uživatel by měl spravovat své projekty respektive rodokmeny. Diagram řešení integrace do systému *DEMoS* je obsažen na obrázku 5.4.

5.3.1 Omezení na straně systému

Je nutné zmínit, že v době vzniku této práce systém *DEMoS* bohužel nepodporoval několik zásadních funkcionalit pro správnou integraci webové aplikace *MoragenWeb*.

První funkcionalitou je samotné získávání dat z databáze matričních záznamů. Jelikož se jedná o značně náročný proces, bude nutné nejprve navrhnout a implementovat **aplikační rozhraní** v rámci sytému *DEMoS*. Takové rozhraní bude efektivní a webová aplikace *MoragenWeb* jej bude využívat jako zdroj dat, které následně pomocí programu *Moragen* zpracuje a sestaví genealogický model. Součástí tohoto problému je také zajistit uživateli výběr zdroje dat na základě zvolených atributů. Těmi mohou být například příjmení, rok narození, typ matriky nebo lokalita. V rámci systému *DEMoS* zatím nepanuje shoda, jakým způsobem uživateli umožnit tento výběr zdrojových dat. Současná integrace aplikace *MoragenWeb* do systému *DEMoS* obsahuje pouze jeden projekt obsahující vzorová zdrojová data poskytnutá vedoucím této práce.

Druhý problém souvisí již se zmíněnou absencí aplikačního rozhraní. Jak je patrné z obrázku 5.4, pro správnou funkci této webové aplikace je zapotřebí spravovat **uživatelské projekty**. Systém *DEMoS* umožňuje registraci uživatelů, ale neumožňuje jim vytvářet vlastní projekty. Proto bude pro integraci využit jeden ukázkový projekt, jak je uvedeno výše. Při dalším vývoji systému *DEMoS* půjde vzniklá aplikace *MoragenWeb* se správou projektů poměrně jednoduše spojit.

5.3.2 Podstatné části integrace

Součástí webové aplikace *MoragenWeb* je i aplikační rozhraní serveru, který je implementován pomocí frameworku *Express*. Aplikační rozhraní tohoto serveru je také využito pro správné zacházení s programem Moragen.

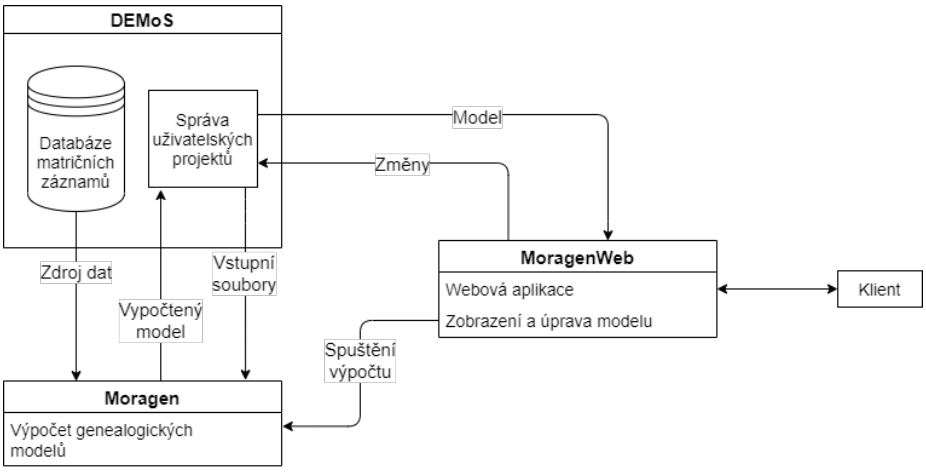
Server běží na portu 9000 a zajišťuje jak zobrazení webové aplikace, tak komunikaci se serverovým prostředím. Pro zobrazení webové aplikace je využita hlavní URL cesta /.

Volání API z webové aplikace zajišťuje cesta */api/moragen/*, kdy existuje několik různých možností podle toho, čeho je nutné docílit. Vyžádání obsahu souboru (například souboru *coreFamilies.mrg*) řeší URL adresa */file/FILENAME/* a zápis do souboru *updates.mrg* neboli uložení projektu využívá adresu */save/*. Pro uložení projektu se využívá metoda *POST*. Implementace této cesty je uvedena v ukázce 5.5. Z prostředí webové aplikace *MoragenWeb* je požadavek vygenerován pomocí knihovny *axios*. [3]

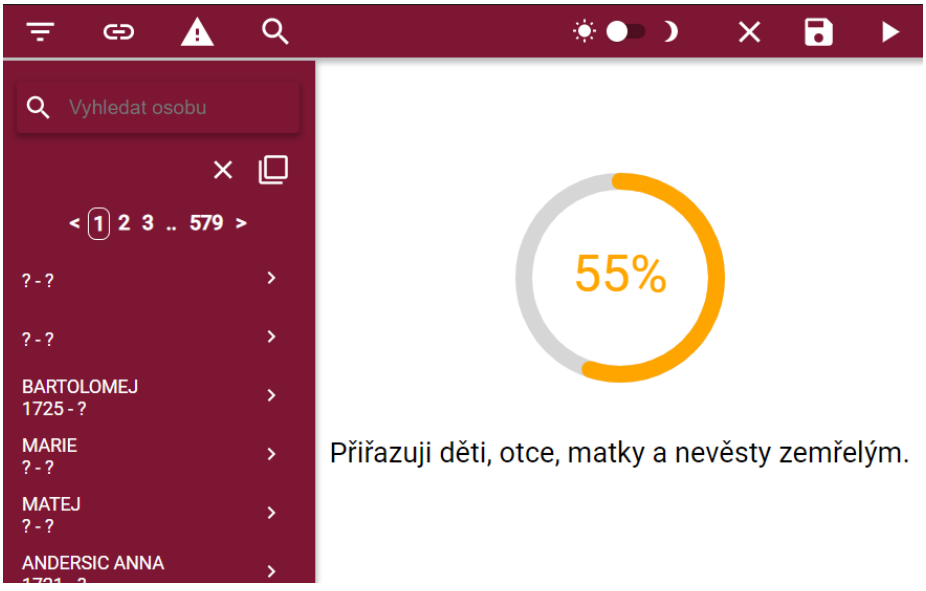
Zdrojový kód 5.5: Implementace *POST* metody pro uložení projektu

```
router.post("/save/", jsonParser, function(req, res) {
    fs.writeFile(
        `${path.join(__dirname, MORAGEN_DIRECTORY)}dataIn/updates.mrg`,
        req.body.string,
        function(err) {
            if (err) res.send("File not saved.")
            res.send("File saved.")
        }
    )
})
```

Pokud chce uživatel spustit **výpočet genealogického modelu**, využije základní adresu */api/moragen/*. V takovém případě server vytvoří nový proces, v němž spustí sestavování rodokmenu. Jakmile tento proces skončí, odešle odpověď zpátky do webové aplikace, která je poté schopna si zažádat o vzniklé soubory pomocí adres zmíněných výše. Problém nastává v případě velmi dlouhého výpočtu, kdy některé prohlížeče automaticky předpokládají, že odpověď již od serveru nepřijde. Z tohoto důvodu je součástí webového serveru také knihovna *express-sse*. [7] Díky tomuto rozšíření je možné se serverem **udržovat spojení** i po dobu výpočtu genealogického modelu a průběžně uživatele o průběhu informovat. Ukázka této zpětné vazby se nachází na obrázku 5.5. Informování uživatele o průběhu výpočtu funguje na principu *Server-sent events*. Jedná se o serverovou technologii, která umožňuje z klientské strany přijímat automatické zprávy vysílané ze serverové strany. Tato standardizovaná technologie je součástí *HTML5* od organizace *W3C*. [23]



Obrázek 5.4: Diagram propojení webové aplikace *MoragenWeb* se systémem *DEMoS*



Obrázek 5.5: Spuštění výpočtu pomocí programu *Moragen* v aplikaci *MoragenWeb*

Kapitola 6

Dosažené výsledky a možnosti dalšího vývoje

Obecně pro každou aplikaci platí, že může být dále zlepšována nebo obohacena o nové funkce. Nejinak je tomu i u webové aplikace *MoragenWeb*. V následujících odstavcích budou uvedeny vylepšení stávajících funkcí nebo úplně nové funkce, které se v rámci této práce nerealizovaly. Je nutné zmínit, že následující možnosti dalšího vývoje se netýkají pouze webové aplikace, ale zaměřují se i na její integraci do systému *DEMoS*. Z tohoto důvodu zde budou uvedeny i některá zlepšení týkají se tohoto systému nebo programu *Moragen*, jenž v něm figuruje.

6.1 Systém DEMoS

Dlouhodobý cíl, který si tento systém určil, byl již popsán v sekci 3.1. Důležité je, že se jej daří realizovat. Každý člověk má možnost si zde založit účet a začít s přepisem matričních záznamů. Může tak přidávat nová data do komunitní databáze, kontrolovat je nebo v nich jen vyhledávat. Vyvíjená webová aplikace *MoragenWeb*, která uživateli poskytuje možnost zobrazení vygenerovaného rodokmenu, v rámci systému funguje s ukázkovými daty. Zatím neumožňuje plné propojení se zmíněnou komunitní databází. Důvody tohoto řešení byly shrnuty v sekci 5.3. V konečném důsledku má uživatel možnost si práci s touto webovou aplikací vyzkoušet a naučit se s ní pracovat. Očekává se, že aplikační rozhraní, které zajistí propojení s databází, bude implementováno v nejbližších měsících.

Jednou z nejvíce zásadních funkcionalit, které současná webová aplikace postrádá, je možnost **editace všech údajů** o jednotlivých osobách či přidání nových osob. Důvodem, proč takovou možnost uživateli nenabízí je fakt, že program *Moragen* zatím neumožňuje definovat nové osoby a pracovat s nimi nebo měnit jejich údaje. Výpočet vychází pouze ze seznamu matričních záznamů a uživatelem definovaných změnách u přiřazení identit k osobám. Aby mohla být přidána některá osoba, musí být v současnosti obsažena v těchto matričních záznamech.

Zmíněný program *Moragen* se stále vyvíjí a je možné, že několik měsíců po dokončení této práce již bude podporovat daleko více funkcí. I přes to zde bude představena ještě jedna, která s tímto programem souvisí a mohla by se stát předmětem dalšího vývoje. Výpočet modelů probíhá ve složce, kde se program nachází. Všechny zdrojové soubory i soubory se záznamy proto musí být obsahem této složky. V praxi to znamená, že pro každý výpočet je nutné přesunout soubor se záznamy do adresáře s programem, spustit výpočet a následně vzniklé soubory z tohoto adresáře znovu přesunout zpět. Pro budoucí vývoj systému *DEMoS* je takové řešení nešikovné a bude vyžadovat, aby tento program podporoval například spuštění s parametrem cesty ke složce nebo podobné řešení.

Aby webová aplikace *MoragenWeb* efektivně a rychle komunikovala s prostředím genealogické databáze, bude zapotřebí takové propojení do systému navrhnout a implementovat. Jak je uvedeno v sekci 5.3, aplikační rozhraní bude řešit především uživatelský výběr množiny záznamů, ze kterých bude program *Moragen* sestavovat genealogické modely. Po dokončení tohoto kroku bude aplikace plně integrovaná a propojená se systémem *DEMoS*.

6.2 Aplikace MoragenWeb

Co se týče samotné webové aplikace, její vývoj zajisté není u konce. V současnosti umožňuje uživateli tvořit rodokmeny a upravovat je s pomocí přesunu identit. Podrobně se tomuto problému věnuje sekce 5.2.2. Cestou budoucího vývoje by mohlo být jistě editace rodokmenu řešená podobně jako v jiných systémech, tedy propojováním jednotlivých osob. Na základě těchto propojení by aplikace mohla automaticky vytvořit a přiřadit novou identitu, samozřejmě s následnou podporou v programu *Moragen*.

Uživatelé by uvítali také **více detailů** u zobrazených osob, příkladem může být chybně vygenerovaná osoba v rodokmenu. V současné verzi aplikace se uživatel o této chybě dozví, nicméně neví, co konkrétně daná osoba porušuje. Všechny tyto chyby zpracovává program *Moragen*, nejprve by tedy bylo nutné tento program rozšířit tak, aby o jednotlivých nalezených chybách dodal více informací. Jakmile tyto chyby budou k dispozici i v samotné webové aplikaci, je v rámci jejího fungování možné uživatelům navrhnout možné řešení těchto chyb a usnadnit jim tak práci.

Poslední věcí, kterou je dobré v této sekci zmínit, je možnost vygenerovat si z vzniklých modelů svůj rodokmen, například v populárním formátu *GEDCOM*.

Kapitola 7

Závěr

Cílem práce bylo vytvořit webovou aplikaci, která by umožňovala zobrazení a úpravu genealogických modelů a také by, pomocí programu Moragen, uživatelům dovolila zahájit nový výpočet nebo kontrolu upraveného modelu. Zmíněný program sestavuje genealogické modely na základě dodaných matričních záznamů. Tato práce navazuje na program MoragenGUI, jehož funkcionalitu integruje do serverového řešení projektu DEMoS.

Zpočítku bylo důležité nastudovat potřebné informace, nejprve údaje o již existujících webových službách, které podobné služby poskytují. Obsahem dalšího kroku bylo seznámení se s prací programů Moragen a MoragenGUI a stavem jejich řešení. Posledním nutným krokem pro pokračování práce zůstalo obeznámení se systémem DEMoS fungujícím v rámci Fakulty informatiky na Vysokém učení technickém v Brně. Po získání všech potřebných údajů práce pokračovala k návrhu webové aplikace.

V rámci tohoto kroku bylo zapotřebí nejprve specifikovat požadavky, které budou pro výslednou aplikaci klíčové. Návrh pokračoval rozmístěním hlavních prvků na stránce a tvorbou prvního mockupu. Testování, které se provádělo na funkčním prototypu webové aplikace, ukázalo některé náměty na zlepšení - ty byly zahrnuty do finální podoby grafického uživatelského rozhraní aplikace.

Co se týče implementace - webová aplikace byla napsána v jazyce JavaScript za pomoci knihovny React. Pro webový server byla využita technologie NodeJS. Aby byl výsledný produkt plně funkční, v průběhu implementace bylo nutné vyřešit různé problémy, které se vyskytly. Důležitou součástí práce byla také integrace do systému DEMoS. Ta byla úspěšně provedena, bohužel se ale nejedná o konečné řešení, jelikož ve zmíněném systému chyběly některé důležité součásti. Kvůli tomu nemohla být integrace provedena optimálním způsobem.

Výsledkem této práce je serverové prostředí s webovou aplikací. Je možné říci, že všechny cíle, které si tato práce předsevzala, byly splněny. Aplikace je plně funkční a svým uživatelům poskytuje přístup ke genealogickým modelům, které sami uživatelé mohou vygenerovat pomocí programu Moragen, který je v serverovém řešení připojen. Aplikace disponuje moderním vzhledem a uživatelsky přívětivým prostředím i ovládáním.

Vývoj systému DEMoS ještě není u konce a jistě bude dále vznikat také propracovanější propojení s novou webovou aplikací. Jmenovitě jde o tvorbu uživatelských projektů nebo o přímé napojení komunitní databáze coby zdroje matričních záznamů pro tvorbu genealogických modelů.

Literatura

[1] *Ancestry: genealogický databázový program.* 2009. Dostupné z: https://ancestry.nethar.cz/index.php.

[2] *Ancestry.com: Genealogy, Family Trees & Family History Records.* Lehi, Utah, USA: Ancestry.com, 2021. Dostupné z: https://www.ancestry.com/.

[3] *Axios.* Npm, 2021. Dostupné z: https://www.npmjs.com/package/axios.

[4] *DEMoS: genealogická databáze.* Brno: Fakulta informačních technologií VUT, 2021. Dostupné z: http://perun.fit.vutbr.cz/.

[5] Dickey, J. *Write Modern Web Apps with the MEAN Stack: Mongo, Express, AngularJS, and Node.js.* 1. vyd. United States of America: PEACHPIT PRESS, 2015. ISBN 978-0-13-393015-3.

[6] *Express: Node.js web application framework.* San Francisco, California: OpenJS Foundation, 2021. Dostupné z: https://expressjs.com/.

[7] *Express-sse: An Express middleware for quick'n'easy server-sent events.* Npm, 2021. Dostupné z: https://www.npmjs.com/package/express-sse.

[8] *FamilySearch: zdarma dostupná rodinná historie a genealogické záznamy.* Salt Lake City, Utah, US: Genealogical Society of Utah, 2021. Dostupné z: https://www.familysearch.org/.

[9] *FamilySearch indexing.* Salt Lake City, Utah, US: Genealogical Society of Utah, 2021. Dostupné z: https://www.familysearch.org/indexing/.

[10] *Incremental Search.* Boston, Massachusetts, USA: Free Software Foundation, 2021. Dostupné z: https://www.gnu.org/software/emacs/manual/html_node/emacs/Incremental-Search.html.

[11] Marquis, M. *Javascript for web designers.* 1. vyd. New York, New York: A Book Apart, 2016. ISBN 978-1-937557-47-8.

[12] *Find As You Type: Type Ahead Find.* Mountain View, Kalifornie, USA: Mozilla Corporation, 2012. Dostupné z: https://website-archive.mozilla.org/www.mozilla.org/access/access/type-ahead/.

[13] *MyHeritage: Vaše rodinná historie.* Or Jehuda, Izrael: MyHeritage, 2021. Dostupné z: https://www.myheritage.cz/.

[14] Nielsen, J. *Web.Design*. 1. vyd. Praha: SoftPress, 2002. ISBN 80-864-9727-5. Dostupné z: https://ndk.cz/uuid/uuid:87f13fe0-7a29-11e4-93df-005056825209.

[15] *Node Package Manager*. Npm, 2021. Dostupné z: https://www.npmjs.com/.

[16] *Java Client Roadmap Update: An update of timelines for Java Deployment and Java UI technologies*. Austin, USA: Oracle Corporation, 2020. Dostupné z: https://www.oracle.com/technetwork/java/javase/javaclientroadmapupdatev2020may-6548840.pdf.

[17] Peterka, J. *Cesta k rodinným kořenům, aneb, Praktická příručka občanské genealogie*. 1. vyd. Praha: Libri, 2006. ISBN 80-727-7307-0.

[18] *React-family-tree*. Npm, 2021. Dostupné z: https://www.npmjs.com/package/react-family-tree.

[19] *React-zoom-pan-pinch*. Npm, 2021. Dostupné z: https://www.npmjs.com/package/react-family-tree.

[20] *Styled-components: Visual primitives for the component age*. 2021. Dostupné z: https://styled-components.com/.

[21] Vipul, A. M. a Prathamesh, S. *ReactJS by Example - Building Modern Web Applications with React: Get up and running with ReactJS by developing five cutting-edge and responsive projects*. 1. vyd. Birmingham B3 2PB, UK.: Packt Publishing Ltd., 2016. ISBN 978-1-78528-964-4.

[22] Wawreczka, V. *Modelování na základně dat z archiválií*. Brno, CZ, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Dostupné z: https://www.fit.vut.cz/study/thesis/22946/.

[23] *World Wide Web Consortium*. Cambridge, Massachusetts, United States: W3C, 2021. Dostupné z: https://www.w3.org/.

[24] Zbořil, F., Rozman, J. a Kočí, R. Algorithmic creation of genealogical models. In: VUT Brno. *Proceedings of ISDA 2018*. Springer International Publishing, 2019, sv. 941, s. 650–658. Advances in Intelligent Systems and Computing. DOI: 10.1007/978-3-030-16660-1_63. ISBN 978-3-030-16659-5. Dostupné z: https://www.fit.vut.cz/research/publication/11849.

Příloha A

Obsah přiloženého paměťového média

- **app/** – adresář se zdrojovými soubory spustitelného serveru
- **doc/** – adresář se zdrojovými soubory technické zprávy
- **moragen/** – adresář se zdrojovými soubory programu Moragen potřebnými pro kompletní běh webové aplikace, obsah tohoto adresáře není výsledkem této práce
- **src/** – adresář se zdrojovými soubory webové aplikace
- **readme.txt** – návod ke spuštění serveru a nápověda k použití aplikace
- **xpatek08.pdf** – text bakalářské práce ve formátu PDF