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 — **VOJTĚCH WAWRECZKA**
AUTHOR

VEDOUCÍ PRÁCE — **Doc. Ing. FRANTIŠEK ZBOŘIL, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Wawreczka Vojtěch**
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.
2. Navrhněte systém, 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. Diskutujte dosažené výsledky a navrhněte další možná rozšíření implementovaného 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 2019
Datum odevzdání: 28. května 2020
Datum schválení: 31. října 2019

Abstrakt

Cílem této práce je vytvořit aplikaci, která zpracovává výsledky programu pro tvorbu genealogických modelů ze záznamů matrik a umožňuje uživateli tato data upravovat a vytvářet pomocí nich sociální modely. Aplikace je vytvořena v jazyce Java na platformě JavaFX. Velký důraz je v práci kladen také na uživatelskou přívětivost při zachování všech stávajících požadavků.

Vytvořené řešení umožňuje uživateli generované rodokmeny zobrazit a upravovat pomocí uzamykání nebo rušení vazeb mezi osobami. U osob jsou zobrazeny případné chyby a nejasnosti, které jsou při tvorbě rodokmenu generovány. V rodokmenech lze vyhledávat a filtrovat na základě daných parametrů.

Abstract

The aim of this work is to create an application that processes the results of program for creating genealogical models from records of register offices and allows a user to edit the data and to create social models using them. The application is created in Java on platform JavaFX. A great emphasis in this work is also placed on user friendliness while maintaining all existing requirements.

The created solution allows a user to display generated family trees and to edit them by locking or breaking links between individuals. Possible errors and ambiguities that are generated while creating a family tree are displayed in person's details. In family trees a user can search and filter information based on the given parameters.

Klíčová slova

genealogie, tvorba rodokmenů, matriky, archiválie, sociální modely, Java, JavaFX, GUI

Keywords

genealogy, creating of family tree, registries, archival materials, social models, Java, JavaFX, GUI

Citace

WAWRECZKA, Vojtěch. *Modelování na základně dat z archiválií*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. František Zbořil, 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.

.....
Vojtěch Wawreczka
28. května 2020

Poděkování

Touto cestou bych chtěl poděkovat vedoucímu této bakalářské práce Doc. Ing. Františku Zbořilovi, Ph. D. za ochotu, cenné rady a připomínky.

Obsah

Kapitola 1

Úvod

Digitalizace dnešního světa nás stále více ovlivňuje ve všech oblastech našeho života. A s trochou nadsázky lze říci, že začíná pronikat i do historie. Do veřejného prostoru se totiž dostávají digitalizované matriky a zkoumat vlastní historii je tak jednodušší než kdy dřív. Není proto divu, že roste popularita tvorby vlastních rodokmenů především mezi laickou veřejností.

A zde přichází příležitost pro informatiku, která v oblasti genealogie může výrazně usnadnit práci při zpracovávání digitalizovaných matrik. Jak bude v práci uvedeno, současné genealogické systémy však reagují pouze na onu populární stránku věci, kdy umožňují uživatelům jednoduše tvořit rodokmeny a přidávat do nich další osoby. Nijak nereflektují matriční záznamy a není možné zde tato data vkládat a pracovat s nimi.

Tato práce navazuje na vývoj programu [22], který dokáže zpracovat záznamy z matrik, pomocí algoritmu je spojit vazbami a vytvořit tak rodokmen osob, které jsou na rozdíl od současných systémů navázány přímo na záznamy. Celý rodokmen tak není postaven na konkrétních osobách, ale na jednotlivých záznamech.

Hlavním cílem práce je vytvořit aplikaci s grafickým rozhraním, která dokáže zpracovat výsledky tohoto programu. Jelikož výsledná data nemusí být vždy zcela korektní, umožní uživateli výsledky upravovat a znovu je nechat propočítat zmíněným programem, a tak ve spolupráci s ním tvořit jednoduše a efektivně rozsáhlé rodokmeny. Dále umožní uživateli v rodokmenech vyhledávat a vytvářet sociální modely jednotlivých rodů, profesí apod. Velký důraz práce klade také na grafické rozhraní a jeho přívětivost, jednoduchost a intuitivnost, bez nichž se moderní aplikace neobejdou.

V první kapitole jsou popsány základní teoretické poznatky o spojování záznamů a dále konkrétní způsob, jak probíhá toto spojování v případě programu, na který tato práce navazuje. Druhá kapitola pak ukáže nejznámější genealogické programy pro tvorbu rodokmenů.

Další kapitoly se již zabývají konkrétním vývojem aplikace. V kapitole třetí jsou specifikovány požadavky na aplikaci a popsán kompletní návrh včetně grafického rozhraní. Čtvrtá kapitola shrnuje průběh implementace a vyzdvihuje zásadní implementované algoritmy a grafické prvky. Kapitola pátá ukončuje tvorbu aplikace celkovým testováním funkčnosti i grafického rozhraní, dále vytvořením sociálních modelů z dostupných dat a nastíněním možného směřování dalšího vývoje.

Kapitola 2

Algoritmy pro tvorbu genealogických modelů

Pro vytvoření genealogického modelu je zapotřebí algoritmus, který by dokázal propojit matriční záznamy, ve kterých participují stejné osoby. K tomu lze využít teorii o propojování záznamů. Ta se uplatňuje nejen v genealogii, ale v mnoha dalších oblastech. Příkladem může být finančnictví, podnikání či genetika. Tuto teorii formalizovali už v roce 1969 Ivan P. Fellegi a Alan B. Sunter a popsali pravděpodobnostní rozhodovací pravidlo, které je při srovnávání nezávislých atributů optimální. Vytvořili matematický model popsaný v *A theory for record linkage* [10]. Dodnes na tomto modelu staví velká část algoritmů.

V následujících odstavcích bude tato teorie velmi krátce uvedena. Program Moragen, který je pro tuto práci zásadní a který je popsán ve druhé části této kapitoly, se touto teorií totiž spíše pouze inspiruje, než by na ní přímo stavěl.

2.1 Teorie spojování záznamů

Matematický model popsaný Fellegi a Sunterem řeší následující problém. Existují soubory A a B, které obsahují nějakou množinu záznamů. Pro každou dvojici, která obsahuje jeden záznam ze souboru A a jeden záznam ze souboru B, je potřeba určit, zda se jedná o záznamy popisující identickou entitu či nikoli.

Pro účely této kapitoly bude používáno toto označení:

- A, resp. B je množina všech záznamů ze souboru A, resp. B.
- a, resp. b je jeden záznam, pro který platí $a \in A$, resp. $b \in B$
- $[a, b]$ je uspořádaná dvojice záznamů, pro kterou platí $[a, b] \in A \times B$

Pro každou dvojici $[a, b]$ platí tedy jedno ze dvou tvrzení:

1. Záznamy a, b označují identickou entitu.
2. Záznamy a, b neoznačují identickou entitu.

Cílem algoritmu je jedno z těchto dvou tvrzení dokázat. Mezi zásadní kroky algoritmu patří srovnání jednotlivých atributů záznamů a následně aplikování rozhodovacího pravidla založeného na teorii pravděpodobnosti. Výsledkem je pro každou dvojici $[a, b]$ jedno ze tří uvedených tvrzení:

(a) Záznamy a, b označují identickou entitu.

(b) Záznamy a, b neoznačují identickou entitu.

(c) Záznamy a, b možná označují identickou entitu, ale chybí dostatek důkazů.

Algoritmus může také generovat dva druhy chyb:

- Algoritmus pro dvojici $[a, b]$ vrátil tvrzení (a), ale ve skutečnosti je platné tvrzení (2)
- Algoritmus pro dvojici $[a, b]$ vrátil tvrzení (b), ale ve skutečnosti je platné tvrzení (1)

Výsledkem jsou tedy dvojice $[a, b]$, ke kterým je přiřazeno jedno z tvrzení (a), (b), (c). Následně nastává prostor pro uživatele, který dokáže charakteristiky každého záznamu lépe posoudit, aby rozhodl dvojice, u nichž pro rozhodnutí algoritmem bylo málo důkazů, a případně opravil chyby, které model vygeneruje. I přes tento nutný zásah uživatele je však algoritmus mnohem efektivnější, co se týče času potřebného na výpočet a množství chyb, než ruční práce.

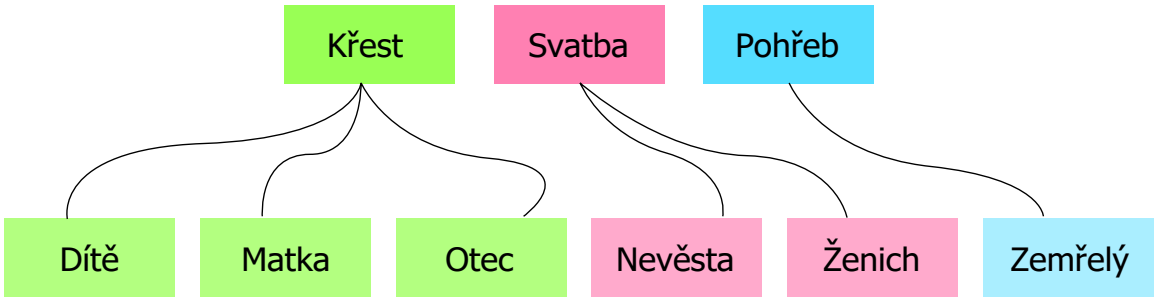
2.2 Projekt Moragen

Jedná se o program pro vytvoření genealogického modelu z matričních záznamů. Na tento projekt [22] navazuje tato práce, a proto mu bude věnována celá tato podkapitola. Jak již bylo řečeno, uvedenou teorií o propojování záznamů se algoritmus implementovaný v programu spíše pouze inspiruje. Zásadním rozdílem je fakt, že se zde nepracuje s teorií pravděpodobnosti.

Nicméně je třeba začít od začátku. Každý *záznam* v matrice je vytvořen k nějaké události. Existují tři druhy záznamů, každý záznam obsahuje datum události a jména lidí, kteří na ní participovali (případně další méně podstatné informace). Vstupem do programu jsou tedy následující data:

- Záznam o křtu – obsahuje datum, jméno a příjmení dítěte, matky a otce
- Záznam o svatbě – obsahuje datum, jméno a příjmení nevěsty a ženicha
- Záznam o pohřbu – obsahuje datum, jméno a příjmení zemřelého

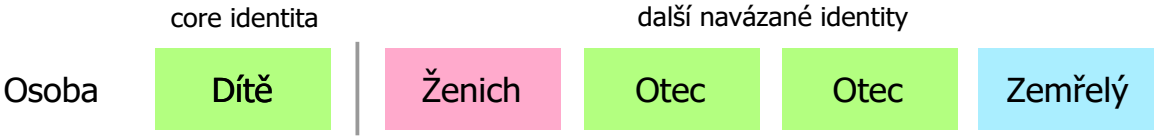
V každém ze záznamů participují lidé v tzv. *rolích*. Jak je znázorněno na obrázku 2.1, u záznamu křtu se jedná o role dítěte, matky a otce, u záznamu svatby o roli nevěsty a ženicha a u záznamu pohřbu o roli zemřelého. Formálně je role struktura, která obvykle obsahuje datum události, jméno a příjmení osoby, případně další údaje. Díky záznamům jsou mezi rolemi navázány vazby. Dítě se narodí matce a otci. Nevěsta se provdá za ženicha.



Obrázek 2.1: Ukázka vztahu mezi záznamy (nahoře) a rolemi (dole).

Prvním krokem algoritmu je tak vytvoření množiny rolí, kde každá je navázána k záznamu, ke kterému patří.

V druhém kroku je zapotřebí vyhledat role, ve kterých vystupuje stejná osoba, a propojit je vazbami. Pro každou roli je proto vytvořen objekt zvaný *identita*. Tyto identity jsou pak pomocí algoritmu spojovány do seznamů, které přísluší jedné osobě. Každý seznam je uvozen tzv. *core identitou*, která identifikuje celý seznam. Jedná se o identitu, která obsahuje první roli dané osoby v záznamech. Obvykle jde o roli dítěte při křtu. Jak je vidět, rozdíl mezi rolí a identitou je malý a pro tuto práci nepodstatný. Proto bude v následujících kapitolách používán pojem *identita* pro označení role osoby v záznamu a pojem *osoba* pro seznam identit příslušejících jedné osobě, jak je prezentováno v obrázku 2.2.



Obrázek 2.2: Ukázka vztahu mezi osobou a identitami.

Takto vznikne model. Ovšem, aby byl model genealogický, musí platit některé podmínky. Po vytvoření všech identit proto algoritmus projde každou identitu a ověří, zda platí všechna pravidla. Mezi pravidla patří například to, že žena může mít dítě 250 dní po předchozím dítěti, nebo že matka musí být mladší než 63 let a otec mladší než 83 let.

Pokud některá z rolí poruší některé z pravidel, hledá se lepší řešení pomocí algoritmu BestFS. Pokud se dané řešení nenajde, označí se daná identita jako chybná. Tyto chyby jsou pak jedním z výstupů algoritmu.

Tímto způsobem tedy z matričních záznamů vzniknou role a z těch spojováním pomocí identit vzniknou osoby. Případně mohou být generovány také nějaké chyby. Výstupem programu Moragen jsou množiny rolí/identit, množiny osob, tedy seznamů propojených identit, a množiny chyb. Nyní je potřeba umožnit uživateli s těmito výsledky pracovat, upravovat je a následně na základě provedených změn znovu přepočítat všechny vazby mezi identitami. Vyvíjením takové aplikace se zabývají následující kapitoly.

Kapitola 3

Existující systémy pro zpracování historických dat

Před začátkem vývoje aplikace je vždy důležité se důkladně seznámit se současnými systémy, které v dané oblasti existují. Může se totiž stát, že nějaká aplikace, která by částečně či zcela vyhovovala požadavkům, již existuje. V tom případě stačí použít existující aplikaci, případně ji přizpůsobit konkrétním požadavkům. Tvorba nové aplikace by byla jen nošením dříví do lesa.

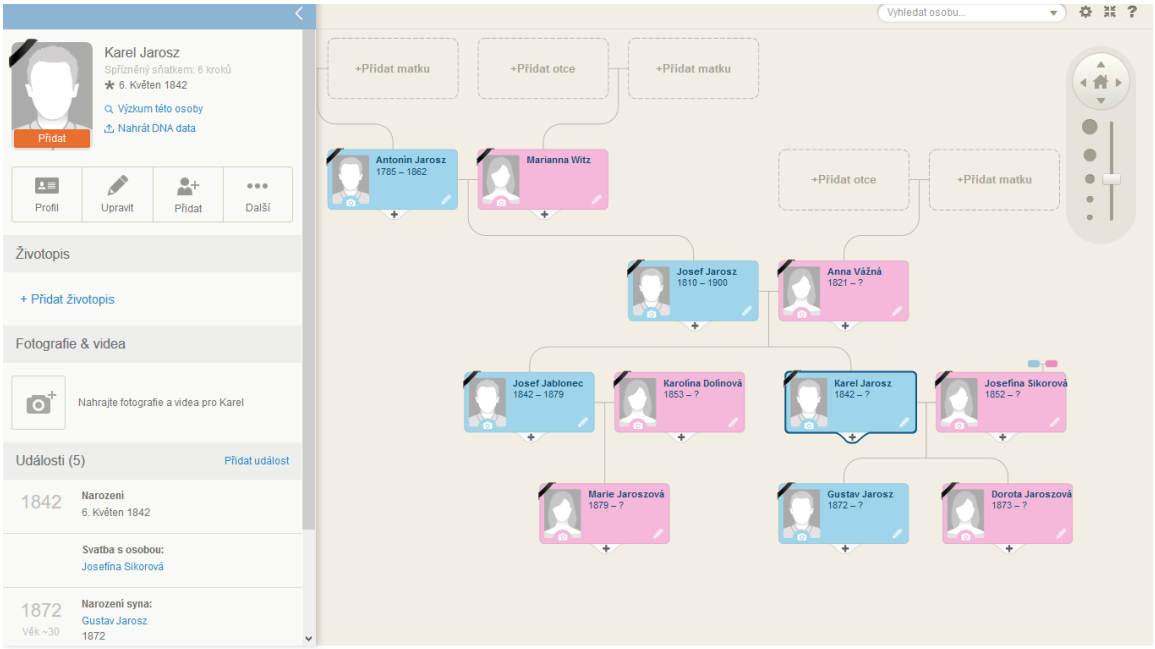
V oblasti genealogie a tvorby rodokmenů existuje dnes velké množství webových i desktopových aplikací. Drtivá většina je však velmi jednoduchá a nabízí pouze vytvoření rodokmenového stromu vkládáním jednotlivých osob. V následujících podkapitolách budou představeny tři aplikace, které jsou v dnešní době populárnější a to možná právě proto, že vybočují z trendu jednoduchých aplikací a nabízejí komplexnější řešení tvorby rodokmenů. Jak bude uvedeno, ani jednu z nich však není možné nijak využít k účelům, kterými se zabývá tato bakalářská práce.

3.1 MyHeritage

MyHeritage [14] je softwarová platforma, která umožňuje tvorbu rodokmenů. Je jedinečná v propojení rodokmenů s výzkumem DNA, díky tomu lze zjistit například etnický původ majitele daného vzorku DNA a pomocí sdílení DNA vyhledat příbuzné. V dnešní době je aplikace rozšířená do celého světa. Podle údajů na svých webových stránkách je dnes přeložena do 42 dvou jazyků a obsahuje více než 12 miliard záznamů. Ukázka aplikace je na snímku 3.1.

Kromě klasické tvorby rodokmenů, kterou nabízejí všechny podobné aplikace, MyHeritage umožňuje vyhledávání shod mezi rodokmeny jednotlivých uživatelů a propojuje tak rodokmeny uživatelů po celém světě. Umožňuje také vyhledávání v záznamech. Nejsou to jen matriky, ale různé profesní ročenky, údaje o sčítání lidu, seznamy osob z oblastí finančnictví nebo státní správy. Jak bylo uvedeno, dnes je zde 12 miliard takových záznamů. Protože jde však o celosvětovou aplikaci, jedná se jen o malý zlomek všech dostupných materiálů.

Každá osoba má v rodokmenu podrobný detail, který může obsahovat také fotografie, či rozsáhlý životopis. Osoby lze přizvat k úpravám rodokmenu pomocí e-mailových adres. Díky



Obrázek 3.1: Ukázka rodokmenového stromu v aplikaci MyHeritage. Screenshot z aplikace dostupné na [14].

tomu MyHeritage může fungovat také jako jakýsi upomínkovač narozenin a výročí. Kromě webového rozhraní existuje desktopová i mobilní aplikace.

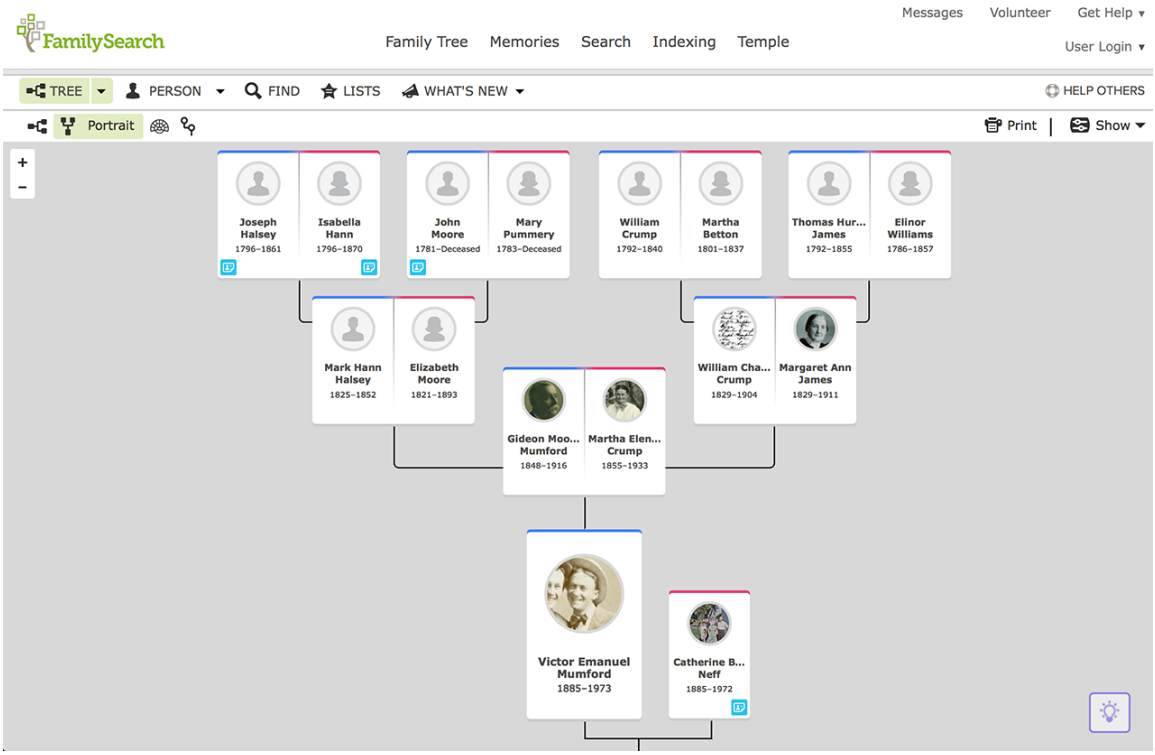
MyHeritage nabízí spoustu možností, umožňuje i vyhledávání v historických záznamech a hledání shod s osobami, které jsou v rodokmenu. Přesto osoby zůstávají hlavním prvkem rodokmenu. Není tedy možné tvořit rodokmen na základě záznamů, ale pouze v nich dohledávat potřebné informace.

3.2 FamilySearch

FamilySearch [9] je v první řadě celosvětová genealogická databáze. Obsahuje aktuálně přes čtyři biliony jmen. Mezi zdroje dat patří matriky, záznamy obyvatel ze státní správy, záznamy zaměstnanců ze soukromého sektoru, dále válečné záznamy, pozemkové soupisy apod. Záznamy jsou indexované a lze v nich vyhledávat. Tento projekt sponzoruje *Církev Ježíše Krista Svatých posledních dní*. Na vývoji se však podílí tisíce dobrovolníků různých národností i vyznání.

Přestože zpřístupnění digitalizovaných záznamů je hlavním cílem webu a celého projektu, druhou funkcionalitou je tvoření společného online rodokmenu. FamilySearch se snaží nabourat tradiční představy, kdy si každý uživatel tvoří svůj vlastní rodokmen. Tento projekt vytváří pouze jeden celosvětový rodokmen, do kterého může každý přidávat, upravovat a spojovat duplicity. Cílem je mít zde každou osobu pouze jednou. Obrázek 3.2 obsahuje ukázku tohoto rodokmenu.

Zdálo by se, že je to požadované řešení a tato bakalářská práce i program, na který navazuje, začínají ztrácet smysl. Není tomu ale tak. Přestože FamilySearch obsahuje jak záznamy, tak



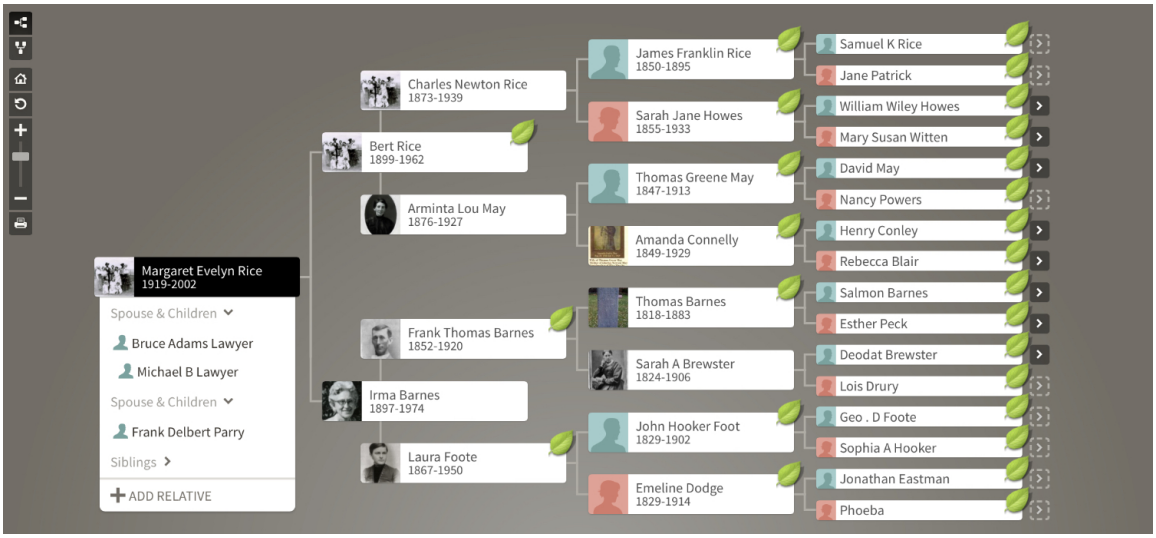
Obrázek 3.2: Ukázka rodokmenu v aplikaci FamilySearch. Obrázek převzat z [20].

rodokmen, chybí zde mezičlánek, který právě řeší projekt Moragen. A to, jak ze záznamů vytvořit rodokmeny. Ve FamilySearch je možné vkládat záznamy a vyhledávat v nich, je možné vkládat a upravovat osoby v rodokmenu. Chybí zde však možnost rodokmeny ze záznamů vytvořit. Celá genealogická databáze tak slouží pouze jako databáze, ve které lze vyhledávat a na základě vyhledaných údajů vytvářet rodokmen.

3.3 Ancestry

Ancestry [3] je projekt, který se ve velkém podobá dříve popsanému programu MyHeritage. Umožňuje standardní tvorbu a úpravy rodokmenu vkládáním jednotlivých osob. Každý detail osoby může obsahovat mnoho informací včetně fotografií. Aktuálně obsahuje Ancestry podle svých webových stránek cca 20 bilionů záznamů z matrik a dalších zdrojů, ve kterých mohou uživatelé vyhledávat a na základě výsledků pak tvořit rodokmen.

Stejně jako MyHeritage obsahuje i Ancestry propojení s DNA. Slouží zde ke stejným účelům, tedy vyhledání např. etnického původu, porovnání a hledání příbuzných napříč celým světem a nabízí také prostudování vložené DNA po zdravotní stránce. Jediný podstatný rozdíl oproti MyHeritage je dostupnost, resp. dostupnost funkcí zdarma. U MyHeritage existuje placená Premium verze, ale většina funkcí je dostupná zdarma. U Ancestry je počet bezplatných funkcí výrazněji omezen. Ukázka aplikace je na snímku 3.3.



Obrázek 3.3: Ukázka rodokmenu v programu Ancestry. Obrázek převzat z [2].

Závěrečné shrnutí

Celkově byly v této kapitole popsány komplexní systémy, které dnes existují pro práci s genealogickými daty. Kromě nich existují další projekty jako *Wiki Tree*, *Legacy Family Tree*, *Family Tree Maker* a mnoho dalších. Všechny tyto systémy mají jedno společné. Ty jednodušší vůbec neobsahují historické záznamy a pouze umožňují jednoduše vytvořit rodokmen vkládáním osob a vazeb mezi nimi. A ty složitější, které obsahují historická data z matrik a dalších materiálů, umožňují v těchto datech pouze vyhledávat, ne z nich tvořit rodokmen.

Protože potřebná aplikace zatím neexistuje, nic nebrání započetí vývoje. V následující kapitole bude popsána jeho první část, a to analýza a specifikace požadavků a návrh celé aplikace.

Kapitola 4

Analýza a návrh aplikace

Dříve než se začne vznikající aplikace přímo vytvářet, je podstatné analyzovat a specifikovat veškeré požadavky, které musí aplikace splnit. Dále je potřeba navrhnout řešení všech problémů, které aplikace řeší, a to včetně případných grafických výstupů a datových struktur. Touto částí vývoje se zabývá tato kapitola.

V první podkapitole je popsána analýza a specifikace požadavků na aplikaci. Druhá podkapitola se věnuje objektovému návrhu včetně zvolené architektury. Třetí podkapitola se zaměřuje na návrh grafického rozhraní. Poslední podkapitola se zabývá vstup-výstupním rozhraním celé aplikace, jelikož aplikace musí spolupracovat s programem Moragen.

4.1 Analýza a specifikace požadavků

Bez řádné analýzy požadavků nemůže vzniknout kvalitní aplikace. Zásadní je zejména specifikovat typ uživatele, který bude aplikaci používat, specifikovat problém, který aplikace řeší, a určit prioritu požadavků. Při vývoji se pak může totiž stát, že požadavky budou protichůdné. Následující požadavky byly analyzovány ve spolupráci s vedoucím této bakalářské práce.

Cílové uživatele výsledné aplikace můžeme rozdělit do dvou skupin. V první skupině jsou lidé, kteří budou aplikaci využívat v rámci své profese, tedy historikové, genealogové, zaměstnanci archivů apod. Tato skupina uživatelů bude náročná v požadavcích na aplikaci. Zcela zásadní bude rychlost aplikace, ideálně co nejmenší zanoření ovládacích prvků, používání klávesnice při práci s aplikací. Nebude tolik kladen důraz na příliš moderní grafické rozhraní. A vzhledem k tomu, že tito uživatelé budou s aplikací pracovat velmi často, po prvotním pochopení konceptu a ovládání programu nebude zapotřebí žádných dalších nápověd či vysvětlivek.

Aplikace může být v budoucnu veřejně dostupná. Pak budou druhou skupinou uživatelů laici, kteří se nezabývají profesně historií ani genealogií a do kontaktu s ní přišli v drtivé většině maximálně v populárně-naučné rovině. Pro tuto skupinu bude proto mnohdy těžší pochopit princip aplikace a funkce, které nabízí. Aplikace by však měla tento problém vyřešit a nabídnout jim co nejvíce pohodlné řešení, jak se s ní seznámit. Tito uživatelé s aplikací budou pracovat sporadicky, proto je zapotřebí mít vždy možnost, jak si její

užívání osvěžit. Naopak pro ně nebude tolik důležitá rychlost aplikace a efektivita při práci s ní.

U obou skupin cílových uživatelů lze očekávat, že nebudou mít velké zkušenosti v práci s počítačem. Je nutné omezit úkony uživatele spojené s instalací aplikace a propojením na program Moragen na nezbytné minimum.

Uživatel bude aplikaci typicky používat ke třem různým účelům. Tím zásadním je tvorba rodokmenů z matričních záznamů. Tedy uživatel bude požadovat, aby se mu zobrazil vytvořený rodokmen, aby se zobrazily chyby, které výpočet vygeneroval, a aby mohl rodokmen upravovat. Úprava rodokmenu znamená *uzamčení* nebo *odstranění* vazeb mezi identitami a osobami. Uzamčení identity znamená, že identita stoprocentně patří k osobě, ke které je přiřazena. Odstranění identity znamená, že identita stoprocentně nepatří k osobě, od které byla odstraněna. Po úpravě zde musí být možnost opětovně spustit program Moragen, který na základě provedených změn přepočítá vazby v celém rodokmenu. Úprava rodokmenu je hlavní problém, který má aplikace řešit.

Mezi další příklady typického použití patří vyhledávání a vytváření sociálních modelů. Uživatel bude typicky v aplikaci vyhledávat, a to na základě různých parametrů. Bude očekávat, že se mu podle zadaných parametrů zobrazí všechny výsledky, které jsou v daném rodokmenu k dispozici, a to včetně možnosti zobrazení dalších podrobných informací. Filtrování spočívá v podobné funkcionalitě. Výsledky filtrování se však nejen zobrazí, ale ostatní entity, které nesplňují parametry, se v rodokmenech zobrazovat nebudou. Typicky bude uživatel chtít pracovat pouze s jedním rodem a bude nežádoucí zobrazování větví jiných rodů, které jsou připojeny na základě sňatků.

V oblasti grafického návrhu bude uživatel obvykle vyžadovat interaktivní prostředí aplikace, jednoduché a intuitivní ovládání, zároveň ale také možnost se jednoduše dostat ke všem potřebných informacím. U druhé definované skupiny uživatelů bude také požadována nápověda, která by usnadnila ovládání aplikace.

Celkově lze všechny požadavky shrnout do těchto zásadních bodů:

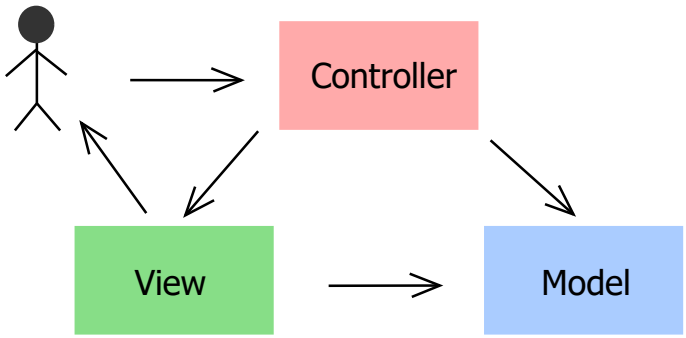
- možnost zobrazit a upravovat výsledky programu Moragen
- možnost vyhledávat v těchto výsledcích
- možnost vytvářet sociální modely
- jednoduché a intuitivní grafické rozhraní

4.2 Architektura a objektový návrh

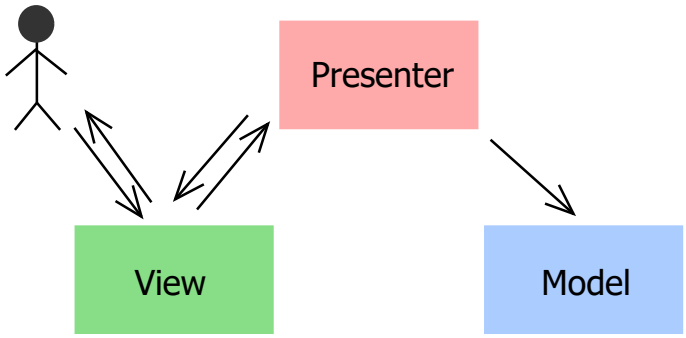
Na základě podrobné specifikace požadavků může být tvořen návrh aplikace. Pro návrh i následnou implementaci byl vybrán objektově orientovaný přístup. Po zvolení architektonického vzoru byly v modelovacím jazyce UML vytvořeny diagramy tříd. Ten základní, který popisuje datovou část aplikace, bude uveden v následujících odstavcích.

Zvolená architektura

Pro tvorbu aplikace byla vybrána architektura *Model-View-Presenter* (MVP). Následující popis vychází z [6], [5], [4] a [17]. Jak může být už z názvu patrné, architektura vychází ze známějšího vzoru *Model-View-Controller* (MVC) a někdy je mylně označována také jako MVC. Nejprve tedy několik slov k MVC a následně budou ukázány rozdíly ve srovnání s MVP. Architektura MVC je založena na trojici komunikujících objektů. *Model* reprezentuje data a metody, které s daty pracují. *View* zajišťuje grafický výstup aplikace a má vazbu na model, ze kterého načítá data ke zobrazení. *Controller* má řídící funkci, rozhoduje, kdy se který View zobrazí a jaký Model se použije. Proto má vazbu na View i Model. MVP je v tomto téměř shodné, pouze Controller se zde jmenuje *Presenter*.



Obrázek 4.1: Schéma architektury Model-View-Controller (MVC).



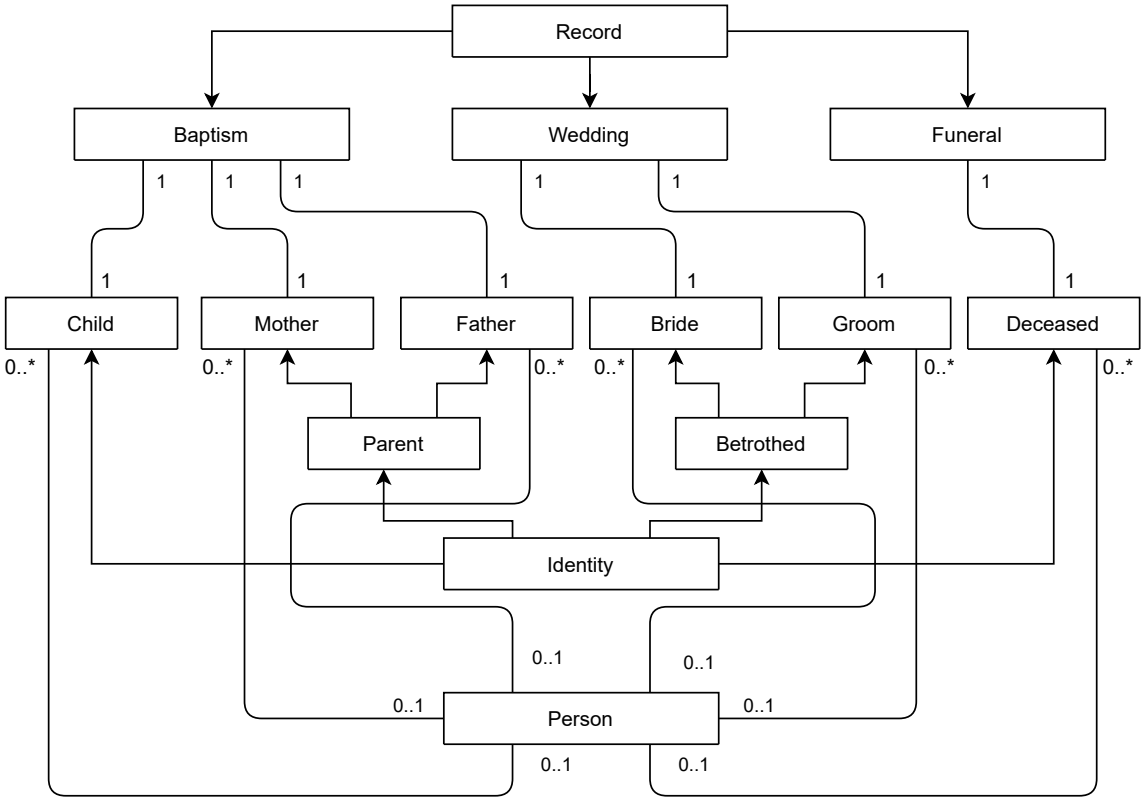
Obrázek 4.2: Schéma architektury Model-View-Presenter (MVP).

Jak můžeme vidět na obrázcích 4.1 a 4.2, zásadní rozdíl mezi MVC a MVP, tedy Controllerem a Presenterem, je to, kdo ošetřuje vstup od uživatele. U architektury MVC zachytává vstup od uživatele přímo Controller a díky němu rozhoduje, která část programu se spustí. Tento přístup je vhodnější tam, kde jsou oddělené vstupy a výstupy, což je například u PHP (vstupy jsou čteny z HTTP požadavku, výstupem je obvykle HTML). Naopak architektura MVP je vhodná tam, kde je vstup a výstup sjednocen. View totiž na zobrazeném vstupu také zachytává veškeré interakce s uživatelem a volá pak metody Presenteru. Proto View musí mít vazbu i na Presenter. Druhý zásadní rozdíl je vazba mezi Modelem a View, která u MVP byla zrušena. Veškerá komunikace mezi View a Modelem tak musí jít vždy přes Presenter. Jak bude uvedeno později, aplikace byla implementována na platformě JavaFX, kde vstupním objektem interakce s uživatelem je View, nikoli Controller/Presenter. Proto je vhodnější použití architektury MVP.

Zvolená architektura byla však ještě uzpůsobena potřebám aplikace. Jelikož se jedná o relativně malou a nikterak náročnou aplikaci, bylo zvoleno sloučení View a Presenteru. Díky zachytávání interakcí uživatele přes View, Presenter neplnil téměř žádnou funkci a pouze komplikoval volání vesměs jednoduchých metod Modelu nebo dalších Views. Toto sloučení přispělo větší efektivitě celé aplikace i lepší přehlednosti kódu. Ve výsledku tak byla použita architektura nazývaná Model-View, kterou využívá také například framework Qt [21].

Objektový návrh Modelů

První navrženou částí byly Modely a jejich vzájemná komunikace. V diagramu tříd 4.3 můžeme vidět návrh tříd reprezentující základní datové struktury aplikace, tedy záznamy, identity a osoby. Celkový diagram tříd je samozřejmě obsáhlejší, pro lepší pochopení komunikace objektů těchto tří tříd je zde uvedena pouze jeho část. V diagramu jsou uvedeny pouze názvy tříd, nikoli jejich atributy a metody.



Obrázek 4.3: Diagram tříd zobrazující základní strukturu modelů a vazby mezi nimi. Neobsahuje všechny třídy ani atributy a metody zobrazených tříd.

Pro každý typ záznamu byla vytvořena třída, která jej definuje – `Baptism` pro křest, `Wedding` pro svatbu, `Funeral` pro pohřeb. Každá třída má své specifické atributy na základě toho, co je v matrice u daných záznamů typicky uvedeno. Pro zjednodušený přístup k záznamům bez ohledu na typ, byla vytvořena třída `Record`, ze které všechny záznamy dědí.

Stejně byly navrženy i identity. Opět existuje pro každý typ identity speciální třída – `Child` pro dítě, `Mother` pro matku, `Father` pro otce, `Bride` pro nevěstu, `Groom` pro ženicha, `Deceased` pro zemřelého. Kromě toho matka a otec mají některé vlastnosti shodné, stejně

jako nevěsta a ženich. Pro lepší přístup k identitě rodiče, resp. snoubence, bez ohledu na pohlaví byly vytvořené třídy `Betrothed` pro snoubence/novomanžele a `Parent` pro rodiče. Tyto třídy získaly několik atributů na úkor matky a otce, resp. nevěsty a ženicha, a třídy `Mother`, `Father`, `Bride` a `Groom` z nich dědí. Stejně jako u záznamů i zde je vytvořena obecná třída `Identity` a všechny identity dědí z této třídy.

Každý záznam pak vlastní příslušné identity. `Baptism` obsahuje identitu `Child`, `Mother`, `Father`. `Wedding` obsahuje `Bride` a `Groom`. `Funeral` vlastní objekt třídy `Deceased`. A obráceně, každá identita patří jednomu záznamu.

`Person` je poslední třída v diagramu. Popisuje konkrétní osobu v rodokmenu. Osoba je definována identitami, proto i třída `Person` vlastní seznam identit, které ji přísluší. Naopak každá identita je obvykle přiřazena k jedné konkrétní osobě, případně není přiřazena k žádné osobě. Kromě identit má osoba jen velmi málo charakteristik, protože veškeré vlastnosti vycházejí z přiřazených identit.

Jak bylo řečeno kromě tříd `Record`, `Identity` a `Person` a tříd, které z nich dědí, existují ještě další Modely. Jednou z nich je třída `Model`, ze které dědí všechny tři již představené rodičovské třídy. Tato třída obsahuje například `id` každého modelu, podle něj pak lze identifikovat jednotlivé záznamy, identity a osoby. Další je třída `FamilyTree`, která reprezentuje celý rodokmen a drží seznamy všech záznamů, identit a osob. Třída `Submodel` obsahuje jméno submodelu a objekt třídy `Filter`. Ten nese všechny parametry daného vyhledávání či filtrování. Vstupy a výstupy reprezentují třídy `RecordFile`, `IdentityFile`, `PersonFile`, `ErrorFile` a `UpdateFile`, o nich bude řeč později. Poslední důležitou třídou je třída `Prolog`, která slouží pro komunikaci s programem Moragen.

Objektový návrh Views

Druhou částí objektového návrhu byly třídy zastávající v architektuře funkci Views. Každá taková třída dědí z třídy `View`, resp. `Views`. Tato základní rodičovská třída má jako jeden z atributů `node` datového typu `Node`, který vrací rodičovský uzel grafického zobrazení, a je tak možné vytvořenou část grafiky zobrazit na libovolném místě vložením uzlu `node`. Třída `Views` může těchto uzlů obsahovat více.

Jak již bylo řečeno, třídy typu View a Presenter byly sloučeny do jednoho. Viditelné to je však pouze u některých tříd typu View, které obsahují i řídící funkce. Typicky se jedná o třídy, které mají jako jeden z atributů objekt třídy typu Model. Takové třídy pak také obsahují metody `getParent()` pro vrácení objektu, který vytvořil volaný objekt, a `getModel()` pro zpřístupnění modelu. Je tak zajištěna dostupnost hlavních objektů a omezena redundance atributů u jednotlivých tříd. Kromě těchto tříd je zde však velká část tříd typu View, které pouze zobrazují nějakou část grafického rozhraní a nemají žádnou vazbu na Modely ani žádné řídící funkce. Jedná se tedy čistě o třídy typu View, nikoli Presenter.

4.3 Návrh grafického rozhraní

Návrh vhodného grafického rozhraní je u aplikace tohoto typu zcela zásadní. Pro začátek je potřeba shrnout klíčové prvky GUI, které musí být nějakým způsobem zohledněny v návrhu, a na kterých pak bude stát akceptační testování. Tím hlavním řešeným problémem je zob-

razení výsledků programu Moragen a jejich úprava. Ideální řešení by obsahovalo rodokmen sestavený podle osob. Každá osoba by měla přiřazené identity. Tyto identity by se daly přesouvat k jiným osobám. Možné je využití přístupu *Drag&Drop*. Jedná se o techniku, kdy uživatel přesouvá objekt po obrazovce tažením myši, uvolněním tlačítka myši zvolí správnou pozici objektu [23]. Každá identita by měla jít také nějak uzamknout k dané osobě. Toto uzamčení by mělo být graficky viditelné. Odstranění identity by mohlo proběhnout právě přesunutím identity jinam. Následně by bylo možné jednoduše spustit výpočet.

Druhá část se týká vyhledávání. Zde by uživatel mohl požadovat jak rychlé vyhledávání zadáním pouze několika písmen, tak komplexní variantu hledání s vícero konkrétními parametry, které by bylo potřeba zadat. U výsledků by mělo být možné, jednak je vykreslit jako rodokmen, jednak zobrazit detail každé osoby. Třetí část typického využití obsahuje filtrování a vytváření sociálních modelů. Tato oblast byla nejméně specifikována v požadavcích. Hlavním cílem tak bude zobrazení pouze omezené části rodokmenu.

V následujících odstavcích bude popsán vznik prvního návrhu GUI a jeho prototyp. Následně testování návrhu, které proběhlo a jaké byly jeho výsledky. Poslední částí bude finální návrh a s ním druhý prototyp grafického rozhraní.

Návrh a první prototyp

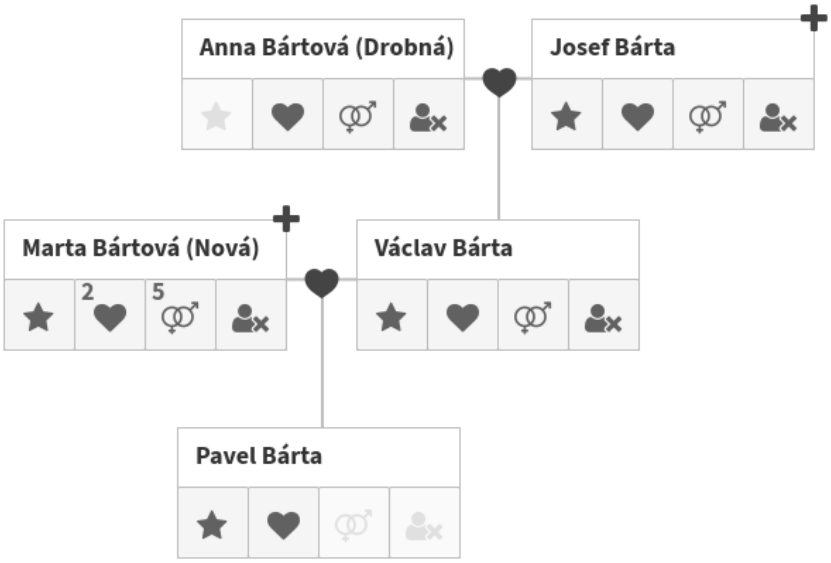
Z uvedených klíčových prvků vyplynula potřeba mít v grafickém rozhraní plochu pro zobrazení rodokmenu a ideálně co nejvíce akcí moci spustit přímo z rodokmenu. Zároveň zde ale byly různé funkcionality, které bylo potřebné spouštět z jakéhosi ovládacího panelu, a data, která bylo potřeba zobrazit mimo rodokmen (například detaily osob a záznamů nebo výsledky vyhledávání).

První otázkou tedy bylo, jak tyto tři oblasti uspořádat. Zvažováno bylo umístění ovládacího panelu v horizontální poloze v horní části obrazovky. Tím by totiž vznikla větší plocha pro úpravy rodokmenu. Zároveň data mimo rodokmen by se musela zobrazovat ve vyskakovacích oknech a tím by se plocha pro zobrazení rodokmenu výrazně omezila. Také většina ovládání by byla hůře dostupná a neviditelná na první pohled. Z těchto důvodů převážily výhody umístění ovládacího panelu vertikálně na levou stranu obrazovky. Sloučením ovládacího panelu a prostoru pro zobrazení dat mimo rodokmen vznikne panel, který sice trochu omezí plochu s rodokmenem, ale zase je veškeré ovládání pohodlnější. Díky tomu, že celý panel půjde zavřít, bude možné mít plochu s rodokmenem na celé obrazovce.

Následně bylo veškeré potřebné ovládání a zobrazitelná data rozděleny do tří záložek. Záložka *Rodokmen* obsahuje akce spojené s celým projektem, jako Otevřít, Uložit, Zavřít. Záložka *Výpočet* zobrazí kromě možnosti nového výpočtu rodokmenu také seznam chyb. Záložka *Modely* pak umožní vytvářet sociální modely. Bokem od těchto tří záložek je vyhledávání, které může být rychlé nebo rozšířené. Výsledky vyhledávání i detaily osob se zobrazí také v levém bloku.

Zobrazení rodokmenu bude stejné, jak jsou na něj uživatelé zvyklí z jiných programů a jak je vykreslen na obrázku 4.4. Tedy jednotlivé bloky se jménem symbolizují osoby a pomocí čar jsou tyto osoby propojeny vazbami. Věkové rozložení je zvoleno takto – nahoře nejstarší generace, dole nejmladší generace. Každá osoba obsahuje kromě jména také identity. Ty mohou být u každé osoby čtyř typů – Dítě, Nevěsta nebo Ženich, Matka nebo Otec, Zemřelý. Každý typ je symbolizován jedním blokem s ikonou. Pokud je identit stejného typu k osobě

přiřazeno více, je zde také číslo, které udává daný počet. Po rozkliknutí jsou vidět jednotlivé identity.



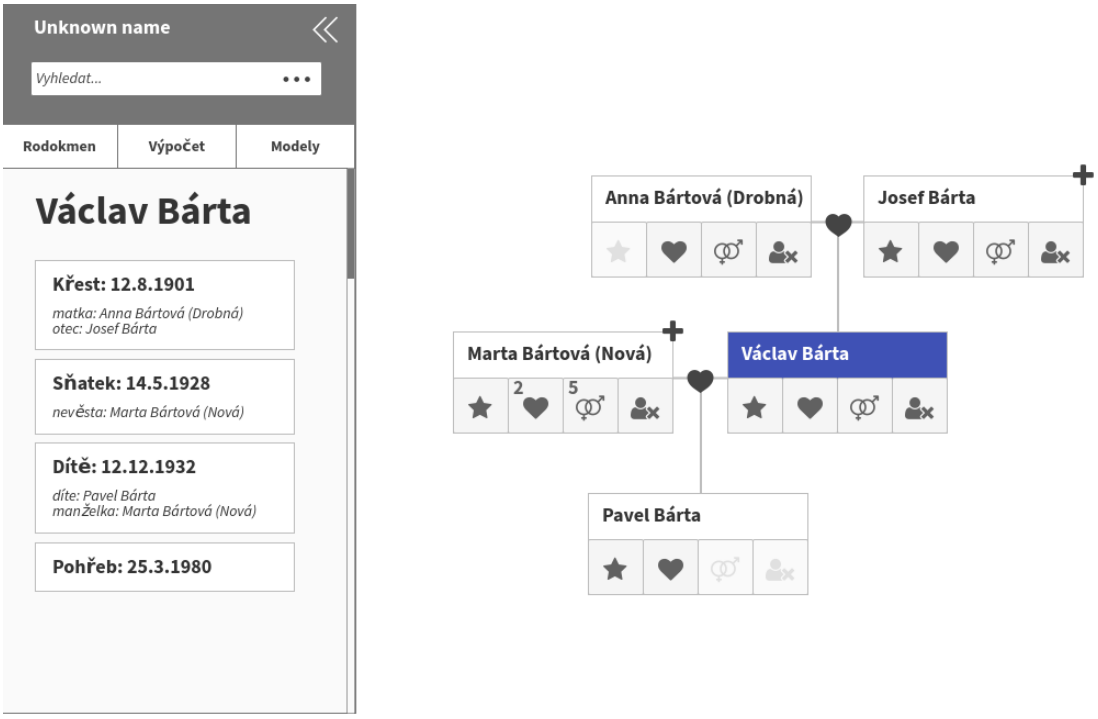
Obrázek 4.4: Návrh zobrazení osob a identit v rodokmenu.

Po vytvoření prvních papírových náčrtů byl vytvořen prototyp grafického rozhraní pro deset různých obrazovek. Prototyp byl vytvořen pomocí webu https://mockflow.com/. Pro ilustraci prvního prototypu je zde vložen snímek 4.5 s obrazovkou, na které je zobrazen detail pro jednu z osob v rodokmenu.

Testování a hodnocení prototypu

Na vyhotoveném prototypu bylo provedeno testování. Pro testování byla vybrána skupina sedmi uživatelů různého pohlaví i věku s různým zájmem o historii a různými zkušenostmi v práci s počítačem. Nejprve byla každému uživateli představena aplikace a vysvětlena souvislost mezi záznamy, identitami a osobami. Následně každý obdržel dotazník s ukázkami prototypu. Otázek bylo celkem třicet a daly by se rozdělit do čtyř okruhů:

- smysl ikon a tlačítek – zda se pod tlačítky a ikonami skrývala funkcionalita, kterou tam uživatel očekával
- srozumitelnost zobrazených dat – zda ze zobrazených dat byl uživatel schopen správně vyčíst požadované informace
- intuitivní reakce – kde by uživatel očekával danou funkcionalitu, co by byla jeho první reakce
- grafické rozhraní – jak byl uživatel celkově spokojen s grafickou stránkou aplikace, zda na něj působila dobrým dojmem a rád by se k ní vracel



Obrázek 4.5: První prototyp grafického návrhu – obrazovka s detailem osoby.

Poznatky, které vyšly z výsledků, budou rozebrány opět podle jednotlivých oblastí. Ikony reprezentující identity byly správně přiřazeny v přibližně polovině případů. Nejasný byl zejména rozdíl mezi ikonou snoubence a rodiče. Názvy jednotlivých záložek také nejsou příliš vystihující, tedy kromě záložky Výpočet, jejíž význam je pro uživatele zcela jasný. U jednotlivých položek ve vyhledávání nebo modelů je pro uživatele zřejmé, jaké akce s nimi lze provádět.

Ze zobrazeného rodokmenu byla část uživatelů zmatena. Nedokázali správně vyčíst požadované informace ve více než polovině případů. Naopak výsledky vyhledávání byly pro drtivou většinu naprosto zřejmé. Do této části dotazníku se také mohlo výrazně promítat nepochopení konceptu záznamů, identit a osob. Intuitivní reakcí uživatele bylo často pouhé kliknutí na daný objekt. A to i pro dvě různé funkcionality. V rodokmenu nebylo také zřejmé, jak vykreslit rodokmen pro danou osobu. Co se týče grafické stránky aplikace, byla hodnocena spíše kladně. Vytýkána jí byla chybějící barevnost. Většina uživatelů měla z aplikace smíšené pocity.

Z uvedených výsledků testování byly vyvozeny následující závěry:

- ikony a tlačítka mají často nesrozumitelný význam, nejsou uživateli pochopeny
- funkce by mělo být možné spustit jednoduše kliknutím na daný objekt, ne složitým nastavováním
- aplikace by mohla mít modernější a barevnější vzhled, aby se s ní lépe pracovalo

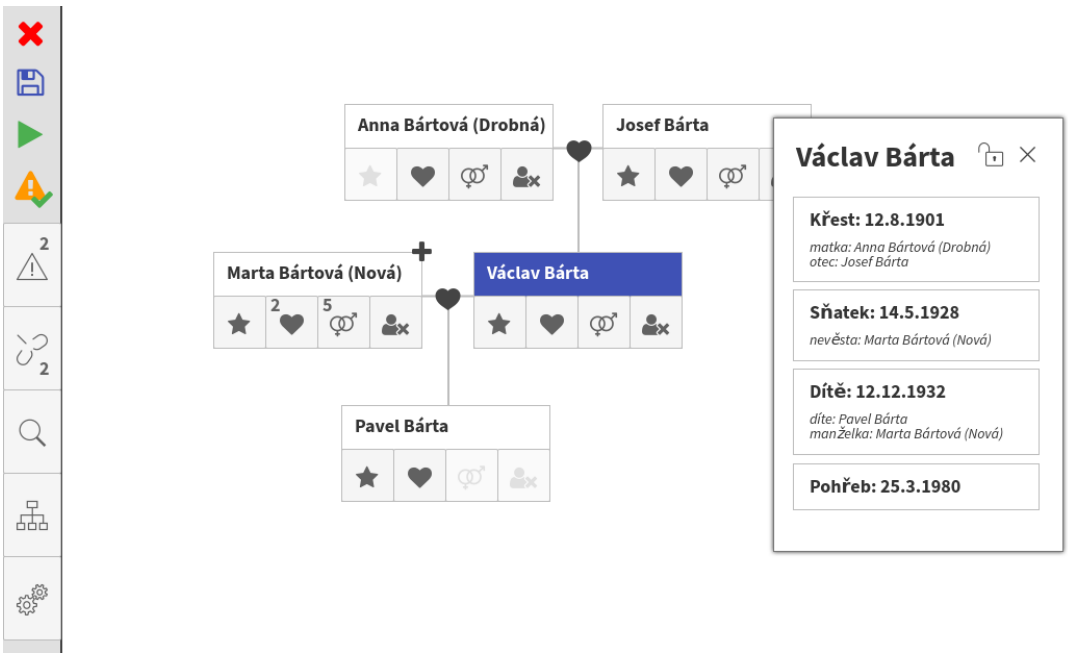
Tyto body byly následně zohledněny a zapracovány do návrhu grafického rozhraní a projevily se ve druhém prototypu.

Úpravy návrhu a druhý prototyp

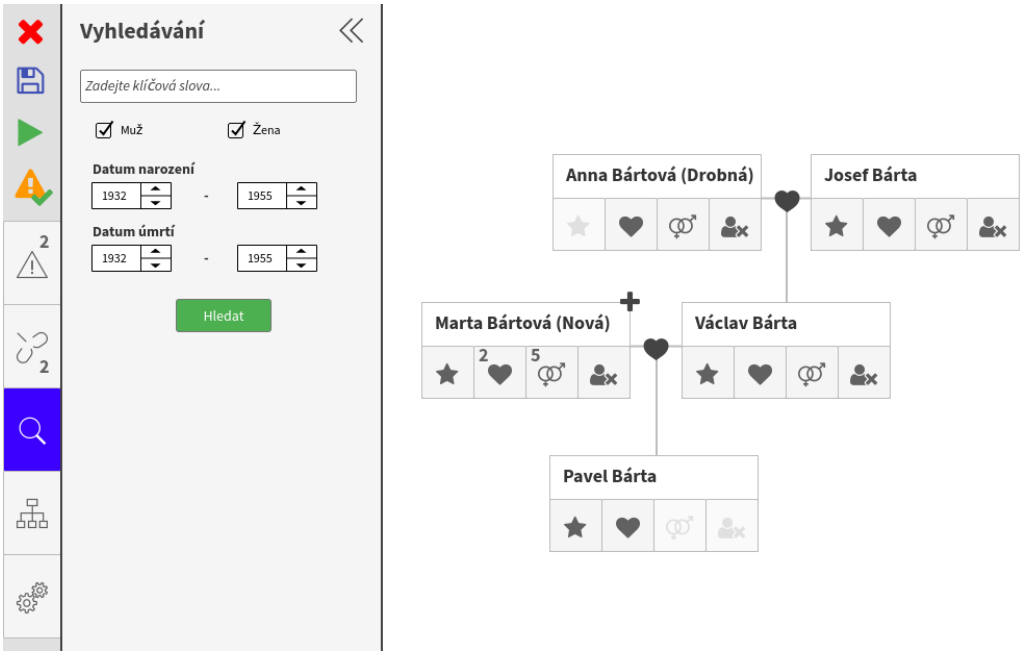
Zásadních úprav po testování návrhu GUI bylo několik. První úprava zcela přeskupila ovládání a vznikl úzký pruh ovládacího panelu na levé straně. Zde byla umístěna samostatná tlačítka, která souvisí s celým projektem – *Zavřít*, *Uložit*, *Vytvořit rodokmen*, *Zkontrolovat rodokmen*. A pod nimi jsou záložky s obsahem – *Seznam chyb*, *Nepřiřazené identity*, *Vyhledávání*, *Sociální modely* a *Nastavení*. Zmizelo tedy smíchání ovládacích prvků a dat v jednotlivých záložkách. Nyní je jasně vidět, kde jsou data a kde ovládání aplikace. Díky tomuto ovládacímu panelu se také opticky zvětšil prostor pro rodokmen, protože záložky není nutné mít otevřené tak často, jako tomu bylo v předchozím návrhu u celého panelu. U ovládacího panelu a záložek byla také změněna grafika, aby aplikace působila lepším dojmem.

Dále pro pohodlnější práci s rodokmenem byly vytvořeny detaily v režimu vyskakovacích oken. Detail osoby či záznamu je možné zobrazit bez otevření záložky. Přesouvání identit je pak navrženo přístupem Drag&Drop. Identitu tedy lze vzít a přesunout k požadované osobě. U každé osoby a identity budou po kliknutí zobrazeny možnosti. To reflektuje intuitivní reakci uživatelů při provádění jakékoli akce. U identit byly navržené odlišné ikonky, aby bylo zřejmé, zda se jedná o rodiče či snoubence.

Na základě těchto změn byl vyhotoven druhý prototyp. Pro ukázku jsou zde uvedeny dva snímky. Na snímku 4.6 je opět vykreslen otevřený detail dané osoby. Na snímku 4.7 je otevřena záložka Vyhledávání.



Obrázek 4.6: Druhý prototyp grafického návrhu – obrazovka s otevřeným detailem osoby.



Obrázek 4.7: Druhý prototyp grafického návrhu – obrazovka s otevřenou záložkou Vyhledávání.

4.4 Vstupy a výstupy programu

Vstupy a výstupy jsou pro vyvíjenou aplikaci zcela zásadní, bez nich by nemohla fungovat. Vstupem jsou jednotlivé soubory. Ten první obsahuje opsané záznamy z matriky. Ostatní tři jsou výstupem programu Moragen. Jak bylo uvedeno ve druhé kapitole, výstupem programu jsou tři množiny – množina rolí, množina identit a množina chyb. Každá tato množina je reprezentována jedním souborem. Výstupem této aplikace je soubor se změnami, který slouží pro přepočítání rodokmenu v programu Moragen.

Původně byl jako vstup i výstup zamýšlen ještě jeden soubor, ve kterém by se nacházel uložený projekt. Pro uživatele je totiž důležité mít možnost uložit rozdělanou práci a následně ji znovu načíst. Nicméně po zvážení byl tento soubor sloučen se souborem se změnami, který se vytváří pro potřeby programu Moragen. Uložená data totiž mohou být ve stejném formátu a je zde potřeba navíc ukládat pouze jeden typ dat, který může program Moragen ignorovat. Celkem jde tedy o pět souborů, které budou v této podkapitole popsány.

Vstupy

Soubor se záznamy má povinnou příponu GPM. Formát souboru je závislý na programu Moragen, který je imlementovaný v SWI Prologu, proto jsou jednotlivé záznamy uvedeny formou predikátů. Soubor obsahuje na každém řádku jeden predikát se záznamem. Podle typu záznamu existují predikáty `brecord` pro křest, `mrecord` pro svatbu a `zrecord` pro pohřeb. Každý záznam obsahuje jednotlivé charakteristiky uvedené v argumentech záznamu. Pro propojení s ostatními soubory je důležitý zejména první argument, který uvádí ID záznamu. Zde jsou uvedeny ukázky všech tří predikátů:

```
brecord(689,",16,9,1809,",'Jan','Svoboda','Tomas',",'Marie','Grec',",",").
```

```
mrecord(1850,'O2707_P298',22,11,1882,'TOMAS','SALAMOUN','JOSEF','KATERINA',
",23,",'KUZMIC',",",",",25,'N2703_CH948','N2703_CH844').
```

```
zrecord(268,11,7,1695,'MANZELKA','HELENA',",'SALAMOUN',",41,0,0,").
```

Soubor s rolemi je generován programem Moragen. Vždy je pojmenován jako *roles.mrg*. Přípona MRG je stejná pro všechny soubory generované programem. Soubor s rolemi obsahuje všechny role/identity, které byly vytvořeny z vložených záznamů. Data jsou uvedena také formou predikátů. Důležité pro vyvíjenou aplikaci jsou tři atributy, ten první u predikátu `identity`, který udává ID role/identity, a první a druhý u prvního predikátu `role`, ty udávají ID záznamu, ke kterému role patří, a typ role. Následuje opět ukázka:

```
identity(25,role(27,child,1,5,[1667,1667],'Urban','Kura',[]),
role(25,100),[identity(25,100)]).
```

Soubor s osobami, resp. core identitami, obsahuje jednotlivé identity sloučené pod core identity. Soubor nese název *coreFamilies.mrg* a je opět generován programem Moragen. Identity jsou v souboru identifikovány pomocí ID, které odkazuje na roli v souboru roles. V souboru jsou celkem tři typy predikátů. Pro vyvíjenou aplikaci jsou důležité predikáty `coreIdt`. Jako první atribut je uvedeno ID core identity, jako druhý seznam ID identit, které jsou k core identitě přiřazeny, jako je vidět v následující ukázce:

```
coreIdt(10,[10,1897,1904,1912,1921,1929,1937]).
```

Posledním souborem, který generuje program Moragen je soubor s chybami. Tento soubor nese název *semanticCheck.mrg* a není nutným vstupem do aplikace, pouze umožní zobrazení chyb v rodokmenu. Obsahuje spoustu dat, pro aplikaci jsou důležité řádky s predikáty `mw_errors`. Ty obsahují jako první atribut ID identity, ke které je chyba přiřazena. Ukázku formátu je možné vidět zde:

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

Výstupy

Výstupem programu je soubor nazvaný *updates.mrg*. Ten se vytvoří při uložení programu a z něj pak může program Moragen čerpat informace o provedených změnách. Jak již bylo zmíněno, tento soubor obsahuje všechna data potřebná k uložení projektu. Při opětovném otevření projektu je soubor znovu načten a jedná se tedy i o vstup. Primárně jde však o výstup aplikace, proto je zařazen pod část Výstupy. Obsahuje několik druhů predikátů. Predikát `mw_force` uvádí, která identita byla uzamčena k jaké core identitě. Prvním atributem tohoto predikátu je ID core identity, druhým ID uzamčené identity. Stejné atributy má i predikát `mw_exclude`, ten však udává naopak, u které core identity byla identita zakázána. Predikát `submodel` je pouze pro účely uložení aplikace a obsahuje v atributech parametry jednotlivých submodelů. Zde jsou uvedeny ukázky všech tří predikátů:

```
mw_force(42,215).
```

```
mw_exclude(82,112).
```

```
submodel('rodina Beránek',[",",'Beránek',",",-1,-1,-1,-1,",","]).
```

Kapitola 5

Podstatné části implementace

Cílem této kapitoly není popsat každý řádek kódu. Proto se nebude zabývat celou aplikací a každou třídou či metodou. Nejedná se ani o návod na použití aplikace. V následujícím textu budou pouze vyzdviženy podstatné implementované algoritmy a funkcionality a krátce shrnut průběh implementace a problémy, které se vyskytly. První podkapitola se věnuje vybranému jazyku a prostředí, ve kterém implementace probíhala.

5.1 Vybraný programovací jazyk a prostředí

Pro implementaci této aplikace byl vybrán jazyk Java [15]. Následující text vychází z článků [8], [7] a [18] a také z oficiálních webů jednotlivých jazyků a frameworků, které budou zmíněny na příslušných místech v textu. Jedná se o jazyk s virtuálním strojem, který vyvíjí firma Oracle. Znamená to, že při kompilaci je vytvořen mezikód. Ten je následně při spuštění aplikace překládán interpretem na strojový kód. A právě daný interpret se nazývá virtuální stroj. V případě Javy se jedná o Java Virtual Machine (JVM). Tento typ jazyka kombinuje kompilované a interpretované jazyky, a je tedy jakýmsi kompromisním řešením. Nese výhody kompilovaných jazyků, jako rychlost výsledné aplikace či nepřístupnost zdrojového kódu, stejně jako výhody interpretovaných jazyků, jimiž jsou snadná přenositelnost, stabilita a editace.

Aby bylo možné spustit programy v jazyce Java, je potřeba mít nainstalované tzv. běhové prostředí Java Runtime Environment (JRE), které má v sobě zabudovaný virtuální stroj. Pro vývoj aplikací je však potřeba Java Development Kit (JDK), které obsahuje JRE a další knihovny a nástroje potřebné pro vývojáře. Pro implementaci této aplikace bylo využito JDK verze 13.0.2. Aplikace však byla zkompilována pro verzi 8. Pro její spuštění je tak potřeba mít nainstalované JRE minimálně verze 8.

Pro implementaci grafického rozhraní se používají různé frameworky. V dnešní době jsou nejznámější Swing a JavaFX [12]. Swing je framework, který umožňuje jednoduchou tvorbu formulářových aplikací. Díky vlastním funkcích implementovaným v Javě může fungovat stejně na všech platformách, je však trochu pomalejší. JavaFX je modernější framework vhodný pro okenní aplikace s vizuálním obsahem (hudba, obrázky, grafy). V dnešní době je však vyvíjen open-source komunitou a není již součástí JRE/JDK. Proto je potřeba jej

doinstalovat. Pro tuto aplikaci bylo vybrán framework JavaFX, a to především proto, že je vhodnější pro obsáhlejší aplikace. Vyvíjena byla aplikace ve verzi JavaFX SDK 13.0.2.

Veškerá implementace probíhala v prostředí Netbeans [19] ve verzi 11.2. Skripty potřebné pro spuštění programu Moragen byly implementovány v jazyce Prolog v prostředí SWI-Prolog [1]. Pro ikony byla využita knihovna Font Awesome [11] v úpravě pro JavaFX ve verzi 8.9.

5.2 Struktura a průběh implemetnace

Práce na aplikaci ve fázi implementace začala stejně jako návrh od tříd typu Model. Nejprve byly implementovány veškeré funkcionality, které aplikace umožňuje. Teprve ve druhé fázi bylo implementováno i grafické rozhraní. Jak bude podrobněji popsáno v příští kapitole, implementací se prolínalo i jednotkové testování dokončených tříd. Během implementace došlo také k drobným změnám a úpravám původního návrhu. Nejviditelnější změnou je režim dvou oken, který bude uveden v podkapitole Užitečné grafické prvky.

Co se týče implementace objektového návrhu, aplikace byla rozdělena do balíků `models`, které obsahují všechny třídy typu Model, a `views` s třídami typu Views. V balíku `views` je další podstatný balík helpers, který obsahuje podpůrné grafické prvky a zobrazení, které v sobě nenesou roli Presenteru. Balíky jsou dále členěny podle potřeby. Celá aplikace nese název `Moragen GUI` a spouští se třídou se stejným názvem. Z důvodů úspěšné kompilace byla vytvořena zvlášť třída `Starter`, která rozšiřuje třídu `Application` a zajišťuje spuštění grafického rozhraní.

Implementace balíku `models` proběhla stejně, jak bylo uvedeno v návrhu. Vyzdvihnout si zaslouží způsob získávání dat o osobě z identit, který v návrhu nebyl zmíněn. Je realizován v celku jednoduše. Třída `Identity` obsahuje metody `get` pro všechny vlastnosti, které může identita osobě poskytnout (jméno, příjmení, datum narození, datum úmrtí atd.). Všechny jsou však nastaveny na hodnotu `null`. U tříd, které z třídy `Identity` dědí, jsou pak přepsány ty metody, které mohou poskytnout data z přiřazeného záznamu. Ve výsledku tak objekt třídy `Person` vlastní seznam identit a nad každou může zavolat například metodu `getBirthDate()`. Ovšem pouze pokud je identita typu `Child`, vrátí se nějaká hodnota, ostatní identity budou vracet hodnotu `null`. Takto může osoba získat všechny potřebné vlastnosti.

Vyhledávání a filtrování je implementováno pomocí třídy `Filter`, která nese parametry vyhledávání a díky metodě `pass` určuje, která identita či osoba daným filtrem projde. Tímto způsobem lze získávat vyfiltrované seznamy osob či identit, které vyhovují daným parametrům. Třída `Prolog` bude podrobněji zmíněna níže.

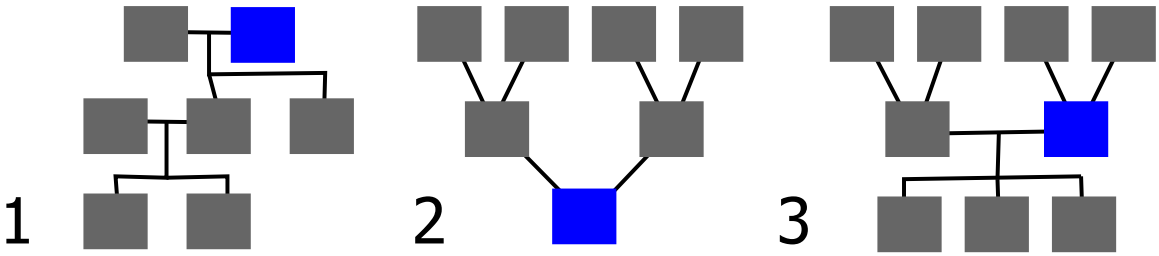
Implementace balíku `views` není v ničem překvapivá. Proběhla, jak bylo uvedeno v návrhu. Speciální prvky či efekty, které se zde používají, budou uvedeny níže.

5.3 Algoritmus pro grafické vykreslení rodokmenu

Algoritmus popsaný v této podkapitole je velmi důležitý pro vzhled celé aplikace. Zajišťuje klasické stromové vykreslení rodokmenu, na které jsou uživatelé zvyklí z jiných aplikací.

Před začátkem tvorby algoritmu bylo stanoveno, že rodokmen bude vykreslen vždy pouze pro jednu osobu, tedy nebude zobrazen celý rodokmen, ale vždy jen jeho část. Zobrazení celého rodokmenu by bylo velmi náročné na vykreslení a logicky by často bylo nemožné vykreslit v rodokmenu každou osobu pouze jednou. Rodiny jsou totiž většinou vzájemně propojeny a různé osoby by bylo potřeba vykreslit na více místech. Rodokmen by tak byl velmi nepřehledný. Také bylo rozhodnuto, že generace budou vykresleny horizontálně a to od nejstarší po nejmladší. Tedy, že nahoře budou nejstarší členové dané větve rodokmenu, o úroveň níže jejich děti, poté vnuci atd.

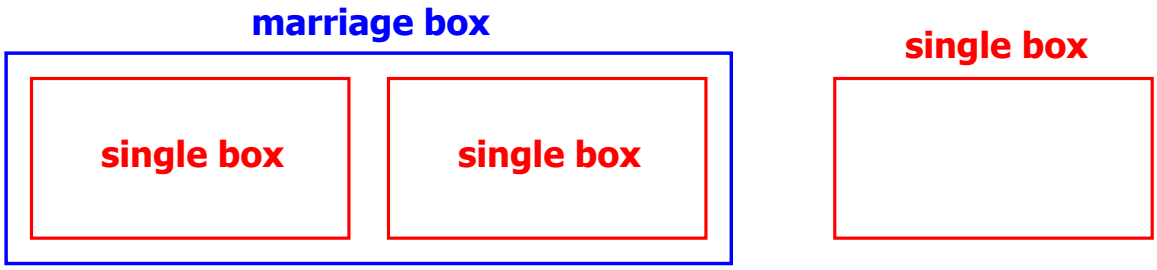
Dále byly zvažovány tři varianty stromu, jak je vidět na obrázku 5.1. První možností bylo vykreslit pouze strom směrem dolů. Tedy, že by nejstarší generaci tvořila vybraná osoba, pro níž je rodokmen vykreslen a její partner/ka. Dále by byly vykresleni jejich potomci – přímí i vzdálení. Jinak řečeno, u každé osoby v rodokmenu by byli vykresleni její přímí potomci. Druhá možnost byla opačná, tedy vykreslení stormu směrem nahoru. Zde by naopak vybraná osoba tvořila nejmladší generaci a zobrazeni by byli její rodiče, prarodiče atd. Můžeme tedy říci, že pro každou osobu v rodokmenu by byli vykresleni pouze její rodiče. Třetí zvažovanou možností bylo spojení obou stromů, tedy že by byli vykresleni jak potomci, tak i předci vybrané osoby a jejího partnera/ky. Protože uživatelé mohou očekávat všechny varianty, byla vybrána třetí, která zahrnuje obě dvě předchozí. Bylo také rozhodnuto, že vykreslení bude omezeno na dvě generace do minulosti a dvě generace do budoucnosti. Celkem tedy bude zobrazeno pět generací (pokud budou osoby v rodokmenu existovat).



Obrázek 5.1: Zvažované možnosti grafického rozložení rodokmenu.

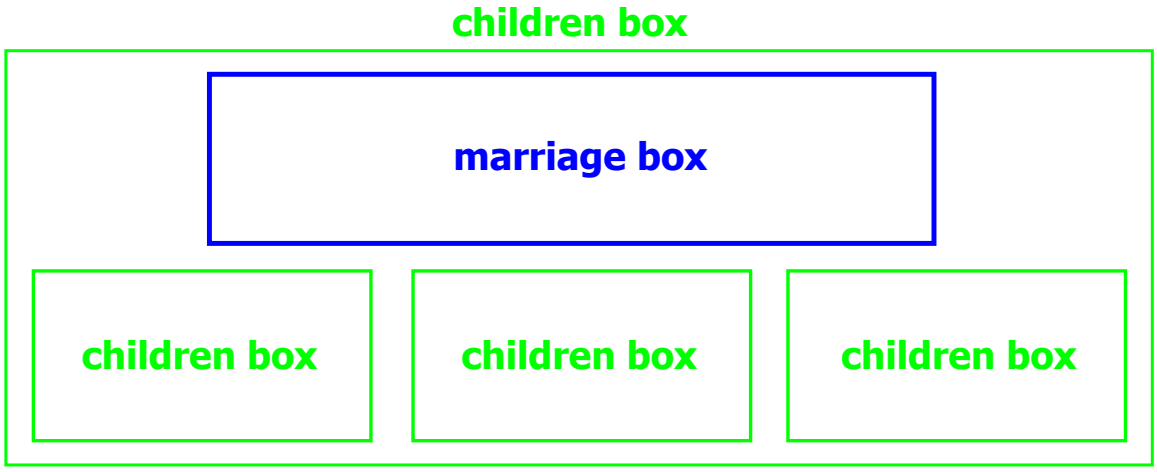
Zásadním požadavkem na algoritmus bylo, aby vykreslil všechny osoby na správných místech, bez zbytečných mezer, ale tak, aby se nepřekrývaly. Zvažovány byly různé varianty výpočtu, například rozřazení osob do jednotlivých generací, a následné vykreslování od generace s největším počtem osob. Další možností bylo postupné vykreslování a udržování seznamů osob v jednotlivých generacích a jejich posouvání v seznamech i na obrazovce při přidávání dalších osob. Vybrán byl nakonec systém boxů, zejména pro svou snadnou implementovatelnost pomocí objektově orientovaného přístupu. Základní princip je tedy postaven na boxech, které mají svou šířku, výšku a souřadnice x a y.

První typ boxu je Single Box, jedná se o box přesně pro jednu osobu. Tyto Single Boxy jsou sdružovány do Marriage Boxů, které reprezentují manželství a obsahují vždy dva Single Boxy umístěné vedle sebe, jak ukazuje obrázek 5.2. Pokud daná osoba neměla partnera, je druhé místo pro Single Box prázdné. Dále vznikají Children Boxy, které symbolizují rodinu. Children Box již má dvě úrovně – dvě generace. V té horní je umístěn jeden Marriage Box, a pokud toto manželství mělo potomky, ve spodní úrovni jsou umístěny další Children Boxy, jeden pro každého potomka a jeho partnera. Takto dochází ke zřetězení, které můžeme vidět na obrázku 5.3. Pokud manželství potomky nemělo, zůstane spodní úroveň prázdná.



Obrázek 5.2: Typy boxů pro grafické zobrazení rodokmenu – Single Box pro osobu, Marriage Box pro manželství.

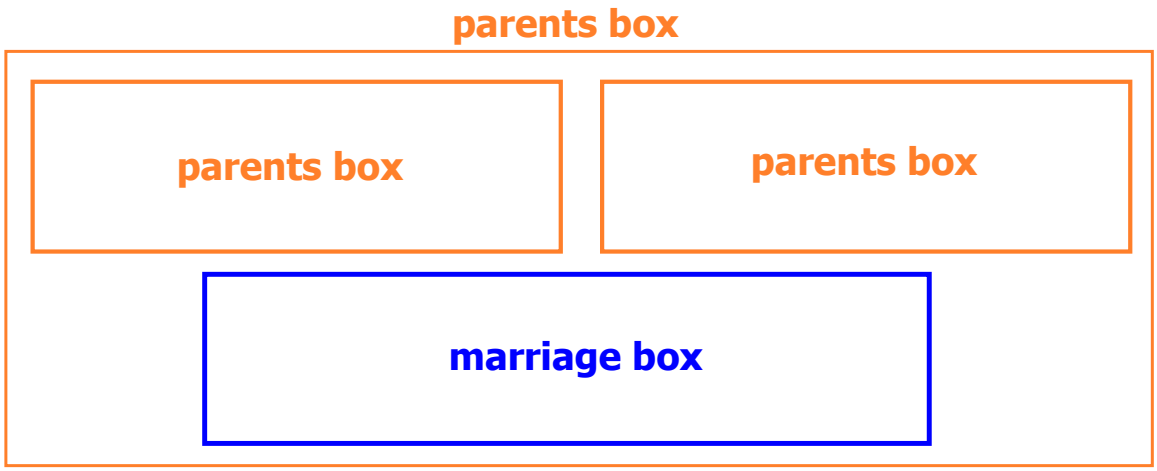
Tento přístup lze využít pro zobrazení stromu s potomky daného páru. Nikoli však pro zobrazení předků. Problém je v tom, že každý Children Box odvozuje svou šířku a výšku podle svého obsahu. Tedy ten první zahrnuje vybraný pár, pro který je zobrazen rodokmen a všechny jeho potomky. Další zahrnují vždy jednoho potomka tohoto páru, jeho partnera a všechny jejich potomky. Díky tomu každý Children Box může spočítat svou velikost a od toho odvodit své umístění na obrazovce. U zobrazení předků však jsou potřeba vždy dva Children Boxy vedle sebe, jeden pro rodiče vybrané osoby a vybranou osobu jako jejich potomka a druhý tak stejně pro partnera vybrané osoby jako potomka jeho rodičů. Takto však Children Boxy neumí zjistit svou pozici a překrývaly by se.



Obrázek 5.3: Typy boxů pro grafické zobrazení rodokmenu – Children Box pro manželství s potomky.

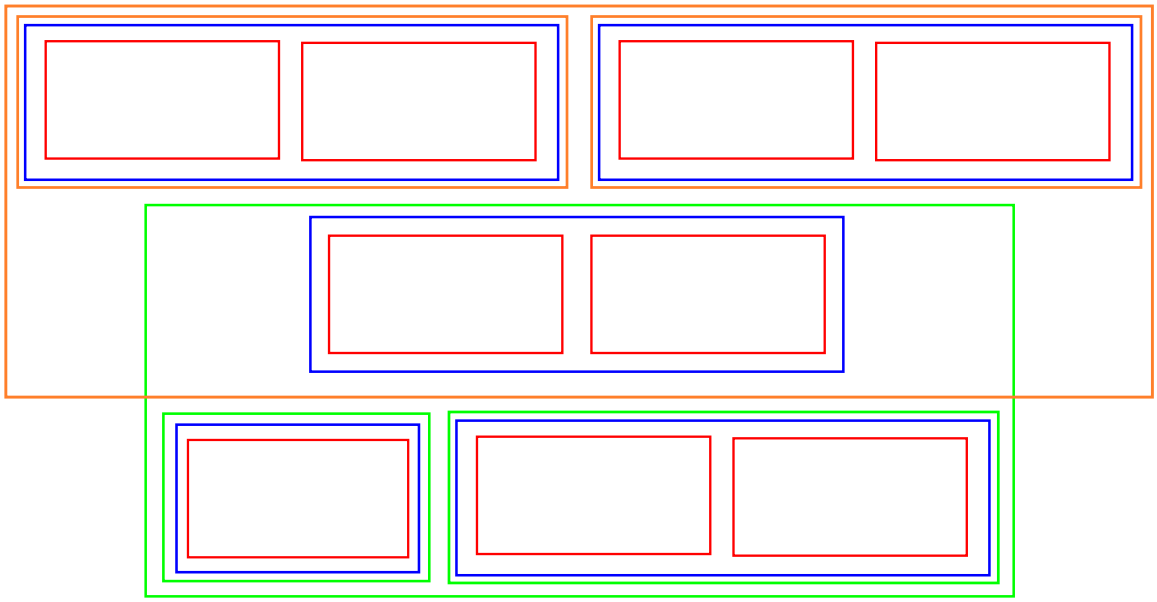
Proto byl vytvořen další typ boxu, Parents Box, který reprezentuje pár a předky obou osob z páru. Obsahuje tedy opět dvě úrovně. V té spodní je Marriage Box pro pár, jehož předky je potřeba zobrazit, v té horní dva Parents Boxy, každý pro rodiče jedné z osob ve spodní úrovni. Názorná ukázka je na obrázku 5.4.

Tento přístup způsobí, že i když algoritmus má vykreslit strom oběma směry, stejně nejprve spočítá oba stromy zvlášť. Pak je potřeba Parents Boxy a Children Boxy propojit do jednoho celku. K tomu slouží Family Box, který obsahuje rodičovský Children Box, rodičovský Parents Box, a také Single Box pro vybranou osobu a Marriage Box pro vybranou osobu a jejího partnera/partnerku. Při vykreslení je potřeba nejprve stromy vycentrovat podle toho, který je širší. A následně je překrýt, protože oba dva obsahují Marriage Box páru, pro



Obrázek 5.4: Typy boxů pro grafické zobrazení rodokmenu – Parents Box pro manželství s předky.

který je rodokmen vykreslen. Takto je tedy algoritmus po jemné úpravě schopen zobrazit i každý strom zvlášť. Případně upravit počty generací předků a potomků, které se vykreslují. Jak vypadá celý rodokmen vykreslený po jednotlivých boxech, je znázorněno na obrázku 5.5.



Obrázek 5.5: Ukázka zobrazení rodokmenu v boxech pro jednu generaci potomků i předků.

Tento algoritmus byl implementován formou tříd, které nesou název jednotlivých boxů a všechny dědí z třídy `Box`, která obsahuje atributy `width`, `height`, `x`, `y`. Nejprve jsou boxy vytvořeny, a to v pořadí nejprve Children Boxy, následně Parents Boxy a pak jsou propojeny do jednoho Family Boxu. Když je rodokmen takto vytvořen v boxech a ty jsou navázány na konkrétní osoby, může se začít s výpočtem zobrazení. Nejprve se spočítá pro každý box jeho šířka a výška. U Single Boxů je známá podle velikosti bloku, ve kterém jsou zobrazeny osoby. Velikost ostatních bloků se podle toho dopočítá. Následně jsou vypočteny souřadnice jednotlivých boxů právě v závislosti na jejich velikosti a velikosti okolních boxů.

Takto jsou všechny osoby vykresleny na správném místě, zbývá jen je propojit čárami reprezentujícími vazby mezi nimi. Vykreslení čar mají na starosti také jednotlivé boxy. Single Box vykreslí vertikální spoj směrem nahoru, pokud obsahuje předky. Marriage Box vykreslí horizontální čáru propojující obě osoby v manželství a pokud manželství mělo potomky, vykreslí vertikální čáru směrem dolů. Children Box a Parents Box vykreslí horizontální čáru, která spojí všechny dosud vykreslené čáry v dané rodině, a propojí tak rodiče se svými potomky.

Drobným nedostatkem na tomto algoritmu zůstává fakt, že z důvodu manipulaci s identitami může přesto nastat situace, kdy bude v rodokmenu zobrazena jedna osoba dvakrát. Proto byla k těmto osobám přidána ikonka, která toto symbolizuje a umožňuje zvýraznit všechny bloky v rodokmenu, které zastupují jednu a tutéž osobu.

5.4 Komunikace s externím programem pro výpočet rodokmenu

Důležitou částí implementace je i propojení s programem Moragen. Je totiž nutné, nebo minimálně pro uživatele přívětivější, mít možnost spustit výpočet programu přímo z vytvářené aplikace. Tímto přístupem aplikace poskytne vskutku plnou podporu pro práci při tvorbě genealogických modelů.

Jak bylo uvedeno v návrhu, pro komunikaci s programem Moragen byla vytvořena třída `Prolog`. Tato třída obsahuje jako jeden ze svých atributů cestu k programu a soubor, který má spustit. Výpočet lze spouštět ve dvou režimech. Ten první celý rodokmen kompletně přepočítá v závislosti na provedených změnách. To znamená, že může zcela přetvořit neuzamčené vazby mezi identitami. Druhý režim výpočtu pouze znovu zkontroluje identitu po identitě, zda po provedených úpravách stále obsahují chyby, které generovaly pravidla pro vytváření genealogických modelů, případně, zda nevznikly nové. Třída `Prolog` také kontroluje instalaci programu, ve kterém se bude program Moragen spouštět.

Program Moragen je implementován v jazyce Prolog. Pro spuštění kódu v tomto jazyce se nejčastěji využívá SWI-Prolog, stejně tak tomu bylo i při vývoji této aplikace. SWI-Prolog obsahuje dvě spustitelné verze programu – *swipl.exe* a *swipl-win.exe*. Ta první spustí pouze příkazovou řádku programu, ta druhá otevře grafické rozhraní. Při vývoji aplikace byl nejdříve používán program *swipl.exe*, nicméně nepodařilo se jej zprovoznit do funkčního řešení. Při spuštění programu Moragen pomocí *swipl.exe* se program vždy v nějaké části z neznámých důvodů zastavil a nedokončil výpočet. Proto byl později využit spustitelný soubor *swipl-win.exe*. Ten sice otevře grafické rozhraní, nicméně po ukončení výpočtu je rozhraní uzavřeno. Za výhodu lze také požadovat, že uživatel také vidí stav výpočtu, který program Moragen poskytuje, a může tak být podrobněji informován o postupu ve výpočtu.

Je tedy zřejmé, že pro spuštění výpočtu je nutné mít nainstalován program, který dokáže spustit kód v jazyce Prolog. Jak bylo zmíněno, implementace probíhala za pomoci SWI-Prolog a u tohoto programu lze zaručit funkčnost, na rozdíl od případných jiných netestovaných programů. Třída Prolog kontroluje, zda je nějaký program pro spuštění nainstalován. Pokud se instalaci nepodaří ověřit, vyzve uživatele, aby pomocí ručního nastavení vybral spustitelný program.

Aby nebylo potřeba spustitelný program vybírat při každém spuštění aplikace, je toto nastavení uloženo pomocí *serializace* [16] objektu třídy `Prolog`. Serializace objektu znamená,

že jeho aktuální stav je uložen do toku bitů (byte stream). Ve výsledku je tento objekt uložen do souboru s příponou SER. Následně pomocí *deserializace* lze z tohoto souboru vytvořit opět funkční objekt. Při nastavení spustitelného souboru se objekt třídy `Prolog` uloží do souboru *pl.ser*. Při každém dalším nastavení se z tohoto souboru vytvoří objekt a z něj se načte nastavení spustitelného programu pro provedení výpočtu. Pokud tento soubor neexistuje, nebo program nelze spustit, bude uživatel znovu vyzván k vybrání programu.

Samotné spuštění probíhá zadáním příkazu do příkazové řádky operačního systému Windows. Protože při spuštění programu Moragen bylo potřeba konzultovat určité soubory a spustit vícero příkazů, byly vytvořeny v jazyce Prolog dva speciální soubory, které toto provedou, a samotná aplikace se o to nemusí starat. Pro vytvoření stromu je to soubor *gui_run.pl*, pro kontrolu chyb je to *gui_check.pl*. Tyto soubory obsahují všechny potřebné příkazy pro provedení výpočtu, resp. kontrolu chyb a, jak již bylo řečeno, po provedení výpočtu ukončí program, ve kterém jsou spuštěny, tedy v tomto případě grafické rozhraní SWI-Prolog.

Vytváření projektů

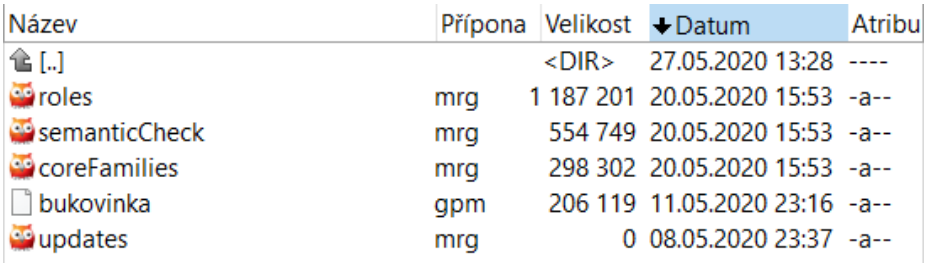
Instalace SWI-Prolog nebo jiného podobného programu není však úplně nezbytná pro spuštění aplikace, jak by se po předchozích odstavcích mohlo zdát. Pokud jsou k dispozici všechny potřebné vstupní soubory, tedy soubor se záznamy s příponou GPM a dříve vypočtené soubory *roles.mrg*, *coreFamilies.mrg* a *semanticCheck.mrg*, může se rodokmen načíst z nich. Pak sice není umožněn výpočet, ale přesto lze rodokmen upravovat, vyhledávat v něm, tvořit sociální modely a podobně.

Aby takto mohl uživatel přistupovat ke všem rodokmenům, tedy otevřít je bez předchozího výpočtu, byly vytvořeny v adresářích tzv. *projekty*. Jedná se o přístup, kdy uživatel při spuštění aplikace vybere soubor se záznamy, ze kterého se má načíst rodokmen, a následně jsou všechny důležité soubory ukládány do stejného adresáře. To umožní později načíst rodokmen bez výpočtu. Uživateli tak vznikají v jednotlivých adresářích projekty, které obsahují všechny soubory potřebné k načtení aplikace s daným obsahem.

Jelikož však program Moragen zatím neumožňuje spuštění s parametrem, který by obsahoval cestu ke konkrétnímu souboru se záznamy, byla aplikace vytvořena tak, aby tento nedostatek uživatel nepocítil. Program Moragen se spouští nad souborem, který je přímo v jeho kódu pevně nastaven. Proto byla zřízena složka *dataIn*, do které se uživatelem vybraný soubor se záznamy překopíruje a přejmenuje na standardní název *records.gpm*. Nad ním program Moragen provede výpočet a všechny soubory vloží do složky *dataOut*. Odtud jsou potřebné soubory opět překopírovány do správného adresáře, aby mohly být později znovu načteny aplikací. Na obrázku 5.6 je ukázka adresáře s projektem.

5.5 Užitečné grafické prvky

Tato podkapitola má za cíl vypíchnout podstatné prvky grafického rozhraní aplikace a upozornit na ně, resp. na jejich implementaci. Týká se to zejména částí implementace, které nejsou na první pohled zřejmé a nejsou až tak obvyklé.



Název	Přípona	Velikost	Datum	Atribu
[..]		<DIR>	27.05.2020 13:28	----
roles	mrg	1 187 201	20.05.2020 15:53	-a--
semanticCheck	mrg	554 749	20.05.2020 15:53	-a--
coreFamilies	mrg	298 302	20.05.2020 15:53	-a--
bukovinka	gpm	206 119	11.05.2020 23:16	-a--
updates	mrg	0	08.05.2020 23:37	-a--

Obrázek 5.6: Ukázka adresáře s projektem. Screenshot z aplikace *Total Commander*.

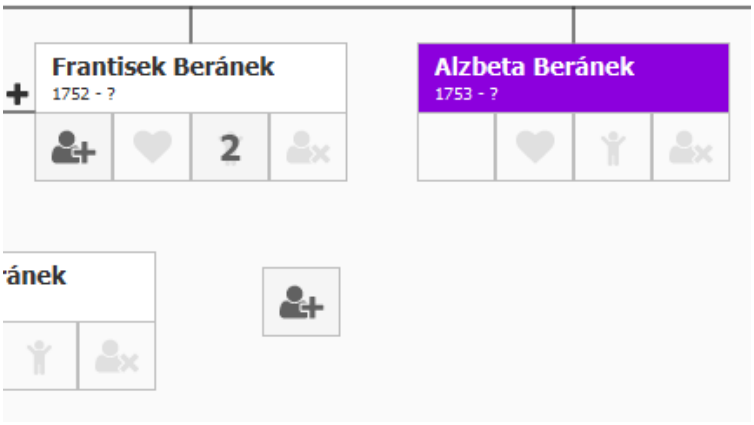
Dříve než se bude text věnovat jednotlivým prvkům, je potřeba zmínit také jednu funkcionalitu, která byla přidána až v průběhu implementace a nebyla obsažena v návrhu. Jedná se o emphrežim dvou oken. Jde o možnost zobrazit dva rozdílné rodokmeny vedle sebe. Jednotlivé identity pak lze přesouvat také mezi oběma okny a uživatel tak má výrazně zjednodušenou práci při úpravách rodokmenu.

Využití metody Drag&Drop

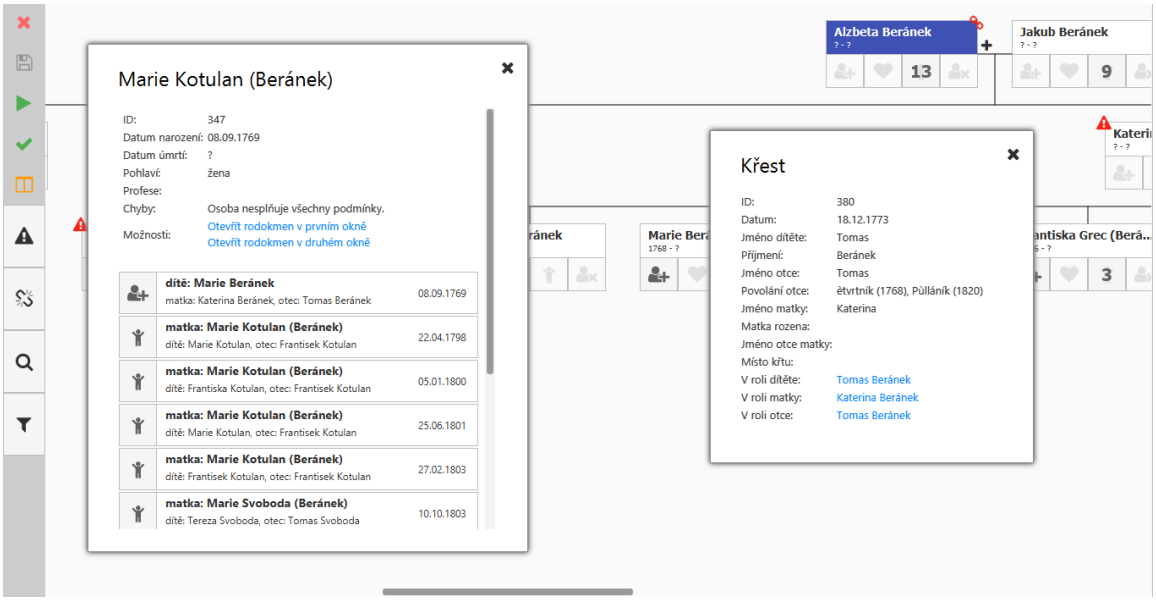
Jak bylo uvedeno, ideálním přístupem k přesouvání identit by byla metoda Drag&Drop. Tato metoda byla nakonec také implementována a to pomocí událostí, na nichž stojí celá interakce s uživatelem ve frameworku JavaFX. Každé kliknutí myši či klávesy vyvolá událost, pokud je na tuto událost navázána konkrétní metoda, provede se. Pro metodu Drag&Drop jsou zásadní události `DragDetected`, `MouseDragged` a `MouseReleased`, postupně zachycují stisknutí tlačítka myši, posun myši se stisknutým tlačítkem, uvolnění tlačítka myši. Díky souřadnicím, které událost poskytne, lze vypočítat, kde se myš nachází a díky tomu přesouvat i jednotlivé bloky s obsahem.

Přesouvat lze tedy bloky s identitami, přesněji řečeno s jednou identitou, jak je možné vidět na snímku 5.7. Nelze přesouvat blok, který obsahuje více identit či blok, který neobsahuje žádnou identitu. Při přesunu bloku jsou zachycovány další dvě události a to na blocích znázorňujících osoby. Jedná se o události `MouseDragEntered` a `MouseDragExited`, které zachycují postupně vniknutí ukazatele myši do bloku při stisknutém tlačítku myši a opuštění bloku ukazatelem myši při stisknutí tlačítka myši. Události zachycují také stisknutí resp. uvolnění tlačítka myši přímo v bloku. Pokud je stisknuto v bloku, je vyvolána událost `MouseDragEntered`, pokud je uvolněno v bloku vyvolá se událost `MouseDragExited`. Díky tomuto přístupu je možné sledovat, kam je identita přesouvána, a následně propojit přesouvanou identitu a osobu, ke které byla přetažena.

Stejný princip byl implementován i u detailů osoby a záznamu. Pomocí metody Drag&Drop je lze přesouvat po celé obrazovce, jak to uživatel zrovna potřebuje. Při více otevřených detailech se každý detail po kliknutí přesune do popředí obrazovky. A stejně jako bloky s osobami, i detaily byly uzpůsobeny k zachytávání přesouvané identity pomocí událostí `MouseDragEntered` a MouseDragExited. Je tedy možné identity přesouvat nejen v rámci rodokmenu, ale také mezi detaily, či mezi detailem a rodokmenem. Ukázka otevřeného detailu v aplikace je na obrázku 5.8.



Obrázek 5.7: Screenshot z aplikace – ukázka přesouvání identit.



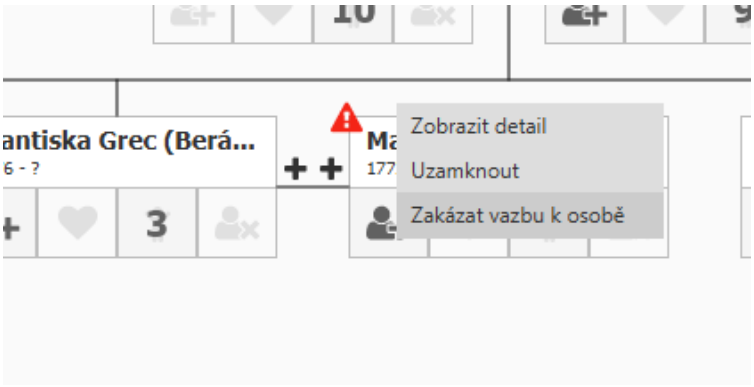
Obrázek 5.8: Screenshot z aplikace – ukázka zobrazení detailů.

Dialogové menu

Při testování návrhu grafického rozhraní bylo zjištěno, že uživatelé při každé události intuitivně klikají na daný objekt. Proto bylo u bloku osoby a identity v rodokmenu vytvořeno *dialogové menu*, které se otevře po kliknutí pravým tlačítkem myši, jak jsou uživatelé zvyklí z mnoha dalších aplikací. Jednotlivé odkazy dialogového menu jsou vytvořeny třídou `DialogButton`. Obsahují všechny akce, které je možné s identitou či osobou provést. Ukázka této funkcionality je na obrázku 5.9.

Toto menu je zobrazeno na základě události `Action`, která se vyvolá při stisknutí tlačítka. U této události je nutné nejprve zkontrolovat, zda se jedná o kliknutí pravým tlačítkem, tato událost je totiž vyvolána při kliknutí libovolným tlačítkem myši. A hlídat se musí i skrytí dialogového menu při kliknutí kamkoli jinam. Pro tento problém byla vytvořena metoda ve třídě `Desktop`, která je navázána na událost `MouseClicked`, která se zavolá při kliknutí

na libovolnou část obrazovky. Tato metoda odstraní případné otevřené dialogové menu z obrazovky. Problém nastane, pokud je stisknuto tlačítko, protože to událost `MouseClicked` nezachytí a je potřeba metodu v třídě `Desktop` zavolat explicitně.



Obrázek 5.9: Screenshot z aplikace – ukázka dialogového menu při kliknutí na identitu.

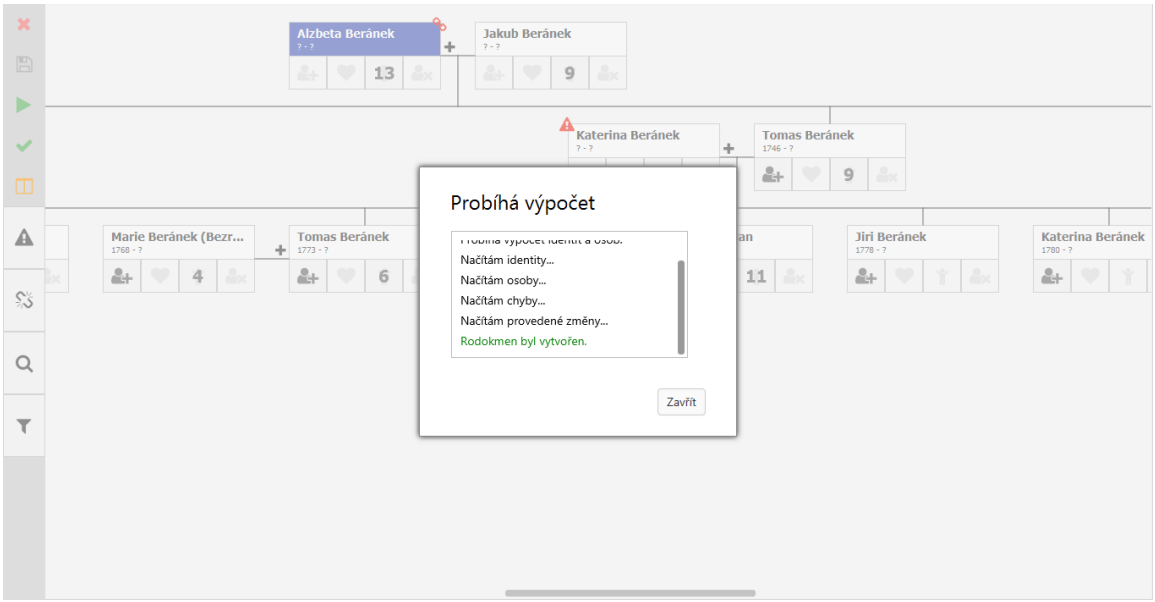
Časově náročné výpočty

Zásadním problémem při implementaci bylo provedení časově náročných výpočtů. Při malém objemu dat, na kterém byla aplikace testována hned při implementaci, jak bude později uvedeno, se delší čas výpočtu nijak neprojeví. Pokud však vložíme už jednotky tisíc záznamů, začíná být zpoždění patrné. Pokud by tato situace nebyla nijak ošetřena, uživatel by to vnímal jako zaseknutí aplikace. Veškerá grafika se totiž vždy zobrazí až na závěr všech prováděných operací.

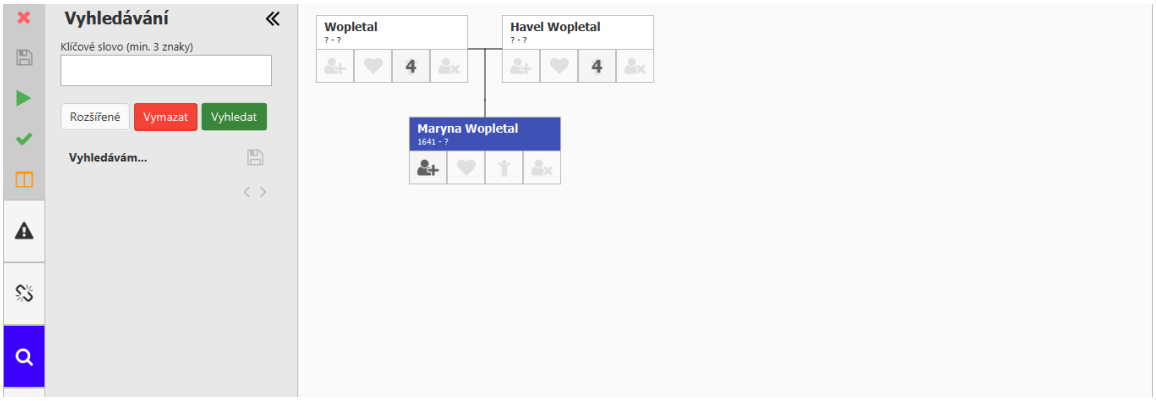
JavaFX obsahuje pro tyto případy třídu `Service`. Jak je uvedeno v dokumentaci [12], třída `Service` slouží pro jednoduchou správu vícevláknových aplikací. `Service` vytvoří úlohu, která probíhá na jiném vlákně, pomocí metody `createTask` a následně ji řídí. Lze ji použít opakovaně pomocí resetování. Pro třídu `Service` jsou vytvořeny speciální události `Success`, `Failed` a `Canceled`, které jsou vyvolány postupně, když je úloha úspěšně dokončena, když je úloha neúspěšně ukončena a když je úloha předčasně ukončena uživatelem.

Implementace této třídy byla použita při načítání a ukládání projektu a při výpočtu nového rodokmenu i kontrole chyb. Vždy se nejprve otevře dialogové okno s informačními zprávami pro uživatele zobrazené na snímku 5.10, příslušné metody se propojí s událostmi a spustí se třída `Service`. Tím je vytvořeno nové vlákno, na kterém probíhá výpočet, ale zároveň je dokončena veškerá činnost na stávajícím vlákně, a proto se zobrazí všechna načtená grafika. Každý výpočet lze zrušit tlačítkem v dialogovém okně, které zavolá metodu `cancel` třídy `Service`.

Jak bude později uvedeno, při načtení většího objemu testovacích dat se časová náročnost výpočtu projevila i u vyhledávání a zobrazení nepřiřazených identit. Zde byla proto také implementována třída Service. V tomto případě však třída umožní pouze otevření prozatím prázdné záložky, jak je vidět na obrázku 5.11, a teprve po ukončení výpočtu jsou v záložce zobrazeny výsledky hledání. Neotvírá se zde žádné dialogové okno.



Obrázek 5.10: Screenshot z aplikace – ukázka průběhu výpočtu při použití třídy Service.



Obrázek 5.11: Screenshot z aplikace – ukázka vyhledávání.

Kapitola 6

Testování a další možný vývoj

Poslední fází vývoje každé aplikace je testování. Nejinak tomu bylo i v této práci. V návrhu již bylo popsáno testování grafického rozhraní. Testování vyvíjené aplikace probíhalo současně s vývojem, na základě provedených testů byly odhaleny chyby, které byly ihned opraveny. Na závěr vývoje bylo provedeno testování na uživatelích, které mělo dva cíle, a to odhalit chyby a nedostatky a zkontrolovat, zda jsou splněny všechny požadavky specifikované v první kapitole této práce. Popis jednotlivých typů testů v této kapitole vychází z [13].

6.1 Testovací data

Pro testování byla potřebná testovací data. Pro prvotní testování při vývoji aplikace byla vytvořena fiktivní data o počtu asi 30 matričních záznamů. Protože testování bylo potřeba dříve, než byla aplikace propojena s programem Moragen, byly ručně vytvořeny i ostatní soubory, které generuje tento program. Cílem bylo také zanést do těchto dat záměrné chyby, se kterými by si měla aplikace umět poradit. A tyto chyby by program Moragen nedokázal zpracovat.

Druhá sada testovacích dat byla poskytnuta vedoucím této práce. Obsahuje ručně opsané záznamy z matrik z jihomoravské obce Bukovinka v okrese Blansko. Všechny záznamy jsou pouze typu narození a to v počtu 1861. Datovány jsou do období 1641 – 1899.

6.2 Průběh testování

Prvním testováním bylo u každé třídy jednotkové testování, tzv. *Unit Testing*. Jedná se o testování každé třídy či složitější metody. Tyto testy byly provedeny po dokončení každé třídy. U třídy typu Model byly porovnávány textové výpisy na konzoli, provedené pomocí metod `toString`, které jsou implementovány téměř ve všech třídách typu Model. U tříd typu View bylo provedeno testování pomocí výstupů na obrazovku. Na základě testů pak byly hledány a opravovány chyby. Dokud daná třída neposkytla očekávané výpisy/výstupy na obrazovku.

Dvakrát také proběhlo *integrační testování*. Jedná se o testy, které se zaměřují na komunikaci objektů mezi sebou. Poprvé se tak stalo po dokončení všech objektů v balíku `models`. Testování bylo opět provedeno pomocí výstupů na konzoli, případně kontrolou obsahu vytvořených souborů. Následně byly na základě chybných výstupů hledány chyby, ve výjimečných případech také přidávány chybějící metody do tříd tak, aby bylo dosaženo korektního stavu. Stejně byly testovány i třídy z balíku `views`. Zde se ovšem už testování částečně krylo se systémovými testy.

Po dokončení implementace celé aplikace proběhlo testování pomocí poskytnuté rozsáhlejší testovací sady. Zde byly odhaleny některé nedostatky související zejména s rychlostí aplikace. Příkladem může být již zmíněné vyhledávání, které z důvodů většího počtu záznamů mělo již viditelné zpoždění a uživatel mohl vnímat zaseknutí aplikace. Proto zde byla přidána třída `Service`, která umožní zobrazit prázdnou záložku a výsledky vypočítat na pozadí. Toto odstranilo však jen část zpoždění. Po důsledném hledání dalších příčin zpoždění bylo konstatováno, že se jedná o zpoždění, které souvisí se samotným frameworkem JavaFX a vykreslením velkého počtu objektů. Část zpoždění způsobovalo i to, že u každého objektu byly obsluhovány události kliknutí myši, to způsobilo prodlevu už při samotném zachytávání události. Proto byl celý problém vyřešen stránkováním. Výsledky u vyhledávání i nepřiřazených identit jsou proto rozděleny na stránky po 50 záznamech a zde již není prodleva téměř patrná.

Na úplný závěr bylo provedeno testování `systémové` a `akceptační`. To první ověřuje, jak spolu spolupracují jednotlivé části systému a jak funguje systém jako celek. Akceptační testování, jak bylo uvedeno v úvodu kapitoly, má za cíl zkontrolovat splnění všech dříve specifikovaných požadavků. Do tohoto testování již byli zapojeni uživatelé. Více o testování na uživatelích uvádí následující podkapitola.

Podstatné je zmínit ještě jednu část testování. Na závěr implementování aplikace a před začátkem finálního testování bylo implementováno logování výjimek. V průběhu implementace byly totiž výjimky odesílány na chybový výstup. Na závěr byly ošetřeny i tyto výstupy a pro zachování informací o výjimkách v programu byl zvolen zápis do souboru. Každá vyvolaná výjimka v programu se zapíše do souboru *gui.log*. Jak lze vidět na obrázku 6.1, log nese přesný datum a čas zápisu, dále typ výjimky, zprávu výjimky a cestu k souboru, který výjimku vyvolal. Zde je potřeba uvést, že pro potřeby aplikace byla vytvořena speciální třída `AppException`, která kromě obvyklé zprávy, která se zobrazí uživateli, nese také zprávu pro vývojáře. Ta obsahuje podrobnější popis chyby i s hodnotami proměnných, které chybu vyvolaly.

```
LOG 2020-05-23T13:29:48.517475200
Type: class moragengui.exception.AppException
Message: Cannot move identity to person because of sex condition (identity: [Identity (type: father, id: 1864, record: 4, pe
Stack Trace: [moragengui.models.FamilyTree.moveIdentityToPerson(FamilyTree.java:354), moragengui.views.tree.IdentityBox.rele
LOG 2020-05-23T13:35:02.342643300
Type: class moragengui.exception.AppException
Message: Cannot move identity to person because of sex condition (identity: [Identity (type: father, id: 1864, record: 4, pe
Stack Trace: [moragengui.models.FamilyTree.moveIdentityToPerson(FamilyTree.java:354), moragengui.views.tree.IdentityBox.rele
LOG 2020-05-23T13:58:46.896581100
Type: class java.lang.NullPointerException
Message: null
Stack Trace: [moragengui.views.details.PersonDetail.<init>(PersonDetail.java:38), moragengui.views.rows.PersonRow.clicked(Pe
```

Obrázek 6.1: Ukázka formátu logů.

6.3 Testování na koncových uživatelích

Pro testování bylo vybráno celkem pět uživatelů. Výběr byl proveden tak, aby zde byly zastoupeny rovnoměrně obě pohlaví a různé věkové kategorie. Zastoupeni byli laici i lidé s větším zájmem o historii, stejně tak lidé s různou zkušeností v práci s počítačem. Před testováním byli uživatelé poučeni o způsobu testování. Byla jim vysvětlena souvislost mezi záznamy, identitami a osobami. Byla jim představena výsledná aplikace, ve zkratce ukázáno její ovládání a také vysvětlen účel vývoje aplikace, jaký problém má aplikace vyřešit. Pro testování bylo zvoleno klidné prostředí a uživatelé byli seznámeni s faktem, že nejde o testování uživatelů, ale o testování aplikace. Tedy, že není chybou, když něco nebudou vědět, nebo něco nepůjde v aplikace splnit.

Následně jim byly zadány dvě komplexní úlohy. Každá úloha se skládala z vícero bodů a byla provedena na jiných testovacích datech. Zásadní body v první úloze byly:

- otevřít příslušný projekt
- zobrazit rodokmen pro danou osobu
- vyhledat požadovaný záznam pomocí vyhledávání
- zobrazit vyhledaný záznam v druhém okně
- přesunout identity mezi záznamy
- uzamknout osobu
- uložit rodokmen

Druhá úloha obsahovala tyto hlavní body:

- vyhledat osobu na základě jména a data narození
- vyhledat jména rodinných příslušníků
- vytvořit nové osoby z nepřiřazených identit
- vyhledat osobu na základě jména a povolání
- vyhledat příslušníky daného rodu
- uložit vyhledané výsledky jako submodel
- zobrazit submodel v rodokmenu
- zakázat vazbu mezi identitou a osobou
- provést nový výpočet

Při testování byl uživatelům měřen čas, který jim testování zabere. Při tomto měření žádná úloha nevybočovala z normálu, kdy by se u ní uživatelé na delší dobu zdrželi. Celkově testování trvalo průměrně 45 minut. V průběhu testování byly dále sledovány intuitivní reakce uživatelů, zda jejich první reakce na daný problém vede správným směrem. Kontrolována

byla také aplikace, zda reaguje správným způsobem a nedochází k neočekávaným stavům. Po ukončení testování měli uživatelé ještě vyplnit dotazník, který zjišťoval, jak je aplikace pochopitelná a přehledná pro běžného uživatele. Také zde byla možnost pro sdělení nápadů na případné vylepšení, nebo pro výtky, co zde chybělo, nebo bylo naopak nadbytečné.

Při vyhodnocování byly využity všechny poznámky z průběhu testu, formuláře, které uživatelé vyplňovali i soubory s logy. Po pečlivé analýze všech materiálů bylo nalezeno několik chyb, které se v aplikaci objevují a které by bylo vhodné opravit, a drobné nedostatky, které kazí dojem uživatele z celé aplikace a které by mohly být napraveny drobnými vylepšeními. V neposlední řadě uživatelé zmínili také větší části aplikace, které jim chyběly a které ale nebyly součástí specifikace požadavků. V této oblasti se typicky jednalo o různé formáty zobrazení dat či způsoby vyhledávání. Jak bude níže uvedeno, je to jeden ze směrů dalšího možného vývoje.

Chyby, které byly při testování nalezeny, jsou:

- filtr v submodelu vyfiltruje i vyhovující osoby
- po novém výpočtu zůstanou zobrazena stará data
- projekt nelze uložit

Následně zde bylo také určité chování aplikace, které uživatelé mohli vnímat jako chybu, ale jde o standardní chování aplikace. Příkladem je ponechání starých a již neplatných výsledků ve vyhledávání při práci s rodokmenem. Do testů zasáhlo také to, že program Moragen může vypočítat vazby mezi osobami v každém výpočtu mírně odlišně. Proto u některých uživatelů nebylo možné splnit jednotlivé úkoly.

Zde jsou uvedeny drobné nedostatky, které mohou být jednoduše implementovány:

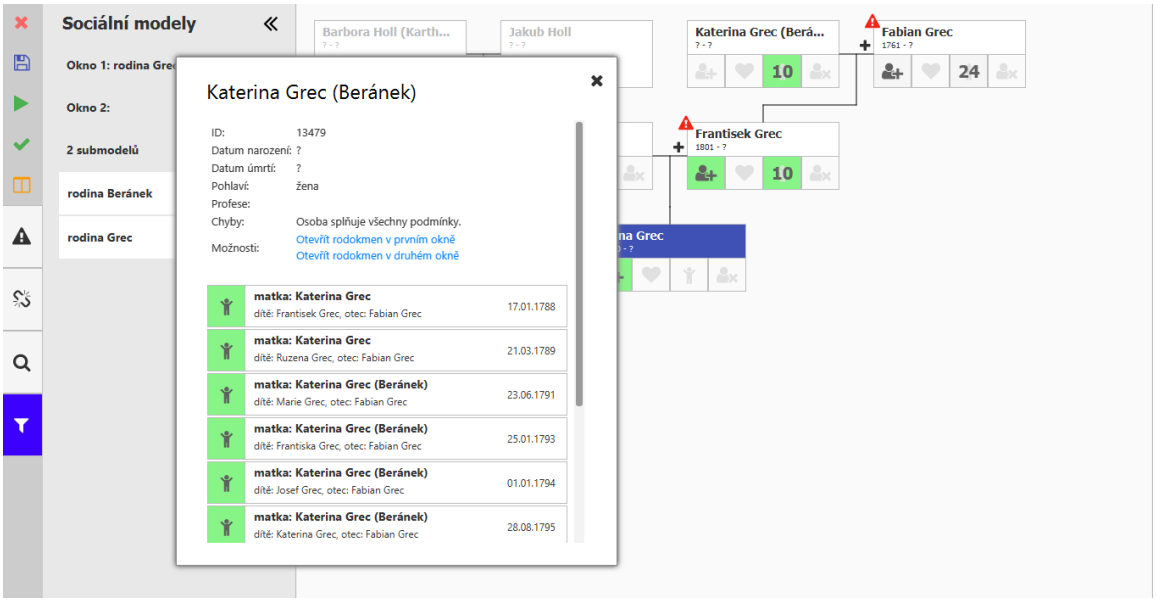
- nebyl pochopen princip submodelů
- uživatelé byli často zmatení a nevěděli, co daná ikona znamená
- uživatelé se snažili intuitivně využívat i klávesnici
- byl špatně interpretován konec výpočtu, uživatelé čekali dále

Z toho byly vyvozeny pro aplikaci následující důsledky. Celkově byla aplikace špatně pochopena zřejmě proto, že neobsahovala dostatek vysvětlivek a některé názvy byly matoucí. Proto byly k jednotlivým tlačítkům přidány vysvětlující popisky. Také byly přidány častější informační zprávy, aby uživatel věděl, že aplikace na něj reaguje. Zřejmě ze stejného důvodu nebyly pochopeny submodely. Celá záložka byla proto rozpracována tak, aby bylo možné zobrazit aktivní submodely. A na závěr byly přidány možnosti použití nejznámějších klávesových zkratek. Tedy klávesu `Enter`, pro potvrzení formuláře vyhledávání a klávesovou zkratku `Ctrl + S` pro uložení projektu.

6.4 Tvorba sociálních modelů a práce s nimi

Po ukončení testování a tím vývoje celé aplikace, byly vytvořeny na základě poskytnutých testovacích dat dva sociální modely. Sociální modely lze tvořit na základě mnoha parametrů – jména, příjmení, místa narození a úmrtí, data narození a úmrtí atd. Vytvořením sociálního modelu má uživatel možnost pracovat pouze s částí rodokmenu, ostatní osoby mimo daný sociální model mu zůstávají skryty.

Vytvořené sociální modely byly modely dvou významných rodin z obce Bukovinka – rodina Grec a rodina Beránek. Na vytvořených sociálních modelech pak byly provedeny úpravy vazeb mezi identitami a osobami tak, aby byly odstraněny všechny chyby, které výpočet programu Moragen vygeneroval. Při úpravách bylo mimo jiné nalezeno také propojení těchto dvou rodin. Upravené vazby byly vždy uzamčeny, aby se již nemohly měnit. Práce na sociálním modelu rodiny Grec je zachycena na snímku 6.2.



Obrázek 6.2: Screenshot z aplikace – ukázka práce na sociálním modelu rodiny Grec.

6.5 Možné směřování dalšího vývoje

Vývoj může směřovat dopředu u každé aplikace. Stejně tak je tomu i zde a už při tvorbě aplikace byly evidovány nápady a možná rozšíření, která by danou aplikaci vylepšila. Celkově by se dalo možné směřování dalšího vývoje vytvořené aplikace rozdělit do tří okruhů.

V tom prvním by byly další možnosti u vyhledávání a filtrování. Je potřebné přiznat, že možností, na základě jakých parametrů lze vyhledávat a filtrovat, je celá řada. Stejně tak je velmi mnoho možností, jak vyhledané výsledky zobrazit, ať už v rodokmenu, nebo v nějakých seznamech. Toto je tedy první směr vývoje, který by uživatelé zcela jistě ocenili.

Druhé směřování aplikace se týká objemu dat, se kterým pracuje. Pro velké objemy dat čítající desítky či stovky tisíc záznamů není navržené řešení zcela ideální a při práci s aplikací

by to mohlo být znát. Proto by bylo možné zvážit použití databáze, kde by se ukládala data ze souborů a aplikace by k nim měla přímý přístup.

Třetí okruh rozšíření se zabývá těsnější vazbou mezi programem Moragen a aplikací. Momentálně zobrazená data v aplikaci totiž nejsou zcela všechna, která lze z výstupů programu Moragen vyčíst. Větší detaily by mohly být zobrazeny například u chybových hlášek. Implementovat by bylo možné také zobrazení návrhů, kam chybnou identitu přesunout.

Kapitola 7

Závěr

Tato práce navazuje na program Moragen, který sestavuje rodokmeny z matričních záznamů vytvářením vazeb mezi jednotlivými záznamy. Cílem bakalářské práce bylo vytvořit aplikaci, která by dokázala zobrazit takto vzniklé rodokmeny, umožnila uživateli je upravit a nechat zkontrolovat programem Moragen a také nabídla možnost v rodokmenech vyhledávat a vytvářet sociální modely.

Nejprve bylo potřeba nastudovat celou problematiku vytváření záznamů a práci, kterou odvádí program Moragen. Následně byly vyhledány existující systémy pro tvorbu genealogických modelů a po prozkoumání všech dostupných funkcionalit bylo možné konstatovat, že žádná podobná aplikace zatím neexistuje. Díky tomu bylo možné začít s tvorbou aplikace. Nejprve byla provedena důkladná analýza požadavků na aplikaci. Následně byla vybrána architektura a navržen objektový model celého systému včetně vstupů a výstupů. Grafické rozhraní bylo po návrhu otestováno na prototypu a výsledky testů byly zohledněny při tvorbě finálního návrhu grafiky.

Aplikace byla implementována v jazyce Java na frameworku JavaFX. V průběhu implementace bylo nutné vymyslet různé potřebné algoritmy a vyřešit řadu problémů, aby byla aplikace plně funkční a uživatelsky přívětivá. Provedeno bylo také propojení aplikace s již existujícím programem Moragen. Už během implementace byla aplikace postupně testována. Na závěr byly provedeny testy na uživatelích. Bylo tak odhaleno několik problémů a chyb, které byly následně opraveny. Testy také poukázaly na potřeby uživatelů a naznačily, jaké funkcionality by v budoucnu bylo možné doplnit. Ve výsledné aplikaci byly na poskytnutých datech vytvořeny sociální modely dvou významných rodů z obce Bukovinka.

Díky této práci vznikla aplikace s grafickým rozhraním, která je dalším krokem v pohodlné tvorbě rodokmenů pomocí matričních záznamů. Na základě reakcí uživatelů je možné říci, že byly naplněny základní cíle, které byly nastíněny v úvodu. Aplikace je plně funkční a poskytuje uživateli veškerou očekávanou podporu při tvorbě rodokmenů. Zároveň obsahuje moderní grafické rozhraní, je uživatelsky přívětivá a intuitivní.

Závěrem je vhodné zmínit, že i přesto je možné aplikaci dále rozšiřovat. Cestami dalšího směřování vývoje může být vylepšení funkcí pro vyhledávání, filtrování a tvorbu sociálních modelů, dále lepší využití všech dostupných výsledků programu Moragen, či navázání na databázi pro rychlejší práci s velkými objemy dat.

Literatura

[1] *SWI-Prolog* [online]. [cit. 2019-05-17]. Dostupné z: https://www.swi-prolog.org/.

[2] Ancestry Team. *Sneak Peek of The New Ancestry Website Coming!* [online]. Ancestry.com Operations Inc., únor 2015 [cit. 2019-05-10]. Dostupné z: https://blogs.ancestry.com/ancestry/2015/02/19/sneak-peak-new-ancestry-website-coming/.

[3] Ancestry.com Operations Inc. *Ancestry* [online]. [cit. 2019-05-10]. Dostupné z: https://www.ancestry.com.

[4] Bernard, B. *Alternativy k MVC a závěrečné poznámky* [online]. Devel.cz Lab s.r.o., květen 2009 [cit. 2019-05-12]. ISSN 1803-5620. Dostupné z: https://www.zdrojak.cz/clanky/alternativy-k-mvc-a-zaverecne-poznamky/.

[5] Bernard, B. *Prezentační vzory z rodiny MVC* [online]. Devel.cz Lab s.r.o., květen 2009 [cit. 2019-05-12]. ISSN 1803-5620. Dostupné z: https://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/.

[6] Bernard, B. *Úvod do architektury MVC* [online]. Devel.cz Lab s.r.o., květen 2009 [cit. 2019-05-12]. ISSN 1803-5620. Dostupné z: https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/.

[7] Čápka, D. *Lekce 1 - Úvod do formulářových aplikací v Java Swing* [online]. David Čápka [cit. 2019-05-16]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/java/formulare/swing/zaklady/java-tutorial-swing-okenni-aplikace-uvod.

[8] Čápka, D. *Lekce 1 - Úvod do jazyka Java* [online]. David Čápka [cit. 2019-05-16]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/java/zaklady/java-tutorial-uvod-do-jazyka-java.

[9] FamilySearch International. *FamilySearch* [online]. [cit. 2019-05-10]. Dostupné z: https://www.familysearch.org.

[10] Fellegi, I. P. a Sunter, A. B. A Theory for Record Linkage. *Journal of the American Statistical Association*. Taylor & Francis. 1969, sv. 64, č. 328, s. 1183–1210. DOI: 10.1080/01621459.1969.10501049.

[11] Fonticons, Inc. *Font Awesome* [online]. [cit. 2019-05-17]. Dostupné z: https://fontawesome.com/.

[12] Gluon. *JavaFX* [online]. [cit. 2019-05-17]. Dostupné z: https://openjfx.io.

[13] Kitner, R. *Typy testování software (třídění testů)* [online]. [cit. 2019-05-20]. Dostupné z: https://kitner.cz/testovani_softwaru/typy-testovani-software-trideni-testu/.

[14] MyHeritage Ltd. *MyHeritage* [online]. [cit. 2019-05-10]. Dostupné z: https://www.myheritage.cz.

[15] Oracle. *Java* [online]. [cit. 2019-05-17]. Dostupné z: https://www.oracle.com/java.

[16] Oracle. *Java Documentation* [online]. [cit. 2019-05-17]. Dostupné z: https://docs.oracle.com/en/java/.

[17] Raj, P., Raman, A. C. a Subramanian, H. *Architectural patterns: uncover essential patterns in the most indispensable realm of enterprise architecture*. First published. Birmingham, UK: Packt Publishing, 2017. 67–70 s. ISBN 978-1-78728-749-5.

[18] Štechmüller, P. *Lekce 1 - Úvod do JavaFX* [online]. David Čápka [cit. 2019-05-16]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/java/formulare/javafx/uvod-do-javafx.

[19] The Apache Software Foundation. *Apache NetBeans* [online]. [cit. 2019-05-17]. Dostupné z: https://netbeans.apache.org/.

[20] The Church of Jesus Christ of Latter-day Saints. *How to Use FamilySearch.org* [online]. [cit. 2019-05-10]. Dostupné z: https://www.churchofjesuschrist.org/topics/family-history/my-family-history/learn-to-use-family-search?lang=eng.

[21] The Qt Company Ltd. *Model/View Programming* [online]. [cit. 2019-05-12]. Dostupné z: https://doc.qt.io/qt-5/model-view-programming.html.

[22] Zbořil, F., Rozman, J. a Kočí, R. Algorithmic creation of genealogical models. In: *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.

[23] Ziff Davis, LLC. *Drag and drop* [online]. [cit. 2019-05-20]. Dostupné z: https://www.pcmag.com/encyclopedia/term/drag-and-drop.

Příloha A

Obsah přiloženého paměťového média

- **app/** – adresář se spustitelnou aplikací
  - **lib/** – adresář s knihovnami potřebnými pro spuštění aplikace
  - **moragen/** – adresář s programem Moragen potřebným pro spuštění aplikace, obsah tohoto adresáře není (kromě souborů *gui_check.pl* a *gui_run.pl*) výsledkem této práce
  - **MoragenGUI.jar** – spustitelná aplikace ve formátu JAR
  - **readme.txt** – návod k instalaci a nápověda k použití aplikace
- **doc/** – adresář se zdrojovými soubory technické zprávy
- **src/** – adresář se zdrojovými soubory aplikace
- **xwawre00.pdf** – text bakalářské práce ve formátu PDF