VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DATABÁZE PRO GENEALOGICKÉ MODELY

DATABASE FOR GENEALOGICAL MODELS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAEL HOFFMANN**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK KOČÍ, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Hoffmann Michael**
Program: Informační technologie
Název: **Databáze pro genealogické modely**
Database for Genealogical Models
Kategorie: Databáze

Zadání:

1. Seznamte se s problematikou práce s genealogickými daty, tvorbou genealogických modelů a existující databází přepisovaných matričních záznamů ČR (DEMoS). Tato výchozí databáze vzniká v rámci projektu na FIT.
2. Proveďte analýzu požadavků na strukturu a ukládání dat pro potřeby genealogických modelů a na základě analýzy vyberte vhodný typ databáze.
3. Navrhněte a realizujte databázi reflektující potřeby ukládání genealogických modelů. Musí být umožněno mapování osob z modelů do výchozí databáze přepsaných záznamů DEMoS.
4. Navrhněte a realizujte uživatelské rozhraní pro správu modelů. Rozhraní integrujte do webového rohraní systému DEMoS.
5. Na datech z databáze DEMoS otestujte systém a diskutujte jeho další vývoj.

Literatura:

- Moravský Zemský Archiv. Digitalizace archivních fondů. http://www.mza.cz/a8web/a8apps1/a8apps1.htm.

Pro udělení zápočtu za první semestr je požadováno:

- První tři body zadání.

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

Vedoucí práce: **Kočí Radek, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Bakalářská práce je součástí projektu DEMoS, který si klade za cíl vytvořit systém pro správu digitalizovaných záznamů starých matrik. Cílem této bakalářské práce je navrhnout a vytvořit aplikaci s databází, která bude umožňovat uživatelům spravovat vytvořené či vygenerované genealogické modely, přičemž bude možné provázat osoby uvedené v modelech se záznamy v databázi digitalizovaných matrik. Pro řešení byly využity technologie PHP, Javascript, HTML, CSS a Neo4j.

Abstract

The bachelor thesis is part of the DEMoS project, aiming to create a system for managing digitized records of old registries. This thesis aims to design and develop an application with a database to allow users to manage created or generated genealogical models. At the same time, it will be possible to link people listed in the models with records in the database of digitized registries. Following technologies were used for the solution and implementation - PHP, Javascript, HTML, CSS, and Neo4j.

Klíčová slova

genealogie, genealogické prameny, genealogické modely, grafová databáze, databáze, webová aplikace, Neo4j, PHP, CSS, HTML

Keywords

genealogy, genealogical sources, genealogical models, graph database, database, web application, Neo4j, PHP, CSS, HTML

Citace

HOFFMANN, Michael. *Databáze pro genealogické modely.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Kočí, Ph.D.

Databáze pro genealogické modely

Prohlášení

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

.....
Michael Hoffmann
5. května 2021

Poděkování

Tímto bych rád poděkoval mému vedoucímu práce panu Ing. Radkovi Kočímu, Ph.D. za čas strávený konzultacemi a za odborné vedení mé bakalářské práce.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce vznikla v rámci výzkumného projektu DEMoS,[1][2] který se zabývá digitalizací a zpracováním informací z historických pramenů. Projekt je řešen na Fakultě informačních technologií Vysokého učení technického v Brně. V rámci projektu DEMoS vzniká databáze přepisů matričních záznamů, především o křtech, svatbách a úmrtích.

Pro vytvoření přesných genealogických modelů je třeba dobře poznat a určit své předky. Člověk sestavující genealogický model tedy musí pátrat v matričních záznamech. Vytvořená aplikace bude uživateli umožňovat vytvářet a spravovat genealogické modely. Osoby z modelů bude možno mapovat do databáze přepsaných matričních záznamů systému DEMoS. Toto propojení uživateli značně usnadní práci a zvýší přesnost vytvořených genealogických modelů.

Cílem této práce je navrhnout a realizovat databázi a obslužné uživatelské rozhraní, které bude sloužit k ukládání a správě genealogických modelů přihlášeným uživatelem. Systém by měl uchovávat genealogická schémata vytvořená uživateli a také navázat jednotlivé předky tvořící modely na příslušné matriční záznamy. Databázi je tedy třeba realizovat s ohledem na tyto požadavky. Aplikace bude disponovat přehledným uživatelským rozhraním, které umožní jednoduše spravovat genealogické modely. Dalším cílem práce je integrace vytvářené aplikace s databází do existujícího systému vznikajícímu v rámci projektu DEMoS.

Tento dokument je rozdělen na několik kapitol. V kapitole 2 jsou vysvětleny jednotlivé teoretické pojmy a termíny, se kterými se v této práci lze setkat. Kapitola 3 vymezuje požadavky na vytvářený systém a na jejich základě popisuje postup analýzy a výběru vhodného typu databáze, obsahuje také informace o již existujících řešeních s vysvětlením podobnosti a rozdílům oproti řešení, které je cílem této práce. Kapitola 4 popisuje technologie použité při implementaci výsledného řešení. Kapitola 5 vysvětluje návrh databáze, webové aplikace a jejího uživatelského rozhraní. Kapitola 6 je poslední kapitolou a je věnována popisu implementace řešení podle návrhu, jeho integraci do systému DEMoS, testování na datech z databáze matričních záznamů a také diskutuje možný další vývoj systému, který je řešením této práce.

[1] Podrobnosti o projektu jsou dostupné na: https://www.fit.vut.cz/research/project/1204/

[2] Aplikace DEMoS je dostupná na stránce: http://perun.fit.vutbr.cz/

Kapitola 2

Studium problematiky

Tato kapitola je zaměřena na teoretické informace, jejichž nastudování bylo nezbytné pro další práci. Je zde vysvětlena problematika práce s genealogickými daty a tvorby genealogických modelů.

2.1 Genealogie

Genealogie je vědní disciplína zabývající se studiem vztahů mezi lidskými jedinci, vyplývajících z jejich společného rodového původu. Je řazena mezi pomocné vědy historické. Název disciplíny vznikl složením dvou slov z řeckého jazyka, a to ze slova „génos“ nebo „genus“, což v českém jazyce znamená „rod“, a ze slova „logos“, které by se v češtině dalo vyložit jako „slovo“ nebo jako „pojem“. Českým synonymem slova „genealogie“ je „rodopis“. [15] [21]

V minulosti se genealogie zabývala téměř výhradně šlechtickými rody a zkoumala příbuzenské vztahy jednotlivých členů rodu. Náležitost k některému šlechtickému rodu přinášela osobě určité výhody, proto nebylo neobvyklé, že docházelo k falšování listin dokládající příbuzenské vztahy. [21]

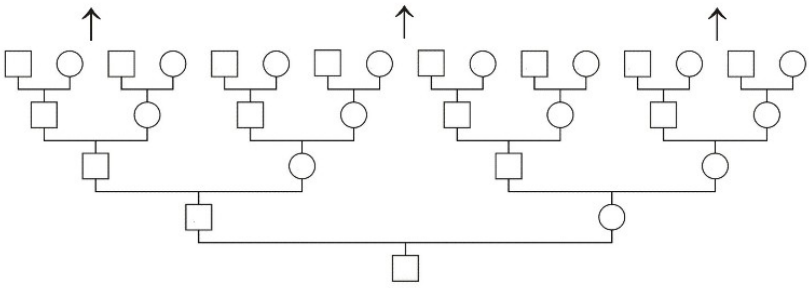
Předmětem jejího studia je tedy výzkum a zaznamenávání lidských rodových linií, přičemž vyzkoumaná data jsou zapisována do schémat, která jsou někdy označována jako genealogické tabulky. V souvislosti s genealogickými tabulkami je známo několik účelných termínů. Nejdůležitějším termínem je termín *střen*, neboli *probant*. Tento termín označuje osobu, která je výchozí osobou genealogického bádání a je tedy i první zaznamenanou osobou schématu, od které se odvíjí větvení schématu. Ve schématu jsou muži obvykle značeni symbolem čtverce a ženy symbolem kruhu.

2.2 Typy genealogických grafů

Existují 3 základní typy genealogických grafů, které se používají k záznamu historie rodu a dále existují další typy, které tyto základní typy kombinují.

2.2.1 Vývod z předků

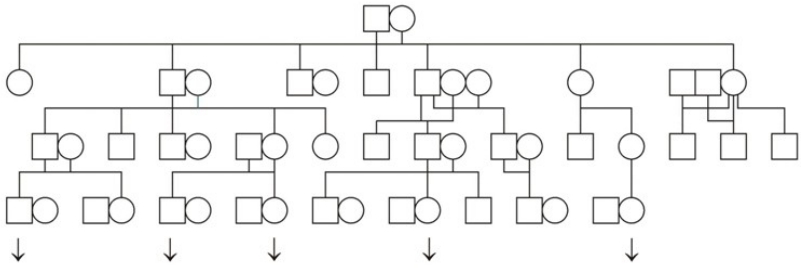
Prvním typem genealogických grafů je graf vývodu z předků. V rámci něho je *střenem* vybraná osoba a do grafu jsou zaznamenáváni přímí předci této osoby tzn. rodiče, prarodiče atd. S každou další generací roste počet jedinců v grafu geometrickou řadou. Schéma vývodu z předků je viditelné na obrázku 2.1.



Obrázek 2.1: Schéma struktury vývodu z předků [14]

2.2.2 Rozrod

Dalším typem grafu je rozrod. Tento graf je tvořen tak, že je zvolen libovolný pár předků, přičemž tento pár nazýváme zakladateli rodu, a do grafu dále zaznamenáváme přímé potomstvo zvoleného páru. Do tohoto grafu bývají zaznamenáváni i nemanželští potomci, protože jsou pokrevně spřízněni se zakladateli rodu. Samotné schéma rozrodu je viditelné na obrázku 2.2.



Obrázek 2.2: Schéma struktury rozrodu [14]

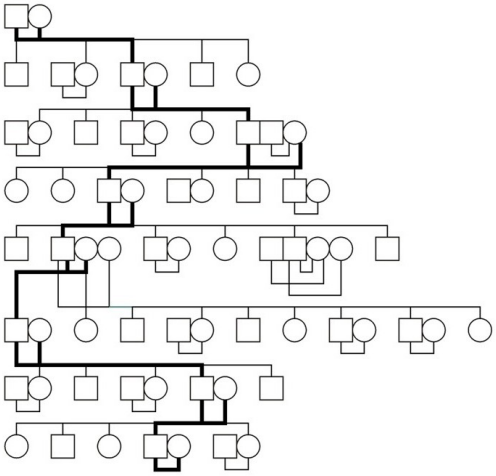
2.2.3 Rodokmen

Rodokmen je posledním ze základních typů genealogických grafů. Je podobný rozrodu v tom, že je vybrán jedinec (*střen*), jehož potomstvo je dále sledováno. Do rodokmenu jsou zaznamenáváni pouze nositelé příjmení, které je stejné jako příjmení vybraného *střena*. To znamená, že při sestavování rodokmenu je sledována pouze mužská linie potomků a to i v případě, že muž změnil příjmení. Dcery jsou zaznamenávány, ale jejich potomstvo už většinou zaznamenáváno není, kvůli odlišnosti jejich příjmení. Schéma rodokmenu je viditelné na obrázku 2.3.

2.3 Matriky

Matriky jsou veřejné úřední knihy sloužící k evidenci životopisných údajů občanů daného státu a bývají základním historickým pramenem, který se využívá při sestavování genealogických grafů. [19]

Knihy bývají spravovány a archivovány matričními úřady, které určí a jejich správní obvody vymezuje Ministerstvo vnitra ČR. Záznamy do matričních knih jsou prováděny



Obrázek 2.3: Schéma struktury rodokmenu [14]

zaměstnanci matričních úřadů. Záznamy jsou psány chronologicky a ručně do připravených knih. [24]

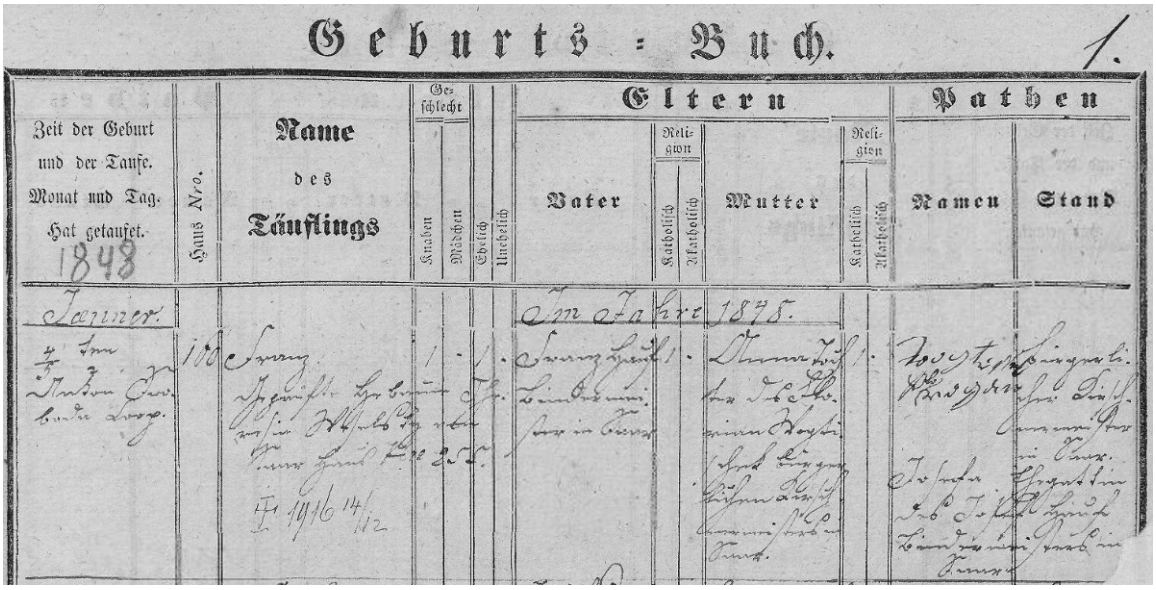
Podle legislativy České republiky existují tři druhy matrik pro tři zaznamenávané typy událostí:

- ***Kniha narození*** – do knihy se zapisuje:
  a) jméno, příjmení, rodné číslo, pohlaví, datum a místo narození dítěte
  b) jména, příjmení, rodná čísla, státní občanství, data a místa narození a místo trvalého bydliště obou rodičů
- ***Kniha manželství*** – do knihy se zapisuje:
  a) jména, příjmení, rodná čísla, data narození, místa narození, osobní stavy a státní občanství obou novomanželů
  b) jména, příjmení, data narození a místa narození rodičů novomanželů
  c) datum a místo uzavření manželství
  d) dohoda novomanželů na příjmení
  e) jména, příjmení a rodná čísla svědků novomanželů
- ***Kniha úmrtí*** – do knihy se zapisuje:
  a) jméno, příjmení, rodné číslo, pohlaví, osobní stav, státní občasntví, místo trvalého pobytu, datum a místo narození a úmrtí zemřelého
  b) osobní údaje pozůstalého (manžela nebo manželky)

Je zavedena také čtvrtá kniha, do které se zapisují záznamy o vzniklých registrovaných partnerstvích. [26]

Každý matriční záznam je opatřen datem zápisu a podpisem odpovědného úředníka (matrikáře). Zároveň je provedený matriční záznam doprovázen vydáním matričního dokladu (rodný list, oddací list, úmrtní list a doklad o registrovaném partnerství), který obsahuje údaje zapsané do matriční knihy v rámci záznamu. [25]

V rámci této práce se vychází ze starých matričních záznamů, které končí počátkem 20. století a mají pouze podobu naskenovaných listů matričních knih. Struktura těchto záznamů je odlišná od těch současných. Navíc se pro zápis používaly jazyky jako němčina nebo latina. Příklad záznamu z roku 1848 je na obrázku 2.4.



Obrázek 2.4: Příklad záznamu z knihy narození z roku 1848 [1]

2.4 Genealogický model

V této práci se pracuje s pojmem genealogický model. Genealogický model je rozšířením rodokmenu. Oproti rodokmenu se totiž v genealogických modelech pracuje navíc i se vztahy typu kmotr, porodník, křtitel a tak dále. Větší množství vztahů rozšíří možnosti vyhledávání ve finální databázi a zároveň v modelu umožní postihnout více osob, které měly něco společného s předky libovolné osoby.

Kapitola 3

Požadavky na výsledné řešení

Tato kapitola je rozdělena na tři části a řeší požadavky kladené na výsledný systém, výběr vhodného typu databáze a také popisuje existující řešení, která se svojí funkcionalitou podobají vytvářenému systému.

První část (3.1) vymezuje funkční požadavky kladené na strukturu dat, ukládání dat genealogických modelů a také na výslednou aplikaci. Další sekce (3.2) je soustředěna na výběr vhodného typu databáze. Jsou zde popsány a srovnány technologie pro tvorbu databáze, které budou vyhovovat požadavkům na ukládání genealogických dat. V poslední části této kapitoly (3.3) jsou rozebrána existující řešení, která se v jistých ohledech podobají řešené situaci.

3.1 Funkční požadavky

Cílem této práce je realizace webové aplikace a databáze pro ukládání genealogických modelů. Vytvořený systém uživateli umožní vytvářet a spravovat vlastní genealogické modely a prohlížet modely ostatních uživatelů. Jednotlivé osoby, které jsou součástí modelů odkazují na záznamy uložené v systému DEMoS. Označení genealogický model je zde použito, protože modely uložené v databázi nebudou obsahovat pouze příbuzenské vztahy, se kterými se lze setkat v klasických rodokmenech. V rámci databáze se totiž bude možno setkat také s osobami, které byly například kmotry, porodními bábami, křtiteli a dalšími.

Seznam funkčních požadavků je tedy takový:

- Možnost spravovat jednotlivé osoby v modelu a jejich vztahy s ostatními.
- Přítomnost vztahů jako kmotr, porodní bába, křtitel, svědek, oddávající, sourozenec a klasické vztahy, se kterými se lze setkat v rodokmenu, tzn. manžel, manželka, potomci.
- Propojení osob ve spravovaných modelech s databází přepsaných matričních záznamů pomocí identifikátorů těchto záznamů a v budoucnu také možnost doplnění dat k osobě na základě matričních záznamů.
- Kontrola uživatelských vstupů, kvůli zamezení případným chybám.
- Správa vícero genealogických modelů jedním uživatelem.
- Rozlišení jednotlivých uživatelů v systému a jejich přihlašování.
- Možnost sdílení vybraných modelů s ostatními uživateli.

- Umožnění exportu a importu dat genealogických modelů, aby je bylo možno zpracovat pomocí automatizovaných nástrojů.

3.1.1 Uživatelské vstupy

Uživatelské vstupy v aplikaci by měly být kontrolovány, aby se zamezilo případným chybám. Kontrolovány budou například sňatky osob stejného pohlaví, zda matka není zároveň svojí dcerou a podobně. Kontrola vztahů bude ztížena tím, že v databázi se budou moci objevit vztahy, se kterými se v klasickém rodokmenu není možné setkat. Kontrola informací, které budou přiřazeny k osobě pravděpodobně nebude tak striktní, protože objevená data mohou být neúplná. Jedinou povinnou informací o osobě bude její identifikátor v rámci databáze a její příslušnost k modelu, aby mohla být zobrazena ve webové aplikaci.

3.1.2 Možnosti konkrétního uživatele

Uživatel bude moci spravovat vícero genealogických modelů v rámci jednoho uživatelského účtu. Při vytváření modelu uživatel zadá jméno modelu a nastaví, jestli má být tento model sdílen s ostatními uživateli. Po vytvoření modelu bude moci uživatel spravovat tento model pomocí editoru, který bude sloužit i k prohlížení spravovaného modelu.

Uživateli bude umožněno editovat pouze jeden genealogický model současně. Tento model bude moci obsahovat více rodin. V jednom modelu se tedy bude možno setkat s větším množstvím grafů, které nebudou nijak propojeny. To bude plně v kompetenci uživatele. Správce modelu tedy bude moci přidávat osoby do modelu, editovat informace o nich a také mezi nimi vytvářet vztahy.

V rámci uživatelského účtu bude také možno prohlížet modely ostatních uživatelů. Tento přístup bude umožněn na základě nastavení, se kterým bude genealogický model vytvořen. V rámci prohlížení modelů bude možno klikat na jednotlivé osoby a prohlížet informace, které se k nim vážou. Dále se zde uživatel setká s možností exportu dat prohlíženého modelu. Exportovaná data modelu potom bude moci importovat do svého vlastního modelu, kde si jej bude moci upravovat například podle subjektivní relevance dat nebo podle dostupných historických pramenů.

3.1.3 Vztahy

Vztahy, se kterými se bude možno v databázi setkat jsou například vztahy muž a žena nějakého svazku, potomek, kmotr, svědek, křtitel, porodní bába a další. Je tedy nutno počítat se širším spektrem vztahů než, se kterým bychom se setkali v systému zabývajícím se správou klasických rodokmenů. Aby bylo toto spektrum v budoucnu možno ještě rozšířit, bude tomu muset být přizpůsoben návrh databáze. Vztahy budou v rámci modelu vytvářeny, tak, že při zvolení osoby bude nastaveno, s kým má tato osoba nějaký vztah.

3.1.4 Osoby

V jednom modelu bude obsažen libovolný počet osob. Tyto osoby bude moci správce modelu přidávat a odebírat v editoru modelů aplikace. Nová osoba bude vždy vytvořena s nějakým pohlavím. Pokud pohlaví osoby nepůjde určit, bude zde možnost přidat osobu s neurčitým pohlavím, přičemž pohlaví osoby s neurčitým pohlavím půjde změnit, pohlaví osob s určitým pohlavím nikoli.

Po přidání nové osoby do modelu bude možné k osobě doplnit informace o ní jako je například jméno, příjmení, rodné příjmení, datum narození, datum úmrtí, místo narození, místo úmrtí a další.

3.1.5 Propojení s databází přepsaných matričních záznamů

Osoby v modelu bude možno propojit s přepsanými matričními záznamy pomocí identifikátorů těchto záznamů. Pomocí identifikátoru, který bude přiřazen k osobě bude možno nahlédnout na připojené matriční záznamy a také zde bude možnost doplnit data o osobě v modelu na základě informací z těchto záznamů. Data, která budou doplněna k osobě na základě matričních záznamů, bude stále moci upravit správce toho modelu, ve kterém se tato osoba bude nacházet.

Toto propojení je hlavním požadavkem, který aplikace musí splňovat. Databáze digitalizovaných matrik, která bude obsahovat záznamy, se kterými budou osoby propojeny však není předmětem této práce, ale vzniká jako projekt na Fakultě informačních technologií Vysokého učení technického v Brně.[1]

Referenční identifikátor bude možno osobě přiřadit při editaci informací o této osobě. Při přiřazování identifikátoru bude správci modelu zobrazena nabídka sjednocení všech záznamů, které by se dané osoby mohly týkat, a po jeho zvolení dojde k uložení identifikátoru sjednocení matričních záznamů do dat osoby.

Je nutné počítat s faktem, že aplikace bude pracovat s informacemi, které nemusí být zcela přesné a mohou být i neúplné. Je tomu tak, neboť jedna osoba může mít v matrikách více záznamů a nelze tedy rozhodnout, který záznam obsahuje zcela pravdivé informace o osobě, a navíc může být například u data narození osoby uveden pouze měsíc a rok.

3.2 Výběr vhodného typu databáze

Pro výběr vhodného typu databáze bylo nutné analyzovat požadavky na strukturu a ukládání genealogických dat. Typy databází vhodné pro tento účel jsou relační databáze a grafové databáze, proto jsou analyzovány a popsány dále v této sekci. Nejvhodnějším typem databáze je takový, u kterého je zajištěna integrita dat. Aby byla zajištěna, je třeba aby transakce prováděné v databázi dodržovali vlastnosti ACID.[2]

3.2.1 Relační databáze

Jedná se o typ databáze, který již existuje déle, ale je stále velmi používaný. Čtyři nejpopulárnější databáze jsou relační databáze. Konkrétně se jedná o databáze Oracle, MySQL, MS SQL Server a PostgreSQL. [11]

Tento typ databází je založen na relační algebře a data jsou zde organizována jako sloupce a řádky v tabulkách. Každý řádek tabulky je identifikován unikátním klíčem, pomocí kterého se lze na data v tabulce odkázat. Základem pro práci s daty v relačních databázích je znalost jazyka *Structured Query Language (SQL)*. Mezi největší výhody tohoto jazyka patří jeho popularita a tedy i znalost jazyka SQL a základů relačních databází v populaci. Dalšími výhodami je například to, že databáze zachovává integritu dat a její struktura je snadno modifikovatelná.

Dvěma nejpopulárnějšími relačními databázemi jsou tyto:

[1] Podrobnosti o projektu jsou dostupné na: https://www.fit.vut.cz/research/project/1204/
[2] Atomicity, Consistency, Isolation, Durability

- ***Oracle*** – jde o databázi s pokročilými možnostmi zpracováním dat a s vysokým výkonem. Podporuje standardní jazyk SQL, ale také jazyk PL/SQL, který rozšiřuje funkčnost standardního SQL.
- ***MySQL*** – je databáze dostupná pod bezplatnou licencí GPL i pod komerční licencí. Využívá jazyk SQL s různými rozšířeními. Patří mezi nejpoužívanější databáze při tvorbě webových aplikací a webových informačních systémů.

3.2.2 Grafové databáze

Jde o databáze, které využívají grafové struktury k reprezentaci a uskladnění dat. Grafové struktury jsou tvořeny uzly a hranami. Uzly reprezentují jednotlivé datové objekty, ke každému objektu mohou přiléhat nějaké atributy, které jsou v uzlu obsaženy. Hrany mezi uzly obsahují vztahy, které mezi sebou uzly mají. Hrany mohou být orientované i neorientované a mohou nést informace o typu vztahu a stejně jako uzly mohou obsahovat nějaké atributy.

Transakce v grafových databázích ve většině případů nemají vlastnosti ACID, takže integritu dat musí zajišťovat programátor. Existuje však i grafová databáze, jejíž transakce mají vlastnosti ACID, jedná se například o databázi Neo4j.

Oproti relační databázi je zde jednodušší přidávat a upravovat vztahy (hrany) mezi jednotlivými položkami, protože kvůli každému novému vztahu není třeba upravovat celé schéma databáze. Genealogický model (rodokmen) je ve většině případů prezentován jako graf, takže grafová databáze je pro implementovanou databázi vhodnějším typem databáze. Navíc je v nich možné, na rozdíl od relačních databází, rychleji vyhledávat ve velkém množství dat. Grafové databáze používají i velké společnosti jako je například Google nebo Amazon.

Dvěma nejpopulárnějšími grafovými databázemi jsou tyto:

- ***Neo4j*** – je open-source databáze, která je implementována v jazyce Java. K práci s daty lze využít jazyka Cypher Query Language nebo Gremlin Language, což jsou oba jazyky inspirované jazykem SQL.
- ***ArangoDB*** – jde o open-source databázi implementovanou v jazycích C++ a Javascript. Pro manipulaci s daty je nutné naučit se jazyk AQL (ArangoDB Query Language), který je v mnoha ohledech podobný jazyku SQL. Narozdíl od databáze Neo4j však její transakce nemají vlastnosti ACID.

3.3 Existující řešení

Existuje velké množství řešení, které slouží k vytvoření a správě vlastních rodokmenů. Tato řešení však pracují pouze s klasickými rodokmeny a ve většině případů nejsou nijak provázána s databází matričních záznamů. Jejich prozkoumání však pomůže vyvinout aplikaci, která obsáhne jejich kladné vlastnosti a vyvaruje se jejich nedostatků.

3.3.1 Family Echo

Jedním z existujících řešení, které poskytuje možnost správy rodokmenu je webová aplikace *Family Echo*.[3] Jedná se o projekt, který lze využívat zdarma a slouží k soukromému použití. Uživatel zde má možnost importovat si do aplikace rodokmen v podobě souboru ve

[3] https://www.familyecho.com/

formátu GEDCOM nebo FamilyScript a dále jej upravovat. Pokud se uživatel zaregistruje, má možnost kromě přidávání členů a tisku výsledného stromu také ukládat rodokmen, přidávat fotografie jednotlivých osob rodokmenu, sdílet rodokmen s ostatními a také stáhnout rodokmen jako soubor ve formátu HTML, GEDCOM, CSV a jiné.

Rodokmen je graficky zobrazován a rozhraní je vcelku přehledné. Nevýhodou je, že uživatel má možnost spravovat pouze jeden rodokmen a také to, že exportovat rodokmen je možné pouze po přihlášení, ale importovat jej lze i bez přihlášení. Další nevýhodou je to, že výchozí osobou rodokmenu je vždy pouze osoba uživatele, tudíž není možné vytvořit rodokmen někoho jiného.

3.3.2 Family Tree Creator

Dalším řešením je nástroj *Family Tree Creator*,[4] který je dostupný v rámci webových stránek *DNAweekly*. Tento nástroj umožňuje uživateli spravovat rodokmen bez nutnosti registrace. Výchozí osobou je stejně jako u předchozího řešení opět osoba uživatele. Rodokmen je možno uložit a po uložení je vygenerován link, pod kterým je možné k uloženému rodokmenu přistupovat. Nástroj je celkově možná až přehnaně jednoduchý a správce rodokmenu asi spíše zvolí propracovanější nástroj. Hlavní nevýhodou nástroje je pouze malé množství údajů, které je možné k osobě doplnit. Mezi tyto údaje patří pouze jméno, pohlaví, datum narození, datum úmrtí a fotografie osoby. Dále hodnotím negativně to, že není možné vyexportovat rodokmen například ve formátu GEDCOM. Export je možný pouze graficky jako PDF nebo JPEG. Import rodokmenu taktéž není možný.

3.3.3 FamilySearch

FamilySearch[5] je služba, kterou spravuje nezisková organizace přidružená k Církvi Ježíše Krista Svatých posledních dnů. Jde o službu, která je zdarma a vyžaduje pouze registraci. Výhodou je, že pokud má v rámci této služby nějaký rodokmen vytvořen osoba, kterou uživatel vloží do svého rodokmenu, je mu do rodokmenu přidán i rodokmen vkládané osoby. Toto propojení šetří práci a urychluje vytvoření rozsáhlého rodokmenu. Systém pracuje se záznamy, které jsou popsány na vyhrazené stránce[6] a po registraci je k nim umožněn přístup. Na základě těchto záznamů je možno vložit osobu do rodokmenu nebo je možné přidat osobu, která žádný takovýto záznam nemá. Nevýhodou je opět to, že v rámci jednoho uživatelského účtu je možno spravovat pouze jeden rodokmen, který se navíc týká pouze osoby uživatele.

3.3.4 MyHeritage

Posledním z prozkoumaných řešení je služba *MyHeritage*,[7] která kromě správy rodokmenů umožňuje také provedení DNA testu pro určení původu uživatele nebo vylepšení a obarvení starých fotografií pomocí strojového učení. Jde o nejznámější službu s podobným zaměřením jako je vypracovávaný projekt. Řešení, ale na rozdíl od předchozích není zdarma. Pro zájemce je dostupná pouze 14 denní trial verze. Předplatné je poté děleno na balíčky: Kompletní, Data, Premium a Premium Plus.

[4] https://www.dnaweekly.com/tools/family-tree-maker/
[5] https://www.familysearch.org/cs/
[6] https://www.familysearch.org/search/collection/list/
[7] https://www.myheritage.cz/

Kompletní balíček obsahuje vše co služba nabízí. Balíček Data umožňuje předplatiteli přístup k historickým záznamům sloužícím k bádání a také umožňuje spravovat rodokmen do velikosti 250 osob. Balíček Premium umožňuje správu rodokmenu do velikosti 2500 osob a také propojení s výše zmíněnými DNA testy. Poslední balíček Premium Plus navíc oproti balíčku Premium nabízí možnost správy neomezeně velkého rodokmenu a také kontrolu konzistence rodokmenu včetně návrhu oprav. Výhodou služby je, že osoby lze propojit s digitalizovanými historickými prameny.

3.4 Význam řešení

V sekci 3.3 jsou shrnuta pouze některá z existujících řešení, která se zaměřením podobají aplikaci vznikající v rámci této práce. Co je tedy motivací pro vývoj nové aplikace s tímto zaměřením?

Výsledná aplikace se oproti většině existujících řešení liší především v provázání s databází digitalizovaných matričních záznamů. Toto provázání umožní přesněji vytvářet výsledné genealogické modely, protože osoby přidávané do modelu budou moci být založené na reálném historickém pramenu bez nutnosti dešifrovat mnohdy obtížně čitelné písmo. Pokud by se však správci modelu nezamlouvala data získaná z matričních záznamů, bude mít možnost si tato data upravit například podle jiného historického pramenu. Uživatel se díky tomuto provázání také snadno dostane ke genealogickému záznamu a bude jej tak moci podrobněji prozkoumat v případě, že data o osobách zobrazená ve vyvíjené aplikaci mu nebudou postačovat.

Další výrazný rozdíl oproti existujícím řešením je fakt, že aplikace bude pracovat s genealogickými modely a ne pouze s rodokmeny. To do aplikace přidá spoustu pro rodokmen neobvyklých vztahů. Pojem genealogický model byl již vysvětlen v sekci 2.4. Import a export modelů navíc umožní v databázi uložit a v aplikaci zobrazit sestavené modely, které mohly vzniknout pomocí automatizovaných nástrojů. Aplikace se bude pravděpodobně dále vyvíjet i nad rámec této práce. A bude doplněna o další vztahy, které bude možno zachytit v genealogických modelech a tudíž i v databázi vznikající v rámci této práce.

Kapitola 4

Použité technologie

Hlavním účelem této kapitoly je seznámit čtenáře s technologiemi, které byly využity při realizaci výsledného řešení. Mezi tyto technologie patří programovací jazyky, software atd.

4.1 Webové aplikace

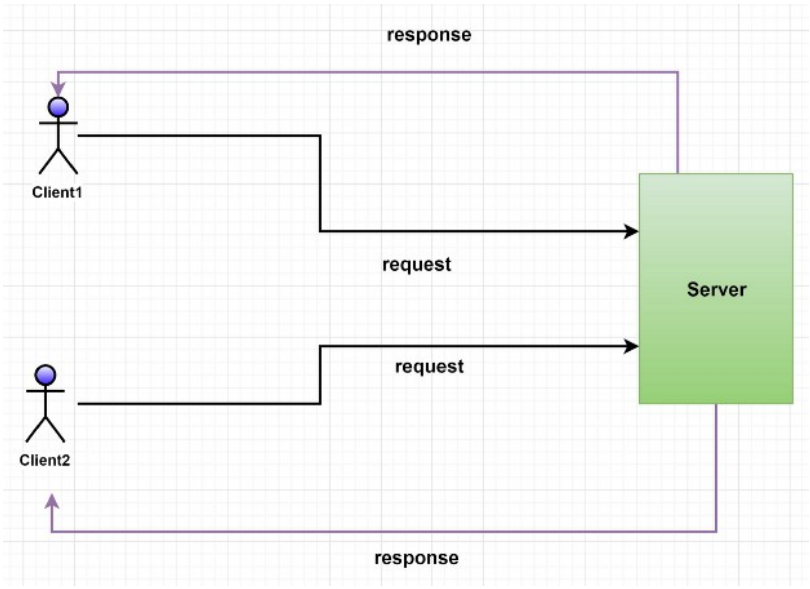
Na začátek je třeba nutný stručný úvod do teorie související s tvorbou webových aplikací, neboť její znalost je nutná k pochopení funkcionality aplikace vzniklé v rámci této práce.

Existují dva typy webových stránek a to stránky statické a stránky dynamické. Rozdíl mezi nimi je v proměnlivosti obsahu. U statických webových stránek je jejich obsah pevně daný, a proto jsou stránky tohoto typu užité většinou k informačním účelům např. při tvorbě osobních blogů. Veškeré dynamické funkce probíhají na straně klienta a jde pouze o funkce, které slouží k triviálním úkonů jako je např. odhalení skrytého obsahu po kliknutí na tlačítko. Pro tvorbu statických webových stránek nám stačí znalost 3 základních programovacích jazyků, které jsou zpracovávány prohlížečem a to: HTML, CSS a JavaScript. [9]

Dynamické webové stránky oproti tomu mohou měnit obsah a tato vlastnost je využita například na stránkách internetových obchodů, ve webových informačních systémech atd. Funkce zde už neprobíhají pouze na straně klienta, ale také na straně serveru. Server pracuje s nějakou datovou vrstvou (např. databází, souborovým systémem) a na základě toho generuje obsah na stránce, která je vracena na stranu klienta. V rámci dynamických webových stránek se pracuje s termínem *CRUD*. Tato písmena označují operace, které server provádí v datové vrstvě, jsou to operace *Create*, *Read*, *Update*, *Delete* (vytvořit, číst, aktualizovat, smazat). Při tvorbě dynamických webových stránek jsou jazyky HTML, CSS a JavaScript doplněny o jazyky zpracovávané serverem jako například PHP, Python, ASP.net aj. [9]

4.1.1 Architektura klient-server

Síťová architektura, která je založena na komunikaci klientů (počítač, mobilní telefon a další) a serveru, se nazývá *architektura klient-server*. Tato komunikace probíhá tak, že klient požádá server o data nebo službu, načež server nějakým způsobem reaguje a vrací odpověď klientovi. Klient může být připojen k vícero serverům současně a server ve většině případů může zpracovávat žádosti vícero klientů současně. Architektura je naznačena na obrázku 4.1.



Obrázek 4.1: Princip architektury klient-server [10]

4.1.2 Třívrstvá architektura

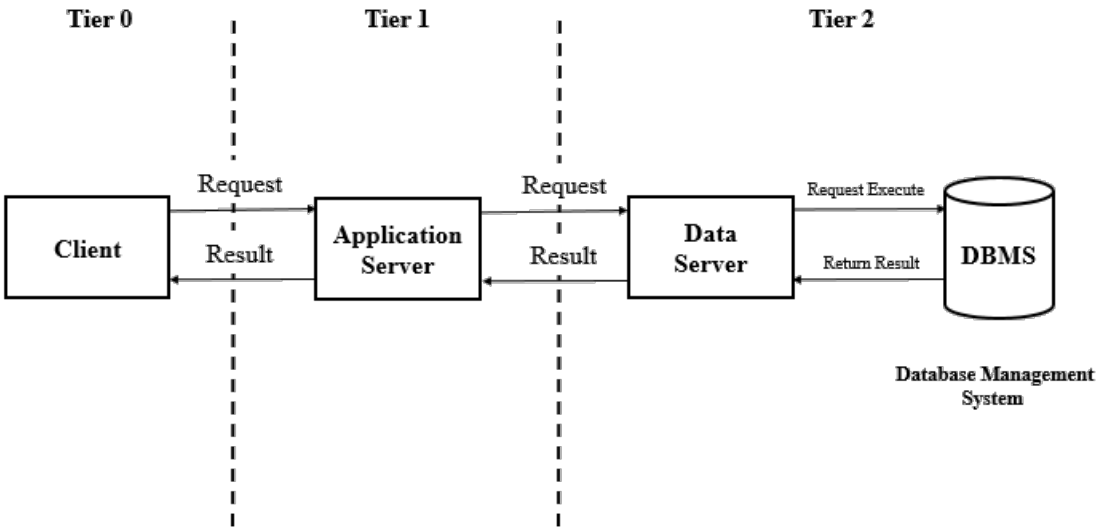
Princip třívrstvé architektury je použit v tomto řešení a jedná se o hojně používanou architekturu webových aplikací. Jak již název napovídá, aplikace je dělena do 3 logických a fyzických vrstev, které mohou být vyvíjeny nezávisle na sobě:

1. ***Prezentační vrstva*** – zahrnuje uživatelské a komunikační rozhraní aplikace. Zobrazuje informace uživateli a předává uživatelské vstupy aplikační vrstvě. V rámci vrstvy může být implementována také kontrola uživatelských vstupů. Vrstvu je možno implementovat ke zobrazení ve webovém prohlížeči, nebo jako počítačovou aplikaci. Prezentační vrstva určená pro webový prohlížeč bývá obvykle implementována v jazycích HTML, CSS a JavaScript.

2. ***Aplikační vrstva*** – říká se jí také logická vrstva a obsahuje veškerou logiku aplikace. Slouží jako most mezi prezentační a datovou vrstvou. Uživatelské vstupy zachycené na prezentační vrstvě jsou zde zpracovány a na jejich základě může dojít ke čtení, mazání, úpravě nebo zápisu dat na datové vrstvě. Pro implementaci lze použít jazyky Python, PHP, Ruby aj.

3. ***Datová vrstva*** – slouží k uchování a zpracování informací spravovaných v rámci aplikace. Říká se jí také databázová vrstva, protože bývá velmi často tvořena databází. Může jít například o databázi MySQL, PostgreSQL, Neo4j a další.

Mezi hlavní výhody této architektury patří možnost nezávislého vývoje jednotlivých částí aplikace a s tím související rychlost vývoje, bezpečnost a integrita dat ukládaných do datové vrstvy, která vychází s předpokladu, že prezentační vrstva nekomunikuje přímo s datovou vrstvou. Diagram znázorňující architekturu je možné vidět na obrázku 4.2. [13][16]



Obrázek 4.2: Princip architektury klient-server [6]

4.2 Frontend

Frontend je označení pro prezentační vrstvu aplikace. Popis prezentační vrstvy lze nalézt v části 4.1.2. Jde tedy o vrstvu webové aplikace, která bude zobrazovat uživateli data interpretovaná tak, aby jim rozuměl. Navíc bude sloužit k získávání uživatelských vstupů, jejich kontrole a k jejich předávání aplikační vrstvě, která je dále bude zpracovávat. Všechny komponenty patřící do této vrstvy jsou dostupné klientovi, vrstva tedy není vhodná například k nastavování přístupových údajů. V rámci vyvíjeného řešení do této vrstvy budou patřit části napsané v jazycích HTML, CSS a JavaScript. [12]

4.2.1 HTML

HTML je značkovací jazyk, který se používá pro vytváření internetových stránek. Dokumenty napsané v tomto jazyce jsou po dotazu klienta na server vráceny klientovi a určují jaký obsah mu bude zobrazen a jakým způsobem. Tento jazyk je podporován všemi internetovými prohlížeči.

HTML dokument je textový soubor, který má specifický syntax a bývá uložen s koncovkou `.html`. Na začátku dokumentu obvykle bývá návěští `<!DOCTYPE html>`, které slouží k oznámení, že od tohoto řádku dolů se jedná o HTML dokument.

Syntax vypadá tak, že v rámci jazyka existují značky, které se používají k označení různých prvků dokumentu. Obsah, který je vymezen značkami, může být text, další značky nebo značkovací jazyky. HTML je také možno kombinovat s dalšími značkovacími jazyky, kde příkladem může být jazyk SVG. Prvky, se kterými lze pracovat v rámci dokumentu jsou dvojího typu: [17][20]

1. ***blokové prvky*** – začínají na novém řádku dokumentu a vymezují prostor, který je rozložen na vícero řádků. Příkladem značek pro vymezení může být `<div>` pro sekci dokumentu, `<table>` pro tabulku nebo `<nav>` pro navigační odkazy.

2. ***řádkové prvky*** – nezačínají na novém řádku a vymezují pouze nezbytný prostor. Jejich využití spočívá především v určení formátu obsahu blokových prvků. Značky řádkových prvků jsou například `<button>` pro tlačítko, `<a>` pro odkaz nebo `<img>` pro obrázek.

Je možné vytvořit jednoduché statické webové stránky pouze v HTML, ale pokud chceme vytvořit stránky, které jsou responzivní a pěkně vypadají, tak je nutné jazyk HTML kombinovat s CSS a s jazykem JavaScript.

4.2.2 CSS

CSS je jazyk, který se používá ke specifikaci vzhledu a chování prvků dokumentů psaných ve značkovacích jazycích. Nejčastěji je používán v kombinaci s jazyky HTML a JavaScript při vytváření webových stránek. Pomocí jazyku je možno zarovnávat obsah na stránce, zakulacovat hrany obrázků, měnit font a barvu textu, upravovat vzhled pro chytré telefony a další. Je také možno specifikovat, jak se vzhled prvků dokumentu změní například při kliknutí nebo přejetí na ně. V CSS se využívá tzv. selektorů, které slouží k výběru prvku značkovacího jazyka, který bude upravován. [8]

4.2.3 JavaScript

JavaScript je označení pro interpretovaný skriptovací jazyk vyvinutý společností *Netscape*. Zpočátku byl užíván především v prohlížeči Netscape a později byl začleněn do ostatních prohlížečů. Je to objektově orientovaný a dynamicky typovaný jazyk. Jeho syntaxe je podobná jazykům Java, C/C++, ale sémanticky se jim vůbec nepodobá. [7]

Používá se jak na straně klienta, tak na straně serveru. Na straně serveru se s ním můžeme setkat například v rámci platformy Node.js, která nahrazuje klasické PHP a liší se od něho například ve zpracování požadavků klienta, ty jsou v případě Node.js zpracovávány asynchronně a nedochází tedy k zablokování serveru. [18]

4.2.4 jQuery

jQuery je JavaScript frameworky obvykle používaný k vývoji klientského rozhraní aplikace. Je vyvinut v rámci *The jQuery Project*. V rámci tohoto projektu vznikají také frameworky jako například jQuery Mobile, sloužící ke tvorbě mobilních aplikací, nebo QUnit, sloužící k provádění jednotkového testování při vývoji. jQuery výrazně usnadňuje práci především tím, že pro manipulaci s objekty v aplikaci se používají CSS selektory. Výhodou je také možnost řetězení volání metod pro vybraný prvek. jQuery snižuje objem vyprodukovaného kódu oproti čistému JavaScriptu při zachování funkcionality. [23]

4.2.5 Bootstrap

Bootstrap je otevřený framework užívaný k tvorbě uživatelských rozhraní webových stránek. Skládá se z komponent napsaných v jazycích HTML, CSS a JavaScript. Uveden byl roku 2011 společností *Twitter*, proto je také známý jako *Twitter Blueprint*. Používá se především pro zajištění responzivity webových stránek, aby byly dobře čitelné i na telefonních zařízeních a na zařízeních s jiným rozlišením, než na kterém je webová aplikace vyvíjena. Často jej proto nalezneme v kombinaci například s jQuery (viz. 4.2.4) a s dalšími frameworky. [22]

4.2.6 Vis.js Community Edition

Sada knihoven pro dynamickou vizualizaci dat s názvem Vis.js je složena s částí, které se zaměřují na různé typy vizualizace. Dělí se tedy na části: *Network*, *Timeline*, *Graph3d* a *Graph2d*. V rámci tohoto projektu je použita pouze knihovna pro vizualizaci sítí, tudíž Vis.js Network. Vis.js Community Edition má 2 licence, jednou je licence *Apache 2.0* a druhá je licence *MIT*, tudíž může být využita v tomto projektu.

Vis.js Network umožňuje vývojáři měnit tvar, styl, velikost a další vlastnosti uzlů a hran sítě. Podporuje shlukování uzlů, což se hodí při větším množství dat. Knihovna podporuje všechny moderní prohlížeče a k vizualizaci používá HTML prvek *canvas*. Více informací o projektu a knihovnách lze nalézt na stránkách [2].

4.3 Backend

Backend je aplikační vrstvou aplikace. Popis aplikační vrstvy lze nalézt v části 4.1.2. Zkráceně jde o část aplikace, která nebude dostupná běžnému uživateli webové aplikace. Vrstva slouží pro zpracování a zprostředkování komunikace mezi datovou vrstvou (databází) a uživatelem. V rámci vyvíjené aplikace sem lze zařadit části napsané v jazyce PHP, které budou sloužit k sestavování a předávání dotazů datové vrstvě na základě uživatelského vstupu a také k ukládání a formátování dat získaných z databáze do prezentační vrstvy. [12]

4.3.1 PHP

PHP nebo také *PHP: Hypertext Preprocessor* je otevřeným imperativním skriptovacím jazykem, který se hojně využívá k vývoji dynamických webových stránek a lze jej prokládat jazykem HTML, od něhož je oddělen pomocí instrukcí `<?php` a `?>`, které označují ohraničují začátek a konec prováděného PHP kódu.

Úkony popsané skriptem jsou prováděny na straně serveru a zpravidla generují HTML kód, který je poté odeslán klientovi. Protože se jazyk používá pro vývoj webových aplikací, tak podporuje velké množství databází a také umí pracovat s různými síťovými protokoly jako je například IMAP, POP3, LDAP aj. Dále je jazyk PHP vhodný ke zpracování textových souborů díky možnosti práce s regulárními výrazy. PHP je také mimo vývoje webových aplikací možné použít například pro vývoj desktopových aplikací. A při vývoji v jazyce PHP je také možno kombinovat procedurální i objektově orientované programování. [4][5]

Kapitola 5

Návrh systému

V rámci řešení bylo nutné navrhnout jednotlivé části implementované aplikace. Strukturu databáze, uživatelské rozhraní a také vrstvu pro komunikaci mezi nimi. Návrh databáze je založen na požadavcích na strukturu a možnostech činností prováděných uživatelem ve výsledné aplikaci. Vzhled a rozložení uživatelského rozhraní je ovlivněno existujícími řešeními z části 3.3 a také funkčními požadavky v části 3.1, přičemž bylo nutné inspirovat se pouze kladnými vlastnostmi rozhraní existujících řešení a vyvarovat se záporným vlastnostem. V rámci návrhu vrstvy mezi databází a GUI bylo nutné zvolit vhodné příkazy pro zápis a načítání dat z databáze.

5.1 Návrh databáze

První řešenou částí návrhu byla databáze. Vzhledem k faktu, že se jedná o grafovou databázi, data jsou ukládána v uzlech, konkrétně v podobě vlastností uzlů. Mezi uzly vznikají různé druhy vztahů, které také bývají doplněny o vlastnosti. Uzly v databázi mají stejně jako vztahy více možných typů. V rámci databáze nakonec vzniklo 6 druhů uzlů a 11 druhů vztahů mezi uzly.

5.1.1 Uzly

Tato část je zaměřena na uzly, které mohou vznikat v rámci databáze.

Prvním druhem uzlů, které jsou použity nesou označení `User`, ty obsahují informace o jednotlivých uživatelích webové aplikace. Vlastnosti, které jeden uzel obsahuje jsou tři:

- **`username`** – nese unikátní uživatelské jméno uživatele v systému
- **`password`** – obsahuje uživatelské heslo, které je zashashováno Bcrypt algoritmem
- **`uuid`** – univerzální unikátní identifikátor uživatele, který uzlu přiřazuje databáze

Dalším typem uzlů je `Model`, ten nese informace o každém genealogickém modelu, který je v rámci databáze vytvořen. Vlastnosti tohoto druhu uzlu jsou:

- **`name`** – název genealogického modelu, který je zvolen uživatelem
- **`visibility`** – viditelnost modelu pro ostatní uživatele (0 - soukromý, 1 - veřejný)
- **`creation`** – časové razítko vytvoření modelu

- **last_edit** – časové razítko poslední úpravy modelu
- **uuid** – univerzální unikátní identifikátor modelu použitý vzhledem k volatilitě identifikátoru, který uzlům přiřazuje databáze

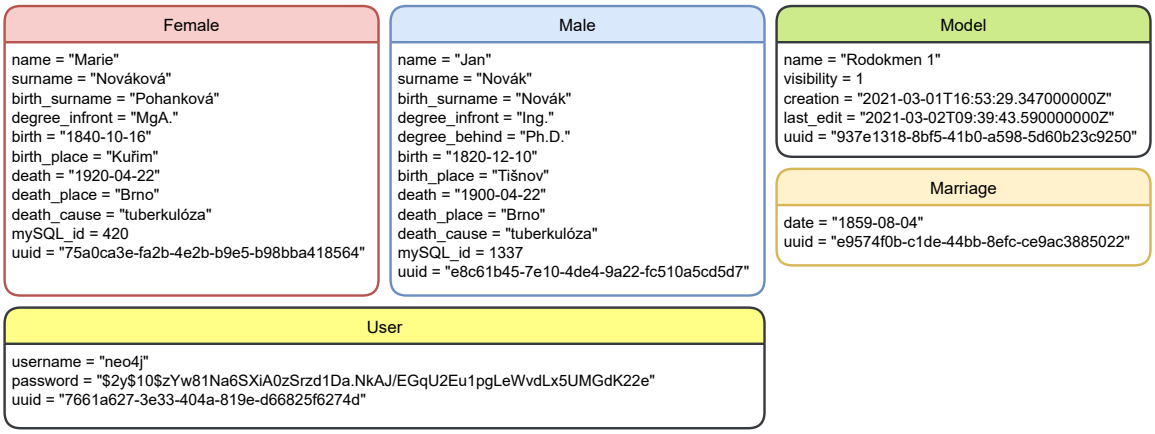
Dalšími třemi druhy uzlů jsou uzly, které slouží k uchování informací o osobách. Jde o uzly s označením `Male`, `Female` a `Unknown`, do kterých jsou osoby uloženy na základě svého pohlaví. V případě uzlu typu `Unknown` se bude jednat o osobu s pohlavím, které podle pramenů nelze bezpečně určit. Doplnitelné vlastnosti těchto tří druhů uzlů jsou stejné. Záznamy o osobách nemusí být vždy kompletní a uzel osoby tudíž nemusí obsahovat všechny vlastnosti. Jedinou vlastnost, kterou bude třeba určit při vytvoření osoby bude její pohlaví a při vytvoření bude do uzlu doplněn univerzální unikátní identifikátor osoby daného pohlaví. Všechny vlastnosti, které mohou uzly `Male`, `Female` a `Unknown` mít jsou:

- **name** - jméno
- **surname** - příjmení
- **birth_surname** - rodné příjmení
- **degree_infront** - titul osoby psaný před jménem
- **degree_behind** - titul osoby psaný za jménem
- **birth** - datum narození
- **birth_place** - místo narození
- **death** - datum úmrtí
- **death_place** - místo úmrtí
- **death_cause** - příčina úmrtí
- **mySQL_id** - identifikátor příslušící záznamu osoby v rámci existující databáze digitalizovaných matričních záznamů
- **uuid** - univerzální unikátní identifikátor osoby (unikátní v rámci uzlů `Male` a v rámci uzlů `Female`)

Posledním typem uzlu v databázi je uzel se značkou `Marriage`, který označuje svazek vzniklý mezi 2 osobami, který může a nemusí být manželský. Kromě osob náležících do svazku je spojen s osobou oddávajícího a s potomky, kteří vznikly z tohoto svazku. Pokud bude například otec potomka neznámý, bude mezi matkou a potomky stejně existovat tento uzel. Vlastnosti uzlu jsou:

- **date** - je datum vzniku svazku nebo datum svatby
- **uuid** - univerzální unikátní identifikátor svazku

Příklad uzlů a vlastností, které mohou mít, je viditelný na obrázku 5.1. Na tomto obrázku není pouze uzel typu `Unknown`, protože jeho vlastnosti jsou totožné s vlastnostmi uzlů typu `Male` a `Female`.



Obrázek 5.1: Příklady typů uzlů s některými vlastnostmi, které mohou mít

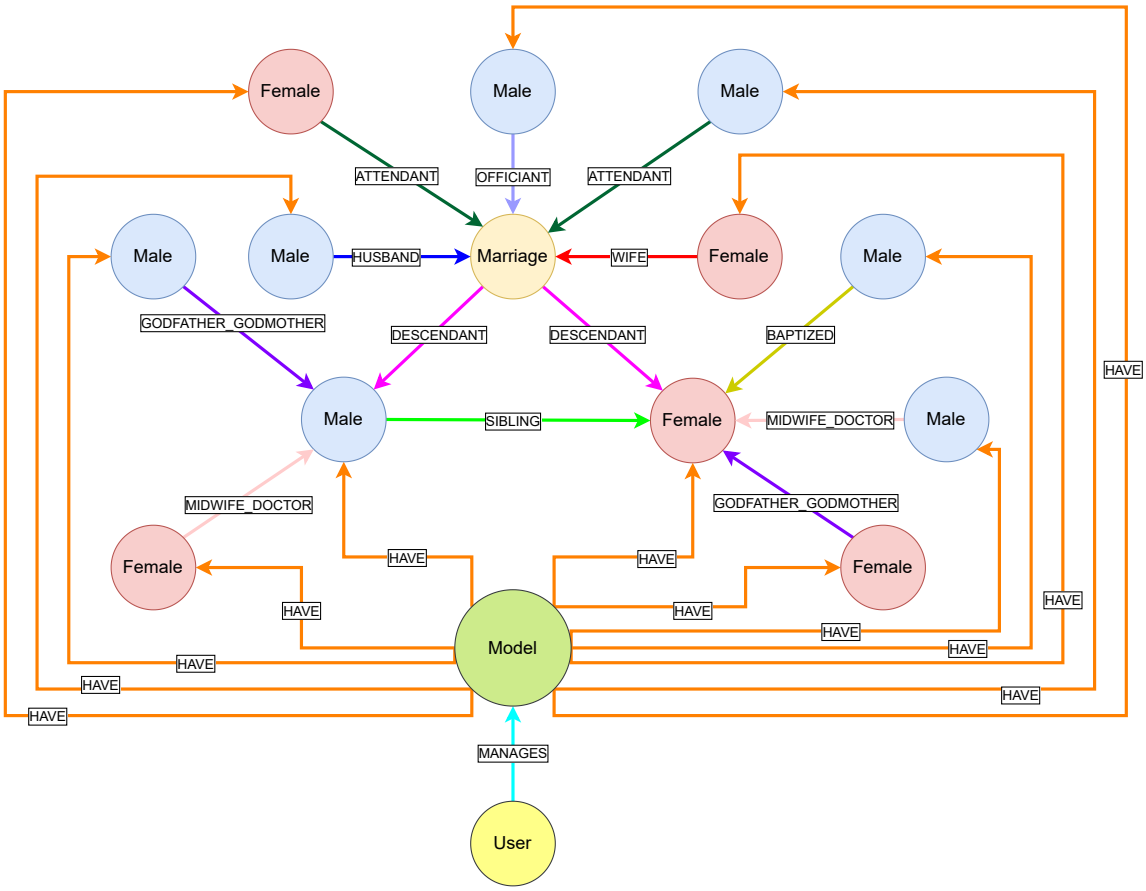
5.1.2 Vztahy

Mezi jednotlivými uzly v databázi může vznikat 11 druhů vztahů, konkrétně se jedná o vztahy:

- `MANAGES` – ukazuje z uzlu typu `User` na uzel typu `Model` a specifikuje, který uživatel spravuje jaký genealogický model.
- `HAVE` – spojuje uzel typu `Model` s uzly typu `Female`, `Male` a `Unknown`, přičemž směřuje z uzlu modelu na osoby a specifikuje tak, které osoby náleží do genealogického modelu.
- `HUSBAND` – je vztah, který vzniká mezi uzlem typu `Male` a uzelem typu `Marriage`. Směřuje z uzlu osoby muže na uzel svazku a specifikuje tak mužského partnera svazku.
- `WIFE` – je vztah, který vzniká mezi uzlem typu `Female` a uzelem typu `Marriage`. Směřuje z uzlu osoby ženy na uzel svazku a specifikuje tak ženského partnera svazku.
- `DESCENDANT` – vzniká mezi uzly typu `Marriage` a uzly typu `Male`, `Female` nebo `Unknown`. Ukazuje z uzlu svazku na potomky tohoto svazku. Tento vztah obsahuje booleovskou vlastnost `biological`, která specifikuje, jestli je potomek biologický nebo osvojený.
- `SIBLING` – vzniká mezi uzly typu `Male`, `Female` a `Unknown`, tedy mezi uzly osob 2 sourozenců. Mezi 2 sourozenci existuje vždy 1 tento vztah, i přes skutečnost, že se jedná o orientovaný vztah. Je tedy vytvořen orientovaně, protože jiným způsobem to databáze nedovoluje. Databáze však povoluje dotazování se bez orientace. Vztah obsahuje booleovskou vlastnost `biological`, která specifikuje, jestli jsou sourozenci vlastní nebo nevlastní.
- `GODFATHER_GODMOTHER` – vztah ukazující z uzlu kmotra nebo kmotry na uzel kmotřence respektive kmotřenky. Vztah nemá žádné vlastnosti a vzniká pouze mezi uzly typu `Male`, `Female` a `Unknown`.
- `BAPTIZED` – vztah vzniká mezi křtěnou osobou a osobou, která křest vykonávala, přičemž ukazuje z osoby provádějící křest. Opět vzniká pouze mezi uzly typu `Male`, `Female` a `Unknown`.

- `MIDWIFE_DOCTOR` – jedná se o vztah, který vzniká mezi osobou dohlížející na porod a osobou, která je výsledkem tohoto porodu. Vzniká pouze mezi uzly typu `Male`, `Female` a `Unknown` a je orientován tak, že vychází z osoby dohlížející na porod.
- `ATTENDANT` – tento vztah slouží k přiřazení osoby svědka respektive svědkyně k manželství (svatbě). Vzniká mezi uzlem typu `Marriage` a uzlem typu `Male`, `Female` nebo `Unknown`. Je orientován tak, že ukazuje na manželství a může mít až 1 vlastnost, která určuje, které straně svazku osoba svědčila. Vlastnost je označena `side` a může nabývat řetězcových hodnot *husband* a *wife*.
- `OFFICIANT` – poslední vztah může vzniknout mezi uzlem manželství (svatby) a osobou, která vykonávala obřad. Je orientován z uzlů typu `Male`, `Female` nebo `Unknown` na uzel typu `Marriage`.

Příklad propojení, které může vzniknout mezi uzly je znázorněno na obrázku 5.2.



Obrázek 5.2: Příklad části grafu, který může vzniknout v databázi

5.2 Návrh architektury aplikace

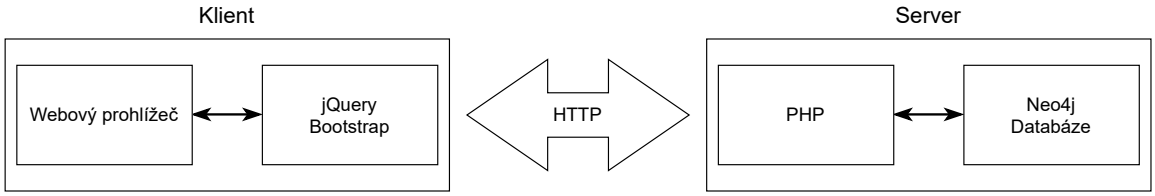
Architekturu aplikace lze rozdělit na tři části. Jednou z těchto částí je datová vrstva, jejíž návrh byl vysvětlen již v sekci 5.1. Datovou vrstvu reprezentuje grafová databáze Neo4j.

Návrh architektury byl řízen modelem třívrstvé architektury, která již byla popsána v rámci části 4.1.2. Na základě tohoto modelu je aplikace rozdělena na tyto části:

- datová vrstva (databáze Neo4j)
- aplikační vrstva (backend aplikace napsaný v jazyce PHP)
- prezentační vrstva (frontend aplikace vytvořen s pomocí frameworků Bootstrap a jQuery)

Jak již bylo vysvětleno v sekci 4.1.2, prezentační vrstva a datová vrstva musí být propojeny vrstvou aplikace, která obsahuje pouze samotnou logiku aplikace a stará se o zpracování a předávání dat mezi uživatelským rozhraním a databází. Touto vrstvou je zmíněný backend aplikace, který představuje vrstvu aplikační.

Na obrázku 5.3 je vyjádřena struktura aplikace, tedy její rozdělení podle třívrstvé architektury a dále také podle architektury klient-server. A je zde patrné, že aplikační vrstva a s ní spojená databáze komunikuje s klientským webovým prohlížečem.



Obrázek 5.3: Schéma návrhu architektury implementované aplikace

Aplikační vrstva zpracovává HTTP požadavky typu GET, POST, SET a DELETE specifikované na základě HTML formulářů a posílané ze strany klienta. Na základě těchto požadavků jsou prováděny implementované operace. Aplikační vrstva se konkrétně stará o kontrolu připojení k databázi, kontrolu zpracovávaných dat, manipulaci s databází a další. Server obvykle na požadavek klienta reaguje odpovědí obsahující například požadovaná data nebo výsledek prováděné operace. V rámci projektu je aplikační rozhraní implementováno pomocí PHP (4.3.1) konkrétně ve verzi 7.4.15.

Prezentační vrstva aplikace je implementována kombinací jazyků HTML (4.2.1), CSS (4.2.2) a JavaScript (4.2.3). Navíc je zde využita vizualizační knihovna pro vykreslení grafu genealogického modelu. Vzhled a responzivita aplikace jsou stanoveny kombinací CSS a využitím frameworku Bootstrap. Technologie jsou zvoleny proto, aby aplikace co nejlépe zapadla do systému DEMoS, kde se tyto technologie také používají. Interaktivita a operace prováděné na straně klienta jsou zajištěny skripty v jazyce JavaScript, který je doplněn o knihovnu jQuery. Skripty na straně klienta se také starají o přesměrování na jiný dokument a někdy jsou do dokumentu vkládány aplikační vrstvou.

5.2.1 Komunikace s databází

Komunikace s databází je prováděna pomocí protokolu Bolt. Pro jazyk PHP neexistuje oficiální knihovna, pomocí které by komunikace byla možná. Bylo tedy nutné vyhledat vhod-

nou komunitní knihovnu, která je doporučena v návodu pro vývojáře.[1] Zvolená knihovna usnadňuje komunikaci s databází a je pojmenována jednoduše *Bolt*.[2]

5.3 Návrh uživatelského rozhraní

Návrh byl inspirován rozhraním existujících řešení popsaných v sekci 3.3. Styl návrhu je také ovlivněn systémem DEMoS a je tedy navržen tak, aby barevně zapadl do tohoto systému. Responzivita aplikace je zajištěna využitím sady nástrojů Bootstrap, která je popsána v sekci 4.2.5. Pro tvorbu je mimo toho využita knihovna jQuery (viz. 4.2.4).

5.3.1 Požadavky na uživatelské rozhraní

Hlavními cíli uživatelského rozhraní aplikace, na které bylo třeba se zaměřit jsou:

1. Přihlášení a odhlášení uživatele a s tím spojená identifikace.
2. Vytváření a správa vícero genealogických modelů jedním uživatelem.
3. Možnost nastavení, jestli genealogický model bude sdílen s ostatními uživateli a jak se bude jmenovat.
4. Zobrazování modelů cizími uživateli.
5. Vykreslení zvoleného genealogického modelu.
6. Výběr vztahů, které se uživateli v grafu zobrazí.
7. Správa jednotlivých osob v rámci modelu.
8. Možnost provázání osoby se záznamem v databázi matričních záznamů.
9. Export a import genealogických modelů.

K vykreslení grafu bylo třeba zvolit vhodnou knihovnu, která umožní přiblížení a oddálení modelu a pohyb v něm, výběr osoby nebo manželství v modelu a také zobrazení nejnutnějších informací o osobě jako je jméno, příjmení, příjmení za svobodna a datum narození a úmrtí. K tomuto účelu byla zvolena knihovna ***Vis.js***, která všechno toto umožňuje a je doporučenou knihovnou pro vizualizaci grafu. Další doporučené vizualizační nástroje lze nalézt zde [3].

5.3.2 Popis uživatelského rozhraní

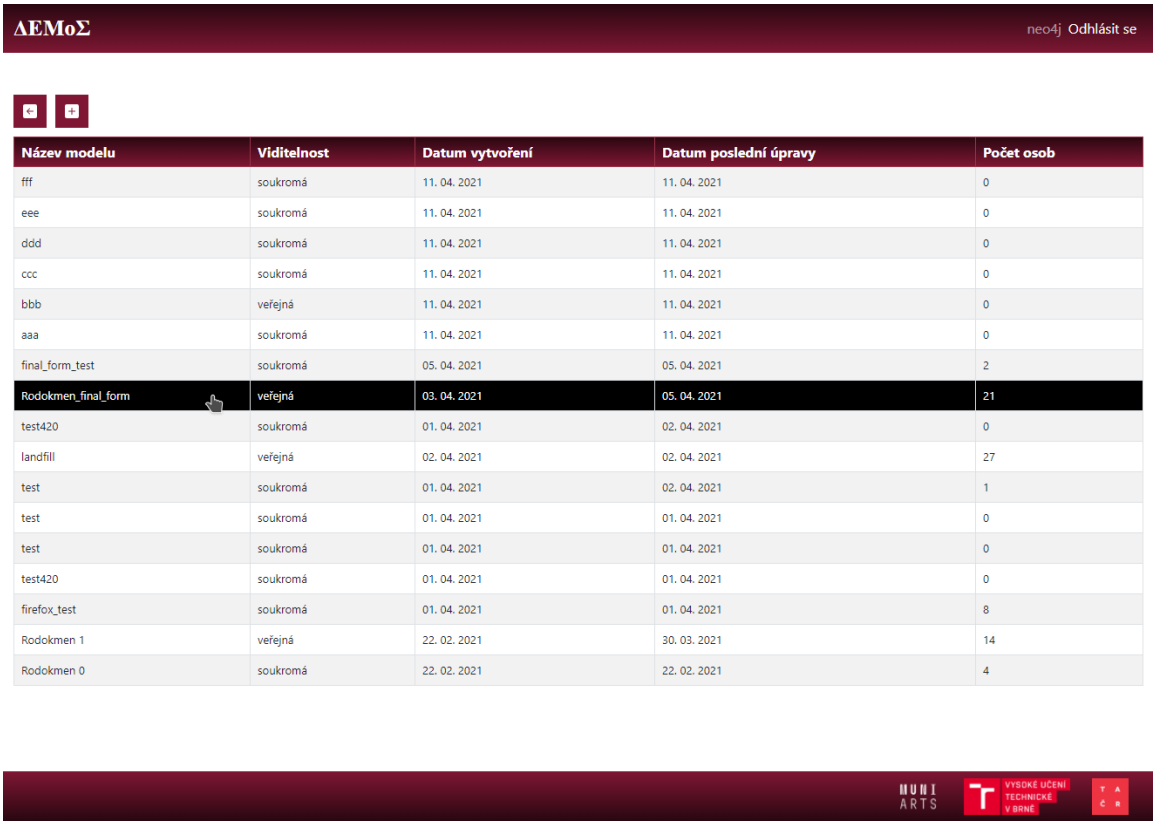
Při přístupu na stránku aplikace pro správu genealogických modelů se uživateli nejprve zobrazí přihlašovací obrazovka, kde uživatel vyplní své uživatelské jméno a heslo.

Hlavní nabídka webové aplikace obsahuje pouze možnosti, ze kterých si má uživatel možnost vybrat. Jedná se o možnost správy osobních modelů (vytvořených uživatelem) a možnost zobrazení modelů, které nejsou uživatele, a jejich správce nastavil, že je možné, aby byly viditelné cizím uživatelem (mají nastaven veřejný přístup). Tato nabídka je zpracována

[1] Návod je dostupný na: https://neo4j.com/developer/php/
[2] Dostupná z: https://github.com/neo4j-php/Bolt

podobným stylem jako hlavní nabídka, která se zobrazí po přístupu na hlavní stránku systému DEMoS.[3]

Pokud bude uživatel chtít spravovat osobní strom, zobrazí se mu seznam stromů, které spravuje ve formě tabulky. Nad tabulkou jsou 2 tlačítka se symbolem šipky vlevo a se symbolem plusového znaménka. Tato tlačítka slouží k návratu do hlavní nabídky a k vytvoření nového modelu. V již zmíněné tabulce pod tlačítky jsou uvedeny názvy modelů, jejich viditelnost pro ostatní uživatele, datum vytvoření, datum poslední úpravy a počet osob, ze kterých je model složen. Po kliknutí na model v tabulce je uživatel přesměrován na hlavní část aplikace, jíž je editor modelů sloužící ke správě a zobrazování vlastních modelů. Nabídka správy osobních modelů je viditelná na obrázku 5.4



Obrázek 5.4: Nabídka osobních modelů s možností vytvoření modelu

V případě, že uživatel v nabídce osobních modelů klikne na tlačítko se symbolem +, tak je odkázán na obrazovku pro vytvoření modelu. Obrazovka obsahuje kolonku, do které uživatel doplní jméno, které modelu vymyslí, a také obsahuje škrtátko, které zaškrtne, pokud bude mít zájem sdílet model s ostatními uživateli. Možnost sdílení se v uživatelském rozhraní už nedá změnit, takže si uživatel musí rozmyslet jestli chce model sdílet už při jeho vytváření. Po vyplnění má uživatel možnost vytvořit model s těmito parametry tlačítkem s popiskem „Vytvořit model“ nebo se vrátit zpět na nabídku osobních modelů tlačítkem s popiskem „Zpět“. K parametrům pro vytvoření stromu by později bylo možné doplnit například heslo k přístupu cizích uživatelů k modelu.

[3]Dostupné na: http://perun.fit.vutbr.cz/

5.3.3 Editor a prohlížeč modelu

Editor a prohlížeč modelů jsou koncipovány totožně. Rozdíl mezi nimi je pouze v tom, že editor je určen správci modelu a umožňuje úpravu modelu. Prohlížeč je poté upravenou verzí editoru postrádající veškeré možnosti úpravy a vše co lze v editoru upravit, to prohlížeč pouze zobrazí. Editor je složen z navigační lišty nahoře, která obsahuje vícero rozbalovacích menu s možnostmi, nabídky, která slouží ke zobrazovaní a upravování dat a vztahů uzlů modelu, a hlavní plochy, která zobrazuje graf aktuálního modelu.

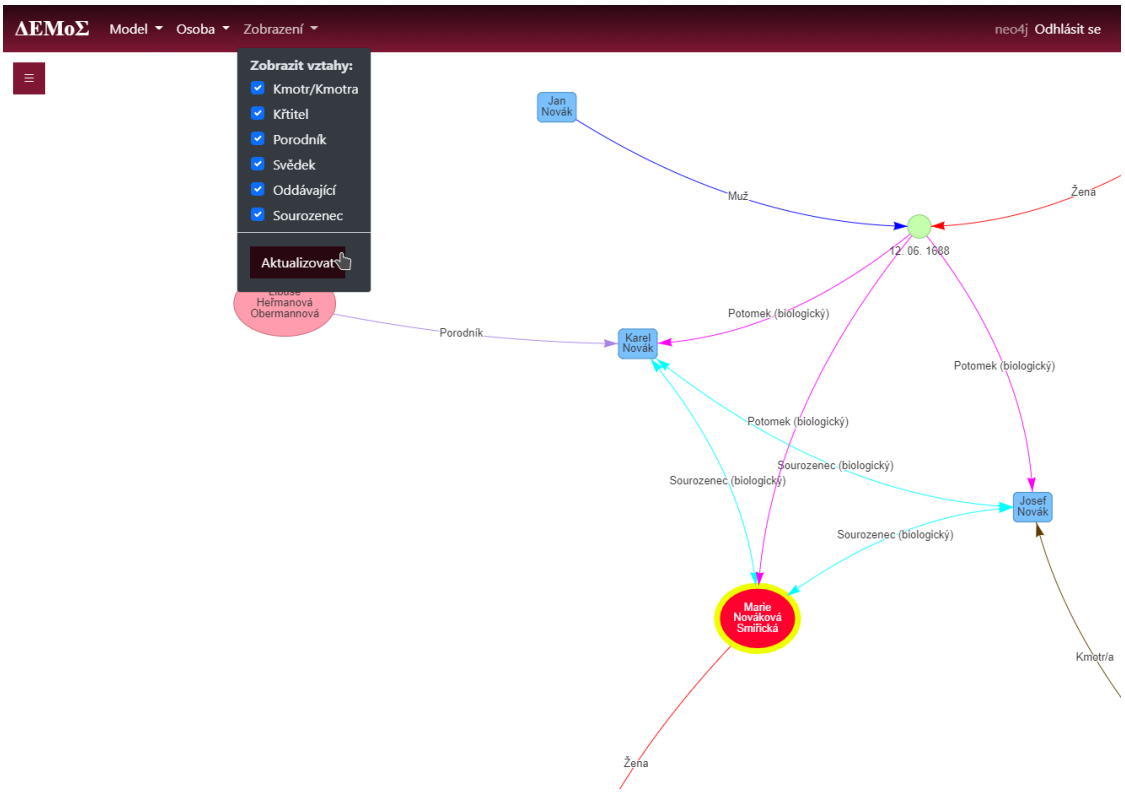
Horní lišta umožňuje odhlášení z aplikace, výběr vztahů, které se zobrazí v grafu, vytvoření nové ženy nebo muže, export nebo import modelu a smazání modelu.

Nabídka, zobrazující se na levé straně okna prohlížeče slouží k úpravě a zobrazení informací o uzlu a jeho vztazích. V případě, že není vybrán žádný uzel v grafu nebo žádný neexistuje, tak nabídka obsahuje pouze tlačítko se šipkou ukazující vlevo, název spravovaného modelu a tlačítko s třemi čárkami. Tlačítko se šipkou slouží k návratu na výběr modelů a tlačítko se třemi čárkami slouží ke skrytí nabídky a zobrazení grafu na šířce celého okna prohlížeče.

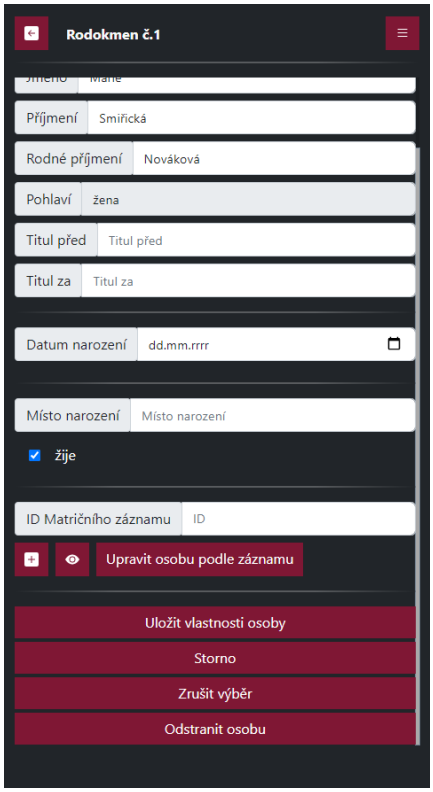
Hlavní plocha editoru slouží k vykreslení grafu modelu. Graf je vykreslován pomocí knihovny *Vis.js* a zobrazuje všechny uzly uložené v rámci modelu a vztahy typu `HUSBAND`, `WIFE` a `DESCENDANT` doplněné o další vztahy databáze, které jsou již však určeny výběrem uživatele v rozbalovacím menu horní nabídky. Pokud je skryta nabídka pro editaci uzlů, tak se v levém horním rohu plochy s grafem zobrazí tlačítko s třemi čárkami, které tuto nabídku zobrazí.

Na obrázcích 5.5 a 5.6 je vidět graf a nabídka pro editaci osoby editoru modelů, editovaná osoba je označena v grafu žlutým orámováním. Dále je na obrázku vidět rozbalovací menu „Zobrazení“, kde si může uživatel nastavit, které vztahy se zobrazí, přičemž ve výchozím stavu jsou zobrazeny všechny vztahy. Pokud na osobu v grafu uživatel najede kurzorem, zobrazí se mu datum narození a úmrtí nebo oznámení, že u osoby není uvedeno ani jedno z toho. Uzly mužů jsou v grafu vyznačeny modrou barvou a mají tvar obdélníku se zakulacenými rohy a uzly žen jsou vyznačeny červeně a mají tvar elipsy. Uzly osob s neurčitým pohlavím mají šedou barvu a tvar obdélníku se zakulacenými rohy. Manželské svazky nebo svazky, ze kterých vznikly potomci mají barvu zelenou. Jednotlivé vztahy v grafu jsou barevně odlišeny a mají popisek a orientaci. Každý uzel v grafu lze vybrat kliknutím, čímž lze přistoupit k jeho editaci. Editace samotných osob se provádí v nabídce (viz. obr. 5.6), která se zobrazuje na levé straně obrazovky a lze ji skrýt nebo zobrazit tlačítkem se třemi horizontálními čarami.

V nabídce na obrázku 5.6 se mohou zobrazit 3 různé editační formuláře. Jsou jimi: *formulář pro editaci dat osoby*, *formulář pro editaci vztahů osoby* a *formulář pro editaci svazku*. Tyto formuláře se zobrazí na základě typu vybraného uzlu. Mezi formulářem dat a formulářem vztahů osob lze přepínat záložkami v horní části formulářů. Trojice tlačítek nacházejících se pod vstupem „ID Matričního záznamu“ slouží k přidání odkazu na matriční záznam, k přístupu na záznam a k úpravě dat osoby v grafové databázi na základě zadaného záznamu.



Obrázek 5.5: Ukázka grafu v editoru modelů



Obrázek 5.6: Ukázka nabídky pro úpravu osoby v editoru modelů

Kapitola 6

Implementace a další vývoj

V této kapitole jsou vysvětleny body implementace výsledné webové aplikace a také užité implementační postupy. Implementace je provedena na základě třívrstvé architektury (4.1.2) a implementace jednotlivých vrstev je popsána v takovém pořadí, v jakém byly realizovány.

6.1 Datová vrstva

Datová vrstva je v tomto případě databáze Neo4j, k jejíž obsluze je využíváno jazyka Cypher Query Language.[1] Příkazy v tomto jazyce jsou předávány databázi v podobě řetězců aplikační vrstvou napsanou v jazyce PHP, která ke komunikaci využívá komunitní knihovnu zmíněnou v sekci 5.2.1. Navázání spojení z databází je implementováno v souboru `bolt_init.php` a bude dále zmíněno v popisu implementace aplikační vrstvy.

6.1.1 Dotazy pro komunikaci s databází

Pro komunikaci s datovou vrstvou bylo také nutné vymyslet vhodné dotazy. Některé jsou dynamicky sestavovány na základě uživatelem poskytnutých dat a při jejich návrhu bylo třeba přemýšlet nad tím, jestli je nelze napsat ještě lépe, aby došlo k minimalizaci počtu přístupů k databázi a k maximalizaci počtu provedených operací nebo získaných informací jedním dotazem. Několik příkladů těchto dotazů je zde:

1. Dotaz sloužící k získání informací o osobě a všech osobách, se kterými je v nějakém vztahu. V tomto případě je hledaná žena v dotazu označena proměnnou `p`. A je hledána v modelu označeném proměnnou `m`. Dotaz vrací struktury uzlů osoby, jejích sourozenců, kmotrů, porodníků a křtitelů. Dále vrací struktury uzlů svazků, ve kterých osoba figuruje jako manžel nebo manželka, biologický potomek, nebiologický potomek, oddávající a nebo svědek. Tento dotaz je uveden zde:

```
MATCH
    (m:Model {uuid: '9f2e9dc9-68a7-4941-8b96-2b61db5b853e'})<-[:MANAGES]-
    (:User {uuid: '7661a627-3e33-404a-819e-d66825f6274d'})
OPTIONAL MATCH
    (m)-[:HAVE]->(p:Female {uuid:'7d39b9a7-34a8-4738-946f-091c309a5743'})
OPTIONAL MATCH
    (p)-[sib_rel:SIBLING]-(sibling)
OPTIONAL MATCH
    (p)<-[:GODFATHER_GODMOTHER]-(godfather)
```

[1] Více informací na: https://neo4j.com/developer/cypher/

```
OPTIONAL MATCH
    (p)<-[:MIDWIFE_DOCTOR]-(midwife_doctor)
OPTIONAL MATCH
    (p)<-[:BAPTIZED]-(baptist)
OPTIONAL MATCH
    (p)-[marr_rel]->(as_husb_wife)
WHERE marr_rel:WIFE
OPTIONAL MATCH
    (p)<-[:DESCENDANT {biological: true}]-(as_biodescendant:Marriage)
OPTIONAL MATCH
    (p)<-[:DESCENDANT {biological: false}]-(as_nonbiodescendant:Marriage)
OPTIONAL MATCH
    (p)-[:OFFICIANT]->(as_officiant:Marriage)
OPTIONAL MATCH
    (p)-[:ATTENDANT]->(as_attendant:Marriage)
WITH
    p,
    {rel: sib_rel.biological, s: sibling} as sibling,
    godfather,
    midwife_doctor,
    baptist,
    as_husb_wife,
    as_biodescendant,
    as_nonbiodescendant,
    as_officiant,
    as_attendant
RETURN
    collect(DISTINCT p),
    collect(DISTINCT sibling),
    collect(DISTINCT godfather),
    collect(DISTINCT midwife_doctor),
    collect(DISTINCT baptist),
    collect(DISTINCT as_husb_wife),
    collect(DISTINCT as_biodescendant),
    collect(DISTINCT as_nonbiodescendant),
    collect(DISTINCT as_officiant),
    collect(DISTINCT as_attendant)
```

2. Dotaz pro vytvoření osoby muže se základní vlastností (uuid) nejprve vyhledá model podle jeho identifikátoru a poté vytvoří muže a naváže jej na vyhledaný model. Nakonec je v uzlu modelu upraven čas poslední editace modelu a dotaz vrátí identifikátor nově vytvořené osoby pro další zpracování. Dotaz je vypsán níže:

```
MATCH(model:Model {uuid:'9f2e9dc9-68a7-4941-8b96-2b61db5b853e'})
CREATE
    (p:Male
        {
            uuid: apoc.create.uuid()
        }
    ),
    (model)-[:Have]->(p)
SET model.last_edit = datetime({epochMillis: timestamp()})
RETURN p.uuid;
```

3. Tento příklad ukazuje dotaz, který slouží k vyhledání všech uzlů osob, všech uzlů svazků a všech vztahů nacházejících se v konkrétním modelu. Nejprve dojde k vyhledání modelu podle jeho identifikátoru a poté jsou vyhledány všechny napojené osoby,

svazky napojené na tyto osoby a vztahy (hrany), které všechny uzly propojují. Dotaz může vypadat takto:

```
MATCH (m:Model {uuid: '9f2e9dc9-68a7-4941-8b96-2b61db5b853e'})
OPTIONAL MATCH (m)-[:HAVE]-(people)
OPTIONAL MATCH (people)-[]-(marr:Marriage)
OPTIONAL MATCH (people)-[rels]-()
WHERE NOT rels:HAVE
RETURN
    collect(DISTINCT m),
    people,
    marr,
    rels
```

6.2 Aplikační vrstva

Tato sekce je zaměřena na detailnější popis implementace jednotlivých částí aplikační vrstvy aplikace. Je zde popsána i část prezentační vrstvy zabývající se vykreslením grafu na základě dat, které obdrží od aplikační vrstvy.

Aplikační vrstva implementována pomocí jazyka PHP. Eviduje uživatelské jméno a UUID uživatele sezení, komunikuje s datovou vrstvou pomocí Bolt protokolu, stará se o načtení a zpracování dat pro vykreslení grafu atd. Kód aplikační vrstvy prokládá HTML kód na každé stránce. Dále budou popsány jednotlivé logické části, na které je celá aplikace rozdělena.

6.2.1 Připojení k databázi

Připojení k databázi je provedeno pomocí skriptu `bolt_init.php`, který používá knihovnu[2] pro komunikaci s databází pomocí Bolt protokolu. Po spuštění skriptu dojde k vytvoření Bolt instance a k nastavení verze protokolu na verzi 4.2. Následně dojde k inicializaci příznaku chyby připojení boolovskou hodnotou *false*. Poté se skript pokusí připojit k databázi pomocí příkazu `bolt->init($name, $user, $password)`, pokud se to nepovede, je příznak chyby změněn na *true*.

6.2.2 Přihlášení uživatele

Po přístupu uživatele na dokument `index.php` aplikace je uživatel přesměrován na dokument `login.php`. Pokud je přihlášen uživatel (existuje-li proměnná `$_SESSION['user']`), dojde k přesměrování na dokument `main.php` (hlavní nabídka aplikace). Jinak je uživateli zobrazena přihlašovací obrazovka, kde do formuláře vyplní svoje uživatelské jméno a heslo. Formulář odešle tlačítkem *Přihlásit se*.

Pokud uživatel nevyplnil uživatelské jméno nebo heslo, je vypsáno upozornění: „Nebylo zadáno uživatelské jméno nebo heslo.“ V případě, že se nelze připojit k databázi, zobrazí se upozornění: „Nelze se připojit k databázi.“ Když se provedou tyto kontroly, dojde k sestavení dotazu na databázi ve tvaru řetězce. Dotaz vrací heslo a UUID uživatele na základě vyplněného uživatelského jména. Pokud databáze nevrátí žádná data, vypíše se: „Uživatel neexistuje.“ V opačném případě dojde k porovnání získaného a uživatelem zadaného hesla funkcí `password_verify()`. Jestliže funkce vrátí boolovskou hodnotu `true`, dojde k uložení uživatelského jména a UUID uživatele do proměnné sezení (`$_SESSION['user']`

[2]Knihovna dostupná na: https://github.com/neo4j-php/Bolt

a `$_SESSION['user_id']`) a stránka je aktualizována. Za předpokladu, že funkce vrátí `false`, vypíše se uživateli: „Chybné heslo.“

6.2.3 Osobní modely

Pokud si uživatel na hlavní obrazovce aplikace, tedy v dokumentu `main.php`, zvolí možnost „Osobní modely“, je přesměrován na dokument `personal_models.php`. Zde má možnost vybrat si model, který bude upravovat nebo prohlížet, vytvořit nový model nebo se vrátit zpět do hlavní nabídky.

Modely jsou do tabulky načítány pomocí dotazu, do kterého je doplněn identifikátor přihlášeného uživatele. Dotaz vrací objekty modelu a počty osob, které obsahují. Složení dotazu v aplikační vrstvě vypadá následovně:

```
MATCH
    (:User {uuid:'". $_SESSION['user_id'] ."'})-[:Manages]-
    (model:Model)
OPTIONAL MATCH
    (model)-[r:Have]->()
WITH model, count(r) as people_count
RETURN model, people_count
ORDER BY datetime(model.last_edit) DESC
```

Pokud si uživatel zvolí model, je přesměrován na dokument `tree_editor.php` s parametrem `$_GET['uuid']`, kde hodnotou je identifikátor modelu. Editoru je potom věnována sekce 6.2.5. Za předpokladu, že si uživatel bude přát vytvořit nový model, klikne na tlačítko se symbolem plusového znaménka. Následně je přesměrován na dokument `new_model.php`. Zde se objeví formulář, do kterého uživatel vyplní parametry modelu (jméno modelu a jeho viditelnost). Uživatel může stisknout buď tlačítko s popiskem „Vytvořit model“ nebo tlačítko s popiskem „Zpět“. Při stisknutí druhého se uživatel vrátí zpět k výběru osobních modelů na dokument `personal_models.php` bez jakýchkoliv změn. Po stisknutí prvního dojde ke kontrole, zda uživatel vyplnil název modelu (není-li to prázdný řetězec). Za předpokladu, že uživatel jméno modelu nevyplnil, je vypsáno upozornění: „Název modelu je povinný parametr.“ Jinak se s pomocí identifikátoru osoby přihlášené v sezení a vyplněných parametrů sestaví příkaz pro databázi a za podmínky, že se k ní lze připojit, dojde k provedení příkazu v databázi. Pomocí JavaScriptu je do úložiště sezení uložen příznak pro aktualizaci stránky a dojde k vrácení se na dokument `personal_models.php` pomocí JavaScriptového příkazu `window.history.go()`. Na základě příznaku pro aktualizaci stránky je potom tato stránka aktualizována a příznak je smazán. To bylo třeba provést, protože některé prohlížeče načítají historii z paměti, takže nedocházelo k aktualizaci tabulky modelů. Pokud se při vytváření modelu nelze připojit k databázi, vypíše se upozornění.

6.2.4 Modely ostatních uživatel

V případě, že si uživatel na hlavní obrazovce zvolil možnost „Modely ostatních uživatelů“, je mu předložena tabulka modelů, u kterých platí, že správci těchto modelů nastavili, že jsou veřejné. Po kliknutí na jeden z modelů v tabulce se uživateli tento model zobrazí k prohlížení.

Modely se do tabulky načítají pomocí řetězce dotazu, do kterého je doplněn identifikátor uživatele v sezení. Dotaz vrací objekty uzlů modelů, které spravuje jiný uživatel a které mají viditelnost nastavenou na veřejnou. Složení řetězce dotazu tedy vypadá takto:

```
MATCH (model:Model)<-[:MANAGES]-(usr:User)
WHERE
    usr.uuid <> '". $_SESSION['user_id'] ."' AND
    model.visibility = 1
OPTIONAL MATCH (model)-[r:HAVE]->()
WITH
    usr.username as username,
    model,
    count(r) as people_count
RETURN model, people_count, username
ORDER BY datetime(model.last_edit) DESC
```

Po zvolení jednoho z modelů je uživatel přesměrován na dokument prohlížeče, tedy na dokument `tree_viewer.php` s parametrem `$_GET['uuid']` určujícím identifikátor prohlíženého modelu. Prohlížeč je verze editoru s úpravami, které znemožňují editaci modelu a osob.

6.2.5 Editor modelů

Editor modelů je hlavní částí aplikace a je z něj odvozen i prohlížeč modelů. Model je načten na základě GET parametru `uuid`. Tento parametr specifikuje identifikátor modelu v rámci databáze a pro načtení modelu je vložen do řetězce dotazu, který byl ukázán v sekci 6.1. Dotaz vrací uzel modelu, uzly všech osob v modelu, uzly všech svazků v modelu a všechny vztahy, které se nachází v modelu. Pokud dotaz nevrátí nic, je spuštěn dotaz, který načte pouze jméno modelu. Pokud ani tento dotaz nevrátí nic, dojde k vypsání hlášky, že model neexistuje. V případě, že v modelu nějaké uzly jsou, dojde ke zpracování navrácených uzlů pomocí funkce `prepare_for_graph()`, která vrací zpracovaná data jako pole, kde na indexu 0 je pole všech uzlů v grafu a na indexu 1 je pole všech hran v grafu, které se mají vykreslit. Funkce přijímá tři parametry - uzly osob, uzly svazků a všechny vztahy. Data jsou poté předána do JavaScriptu ve formátu JSON jako jediný parametr funkce `visjs_draw()`. V této funkci nejprve dojde k propojení přijatých uzlů přijatými hranami ve funkci `data_postprocessing()`. Když jsou data připravena, dojde k vytvoření datových sad vizualizační knihovny a k nastavení vykreslovaní grafu. Následně je graf inicializován a zobrazen do HTML prvku s identifikátorem `#model_viewer`. Nakonec dojde ve funkci k nastavení reakce na událost kliknutí na uzel, která má za následek přidání, odstranění nebo změnu GET parametru `node`, který specifikuje načítaný uzel jako řetězec ve formátu `'uuid;label'`, a k přenačtení stránky. Pokud uživatel klikne na již vybraný uzel, tak je zrušen jeho výběr, jinak dojde k aktualizaci. Funkce, které se starají o vykreslení grafu na základě dat jsou definovány v souboru `tree_editor_frontend.js`.

Načítání uzlů

Pokud je editor načten s parametrem `$_GET['node']`, tak nejprve dojde ke kontrole, jestli se jedná o uzel osoby nebo uzel svazku. Na základě toho je poté sestaven dotaz na databázi coby řetězec. Po provedení dotazu jsou záznamy zpracovány a nastaví se příznaky, které značí, jestli byla načtena osoba nebo svazek a nebo nic. Získanými daty jsou poté vyplněny příslušné editační formuláře.

Vytváření osob a svazků

V editoru lze osobu grafu vytvořit vybráním položky „Nový muž“ nebo „Nová žena“ na horní liště v nabídce „Osoba“, nebo ji lze vytvořit z formuláře editace vztahů jiné osoby nebo svazku. V prvním případě je osoba vytvořena bez vztahů, v druhém je navázána na vybranou osobu nebo svazek. Svazek lze oproti tomu vytvořit pouze z formuláře editace vztahů osoby, protože každý svazek v databázi musí mít alespoň jeden vztah s nějakou osobou. Existující svazek s existující osobou lze propojit pouze přes formulář editace vztahů svazku.

Rozložení částí editoru

Horní lišta editoru je definována v souboru `editor_header.php`, formuláře pro editaci dat a vztahů osob poté v souboru `person_node_sidebar.php` a formulář pro editaci vztahů svazků je definován v souboru `marriage_node_sidebar.php`. Do hlavního dokumentu tedy do dokumentu `tree_editor.php` jsou vkládány části umístěné v souborech:

- `editor_header.php`
- `modals.php`
- `editor_person_node_sidebar.php`
- `editor_marriage_node_sidebar.php`

Import modelu

Implementaci importu modelů lze nalézt v souboru `import_model.php`. Importní vyskakovací prvek lze vyvolat při výběru možnosti „Import“ v nabídce, která se otevře po stisknutí položky „Model“ na horní liště editoru modelu. Ve vyskakovacím prvku je poté možno nahrát soubor s koncovkou příslušnou formátu. Soubor musí být ve formátu JSON a obsahovat data ve formátu, který bude vysvětlen v části sekce 6.2.5 s nadpisem *Export modelu*.

Import probíhá tak, že po nahrání souboru s daty dojde k přečtení řetězce ze souboru. Řetězec je poté převeden na pole pomocí funkce `json_decode()`. Na základě dat v poli je sestaven importní příkaz, který je následně poslán databázi. Import byl otestován na modelech obsahujících asi 60 uzlů a nějaké vztahy. Při importu nedochází k přemazání dat v původním modelu. Importováním je tedy možné spojit více menších modelů do jednoho.

Export modelu

Implementace této funkce se nachází v souboru `export_model.php`. Export lze vyvolat v nabídce na hlavní liště v nabídce „Model“. Při exportu dochází k přesměrování na dokument skriptu s GET parametrem „uuid“. Zde dojde k sestavení dotazu s pomocí parametru. Poté je dotaz předán databázi a vrácená data jsou zpracována do tří polí. Jedno pole obsahuje údaje o všech uzlech modelu. Druhé nese informace o všech svazcích v modelu a třetí nese informace o všech vztazích v modelu. Tato tři pole jsou poté vložena do dalšího pole s klíči *people*, *marriages* a *relations* a toto pole je převedeno do formátu JSON pomocí funkce `json_encode()` a s pomocí JavaScriptu je vloženo do souboru s kódováním UTF-8, který se nabídne uživateli ke stažení. Export je umožněn i v prohlížeči u cizích veřejně dostupných modelů. Takže je možné část modelu převzít od jiného uživatele.

6.3 Prezentační vrstva

Prezentační vrstva webové aplikace byla implementována s použitím frameworků Bootstrap a jQuery a také zde byla využita vizualizační knihovna Vis.js. V sekci 5.3.2 je možné vidět 2 pohledy na prezentační vrstvu implementované aplikace. Popis vykreslení grafu na základě dat již byl popsán v rámci sekce 6.2.5.

Obecné funkce starající se o zajištění interaktivity rozhraní aplikace se nacházejí v souboru `tree_editor_frontend.js`. Mezi tyto funkce patří například funkce ke skrytí části formuláře, pokud je editovaná osoba naživu.

Vyskakovací prvky

Pro výběr existujících osob do vztahů se používají vyskakovací prvky, jejichž viditelnost je ovládána pomocí funkcí prezentační vrstvy. Existují 2 typy těchto prvků, jedny obsahují tabulku volitelných osob se škrtátky a jedny obsahují pouze tabulku volitelných osob. V prvcích se škrtátky lze zvolit více osob a potvrdit výběr tlačítkem „Zvolit“. V případě prvků bez škrtátek lze vybrat pouze jednu osobu tím, že na ni uživatel klikne. V obou případech dojde k aktualizaci položky formuláře vztahů osoby nebo svazku a vyskakovací prvek následně zmizí. Vyskakovací prvky jsou definovány v souboru `modals.php` a funkce k jejich ovládání se nachází v souboru `modals_frontend.js`. Načítání osob do těchto prvků probíhá tak, že se prochází všechny načtené osoby z grafu a jsou vylučovány osoby, které již s uzlem mají nějaký konfliktní vztah. Osoba například nemůže mít vztah sama se sebou a žena svazku nemůže být zároveň potomkem svazku.

6.4 Testování na datech z databáze DEMoS

Data byla do databáze nahrána na základě matričních záznamů pomocí skriptu napsaném v jazyce Python. Obdržená testovací data v podobě tří typů matričních záznamů byla převedena do formátu CSV a následně se v datech vyhledaly souvislosti mezi osobami v záznamech. Na základě nalezených osob a souvislostí došlo k vytvoření příkazů pro vytvoření příslušných uzlů a hran v databázi.

V průběhu testování došlo k drobným změnám v databázi a v uživatelském rozhraní aplikace. Mezi tyto změny patří přidání uzlu typu `Unknown`, který může mít stejné vlastnosti jako uzly typu `Male` a `Female`. Typ uzlu bylo nutno přidat, protože u některých osob v záznamech nelze určit pohlaví. Jedna z úprav aplikace souvisí s tímto novým typem uzlu. Pokud je v uživatelském rozhraní vytvořena osoba s nejistým pohlaví, lze jej změnit, toto však nelze provést u osob se specifikovaným pohlavím. Další úprava aplikace se týkala ukládání kmotrů a svědků. V záznamech totiž není specifikováno, jestli se jedná o svědka ženy nebo muže, a svatba může mít více svědků než 2, což byl počet, který byl možný přidat v původní verzi aplikace. V nové verzi je také možno přidat více než 2 kmotry jedné osobě.

Závěr testování je takový, že data získaná z digitalizovaných matričních záznamů lze nahrát do databáze, ale bylo nutné udělat pár menších změn. Databáze je schopna uchovat všechny požadované informace a díky implementovaným uzlům svazků v databázi a modifikovatelnosti databáze Neo4j je zde stále prostor pro přidání dalších typů vztahů a uzlů. Data však bylo třeba prohlížet pouze v databázi, protože zobrazení celkem asi 2500 osob vytvořených na základě testovacích matričních záznamů vizualizační knihovně trvalo dlouho.

6.5 Další vývoj

V rámci aplikace i v rámci databáze je stále prostor pro další vývoj. Do databáze je možno přidávat další typy uzlů i vztahů mezi nimi. Totéž se týká také webové aplikace.

Vytvořený systém bude dále vyvíjen v rámci projektu DEMoS a bude lépe integrován do celkového systému. Lepší integrací je myšlena například možnost doplňování dat do uzlů osoby na základě matričních záznamů, kdy by uživateli mohla být zobrazena nějaká nabídka záznamů v původní databázi, ve kterých budou sdruženy informace o osobách, a z nich si uživatel vybere. Další zajímavou vlastností aplikace by v budoucnu mohla být automatizace sestavování genealogických modelů, na které se v rámci projektu DEMoS již pracuje a přinesla první výsledky, nebo možnost přidání větvě rodiny na základě zvolení jednoho matričního záznamu.

Možností vývoje je tedy stále spousta a pokud budou schopnosti vyvíjené aplikace dostupné veřejnosti, věřím, že zájemcům o historii svojí rodiny usnadní práci spojenou s bádáním.

Kapitola 7

Závěr

Cílem této bakalářské práce byl návrh a realizace databáze pro ukládání genealogických modelů, kde bylo třeba počítat kromě klasických vztahů, se kterými se setkáváme při tvorbě rodokmenů také se vztahy netypickými jako jsou kmotr/kmotra, svědek, oddávající, křtitel a další. Větší množství vztahů, které se v databázi nacházejí rozšiřuje možnosti vyhledávání osob v databázi.

Databáze, které byla zvolena pro řešení, Neo4j, je databáze grafová. Při implementaci se bylo nutné seznámit s jazykem, který tato databáze používá k obsluze, s čímž souvisí i pochopení konceptu grafových databází.

Implementovaná databáze je doplněna o klientskou webovou aplikaci, která byla zakomponována do systému DEMoS a je vytvořena v jazycích PHP, JavaScript, HTML a CSS. Při implementaci byly využity frameworky jako například jQuery nebo Bootstrap. Pro implementaci vykreslování grafu byla nejprve zvolena knihovna *dTree*, ale kvůli zakomponování vztahů jako svědek, kmotr a další, a také kvůli nemožnosti propojit vícero rodin byla nakonec zvolena knihovna *Vis.js*, která toto umožňuje.

Do výsledné aplikace je možný přístup po přihlášení, které bylo zpracováno jednodušeji, jelikož nebylo hlavním bodem zadání. Jednotlivé osoby v nově vzniklé databázi je možno mapovat na digitalizované matriční záznamy. Toto propojení je provedeno pomocí identifikátoru záznamu osoby v digitalizovaných matrikách, který může být uložen v uzlu každé konkrétní osoby v nové databázi. Dále je v klientské aplikaci řešena možnost exportu a importu modelu ve formátu, který umožňuje zpracování automatizovanými nástroji.

Vytvořená aplikace i databáze se bude dále vyvíjet společně s databází digitalizovaných matričních záznamů.

Literatura

[1] *Acta publica* [online]. Moravský zemský archiv v Brně [cit. 2021-02-09]. Dostupné z: https://www.mza.cz/actapublica/matrika/hledani_obec.

[2] *Vis.js Community Edition* [online]. Vis.js Community [cit. 2021-04-11]. Dostupné z: https://visjs.org/.

[3] *Graph Visualization Tools* [online]. Neo4j, 2021 [cit. 2021-04-11]. Dostupné z: https://neo4j.com/developer/tools-graph-visualization/.

[4] *What can PHP do?* [online]. The PHP Group, 2021 [cit. 2021-03-15]. Dostupné z: https://www.php.net/manual/en/intro-whatcando.php.

[5] *What is PHP?* [online]. The PHP Group, 2021 [cit. 2021-03-15]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php.

[6] Bakni, M. *3-tier client/server model architecture* [online]. Wikimedia, srpen 2017 [cit. 2021-03-05]. Dostupné z: https://commons.wikimedia.org/wiki/File:Client-Server_3-tier_architecture_-_en.png.

[7] Borůvka, J. *Specializovaný interpret jazyka JavaScript*. Brno, 2008. Diplomová práce. Vysoké učení technické v Brně. Fakulta informačních technologií. Ústav inteligentních systémů. Dostupné z: http://hdl.handle.net/11012/53185.

[8] Bos, B. *A brief history of CSS until 2016* [online]. W3C, prosinec 2016 [cit. 2021-03-15]. Dostupné z: https://www.w3.org/Style/CSS20/history.html.

[9] Castillote, J. *Static vs Dynamic Websites: What's the Difference?* [online]. Adam the Automator, leden 2021 [cit. 2021-03-05]. Dostupné z: https://adamtheautomator.com/static-vs-dynamic-website/.

[10] Chakraborty, A. *System Design Basics: Client-Server Architecture* [online]. Towards Data Science, prosinec 2020 [cit. 2021-03-05]. Dostupné z: https://towardsdatascience.com/system-design-basics-getting-started-with-the-client-server-architecture-b02f9c9daae8.

[11] DB Engines. *DB-Engines Ranking* [online]. [cit. 2020-11-17]. Dostupné z: https://db-engines.com/en/ranking.

[12] Fitzgibbons, L. *Front end and back end* [online]. TechTarget, květen 2019 [cit. 2021-03-15]. Dostupné z: https://whatis.techtarget.com/definition/front-end.

[13] IBM. *Three-Tier Architecture* [online]. IBM, říjen 2020 [cit. 2021-03-05]. Dostupné z: https://www.ibm.com/cloud/learn/three-tier-architecture.

[14] Ivo, S. *Rodokmeny - Kompletní genealogické služby* [online]. [cit. 2021-01-09]. Dostupné z: https://www.sperat.cz/rodokmeny/.

[15] Kroupová, T. *Kniha: Cesta ke kořenům objevování tajemství tvého rodu.* Olomouc, CZ, 2018. Diplomová práce. Univerzita Palackého v Olomouci, Fakulta pedagogická. Dostupné z: https://theses.cz/id/1iqzbf/Kroupova_Tereza_DP.pdf/.

[16] Kukhnavets, P. *How to Organize Application Code With 3-Tier Architecture?* [online]. WellDoneBy, listopad 2019 [cit. 2021-03-05]. Dostupné z: https://welldoneby.com/blog/how-organize-application-code-with-3-tier-architecture/#The_introduction_to_a_3layer_architecture.

[17] Lutkevich, B. *HTML (Hypertext Markup Language)* [online]. TheServerSide, únor 2020 [cit. 2021-03-15]. Dostupné z: https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language.

[18] Mrozek, J. *JavaScript na serveru: Architektura a první Hello World* [online]. Devel.cz Lab s.r.o., říjen 2012 [cit. 2021-04-07]. Dostupné z: https://zdrojak.cz/clanky/javascript-na-serveru-architektura-a-prvni-hello-world/.

[19] Pecháček, J. *Jak vypadají matriční záznamy* [online]. [cit. 2021-02-09]. Dostupné z: https://www.odkudjsme.cz/blog/jak-vypadaji-matricni-zaznamy/.

[20] Shannon, R. *What is HTML?* [online]. HTML Source, srpen 2012 [cit. 2021-03-15]. Dostupné z: https://www.yourhtmlsource.com/starthere/whatishtml.html.

[21] Skřebská, Z. *Genealogie a tvorba rodokmenů.* Brno, CZ, 2017. Bakalářská práce. Masarykova univerzita, Přírodovědecká fakulta. Dostupné z: https://is.muni.cz/th/mrrs2/Bakalarska_prace_-_Skrebska.pdf.

[22] Temere, B. M. *Responsive Web Application Using Bootstrap and Foundation.* Helsinki, 2017. Bachelor thesis. Helsinki Metropolia University of Applied Sciences. Dostupné z: https://www.theseus.fi/bitstream/handle/10024/130524/Befekadu_Temere.pdf?sequence=1.

[23] York, R. Introduction to jQuery. In: *Web Development with jQuery®*. John Wiley & Sons, Ltd, Březen 2015, kap. 1, s. 1–26. DOI: 10.1002/9781119209430.ch1. ISBN 9781119209430. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119209430.ch1.

[24] Česká republika. Předpis 301/2000 Sb. In: *Zákon o matrikách, jménu a příjmení, částka 85.* Praha, Česká republika: Ministerstvo vnitra, 2003, s. 4107–4112. Sbírka zákonů 2003. Dostupné z: https://www.psp.cz/sqw/sbirka.sqw?cz=301&r=2000.

[25] Česká republika. *Nahlížení do matričních knih a sbírky listin* [online]. Ministerstvo vnitra, prosinec 2020 [cit. 2021-02-09]. Dostupné z: https://www.mvcr.cz/clanek/matriky-nahlizeni-do-matricnich-knih-a-sbirky-listin.aspx.

[26] Česká republika. *Uzavření registrovaného partnerství* [online]. Ministerstvo vnitra, srpen 2020 [cit. 2021-02-09]. Dostupné z: https://www.mvcr.cz/clanek/uzavreni-registrovaneho-partnerstvi.aspx.

Příloha A

Instalace a konfigurace

V této příloze bude popsán postup instalace potřebných komponentů systému. Aby mohla být aplikace spuštěna musí být na webovém serveru přítomno PHP, doporučenou verzí je PHP 7.4.15, protože bylo využito při implementaci. Prerekvizitou pro fungování databáze Neo4j je přítomnost OpenJDK nebo OracleJDK ve verzi 11 a vyšší.

Instalace databáze

Kompletní dokumentaci popisující instalaci databáze lze nalézt na stránkách: https://neo4j.com/docs/operations-manual/current/installation/. Doporučená verze Neo4j pro implementaci je Neo4j 4.2.1. K databázi je nutno instalovat také knihovnu APOC, jejíž funkce jsou využívány v implementaci. Postup instalace knihovny APOC ve správné verzi lze nalézt na stránce: https://neo4j.com/labs/apoc/4.2/installation/.

Konfigurace databáze

Při konfiguraci je dobré řídit se těmito body:

1. Před spuštěním instance databáze je třeba upravit konfiguraci v souboru `neo4j.conf`, jehož polohu lze nalézt na základě této části manuálu: https://neo4j.com/docs/operations-manual/current/configuration/file-locations/.

2. Uvnitř konfiguračního souboru je třeba změnit název výchozí databáze, to lze provést upravením položky `dbms.default_database`. Řádek v souboru tedy bude po úpravě na správnou hodnotu vypadat takto:

   ```
   dbms.default_database=GenealogyModels
   ```

3. Dále je třeba na konec konfiguračního souboru přidat řádek:

   ```
   cypher.lenient_create_relationship = true
   ```

4. Všechny možnosti konfigurace jsou popsány na této stránce: https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/.

Import a export databáze

Vytvořenou databázi lze importovat pomocí příkazu:

```
neo4j-admin load --verbose --from=<DUMP_FILE> --database=genealogymodels --force
```

Databázi lze exportovat pomocí příkazu:

```
neo4j-admin dump --verbose --database=genealogymodels --to=<DUMP_FILE_DIR>
```

Obě tyto operace je třeba provádět, když je instance databáze neaktivní. Více informací o zálohování lze nalézt na tomto odkazu: https://neo4j.com/docs/operations-manual/current/backup-restore/.

Přihlašovací údaje

V rámci databáze jsou vytvořeny 2 uživatelské účty, s jejichž přístupovými údaji se lze přihlásit do aplikace. Údaje těchto účtů jsou tedy:

- uživatelské jméno: `neo4j`, heslo: `default_user1`
- uživatelské jméno: `test_user`, heslo: `12345a`