VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

SYSTÉM PRO SPRÁVU REVIZÍ ZÁZNAMŮ DATABÁZE PŘEPSANÝCH MATRIČNÍCH DAT

SYSTEM FOR MANAGING REVISIONS OF THE TRANSCRIPTED REGISTERS DATABASE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MATĚJ GOTZMAN**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Gotzman Matěj**
Program: Informační technologie
Název: **Systém pro správu revizí záznamů databáze přepsaných matričních dat**
System for Managing Revisions of the Transcripted Registers Database
Kategorie: Databáze

Zadání:

1. Seznamte se s problematikou práce s genealogickými daty, tvorbou rodokmenů a existující databází přepisovaných matričních záznamů ČR (DEMoS). Tato výchozí databáze vzniká v rámci projektu na FIT.
2. Navrhněte úpravu systému DEMoS tak, aby umožňoval správu revizí záznamů na principu, který je obdobný wiki stránkám. Systém musí evidovat kdo a kdy provedl jakou změnu a rekonstruovat stav záznamu k určitému datu. Uživatelé mají nastavená oprávnění, která určují podmínky, kdo může editovat který záznam.
3. Navržené úpravy implementujte v existující databázi a začleňte do webového uživatelského rozhraní.
4. 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 dva 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 2021
Datum odevzdání: 11. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Cílem této práce je rozšířit existující systém pro ukládání matričních dat (DEMoS) o možnost správy revize záznamů. V rámci úprav byly v databázi vytvořeny pomocné tabulky, pohledy a procedury, sloužící pro ukládání a správu historie provedených změn. Vytvořené řešení obsahuje také webové rozhraní, zobrazující provedené úpravy včetně času a autora změny, které zároveň podporuje obnovu záznamu do některého z předchozích stavů. Pro usnadnění případného rozšiřování databáze v budoucnu byl vytvořen skript, který automaticky generuje MySQL kód implementovaných obslužných prvků databáze pro zadané originální tabulky.

Abstract

This work aims to create an extension of the existing system for managing master data (DEMoS) with the ability to manage record revisions. As part of the modifications, auxiliary tables, views and procedures were created in the database, which are used to store and manage the history of record changes. The created solution also contains a web interface showing the modifications made, including the time and the author of the change, as well as supporting the restoration of the record to one of the previous states. To facilitate the possible expansion of the database in the future, a script was created that automatically generates the MySQL code of implemented database controls for the specified original tables.

Klíčová slova

genealogie, matrika, přepis matričních záznamů, DEMoS, verzovací systém, obnovení stavu záznamu, MySQL, Nette, PHP

Keywords

genealogy, registry book, transcription of registry records, DEMoS, revision control system, record state restoration, MySQL, Nette, PHP

Citace

GOTZMAN, Matěj. *Systém pro správu revizí záznamů databáze přepsaných matričních dat.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Kočí, Ph.D.

Systém pro správu revizí záznamů databáze přepsaných matričních dat

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.

.....
Matěj Gotzman
9. května 2022

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Radkovi Kočímu, Ph.D. za pomoc a ochotu při vedení mé bakalářské práce.

Obsah

Kapitola 1

Úvod

Pátrání po historii svých předků je v dnešní době oblíbeným koníčkem spousty lidí. Získat informace o tom, odkud vlastně pochází vaše rodina, vaše příjmení či zdali nejste vzdálení příbuzní s někým ze svého okolí, je velmi zajímavé. Zájemci se proto pídí po svých příbuzných v různých historických pramenech, jako jsou například matriky, případně kroniky a jiné. S rozvojem informačních technologií se i tato činnost značně usnadnila. Jednotlivé archivy totiž zpřístupnily veřejnosti naskenované matriky. Zájemci si tak mohou vyhledat danou matriku například podle letopočtu, nebo podle obcí zmíněných v záznamech dané matriky. Jelikož se však stále jedná pouze o naskenované stránky jednotlivých matrik, musí jimi případný amatérský detektiv listovat a pracně hledat požadovanou informaci.

Pro usnadnění a urychlení tohoto vyhledávání vznikl ve spolupráci Vysokého učení technického v Brně a Masarykovy univerzity projekt DEMoS (Database of Early Modern Sources). Jedná se o genealogickou databázi, do které budou moci dobrovolníci přepisovat informace z matrik, lánových rejstříků a dalších historických pramenů. Z těchto dat se následně budou generovat rodokmeny. Uživatelé si pak budou moci jednoduše dohledat nejen své předky, ale také například jejich kmotry, svědky na svatbách a další informace.

Celý systém musí být uživatelsky přívětivý a neměl by odrazovat případné dobrovolníky svou složitostí či jinými negativními faktory. Z toho důvodu již vzniklo několik bakalářských a diplomových prací, které usilují o ulehčení ukládání informací. Jednou z nich je i tato práce. Jejím cílem je vytvořit systém, který uživatelům umožní zobrazit historii provedených úprav daného záznamu a také jej obnovit do některého z jeho předchozích stavů. Tento systém je přínosný pro projekt DEMoS díky dobrovolnictví, na kterém je projekt založen. Počítá se s tím, že méně zkušení uživatelé mohou během přepisu informací udělat chyby. Ty pak mohou další uživatelé opravovat. Kontrola nad jednotlivými úpravami může dané opravy usnadnit.

Ve druhé kapitole je popsána teorie, kterou je potřeba znát pro pochopení návrhu systému. Je v ní mimo jiné probráno, čím se zabývá věda genealogie. Dále se kapitola věnuje tématu matrik a také jejich digitalizaci. Následuje popis technologií, které byly v rámci této práce využity. Kapitola se rovněž zabývá problematikou databázových systémů a systémem správy verzí. Třetí kapitola popisuje současný stav projektu DEMoS. Detailně je zde popsáno schéma databáze, jehož pochopení je velmi důležité pro návrh řešení této práce. V poslední části třetí kapitoly je popsán vlastní cíl této práce. Nachází se zde zhodnocení současného stavu projektu a návrh potřebných úprav pro jednotlivé části systému. Čtvrtá kapitola se zabývá implementací databázových prvků systému. Konkrétně strukturou pomocných tabulek, do kterých se ukládají provedené změny. Následují databázové spouštěče, které se starají o uložení změn do pomocných tabulek. Dále popisuje implementaci dvou

druhů pohledů pro zobrazení provedených změn a získaní jednotlivých verzí záznamů. Posledním implementovaným prvkem jsou procedury, které daný záznam obnoví do zvoleného stavu. Pátá kapitola popisuje implementaci webové části systému. Konkrétně funkcionalitu ukládání provedených úprav záznamů nebo získání jejich historie. Je zde také popsána implementace zobrazení této historie, včetně principu obnovení záznamu do požadovaného stavu. V šesté kapitole je popsán postup testování vytvořeného systému.

Kapitola 2

Studium problematiky

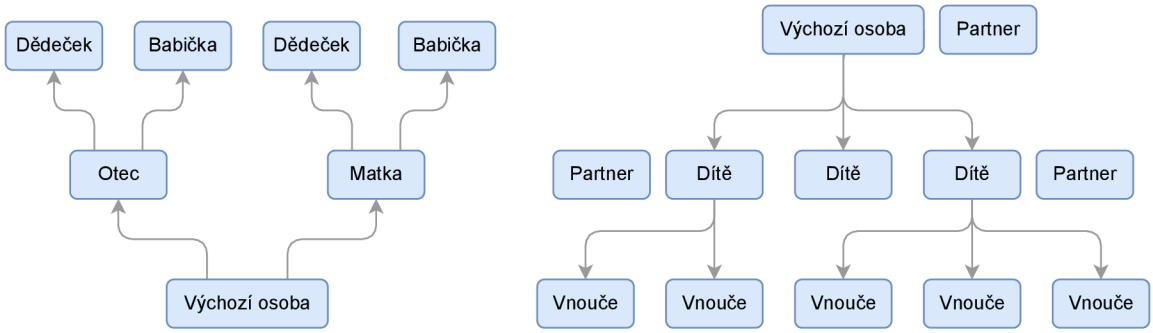
Tato kapitola se zabývá pojmy a technologiemi, které bylo potřeba nastudovat pro následnou implementaci této práce. Nachází se zde vysvětlení oboru genealogie, popis matrik a jejich digitalizací či popis projektu DEMoS. Dále jsou zde popsány technické prvky jako například návrhový vzor MVC nebo framework Nette. Vysvětlení pojmů databázový pohled, spouštěč a další se nachází v kapitole 2.5. Následuje podrobný popis struktury databáze, kterou využívá projekt DEMoS. V poslední části je popsáno zhodnocení současného stavu a také návrh práce, který byl vytvořen na základě získaných informací.

2.1 Genealogie

Název *Genealogie* [5] pochází ze slova *genealogía* v antické řečtině, které v překladu znamená *věda zabývající se studiem rodokmenů*. Jedná se tedy o historickou vědu, která zkoumá vztahy mezi lidmi převážně v rámci jejich rodového původu. Pro získání informací o rodině se využívají rozhovory s jejími členy, záznamy z historických pramenů či genetická data. Ze všech získaných informací se následně sestavuje jakýsi graf znázorňující příbuzenské vztahy mezi jednotlivými členy rodu — tedy rodokmen.

Genealogické pátraní je možné od výchozí osoby provádět dvěma směry. Zkoumání předků výchozí osoby je nazýváno *Vývod* [14]. Pro danou osobu jsou tedy nalezeny informace o jejích rodičích, prarodičích a tak dále. Výsledné schéma bývá znázorňováno ve formátu stromu, kdy je výchozí osoba jeho kořenem a předci osoby jsou poté větveni v rámci jeho koruny. Opačný směr zkoumání se nazývá *Rozrod* [11]. Zde jsou vyhledáváni potomci výchozí osoby. Tato metoda je velmi složitá, jelikož výsledný strom může disponovat tisíci nalezenými osobami. Vyhledávání informací se dělí na dvě metody. První metodou je následovat pouze příjmení výchozí osoby a tím pádem vyhledávat pouze informace o potomcích synů či nemanželských potomcích dcer. Druhou metodou je vyhledávat potomky synů i dcer bez závislosti na příjmení. Výsledné informace opět bývají vyobrazené v podobě stromu. Zde však bývá v rámci kořene kromě výchozí osoby také její partner/ka. Výše zmíněné typy rodokmenů jsou znázorněny pomocí diagramů na obrázku 2.1.

Genealogické pátrání po předcích je jednou z hlavních motivací vytvoření systému DEMoS. Ten bude schopen z přepisovaných informací historických pramenů vytvářet rodokmeny všech zaznamenaných osob.

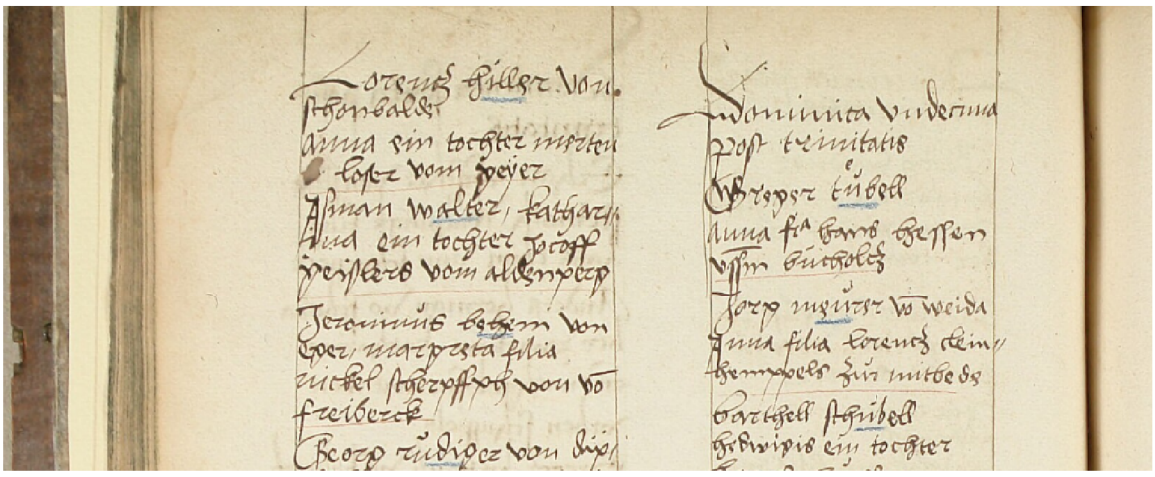


Obrázek 2.1: **Ukázky rodokmenů.** Diagram nacházející se vlevo znázorňuje příklad struktury rodokmenu typu *vývod*. Diagram vpravo znázorňuje příklad struktury rodokmenu typu *rozrod*.

2.2 Matriky

Matriky [7] jsou úřední dokumenty obsahující údaje o obyvatelích. V České republice se uchovávají záznamy o narození, sňatcích či registrovaných partnerstvích a úmrtích. Správu matrik mají na starosti matrikáři na příslušných úřadech. Matriky se v dnešní době dělí na dva druhy. Živé matriky jsou ty, do kterých jsou stále přidávány nové záznamy. Na matričních úřadech zůstávají uloženy ještě 100 až 110 let po vložení posledního rodného záznamu, případně 75 let od posledního oddacího či úmrtního záznamu. Po tuto dobu mají k informacím v matrikách přístup pouze povolané osoby. Po uplynutí výše uvedené doby jsou matriky považovány za mrtvé a přesouvají se do městských, státních či zemských archivů. Zde jsou již jejich informace veřejně přístupné.

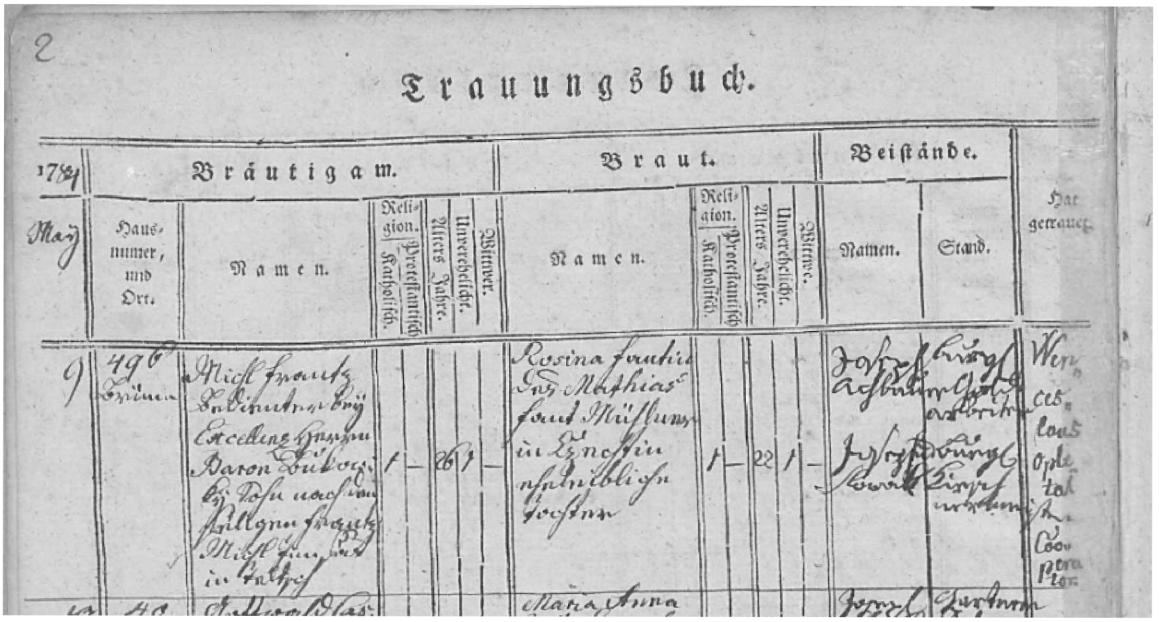
Nejstarší dochovaná matrika[1] na českém území pochází z roku 1531 a obsahuje informace o poddaných z Jáchymova. Na obrázku 2.2 si lze všimnout, že jednotlivé údaje záznamů nebyly psány strukturovaně, ale ve větách.



Obrázek 2.2: **Snímek nejstarší dochované matriky na českém území.** Matrika pochází z roku 1531 a obsahuje záznamy o poddaných z Jáchymova.

[1] Digitální verze nejstarší dochované matriky: https://www.portafontium.eu/iipimage/30063190/

V průběhu historie se měnily jazyky, ve kterých byly záznamy zapisovány. Stejně tak nebyla jednotná ani struktura záznamů. Povinnost všech církví vést matriční záznamy o svých věřících na území Koruny české nařídil Josef II. v rámci jeho reforem z let 1781 a 1784. Současně bylo také zavedeno jednotné vedení matrik. Jak je možné vidět na obrázku 2.3, záznamy z roku 1784 jsou již strukturované do předtištěné tabulky. V 50. letech 20. století pak byla zavedena obecná a povinná matrika pro všechny občany.



Obrázek 2.3: **Ukázka oddacího záznamu.** Tento záznam pochází z roku 1784 a je součásti matriky z Brna, jejíž digitalizovaná verze je dostupná on-line[3].

Od roku 2007 probíhá v České republice digitalizace matrik [4]. Pracovníci jednotlivých archivů postupně skenují všechny matriky a jejich výsledné digitální verze zveřejňují na webových stránkách. Motivací této náročné celonárodní akce není pouze usnadnění přístupu veřejnosti k informacím, ale také zajištění dlouhé životnosti originálních matrik. Případní zájemci totiž nemusí fyzicky manipulovat s originálními matrikami, když si v nich chtějí dohledat informace.

Digitalizací matrik se sice zpřístupnily jejich informace široké veřejnosti, vyhledávání v nich však může být stále velmi zdlouhavé a náročné. Nedají se v nich totiž nijak filtrovat informace či řadit jednotlivé záznamy podle zadaných atributů. Jak je možné vidět na obrázcích výše, záznamy byly zapisovány ručně, což ještě více ztěžuje bádání. V systému DEMoS však budou všechny informace zobrazeny v přehledných tabulkách, a navíc v nich bude možné vyhledávat podle různých atributů, což bádání značně zpříjemní.

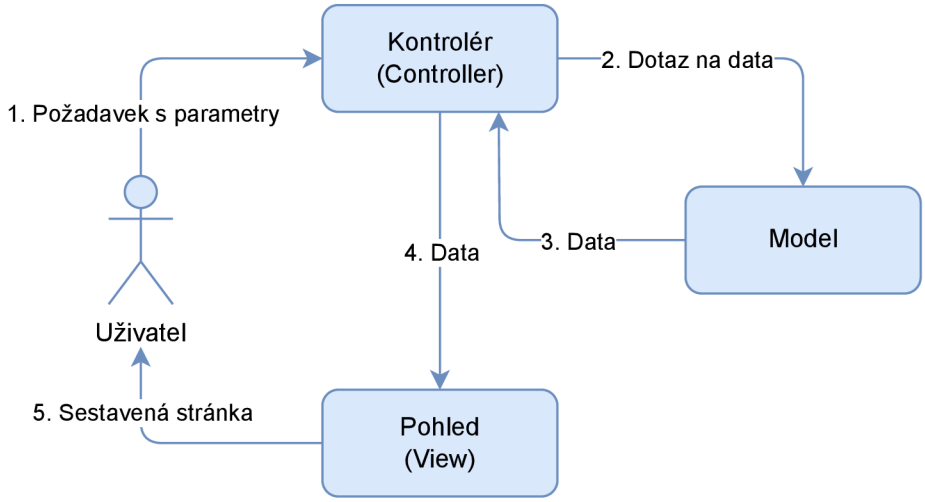
[3]Digitální verze matriky zobrazené na obrázku 2.3: https://www.mza.cz/actapublica/matrika/detail/7823?image=216000010-000253-003381-000000-016890-000000-00-B03490-00010.jp2

2.3 Serverové webové technologie

V této sekci jsou popsány technologie, které je potřeba znát pro pochopení implementace této práce. Je zde popsán návrhový vzor *MVC*, na kterém staví framework *Nette*. Ten byl využit při implementaci již existujícího systému DEMoS a během úprav v rámci této práce. V poslední části této sekce je popsán formát *JSON*, který byl využit pro ukládání informací o provedených úpravách matričních záznamů.

2.3.1 Návrhový vzor MVC

Návrhový vzor MVC [3] je velmi často využíván při vytváření nejen webových aplikací. Systém projektu DEMoS není výjimkou. Hlavní myšlenkou je rozdělit jednotlivé části systému do logických celků, které vykonávají svůj úkol a není pro ně důležité, jak pracují ostatní části. Takovéto rozdělení zvyšuje přehlednost a také ulehčuje implementaci. Architektura MVC tedy kód dělí na tři části a to Model, Pohled (View) a Kontrolér (Controller). Obrázek 2.4 zobrazuje tyto části a také komunikaci mezi nimi.



Obrázek 2.4: **Návrhový vzor MVC.** Diagram zobrazuje jednotlivé logické části MVC a také komunikaci mezi nimi při zpracování požadavku.

Model se stará o logiku systému. Mohou zde být různé výpočty, dotazy na databázi a podobně. V této práci jsou součástí modelu takzvané manažery (Manager), které obsahují metody pro ukládání dat, získání dat z databáze, kontrolu oprávnění uživatele a další. Data z databáze se mapují na objekty, které se následně odesílají k zobrazení.

Pohled (View) slouží k zobrazování výstupu uživateli. Skládá se ze šablon, které obsahují převážně HTML kód, do kterého lze také vkládat proměnné, případně provádět větvení či iterace. Jednotlivé šablony jde také vkládat do sebe. Díky tomu je možné zamezit zbytečnému opakování kódu. Pohled tedy dostane data, která podle šablon naformátuje a odešle je uživateli.

Kontrolér (Controller) je poslední a také řídící částí této architektury. Na základě přijatých parametrů zavolá požadované metody modelu a uloží si získaná data. Následně tato data předá vybranému pohledu, a ten je zformátuje do požadované podoby. V rámci projektu DEMoS se místo názvu kontrolér využívá prezentér (presenter) [13]. Funkčností se však jedná o totožné části.

2.3.2 Framework Nette

Framework Nette [8] je jedním z nejpopulárnějších nástrojů pro tvorbu webových aplikací v jazyce PHP na světě. Jeho autorem je David Grudl a vývoj trvá už více než deset let [10]. Jedná se o český projekt a díky tomu disponuje velkou českou komunitou a také dokumentací dostupnou v češtině. Staví na návrhovém vzoru MVP (Model, View, Presenter), což je upravená verze MVC (Model, View, Controller), kdy prezentér nahrazuje kontrolér [13]. Jejich funkcionalita je však prakticky totožná. Jak již bylo zmíněno v předchozí sekci, metody pro logickou část Model jsou uloženy v souborech, které mají v názvu klíčové slovo Manager.

Šablony pro pohledy se vytvářejí pomocí šablonovacího balíčku s názvem Latte [19] a jsou uloženy v souborech s koncovkou `latte`. Kromě HTML značek Latte podporuje sadu příkazů, které mají stejnou syntaxi jako jazyk PHP. Tyto příkazy se uzavírají do složených závorek. Díky tomu je možné v samotné šabloně iterovat, provádět větvení a další.

Pro ladění je k dispozici nástroj Tracy [12]. Ten vypisuje chyby přímo do prohlížeče, a kromě chybové hlášky zobrazí také část kódu, ve které se chyba nachází. Vypisuje také volání všech funkcí, které chybě předcházely a další informace. V některých případech dokáže navrhnout možné řešení problému. Další oblíbenou funkcí je formátování výpisu proměnných, takzvané „dumpování“. Díky tomu si může programátor nechat vypsat například objekty či pole, kdy mu Tracy k jednotlivým položkám doplní také datové typy a podobně.

Pro testování se využívá nástroj Tester [9]. Soubory s testy disponují koncovkou `phpt`. Tester umožňuje spouštět testy buďto samostatně, nebo také paralelně. Kromě vyhodnocení testů dokáže Tester generovat pokrytí kódu testy.

2.3.3 JSON

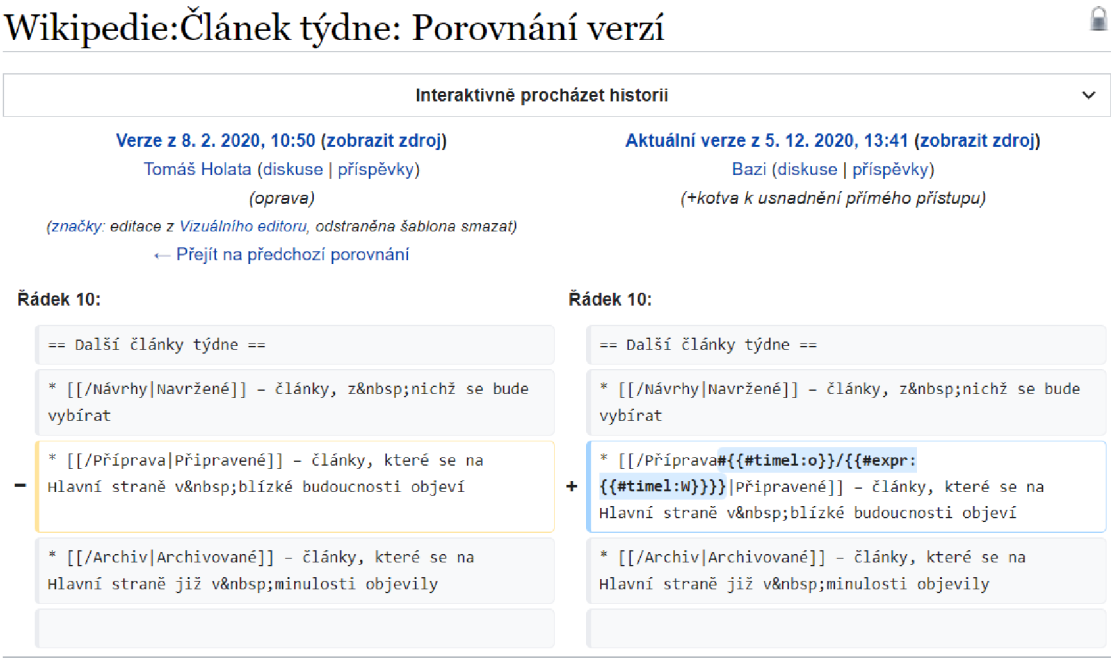
Zkratka *JSON* [6] značí *JavaScript Object Notation* (JavaScriptový zápis objektu). Jedná se o formát zápisu dat pro jejich přenos, nebo uložení. Tento formát je navzdory názvu kompatibilní s většinou programovacích jazyků, nezávisle na počítačové platformě. Jeho zápis není složitý a je srozumitelný jak programátorovi, tak počítači.

Informace se ukládají dvěma způsoby, a to jako dvojice `"identifikátor":data`, nebo pouze samotná data v podobě uspořádaného seznamu. Jednotlivé informace jsou vždy oddělovány čárkou. Soubor dat uzavřený ve složených závorkách symbolizuje objekt. Hranaté závorky zase označují pole prvků. Tyto způsoby zápisu lze různě kombinovat a zanořovat, díky čemuž je teoreticky možné přepsat do formátu *JSON* jakkoliv složité objekty.

Formát *JSON* umožňuje ukládat základní datové typy jako jsou textové řetězce, čísla, logické hodnoty či prázdná hodnota. Textové řetězce jsou ohraničeny uvozovkami. Číselné hodnoty jsou podporovány jako celočíselné, desetinné anebo v podobě exponenciálního zápisu. Logické hodnoty jsou reprezentovány klíčivými slovy `true` a `false`, prázdná hodnota klíčovým slovem `null`. Veškerá data v databázi DEMoS je možné převést do tohoto formátu, proto byl zvolen pro jejich ukládání v rámci této práce.

2.4 Systém správy verzí

Systém správy verzí je software, který se stará o ukládání historie všech změn provedených v souborech či jiných formátech dat [1]. Je hojně využíván při vývoji aplikací pro ukládání změn ve zdrojových kódech. Jeho využití však má smysl při práci s jakýmikoliv daty, zvláště pokud s nimi pracuje více lidí. V dnešní době ho lze nalézt jako součást různých textových či tabulkových editorů, wikipedie a dalších. Obvykle se kromě samotných změn ukládá také jejich autor a čas úpravy. Uživatelé si pak mohou dohledat historii změn a případně také dané informace obnovit do některé z jejich předchozích verzí. Na obrázku 2.5 je zachyceno provedení systému správy verzí článku na webu Wikipedie.



Obrázek 2.5: **Systém správy verzí na webu Wikipedie.** Snímek obrazovky zachycuje zobrazení pro porovnání dvou různých verzí článku na webu Wikipedie[5].

2.5 Relační databázový systém

Jednou z nejdůležitějších částí projektu DEMoS je relační databázový systém (dále označován zkráceným názvem databáze). Databáze je úložiště v počítači, které ukládá informace strukturované do tabulek [18]. Takto uložená data se dají různě seskupovat, třídit a podobně. Řešení téhle práce se výrazně týká databázové části, a proto jsou v následujících sekcích popsány využité prvky databáze.

2.5.1 Tabulka

Tabulka [16] je základním prvkem databáze. Skládá se z jednotlivých sloupců (atributů) a řádků (záznamů). Každý sloupec ukládá určitý typ dat a má daný datový typ. Určení

[5]Snímek obrazovky obsahuje zobrazení webové stránky: https://cs.wikipedia.org/w/index.php?title=Wikipedie%3A%C4%8Cl%C3%A1nek_t%C3%BDdne&type=revision&diff=19234245&oldid=18130620

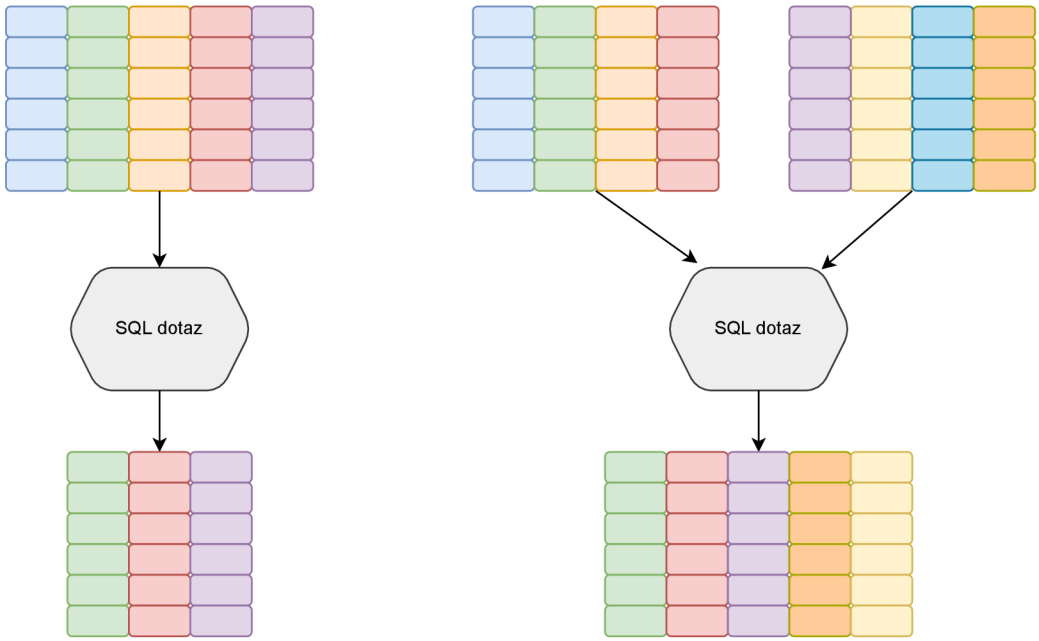
datového typu je důležité pro různá řazení záznamů podle zvolených sloupců, ale také slouží jako kontrola, že jsou do něj ukládány správné hodnoty. Do sloupce s datovým typem `int` tedy mohou být uložena pouze celá čísla a podobně. Dalším důvodem zadávání datového typu je určení velikosti, kterou bude záznam zabírat v úložišti.

Řádek tabulky reprezentuje jeden záznam. Každá tabulka by měla disponovat primárním klíčem, což je jednoznačný identifikátor. Primární klíč může sestávat z jednoho či více sloupců. Díky němu je každý záznam jednoznačně dohledatelný.

Kromě primárních klíčů mohou mít tabulky definované také cizí klíče. Tyto klíče odkazují na jiné tabulky, převážně na jejich primární klíče. Sloupec s cizím klíčem musí mít stejný datový typ jako sloupec s primárním klíčem v tabulce, na kterou odkazuje. Je možné nastavit kontroly či akce, které například zajistí, že pokud je smazán záznam v tabulce odkazované cizím klíčem, bude smazán také záznam, který tímto klíčem disponuje. Naopak lze také zamezit smazání záznamu v odkazované tabulce, je-li vázán k záznamu v tabulce jiné. Propojováním tabulek je možné vytvořit robustní model, aniž by obsahoval redundantní data.

2.5.2 Pohled

Pohled [17] (anglicky view) je podobný tabulce. Také se skládá ze sloupů a řádků. Na rozdíl od tabulek ovšem reálně neuchovává žádná data. Je totiž definován výběry či dotazy. Jeho účelem je vybírat požadovaná data z jedné či více tabulek nebo pohledů a ty pak zobrazit ve formátu jediné tabulky. Záznamy v pohledech lze řadit a také filtrovat. Jelikož však pohledy často kombinují data z více tabulek, nemohou pak disponovat primárními klíči, a proto nad nimi většinou nelze provádět úpravu samotných dat.



Obrázek 2.6: **Příklady fungování databázových pohledů.** Vlevo je naznačeno využití pohledu z důvodu bezpečnosti. Vpravo je zobrazen příklad pohledu, který se skládá z dat z více tabulek.

Pohledy lze využít například z bezpečnostních důvodů, kdy není některým uživatelům zpřístupněna originální tabulka, ale pouze pohled, který obsahuje jen některé sloupce a vynechává citlivá data. Dalším využitím je schování složitých výběrů pro zjednodušení dotazů a tím zlepšení přehlednosti. Příklady využití pohledů jsou znázorněny na obrázku 2.6.

2.5.3 Procedura

Procedura [15] slouží k provedení jednoho či více předem definovaných dotazů. Disponuje třemi typy parametrů. Parametry označené klíčovým slovem `IN` slouží k předání dat příkazům uvnitř procedury. Parametry typu `OUT` naopak slouží k navrácení hodnot po zavolání procedury. Posledním typem jsou parametry `IN OUT`, které, jak název napovídá, kombinují obě funkce. Díky parametrům lze ovlivňovat chování procedur, jelikož kromě samotných dotazů nad databází podporují také větvení, iterování a další funkce.

2.5.4 Spouštěč

Spouštěč [2] (anglicky trigger) je podobně jako procedura složen z jednoho či více dotazů. Na rozdíl od procedury však nedisponuje parametry. Liší se také tím, že je definován pro konkrétní tabulku a volá se pro každý ovlivněný záznam automaticky před, nebo po zvolené akci, jako je vložení, editace či smazání dat z tabulky. Spouštěč má vždy k dispozici dvě verze záznamu. Verzi před vykonáním akce a verzi po ní. Díky tomu může například automaticky ukládat zálohu dat, či počítat počet změn daného záznamu.

Kapitola 3

Popis současného stavu projektu DEMoS

Tato kapitola se zabývá analýzou systému DEMoS před samotnou implementací této práce. V první části jsou shrnuty obecné informace a cíl projektu DEMoS, v rámci něhož stejnojmenný systém vzniká. Následuje podrobný popis současné struktury databáze systému. V další části je zhodnocen stav jednotlivých prvků systému a na základě zjištěných poznatků navrhnuty potřebné úpravy a postup práce.

3.1 Projekt DEMoS

Projekt DEMoS vznikl na Fakultě informačních technologií VUT v Brně ve spolupráci s Filozofickou fakultou Masarykovy univerzity. Jeho cílem bylo vytvořit systém, který umožní přepisovat informace z archivních pramenů do databáze. Mezi archivní prameny patří například matriky, lánové rejstříky, sčítací operáty a další.

Přepisování informací z těchto pramenů je časově náročná věc, a proto je systém založen na dobrovolnické bázi. Zasahovat do záznamů tedy bude moci kdokoliv, kdo se zaregistruje. Aby nemohl čerstvě registrovaný uživatel upravovat záznamy zkušenému přepisovateli, je mu po registraci přidělena nejnižší zkušenostní úroveň. Systém disponuje kontrolou, která zaručí, že uživatelé s nižší úrovní nemohou provádět jakékoliv úpravy záznamů naposledy upravených uživateli s vyšší úrovní.

Na tomto projektu se již podílelo několik bakalářských a diplomových prací a každá postupně přidává určitou funkcionalitu. Jakmile bude systém plně funkční, bude z genealogických dat možné generovat rodokmen všech zaznamenaných osob.

3.2 Schéma databáze aplikace DEMoS

Jedním z úkolů systému DEMoS je ukládat matriční data do databáze. Jelikož se v matrikách nachází velké množství informací, bylo potřeba je roztřídit do jednotlivých tabulek. Databáze nyní disponuje více než čtyřiceti tabulkami. V této podkapitole jsou popsány tabulky a jejich propojení rozdělené do určitých logických částí. Není zde popsaná kompletně celá databáze, ale pouze tabulky důležité pro pochopení v rámci této práce. V diagramech jsou pro větší přehlednost zobrazeny pouze primární klíče, případně cizí klíče, jsou-li důležité.

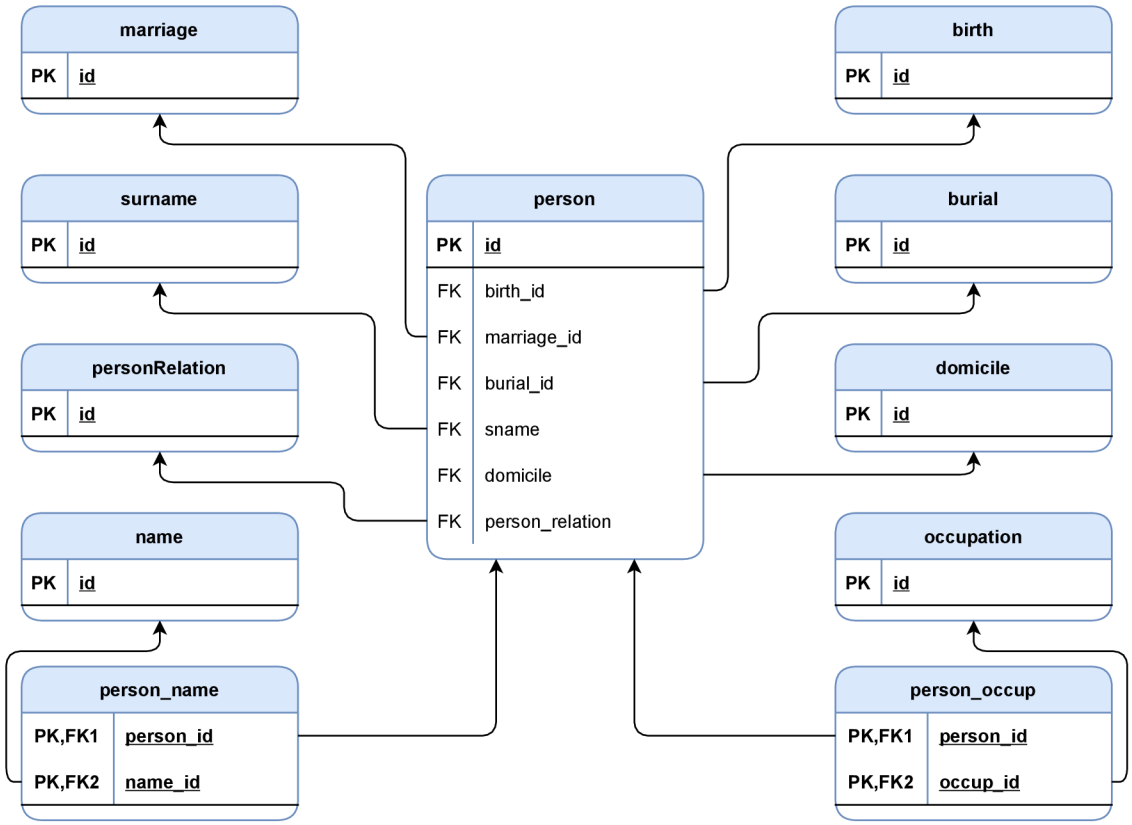
3.2.1 Osoba

Veškeré osoby vyskytující se ve všech typech záznamů se ukládají do databázové části znázorněné na obrázku 3.1. Hlavní tabulka, která má název `person`, uchovává základní údaje o osobách. Například titul, adresu bydliště, víru, datum narození a datum úmrtí. Pro přiřazení osoby k jednomu ze tří typů záznamů (rodný, oddací či úmrtní) slouží tři cizí klíče. Klíč `birth_id` odkazuje na záznam v tabulce `birth`, klíč `marriage_id` odkazuje na tabulku `marriage` a klíč `burial_id` na tabulku `burial`. Vždy je vyplněn pouze jeden z těchto klíčů a daná osoba tedy připadá pouze k jednomu typu záznamu.

Tabulka `person` dále disponuje cizím klíčem `domicile`, který odkazuje na stejnojmennou tabulku. V ní jsou uložené záznamy o obcích. Cizí klíč `sname` odkazuje na tabulku `surname`, která ukládá záznamy s příjmením. Dalším cizím klíčem je `person_relation` odkazující na tabulku `personRelation`, jež uchovává názvy vztahů mezi osobami.

Pro ukládání jmen osob je určená propojovací tabulka `person_name`, jejíž primární klíč sestává ze dvou cizích klíčů. Cizí klíč `person_id` odkazuje na tabulku `person` a klíč `name_id` na tabulku `name`, která uchovává záznamy se jmény. Kromě těchto klíčů dále tabulka `person_name` disponuje atributem `name_order`, který určuje pořadí jmen.

Podobným způsobem se ukládají také povolání dané osoby. K tomu byla vytvořena tabulka `person_occup`, která propojuje tabulku `person` s tabulkou `occupation`. Tabulka `occupation` uchovává informace o povolání. Na rozdíl od jmen zde není důležité pořadí jednotlivých povolání a tabulka `person_occup` tedy nedisponuje atributem `order`.



Obrázek 3.1: **Diagram znázorňující tabulky potřebné pro ukládání informací o osobách.** Tabulky v diagramu disponují pouze těmi atributy, které jsou klíčové k pochopení struktury dané části databáze.

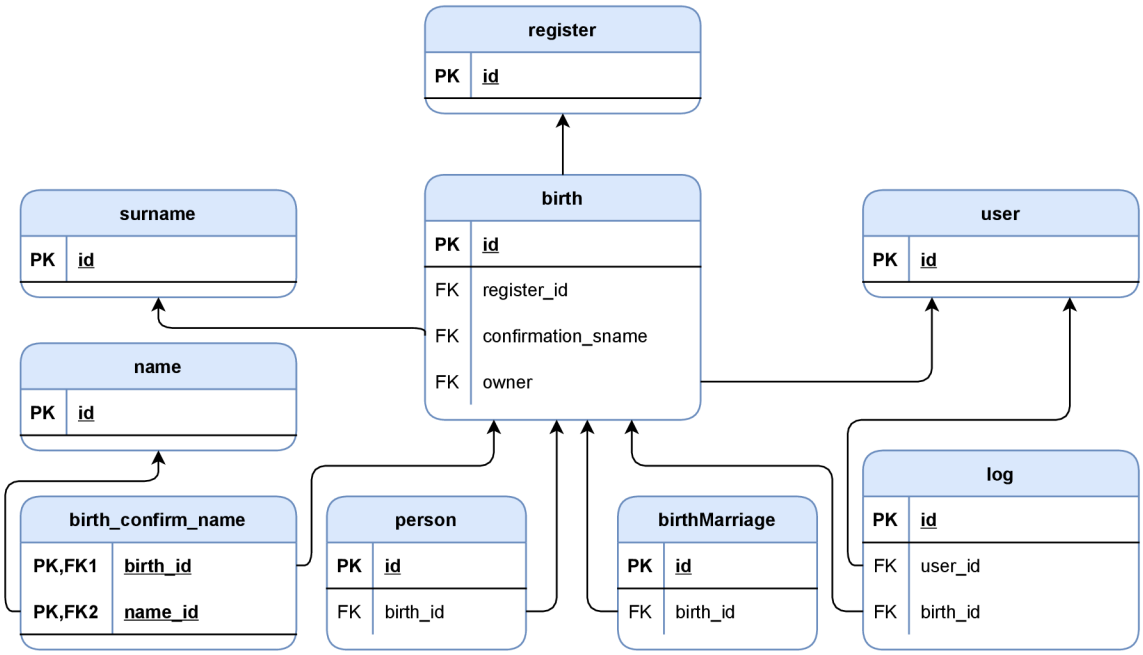
3.2.2 Rodný záznam

Rodné záznamy se ukládají do tabulek zobrazených na obrázku 3.2. Hlavní tabulkou pro rodný záznam je tabulka s názvem `birth`. Ta ukládá základní údaje o narození a křtu. Obsahuje cizí klíč odkazující na tabulku `register`. V tabulce `register` jsou uložené informace o jednotlivých matrikách.

Kromě křtu jsou v tabulce `birth` uložené také informace o biřmování. Konkrétně místo, čas a také jméno a příjmení osoby, která biřmování provedla. Pro uložení příjmení je v tabulce cizí klíč s názvem `confirmation_sname`, který odkazuje na tabulku `surname` ukládající záznamy s příjmením. K ukládání jména je použita propojovací tabulka s názvem `birth_confirm_name`, která disponuje primárním klíčem složeným ze dvou cizích klíčů, kdy klíč s názvem `birth_id` odkazuje na záznam v tabulce `birth` a klíč `name_id` odkazuje na záznam se jménem v tabulce `name`.

Posledním cizím klíčem v tabulce `birth` je klíč s názvem `owner` odkazující na tabulku s uživateli `user`. Sloupec `owner` ukládá informaci o majiteli záznamu. Majitelem záznamu je uživatel, který jako poslední upravil záznam. Kromě sloupců ukládajících informace z matrik disponuje tabulka `birth` dalšími technickými atributy, jako například `score`, jež ukládá výši bodů, které uživatel dostal za vytvoření či úpravu záznamu. Dalším atributem je `last_edited_level`, který ukládá úroveň určující zkušenost uživatele. Tato informace se následně využívá k určení oprávnění uživatelů pro úpravu daného záznamu.

Jednotlivé osoby nacházející se v záznamu narození jsou ukládány do tabulky `person` a dalších tabulek v databázové části určené záznamům o osobách, která je popsána v kapitole 3.2.1. Rodné záznamy také disponují informacemi o svatbách. K tomu je určena další část databáze popsána v kapitole 3.2.3. Informace o provedení změny rodného záznamu jsou ukládány do tabulky `log`. Ta ukládá identifikátor záznamu, autora a čas úpravy.



Obrázek 3.2: **Diagram znázorňující tabulky potřebné pro ukládání informací z rodných záznamů.** Tabulky v diagramu disponují pouze těmi atributy, které jsou klíčové k pochopení struktury dané části databáze.

3.2.3 Sňatek v rámci rodného záznamu

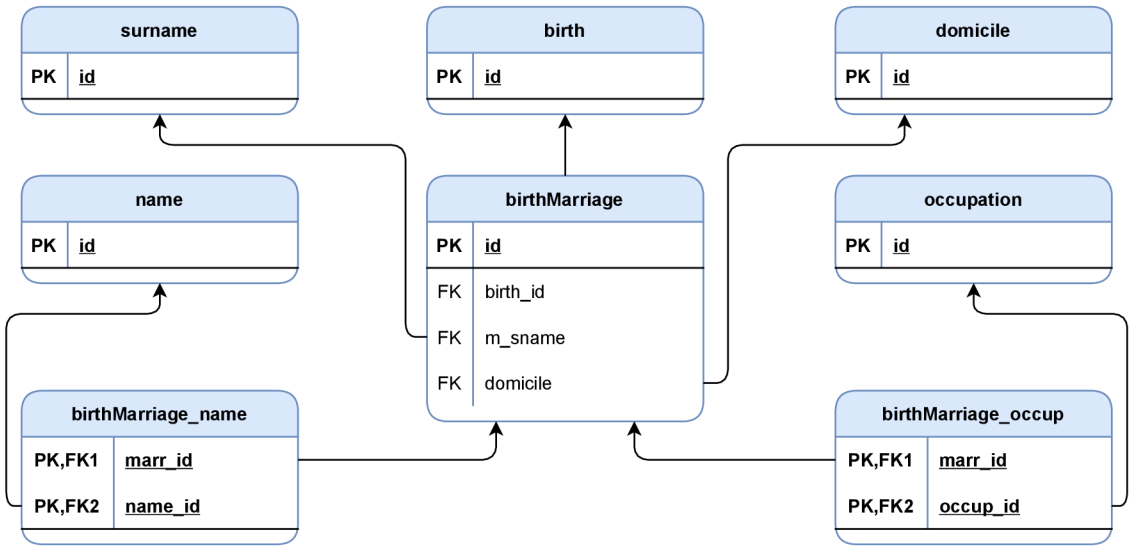
Do rodných záznamů občas bývala dopsána informace o sňatku, nejedná se však o náhradu oddacích záznamů, nýbrž o jakousi dodatečnou informaci. Pro ukládání těchto informací se využívá část databáze, jejíž schéma je znázorněno na obrázku 3.3. Hlavní tabulka má název `birthMarriage`. Ukládá informace jako jsou datum a místo svatby či bydliště partnera. Každý záznam se vztahuje k jednomu rodnému záznamu v tabulce `birth`, na kterou se odkazuje pomocí cizího klíče `birth_id`. Jeden rodný záznam může mít více svatebních záznamů a jejich pořadí určuje atribut `num` v tabulce `birthMarriage`.

Cizí klíč s názvem `m_sname` slouží k uložení příjmení partnera a odkazuje na tabulku `surname`, která ukládá záznamy s příjmením. Tabulka `birthMarrige` také disponuje cizím klíčem s názvem `domicile`. Ten odkazuje na tabulku stejného názvu a ukládá obec jako součást trvalého bydliště partnera.

Pro uložení jmen partnera se využívá propojovací tabulka `birthMarriage_name`. Její primární klíč je složen ze dvou cizích klíčů. Cizí klíč `marr_id` odkazuje na záznam v tabulce `birthMarriage` a klíč s názvem `name_id` odkazuje na tabulku `name`, ve které jsou uloženy záznamy se jmény. Pořadí jmen je určeno atributem `order` v tabulce `birthMarriage_name`.

Databáze také ukládá informace o povoláních partnera. K tomu slouží propojovací tabulka `birthMarriage_occup`. Stejně jako tabulka `birthMarriage_name` disponuje primárním klíčem složeným ze dvou cizích klíčů. Klíč `marr_id` odkazuje na záznam o sňatku narozeného, který se nachází v tabulce `birthMarriage`. Druhý klíč s názvem `occup_id` odkazuje na tabulku `occupation` se záznamy o povoláních.

Záznamy o sňatcích narozeného jsou značně specifické. Díky tomu, že jsou pouze jakousi podčástí rodných záznamů, nemají přímou vazbu na tabulku `register`. Jsou také jediným typem záznamu, který pro ukládání osob nevyužívá tabulku `person`.



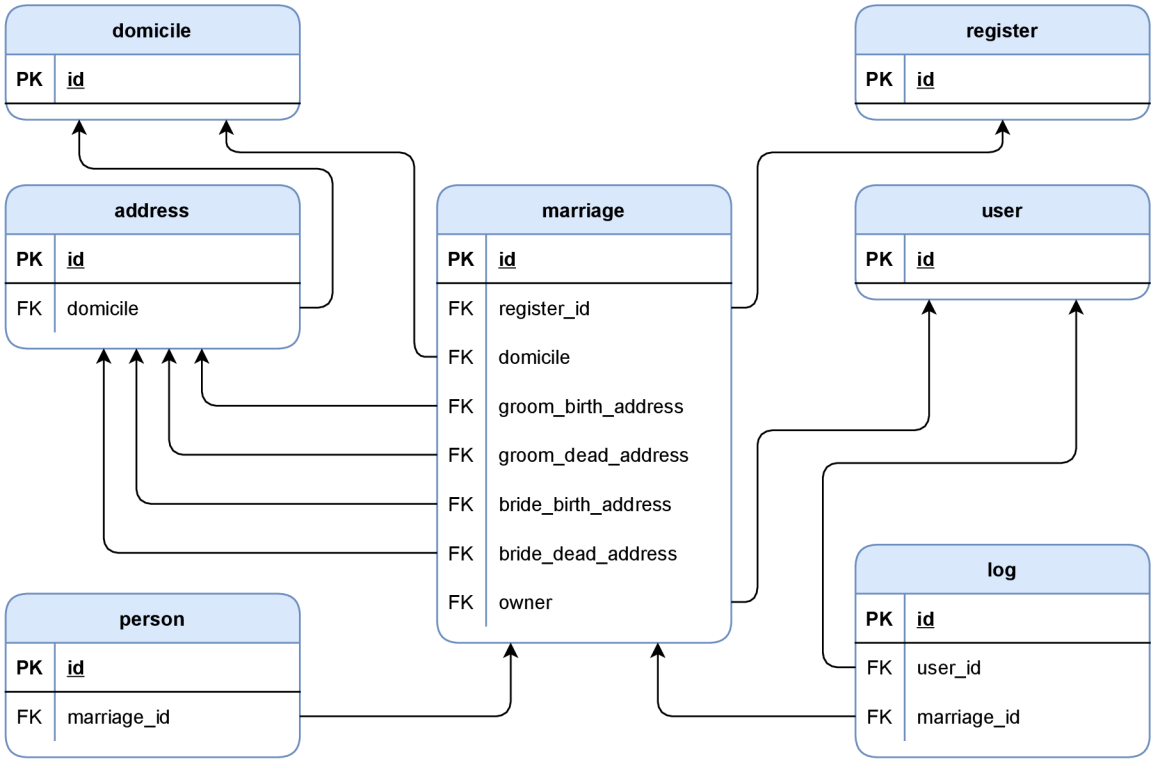
Obrázek 3.3: **Diagram znázorňující tabulky potřebné pro ukládání informací o sňatcích v rámci rodných záznamů.** Tabulky v diagramu disponují pouze těmi atributy, které jsou klíčové k pochopení struktury dané části databáze.

3.2.4 Oddací záznam

Záznamy o sňatcích se ukládají do tabulek, které jsou zobrazeny na obrázku 3.4. Hlavní tabulkou databázové části pro ukládání oddacích záznamů je tabulka `marriage`. Ta ukládá informace o svatbě jako jsou datum svatby, data ohlášek a další. Pro přiřazení záznamu k matrice tabulka disponuje cizím klíčem `register_id`, který odkazuje na záznam s informacemi o matrice v tabulce `register`. Pro uložení obce, ve které svatba proběhla, se využívá cizí klíč `domicile`, odkazující na stejnojmennou tabulku se záznamy o obcích.

Kromě informací o samotné svatbě ukládá tabulka `marriage` také údaje o snoubencích. Mezi ně patří adresy narození a úmrtí nevěsty i ženicha. K tomu slouží čtyři cizí klíče, které odkazují na tabulku `address`. Ta ukládá obec, ulici a číslo popisné. Obec je zde stejně jako v jiných tabulkách zastoupená cizím klíčem `domicile` odkazujícím na stejnojmennou tabulku. Mezi další atributy tabulky `marrige` patří například datum rozvodu, plnoletost snoubenců, nebo stupeň příbuznosti.

Tabulka `marriage` disponuje také technickými atributy, jako jsou číslo skenu a pozice záznamu na něm, jazyk záznamu a jiné. Dalším technickým atributem je `owner`, což je cizí klíč odkazující na tabulku `user`. Tento atribut označuje autora poslední změny. Pro určení oprávnění uživatelů k úpravě záznamu tabulka disponuje sloupcem s názvem `last_edited_level`, který ukládá zkušenostní úroveň autora poslední změny. Informace o provedení změny oddacího záznamu jsou ukládány do tabulky `log`. Ta disponuje cizím klíčem `marriage_id`, který odkazuje na záznam v tabulce `marriage`. Cizí klíč `user_id` odkazuje na tabulku user a ukládá autora. Tabulka `log` mimo jiné zaznamenává čas změny.



Obrázek 3.4: **Diagram znázorňující tabulky potřebné pro ukládání informací z oddacích záznamů.** Tabulky v diagramu disponují pouze těmi atributy, které jsou klíčové k pochopení struktury dané části databáze.

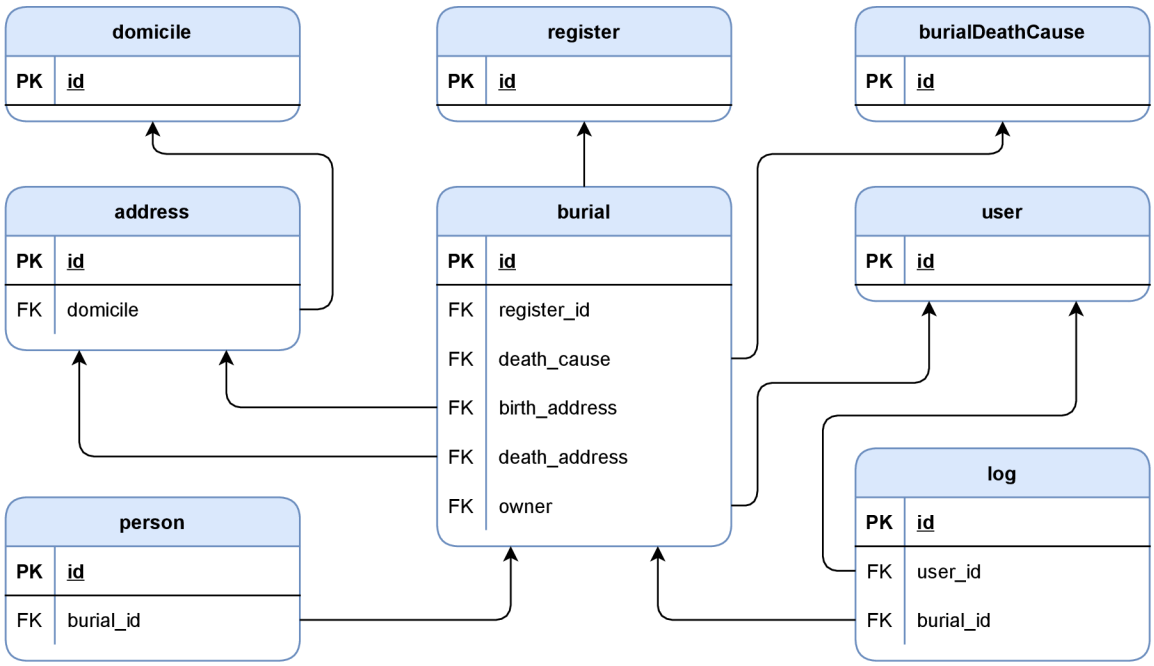
3.2.5 Úmrtní záznamy

Úmrtní záznamy jsou ukládány do tabulek vyobrazených na obrázku 3.5. Hlavní tabulka nese název `burial`. Ta disponuje cizím klíčem `register_id`, který odkazuje na tabulku `register`. Tato tabulka uchovává záznamy o matrikách. Tabulka `burial` ukládá informace o úmrtí, pohřbu a také o samotném zemřelém. Disponuje tedy atributy pro údaje jako jsou například datum úmrtí, datum pohřbu, místo pohřbení, věk dožití s přesností na minuty a další. Nachází se zde také informace o manželství zemřelého. Na rozdíl od rodných záznamů zde lze uložit pouze jedno manželství, které se ukládá přímo do tabulky `burial`.

Úmrtní záznamy obsahují také adresy narození a úmrtí zemřelého. K jejich uložení slouží tabulka `address`. Ta disponuje atributy `street` pro ulici, `descr_num` pro číslo popisné a cizím klíčem `domicile` odkazujícím na stejnojmennou tabulku pro obec. V tabulce `burial` se nachází dva cizí klíče `birth_address` a `death_address`, které na tabulku `address` odkazují.

Pro uložení příčiny úmrtí slouží v tabulce `burial` cizí klíč `death_cause`, který odkazuje na tabulku s názvem `burialDeathCause`. Cizí klíč `owner` odkazuje na tabulku `user`, která uchovává informace o uživatelích. Tento atribut slouží k uložení takzvaného autora záznamu, jímž je autor poslední změny.

Informace o veškerých osobách vyskytujících se v úmrtních záznamech jsou uloženy v tabulce `person` a dalších spadajících do výše popsané databázové části pro ukládání osob. Každá provedená úprava se zaznamenává do tabulky `log`. Ta obsahuje cizí klíč `user_id` odkazující na tabulku `user` pro identifikaci uživatele, který úpravu provedl. Pro přiřazení k záznamu v tabulce `burial` zase slouží cizí klíč s názvem `burial_id`.



Obrázek 3.5: **Diagram znázorňující tabulky potřebné pro ukládání informací z úmrtních záznamů.** Tabulky v diagramu disponují pouze těmi atributy, které jsou klíčové k pochopení struktury dané části databáze.

3.3 Zhodnocení současného stavu a návrh práce

Tato sekce se zabývá zhodnocením současného stavu systému po jeho důkladném prozkoumání a otestování. Na základě získaných informací byl následně navržen postup práce. V první části je podrobně popsán cíl této práce. V následující části jsou vylíčeny zjištěné poznatky z testování současného systému co se ukládání záznamů týče a podle nich pak navržen postup práce pro opravení nalezených chyb a implementaci systému ukládání změn. V kapitole 3.3.3 je zhodnocen současný styl zobrazení uložených záznamů a následně popsán návrh na úpravy, které zobrazení zpřehlední. Dále je zde také charakterizován návrh zobrazování historie úprav záznamu. V poslední části je popsán návrh implementace pro obnovení záznamu do některé z jeho předchozích verzí.

3.3.1 Cíl práce

Cílem této práce je vytvoření systému, který bude automaticky ukládat informace o provedených úpravách záznamů uložených v databázi. Jelikož na projektu DEMoS pracuje více studentů zároveň, není žádoucí přímo zasahovat do současného návrhu databáze. Je však možné přidávat pomocné tabulky, které budou navázány na tabulky s matričními záznamy. Pro urychlení systému je plánováno co nejvíce pracovat s daty na úrovni samotné databáze. Tím se zamezí zbytečná komunikace mezi webovým a databázovým serverem.

Pro zobrazení historie úprav záznamu bude potřeba navrhnout grafické rozhraní, které bude dostatečně přehledné a také intuitivní. Jelikož záznamy obsahují velké množství informací, je potřeba je strukturovat do jedné či více tabulek. Stránka s historií by měla zobrazovat všechny aktuální informace daného záznamu a k nim zobrazit změny seřazené podle času. Uživatel by zde také měl mít možnost obnovit záznam do jednoho z jeho předchozích stavů.

Databázovou a zobrazovací část bude potřeba propojit a tím systém správy revizí napojit na již existující systém DEMoS. K tomu účelu bude potřeba upravit manažery (soubory v logické části model návrhového vzoru MVC) kontrolující obsluhu požadavků týkajících se matričních dat. Konkrétně jejich metody sloužící k ukládání záznamů. Dále v nich bude potřeba vytvořit metody, které z databáze získají informace o úpravách a také metody obsluhující obnovení záznamů. Matriky uchovávají tři typy záznamů, a to rodné, oddací a úmrtní. Pro všechny tři typy bude potřeba provést výše zmíněné úpravy.

3.3.2 Ukládání a editace záznamů

Po prozkoumání a otestování již existujícího systému byly nalezeny chyby v ukládání záznamů. Při ukládání oddacích a úmrtních záznamů byly chyby natolik závažné, že zcela znemožňovaly samotné ukládání. Bylo to způsobeno využitím jiné technologie pro práci s databází, než kterou využívá část systému věnující se rodným záznamům. Pro sjednocení technologií a zaručení bezproblémové funkčnosti bude potřeba přepsat manažery pro oddací a úmrtní záznamy do podoby odpovídající manažeru rodných záznamů. Ukládání rodných záznamů však také není kompletně funkční. Zcela zde chybí implementace ukládání informací o svatbách v rámci rodných záznamů. Výše zmíněné změny bude potřeba dokončit před implementací samotného systému správy revizí záznamů. Společně s řádným otestováním funkčnosti původního systému tyto úpravy tvoří nemalou část práce, přestože se nejedná o předmět původního zadání.

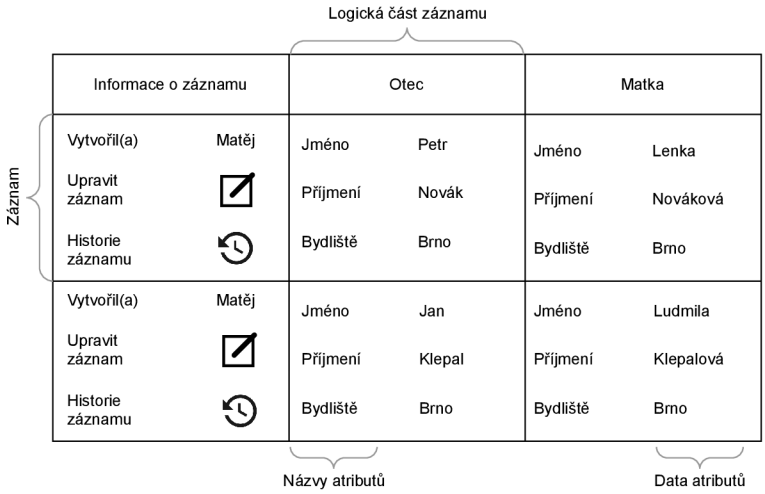
Po zprovoznění základní funkčnosti ukládání dat bude následovat vytvoření pomocných tabulek, do kterých se budou ukládat informace o provedených úpravách. Samotné ukládání

změn bude prováděno pomocí spouštěčů, které bude potřeba implementovat pro všechny hlavní tabulky, které jsou popsány v kapitole 3.2. Ukládání změn v propojovacích tabulkách (například tabulky `person_name`, nebo `person_occup` zobrazeny na obrázku 3.1) se nebude provádět automaticky na straně databáze, jelikož se nad propojovacími tabulkami volají pouze příkazy pro vytvoření a smazání záznamů, nikdy však jejich úprava. Původní záznamy v propojovací tabulce, které se vztahují k upravenému záznamu, se na straně serveru uloží do formátu *JSON*, který bude následně vložen do pomocné tabulky. Některé tabulky, jako například `name`, nebo `occupation`, nepotřebují zálohovat jejich předchozí stavy, jelikož se do nich pouze vkládají nové záznamy. Nikdy se však neprovádí úprava či mazání těchto záznamů, protože na ně mohou být navázány jiné záznamy.

Při každé editaci záznamu se do tabulky `log` uloží informace o autoru a čase změny. V každé pomocné tabulce bude kromě původních dat také atribut odkazující na záznam v tabulce `log`, který zajistí synchronizaci změn v rámci všech pomocných tabulek.

3.3.3 Zobrazení provedených změn

Webová aplikace v současné době disponuje zobrazením seznamu záznamů v podobě tabulky, kdy jeden řádek tabulky reprezentuje jeden rodný, oddací či úmrtní záznam. Jelikož tyto záznamy disponují vysokými desítkami atributů, jeví se tato tabulka jako nekonečná. Uživateli zabere značnou dobu, než se dostane z levé části tabulky do pravé. Tlačítko pro úpravu záznamu se navíc nachází na samotném konci tabulky, což není uživatelsky přívětivé. Z toho důvodu bude potřeba přepracovat tuto tabulku a strukturovat jednotlivé atributy podle jejich příslušnosti. Jeden sloupec tabulky tedy nebude reprezentovat jeden atribut, ale jednu logickou část záznamu jako jsou například jednotlivé osoby. Tento sloupec bude rozdělen na dva vnořené sloupce, kdy v pravém sloupci budou vypisovány názvy atributů a v levém sloupci jejich hodnoty. Jeden řádek tabulky tedy bude rozdělen na několik vnořených řádků podle počtu atributů v dané logické části. Toto řešení sice tabulku vertikálně rozšíří, informace o jednotlivých záznamech však budou značně přehlednější. V prvním sloupci tabulky se bude vždy nacházet autor záznamu, tlačítko pro editaci záznamu a také tlačítko pro zobrazení historie záznamu. Návrh tabulky je zobrazen na obrázku 3.6.



Obrázek 3.6: **Návrh zobrazení záznamů.** Zobrazuje seznam záznamů dané matriky.

Pro zobrazení historie úprav bude potřeba nejdříve sestavit informace z pomocných tabulek do formy odpovídající originálním tabulkám. K tomu budou implementovány databázové pohledy. Tyto pohledy budou mít stejné atributy jako originální tabulka. Navíc přibude atribut s názvem `log_id` odkazující na záznam v tabulce `log`, díky němuž bude možné dohledat autora a čas změny. Jednotlivé řádky v pohledech budou zobrazovat všechny provedené změny, seřazené podle `log_id`. Atributy, které nebyly v daném čase změněny, budou nabývat hodnoty `null`. Pokud daná úprava znamená přidání informace, daný atribut bude obsahovat textový řetězec složený z jedné mezery. To umožní simulaci prázdného záznamu, který však není `null`, jelikož zde došlo ke změně.

Samotné zobrazení historie úprav uživateli bude sestávat z tabulky, jejíž první řádek bude zobrazovat aktuální stav záznamu v podobě, jak bylo popsáno v prvním odstavci této části. Pod ním bude každý řádek symbolizovat sadu úprav v daném čase od jednoho uživatele. Jednotlivé úpravy budou seřazeny chronologicky od nejnovější po nejstarší. Uživatel tak snadno dostane přehled o tom, v jakém stavu se daný záznam nacházel ve zvoleném čase. První sloupec této tabulky bude v řádcích s úpravami obsahovat autora změny, čas změny a tlačítko pro obnovení záznamu do daného času.

3.3.4 Obnovení záznamu do zvoleného předchozího stavu

Jak již bylo zmíněno v předchozí sekci, v prvním sloupci tabulky zobrazující historii úprav se bude nacházet tlačítko, jehož funkcí bude obnovení záznamu do daného stavu. Pro urychlení systému se toto obnovení bude převážně vykonávat na straně databáze. Odpovídající metoda v manažeru obsluhujícím záznam tedy pouze zkontroluje oprávnění uživatele a uloží informaci o provedení změny do tabulky `log`. Následně bude volat databázové procedury, které se postarají o samotné obnovení. Pro každou tabulku, na kterou se obnovení dat vztahuje, bude vytvořena jedna procedura. Tabulky v databázi systému DEMoS lze rozdělit na tři základní typy. Proto budou vytvořeny také tři typy procedur. Prvním typem budou procedury pro takzvané hlavní tabulky záznamu, které ukládají informace o majiteli záznamu. Druhým typem budou procedury pro propojovací tabulky, jejichž primární klíč je sestaven ze dvou cizích klíčů a třetím typem procedury pro obecné tabulky. Změny provedené pro obnovení záznamu z aktuálního do některého z předchozích stavů se uloží stejně jako kdyby uživatel dané údaje přepisoval ručně.

Kapitola 4

Implementace databázových prvků

V této kapitole je popsána implementace databázové části řešení této práce. V první sekci se nachází popis zvoleného návrhu struktury tabulek, do kterých se ukládají informace o provedených úpravách. Další sekce se zabývá automatickým uložením informací do pomocných tabulek pomocí spouštěčů. Následuje popis implementace dvou druhů pohledů. Jeden druh byl implementován pro zobrazení provedených úprav uživateli a druhý typ je využit v procedurách pro obnovení záznamu do některého z jeho předchozích stavů. Popis implementace zmíněných procedur se nachází v poslední sekci této kapitoly.

4.1 Tabulky pro ukládání změn

Každé tabulce v databázi DEMoS, ve které může dojít ke změně dat zapříčiněnou editací záznamu, byla vytvořena pomocná tabulka pro ukládání změn. Ta slouží k uložení původního stavu dat před editací záznamu. Ukládají se zde pouze atributy, které byly změněny.

Jakmile uživatel provede editaci záznamu, je potřeba uložit všechny změněné informace. Jednotlivé atributy tabulek mají různé datové typy a bylo by plýtváním prostředky, kdyby jednotlivé tabulky pro zálohování obsahovaly všechny atributy jako originální tabulka, jejíž změny se ukládají. Data je tedy nutné převést do formátu, který nebude prostorově náročný, a ve kterém nehrozí ztráta žádných informací. Zároveň bylo potřeba jednoznačně určit atribut, jemuž uložená data náleží.

Zvoleným řešením, které splňuje všechny výše zmíněné požadavky, je *JSON* (JavaScript Object Notation). Do něj lze bez problému uložit jakýkoliv datový typ. Změněné informace jsou ukládány ve tvaru `{„název atributu1":data1, „název atributu2":data2, ...}`. Jeden řádek pomocné tabulky s historií změn tedy obsahuje všechna původní data, která byla danou editací v originální tabulce změněna.

Kromě původních dat pomocné tabulky disponují cizím klíčem `original_record_id`, který odkazuje na primární klíč v originální tabulce. Díky tomu jsou původní data přiřazena ke svému záznamu. Dalším důležitým atributem je cizí klíč s názvem `log_id`, který odkazuje na tabulku s názvem `log`. V tabulce `log` se ukládají informace o tom, který záznam byl změněn, kdo změnu provedl a také čas změny. Atribut `log_id` je využíván pro synchronizaci všech změn. Uživatel totiž editací záznamu může ovlivnit několik tabulek najednou. Pro následné sestavení verze záznamu v daném čase je potřeba získat původní data ze všech pomocných tabulek najednou. Díky atributu `log_id` to tedy není problém.

Pojmenování pomocných tabulek se skládá z názvu originální tabulky doplněném o koncovku `„_h"`.

4.2 Spouštěče pro uložení změn

Po provedené editaci záznamu je potřeba zjistit, jaké atributy byly změněny, a na základě těchto informací uložit původní data ve formátu *JSON* do pomocné tabulky. Pro minimalizaci zásahu do manažerů projektu a také pro urychlení sytému byly v databázi implementovány spouštěče, které mají na starosti výše popsané úkoly.

Jak již bylo zmíněno v kapitole 2.5.4, spouštěč je implementován pro každou tabulku zvlášť. Spouští se před, nebo po provedení určité akce s daty v tabulce, a to pro každý ovlivněný záznam. Spouštěč má také k dispozici verzi záznamu před provedením akce a verzi po ní. Tyto vlastnosti byly využity při implementaci.

Pro potřeby systému DEMoS byly implementovány dva druhy spouštěčů. Prvním druhem je spouštěč, který se aktivuje po provedení příkazu `UPDATE`. Nejdříve se vytvoří prázdný objekt *JSON*, do kterého se poté budou ukládat původní data změněných atributů. Tento objekt je uložen v proměnné s názvem `data`. Následuje větvení, jehož podmínky porovnávají atributy tabulky před provedením úprav a po nich. Pokud se dané verze atributů liší, vloží se původní data do *JSON* objektu společně s názvem atributu. Porovnávání dat v MySQL vrací vždy `null`, pokud je alespoň jeden ze vstupů porovnávání roven hodnotě `null`. Z toho důvodu je potřeba vyhodnocovat zvlášť stav, kdy byl záznam prázdný, tedy nabýval hodnotu `null`, a v rámci úprav mu byla přidělena data, nebo naopak. K tomu byla využita funkce `ISNULL`. Pokud byla po kontrole všech atributů nalezena změna, uloží se vytvořený *JSON* do pomocné tabulky společně s `id` záznamu v originální tabulce.

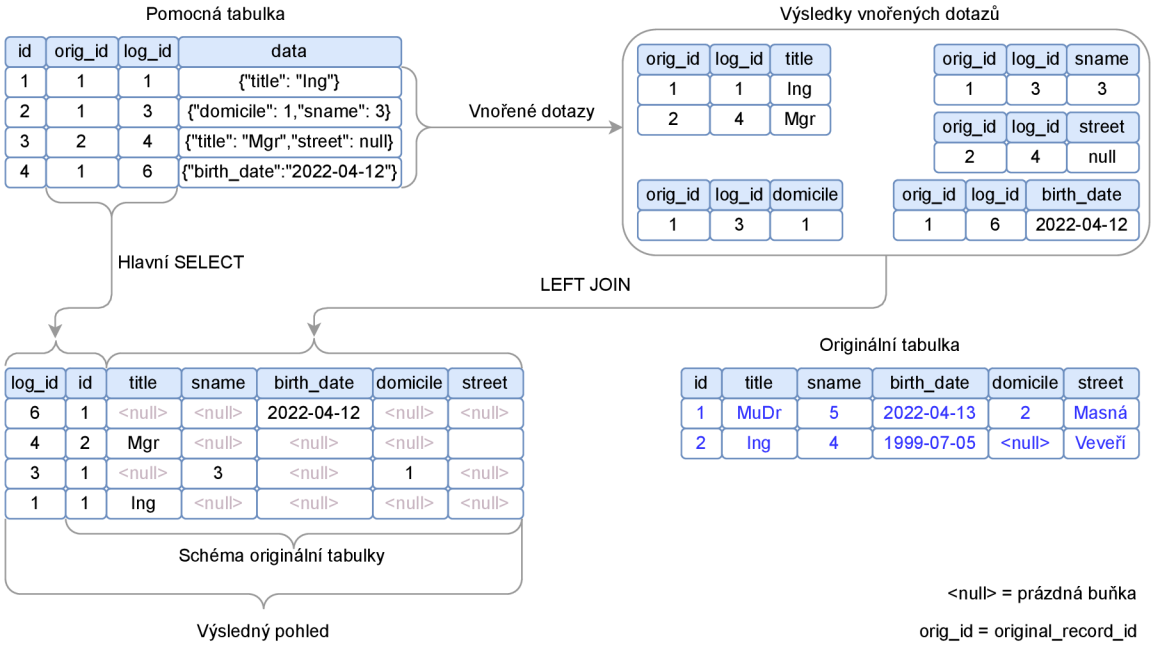
Druhým typem spouštěčů implementovaných v rámci této práce jsou spouštěče, které se aktivují po vložení nového záznamu do originální tabulky, tedy po příkazu `INSERT`. Tyto spouštěče byly implementovány pouze pro dvě tabulky, konkrétně pro tabulku s názvem `person` a tabulku s názvem `birthMarriage`. Tyto tabulky totiž reprezentují jakési menší záznamy vnořené do hlavních záznamů. Tabulka `person` uchovává informace o všech osobách, vyskytujících se v záznamech, a tabulka `birthMarriage` zase informace o sňatcích v rámci rodných záznamů. Pro tyto části databáze je důležité ukládat také informace o vložení záznamu. Díky tomu je možné v historii úprav zobrazit informaci, že byl záznam o dané osobě či sňatku v daném čase vytvořen. Ještě větší význam však má tato informace pro obnovení záznamu do některého z předchozích stavů, což je podrobněji popsáno v kapitole 4.5. Spouštěč pro tabulku `person` kontroluje atributy `birth_id`, `marriage_id` a `burial_id`. Pokud některý z nich není prázdný, uloží do objektu *JSON* název daného atributu a púvodní data nahradí hodnotou `null`. Spouštěč pro tabulku `birthMarriage` kontroluje pouze atribut `birth_id` a pokud není prázdný, uloží do objektu *JSON* `null` stejně jako u tabulky `person`.

4.3 Pohledy pro zobrazení provedených změn

Pokud si chce uživatel zobrazit historii úprav záznamu, je potřeba, aby systém z dat uložených v pomocných tabulkách, uchovávajících předchozí verze atributů, zrekonstruoval jednotlivé úpravy v závislosti na čase. K tomuto účelu byly implementovány databázové pohledy pro všechny tabulky, které disponují pomocnými tabulkami pro ukládání změn. Cílem tohoto pohledu je tedy projít pomocnou tabulku a data uložená ve formátu *JSON* převést do formátu odpovídajícímu původní tabulce. Proces získání dat pro daný pohled je znázorněn na obrázku 4.1.

Pohled pro zobrazení provedených změn je definován dotazem za použití příkazu `SELECT`, který z pomocné tabulky vybírá sloupec `log_id` a také sloupec `original_record_id`, je-

muž přidává alias `id`. Další informace jsou již uložené v atributu s názvem `data`, a to ve formátu *JSON*. Pro extrakci jednotlivých atributů z tohoto formátu byly implementovány vnořené dotazy s příkazem `SELECT`. Tyto dotazy z pomocné tabulky vybírají vždy jeden atribut odpovídající originální tabulce společně s atributy `original_record_id` a `log_id`, které budou využity pro následné spojování záznamu. Samotné atributy originální tabulky jsou z formátu *JSON* extrahovány pomocí dvou do sebe zanořených funkcí, a to konkrétně `JSON_UNQUOTE(JSON_EXTRACT(data, '$.nazev_atributu'))`. Funkce `JSON_EXTRACT` projde uložený objekt ve formátu *JSON* a pokud v něm nalezne klíč shodující se s názvem atributu, vrátí jeho data. Jestliže uložená data obsahují hodnotu datového typu textový řetězec, funkce `JSON_EXTRACT` je navrátí uzavřené v uvozovkách. Pro odstranění uvozovek byla využita funkce `JSON_UNQUOTE()`. Pokud mají uložená data jiný datový typ, funkce `JSON_UNQUOTE` je vrátí beze změny. Každý vnořený dotaz tedy vrací výsledek ve formě tabulky s identifikátory originálního záznamu a úpravy, doplněné jedním atributem originální tabulky.



Obrázek 4.1: **Diagram znázorňující výběr dat při tvorbě pohledu pro zobrazení provedených změn.** Originální tabulka reprezentuje tabulky v databázi DEMoS, jejichž úpravy je potřeba ukládat. Pomocná tabulka reprezentuje tabulky, do kterých se zapisují informace o provedených změnách.

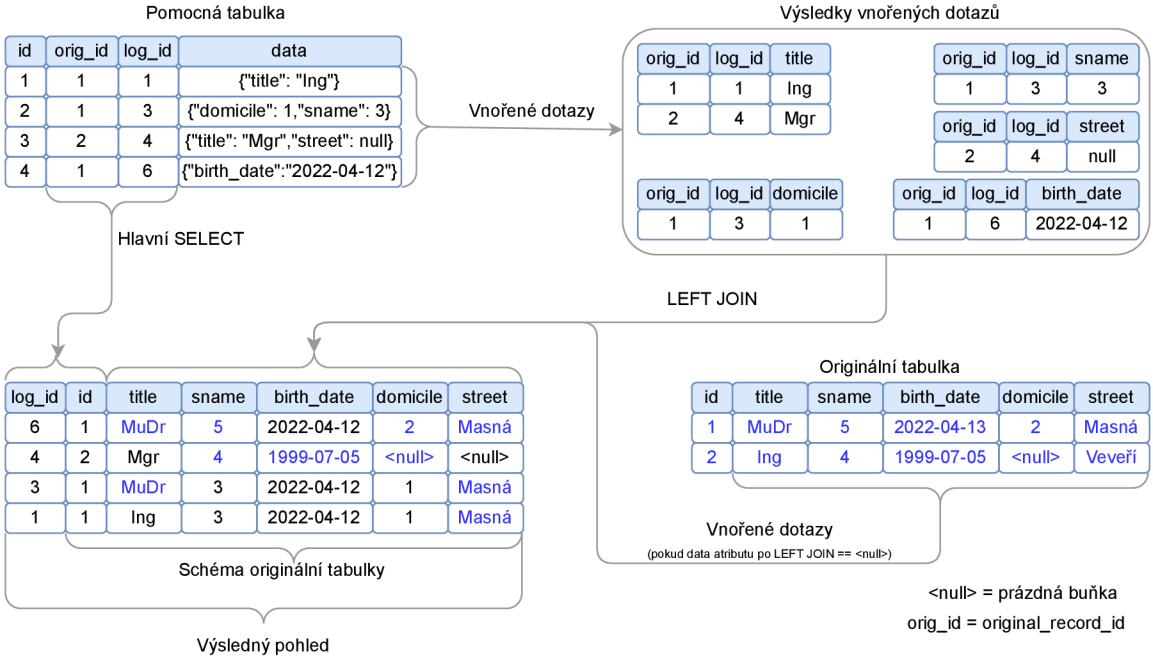
Složení těchto vnořených dotazů s pomocnou tabulkou je implementováno pomocí příkazu `LEFT JOIN` v rámci hlavního dotazu `SELECT`. Příkaz `LEFT JOIN` spojuje tabulky takovým způsobem, že ke všem záznamům pomocné tabulky přiřadí data, nachází-li se ve výsledku vnořeného dotazu. Pokud nebyla nalezena shoda, přiřadí se zde hodnota `null`. Spojování záznamů se řídí podmínkami stanovenými za klíčovým slovem `ON`. Tyto podmínky stanovují, že atributy `original_record_id` a také `log_id` ve vnořených dotazech musí odpovídat stejnojmenným atributům v pomocné tabulce.

Pro každý atribut originální tabulky je tedy v pohledu vytvořen vnořený dotaz, který je následně výše popsaným způsobem připojen k hlavnímu dotazu. Hlavní dotaz v příkazu

`SELECT` vybírá jednotlivé atributy ze vnořených dotazů podle jejich názvů v originální tabulce. Tento způsob si může dovolit díky přiřazování aliasů ve vnořených dotazech. Při tomto výběru se navíc provádí kontrola, zda-li se data daného atributu nerovnají textovému řetězci `'null'`. Pokud ano, znamená to, že byl před úpravou atribut prázdný, a proto se tato data nahradí prázdným textovým řetězcem. Výsledek hlavního dotazu je nakonec sestupně seřazen podle atributu `log_id`. Nejnovější změna je tedy zobrazena jako první.

4.4 Pohledy pro zobrazení jednotlivých verzí záznamu

Dalším typem pohledů využívaných v systému pro správu revizí záznamů je pohled, který zobrazuje stav záznamu před provedením změny v daném čase. Tyto pohledy nejsou využity přímo v systému pro zobrazení dat uživateli, využívají se však v procedurách pro obnovení záznamu do některého z předchozích stavů, jak je blíže popsáno v kapitole 4.5. Tyto pohledy tedy prochází pomocné tabulky a na základě dat uložených ve formátu *JSON* rekonstruují dané záznamy v čase před provedením úpravy. Proces získání dat pro daný pohled je znázorněn na obrázku 4.2.



Obrázek 4.2: **Diagram znázorňující výběr dat při tvorbě pohledu pro zobrazení jednotlivých verzí záznamu.** Originální tabulka reprezentuje tabulky v databázi DEMoS, jejichž úpravy je potřeba ukládat. Pomocná tabulka reprezentuje tabulky, do kterých se zapisují informace o provedených změnách. Modrý text ve výsledném pohledu znázorňuje data, která byla vybrána z originální tabulky.

Podobně jako pohledy pro zobrazení provedených změn je i tento typ pohledů deklarován pomocí dotazu za využití příkazu `SELECT`. Ten z pomocné tabulky vybírá atributy `log_id` a `original_record_id`, jemuž přidává alias `id`. Samotné atributy jsou z formátu *JSON* extrahovány pomocí funkcí `JSON_UNQUOTE(JSON_EXTRACT(data, '$.nazev_atributu'))`. Napojení vnořených dotazů na hlavní dotaz je opět provedeno pomocí příkazu `LEFT JOIN`.

Liší se zde však podmínky ovlivňující propojení. V pomocné tabulce jsou totiž uložené pouze informace o původních datech v čase změny. Pro rekonstrukci celého záznamu je tedy potřeba zjistit, jaká byla poslední změna daného atributu a její data napojit na požadovaný záznam. Pro zajištění této podmínky bylo potřeba implementovat další typ vnořených dotazů, které z pomocné tabulky vybírají pouze atribut `id`, který identifikuje požadovaný záznam v pomocné tabulce. Podmínky pro tento vnořený záznam zajišťují, že data ve formátu *JSON* vybraného záznamu obsahují identifikátor odpovídající požadovanému atributu z originální tabulky. Dále že má `id` odpovídající originálnímu záznamu a také, že hodnota `log_id` v hledaném záznamu je menší, nebo rovna hodnotě `log_id` záznamu, k němuž je připojován.

Pouze toto napojení však není dostačující k celkové rekonstrukci záznamu. Výše popsané vnořené dotazy totiž získají informace pouze pro atributy, které byly alespoň jednou upraveny. Zde jsou však potřeba také aktuální data záznamu, které jsou uloženy v originální tabulce. K tomu byla v hlavním dotazu implementována kontrola, která pro každý napojený atribut zjistí, jestli nabývá nějaké hodnoty. Je-li prázdný, vybere vnořeným dotazem aktuální data z originální tabulky a ty přiřadí prázdným atributům.

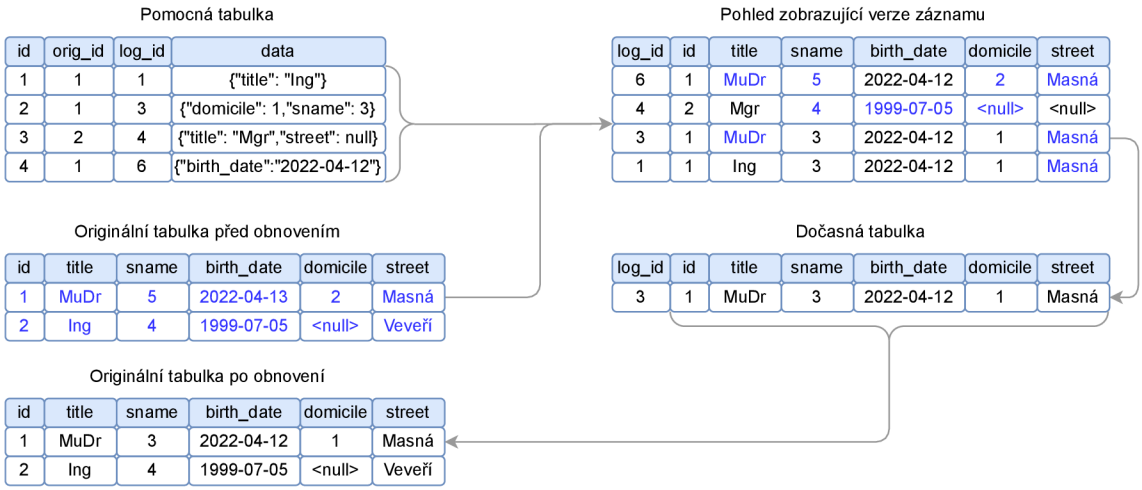
4.5 Procedury pro obnovení záznamu do zvoleného stavu

Při kontrole historie úprav záznamu může uživatel zjistit, že posledních pár změn bylo chybných a bude chtít obnovit záznam do některého z jeho předchozích stavů. Pro umožnění této funkcionality byly v databázi implementovány procedury, které obnovení provedou. Jelikož výše uvedené databázové prvky zálohují informace o provedených změnách pro různé typy originálních tabulek, bylo potřeba implementovat také více druhů procedur. Konkrétně se jedná o 3 druhy, které jsou v následujících odstavcích popsány. Na obrázku 4.3 je vyobrazen obecný postup práce s daty.

4.5.1 Procedury pro obecné tabulky

Procedury prvního druhu obnovují data v tabulkách, které nejsou propojovací a zároveň do nich není potřeba ukládat informace o uživateli, který obnovení provádí. Využity jsou pro tabulky `birthMarriage` a `person`. Všechny jejich atributy kromě primárního klíče je potřeba obnovit. Procedury tohoto typu disponují třemi parametry typu `IN`, tedy vstupních. Konkrétně se jedná o `record_id`, kterým se předává identifikátor záznamu, jenž má být obnoven. Druhý parametr nese název `logId`, pomocí kterého se specifikuje, do jakého stavu má být záznam obnoven. Posledním parametrem je `newLogId`, jenž předává identifikátor změny, který byl na straně serveru vygenerován pro identifikaci provedených změn při obnovení. Po zavolání procedury se jako první provede kontrola, zda-li jsou v pomocné tabulce uloženy relevantní informace o úpravách pro dané `logId`. K tomu se využívá operace `SELECT` nad pohledem jednotlivých verzí, který je blíže popsán v kapitole 4.4. Pokud byla nalezena verze záznamu, jejíž identifikátor změny je nižší nebo roven požadovanému stavu, pokračuje se s obnovením dat. Pokud nebyla nalezena odpovídající verze, procedura ukončí svou činnost, aniž by provedla jakoukoliv změnu. Dalším krokem je vytvoření dočasné tabulky na základě příkazu `SELECT` nad pohledem verzí, pomocí kterého se do ní nahrají data požadované verze. Z této dočasné tabulky se následně vyberou atributy obsahující cizí klíče, které odkazují na hlavní tabulky. Konkrétně `birth_id` pro tabulku `birthMarrige` či `birth_id`, `marriage_id` a `burial_id` pro tabulku s názvem `person`. Pokud některý z těchto atributů v dané verzi obsahuje hodnotu `null`, znamená to, že se má celý záznam vyprázdnit.

Všem atributům originální tabulky kromě primárního klíče a výše zmíněných cizích klíčů je pak přiřazena hodnota `null`. Pokud žádný z cizích klíčů není roven hodnotě `null`, přiřadí se atributům originální tabulky data z tabulky dočasné. K tomu je využit příkaz `UPDATE` volaný nad originální tabulkou s využitím spojení `INNER JOIN` s dočasnou tabulkou. Provedením operace `UPDATE` nad originální tabulkou se automaticky aktivuje spouštěč, který uloží informace o provedených změnách do pomocné tabulky. Následně je potřeba přiřadit provedeným úpravám identifikátor označující danou změnu. To je dosaženo příkazem `UPDATE` nad pomocnou tabulkou, kdy se atributu `log_id` přiřadí již zmíněný parametr `newLogId`.



Obrázek 4.3: **Diagram znázorňující postup obnovení záznamu.** V tomto příkladu probíhá obnovení záznamu v originální tabulce, jehož `id` je rovno jedné. Záznam je obnoven do stavu před editací označenou identifikátorem `log_id` s hodnotou tři.

4.5.2 Procedury pro hlavní tabulky

Procedury druhého typu obnovují data takzvaných hlavních tabulek, konkrétně tabulek s názvy `birth`, `marriage` a `burial`. Tyto tabulky ukládají informace o majiteli záznamu a také o zkušenostní úrovni posledního autora úprav. Z toho důvodu dané procedury kromě tří výše zmíněných základních parametrů disponují parametry `ownerId`, `lastEditedLevel` a `lastEditedHighestLevel`. Po spuštění proběhne stejně jako u prvního typu kontrola dostupných verzí záznamu. Je-li nalezena, následuje vytvoření pomocné tabulky, do které jsou nahrána data dané verze. Poté již neprobíhá kontrola atributů, ale rovnou samotné obnovení dat pomocí příkazu `UPDATE`. Ne všem atributům originální tabulky jsou však přiřazena data z tabulky dočasné. Atributu `owner` je přiřazena hodnota parametru `ownerId`. Atributu `last_edited_level` je přiřazena hodnota parametru `lastEditedLevel` a atributu `last_edited_highest_level` hodnota parametru `lastEditedHighestLevel`. Procedura opět končí přiřazením identifikátoru změny informacím o právě provedených úpravách v pomocné tabulce.

4.5.3 Procedury pro propojovací tabulky

Procedury třetího typu obnovují data v propojovacích tabulkách. Disponují třemi parametry stejnými jako první typ. Po zavolání procedury se opět provede kontrola dostupnosti záloh záznamu. Propojovací tabulky však nedisponují pohledy pro zobrazení jednotlivých verzí a z toho důvodu se dotaz provádí přímo nad pomocnou tabulkou. Následuje uložení aktuálního stavu do pomocné tabulky. Propojovací tabulky totiž nedisponují spouštěči pro uložení informací o provedených úpravách. Nad propojovacími tabulkami se volají pouze příkazy `INSERT` a `DELETE`, nikoliv však `UPDATE`. V rámci operace `SELECT` nad originální tabulkou je tedy vytvořena záloha aktuálního stavu ve formátu *JSON*. K tomu byly využity zanořené funkce `GROUP_CONCAT(JSON_OBJECT())`. Funkce `JSON_OBJECT` převádí zadané parametry do formátu *JSON*. Pro každý záznam v originální tabulce je vytvořen jeden objekt *JSON*. Tyto objekty jsou spojeny do jednoho textového řetězce pomocí funkce `GROUP_CONCAT`, která je navíc odděluje čárkou. Takto vytvořená záloha se následně uloží do pomocné tabulky společně s hodnotou parametru `newLogId`. Poté se vymažou všechny odpovídající záznamy z originální tabulky. Následuje iterace nad všemi objekty *JSON*, kterými daná verze disponuje. V každé iteraci jsou postupně pomocí funkcí `JSON_UNQUOTE(JSON_EXTRACT())` extrahovány jednotlivé atributy, které jsou následně ukládány do originální tabulky. Po dokončení iterování se procedura ukončí.

4.6 Skript pro generování MySQL kódu databázových prvků

Originální tabulky, pro něž byly implementovány výše popsané obslužné prvky, disponují velkým množstvím atributů. S rostoucím počtem atributů také dramaticky narůstá množství MySQL kódu pro vytvoření daných prvků. Z důvodu usnadnění implementace a také minimalizace překlepů byl napsán skript, který dokáže kód obslužných prvků generovat automaticky.

Jedná se o jednoduchý skript napsaný v jazyce PHP, který disponuje prostým webovým rozhraním v podobě formuláře. V něm je nutné zadat adresu databázového serveru a přihlašovací údaje uživatele. Následuje zadání názvu databáze a její tabulky, pro kterou mají být obslužné prvky vytvořeny. Na základě zadaných údajů se skript připojí k databázi, z níž získá informace o atributech zvolené originální tabulky. Následně vygeneruje MySQL kód pro vytvoření pomocné tabulky, implementaci spouštěče pro uložení změn, definici jednotlivých pohledů a také implementaci procedury sloužící k obnovení záznamu.

Skript však neumí generovat kompletně všechny druhy jednotlivých prvků. Generuje například pouze procedury pro obnovení záznamů v obecných tabulkách (popsaných v kapitole 4.5.1). Pro ostatní typy je potřeba vygenerovaný kód upravit. Není také zaručeno, že bude vygenerovaný kód bez chyb. Jedná se pouze o pomůcku při implementaci, u které však není zcela zaručena správnost generovaného kódu. Přesto byl skript velmi užitečný při implementaci této práce a v případě budoucího rozšíření systému DEMoS o podporu ukládání více typů záznamů má potenciál opět pomoct s implementací následujícím autorům.

Kapitola 5

Implementace webové části

V této kapitole je popsána implementace webové části této práce. V současné době systém ukládá tři typy záznamů, a to rodné, oddací a úmrtní. Implementace je pro jednotlivé typy záznamů podobná. V každé sekci této kapitoly je tedy vždy popsán hlavní princip implementované funkcionality pro rodné záznamy a pro zbylé typy jsou již popsány pouze zajímavé části, specifické pro daný typ.

5.1 Ukládání provedených změn

Jakmile uživatel ukončí úpravu rodného záznamu ve webovém formuláři, klikne na tlačítko uložit, a tím spustí funkci, která z informací ve formuláři vytvoří objekt na základě třídy `Birth`. Následně zavolá metodu `save` z manažeru, který obsluhuje daný typ záznamu. Této metodě je jako parametr předán vytvořený objekt. Daná metoda nejprve zkontroluje, zda-li je předaný parametr odpovídajícího typu a také jestli má uživatel dostatečné oprávnění pro úpravu záznamu. Nenarazí-li kontrola na problém, pokračuje příprava pro uložení. Data k uložení se databázi předávají v datové struktuře pole. Metoda tedy převádí daný objekt na pole jeho hodnot. Před započetím ukládání záznamu do databáze se nejdříve vloží informace o provedení změny do tabulky `log`, při kterém je na straně databáze vygenerován identifikátor dané změny, se kterým se v dalších částech pracuje jako s proměnnou s názvem `logId`. Informace jednoho záznamu je ukládána do více tabulek, z toho důvodu byla tato činnost rozdělena do několika privátních metod.

5.1.1 Ukládání jmen

Pro uložení jmen se využívá metoda `saveNamesHelper`, které se předává identifikátor záznamu, jemuž daná jména patří, seznam jmen k uložení, pohlaví, specifikace typu záznamu, kterému jsou jména přiřazena, a také `logId`. Metoda nejdříve zkontroluje, zda se všechna jména vyskytují v tabulce `name`, a případně je uloží jako nové záznamy. Poté vybere z propojovací tabulky současné záznamy, které se vztahují k dané osobě. Následuje porovnání současných a nových propojení záznamu se jmény. Zároveň se generuje záloha aktuálního stavu, která se ukládá do proměnné ve formátu *JSON*. Pokud bylo záznamu nějaké jméno odebráno, je odpovídající propojení jména s daným záznamem smazáno. Pokud naopak nové jméno přibylo, je jeho propojení přidáno. Došlo-li ke změně v propojovací tabulce, uloží se vytvořená záloha do pomocné tabulky pro uchování informací o provedených úpravách, jejíž popis je uveden v sekci 4.1. V případě, že nebyla provedena žádná změna je záloha smazána a nic se do pomocné tabulky neukládá.

5.1.2 Ukládání povolání

K ukládání povolání se využívá metoda `saveOccupationsHelper`. Té se v rámci parametrů předává seznam povolání k uložení, identifikátor záznamu, jemuž mají být povolání přiřazena, typ záznamu a také `logId`.

Metoda kontroluje, jestli se v tabulce `occupation` nachází záznamy všech povolání z předaného seznamu. Pokud některé povolání není obsaženo v tabulce, je mu nový záznam do tabulky uložen. Následuje výběr aktuálních propojení povolání se záznamem z propojovací tabulky. Pokud záznam doposud nedisponoval žádným povoláním, vytvoří se záloha ve formátu *JSON*, do které se tato informace uloží. Poté se porovnávají aktuální propojení s novými povoláními. Zároveň je také vytvářena záloha všech aktuálních propojení. Pokud bylo záznamu některé povolání odebráno, odpovídající propojení je smazáno. Pokud bylo povolání přidáno, je záznam o novém spojení uložen do propojovací tabulky. V případě, že byly při porovnávání nalezeny změny, je dříve vytvořená záloha uložena do pomocné tabulky. V opačném případě není potřeba zálohu ukládat.

5.1.3 Ukládání ostatních atributů

Struktura části databáze určené pro ukládání osob je podobná struktuře části pro ukládání svateb v rámci rodných záznamů. V tomto odstavci je tedy popsán princip ukládání obou typů zároveň. Pro ukládání osob byla implementovaná metoda `savePersons`. Pro ukládání svateb zase metoda s názvem `saveBirthMarriages`. V nich probíhá příprava dat k uložení do tabulek `person` či `birthMarriage`. Pro uložení jmen se volá výše popsaná metoda `saveNamesHelper`, které se specifikuje typ záznamu. K uložení povolání osob se zase využívá metoda `saveOccupationsHelper`. Následuje uložení příjmení, bydliště a dalších atributů záznamu do odpovídajících tabulek. Zde se však neukládají žádné informace o provedených změnách, jelikož jsou do těchto tabulek pouze vkládány nové záznamy. Nikdy se však neupravují, ani nemažou. Uložení informací o změně těchto atributů se tedy provádí v rámci tabulek `person` či `birthMarrige`. Nakonec se aktualizují data těchto tabulek. Uložení provedených změn má na starosti spouštěč v databázi. Nově uloženým informacím v pomocné tabulce je však následně potřeba přiřadit identifikátor změny. Záznamu s informacemi o právě provedených změnách se tedy přidá hodnota z proměnné `logId`.

Rodné záznamy obsahují také informace o konfirmaci. Údaje o osobě, která konfirmaci provedla, se však neukládají do části databáze určené pro osoby. Uchovává se totiž pouze jméno a příjmení. Pro uložení jmen se opět využívá metoda `saveNamesHelper`, které se pomocí parametrů specifikuje typ záznamu. V tomhle případě je využita propojovací tabulka s názvem `birth_confirm_name`. Nakonec se aktualizují data také v tabulkách `birth`, `marriage` nebo `burial`. Uložení informací o provedených změnách realizují příslušné spouštěče na straně databáze. Po aktualizaci dat je identifikátor změny (uložený v proměnné `logId`) přiřazen záznamu o právě provedených úpravách v pomocné tabulce.

5.2 Získání historie provedených změn

Systém vytvořený v rámci této práce umožňuje uživateli zobrazit historii úprav zvoleného záznamu. Pro získání těchto informací z databáze byly v manažerech jednotlivých typů záznamů implementovány metody s názvem `getHistory`. Tyto metody podle identifikátoru záznamu, který je metodě předán pomocí parametru, získají data z pomocných tabulek v databázi a vytvoří z nich objekty odpovídající typu záznamu.

Nejprve metoda vyhledá v tabulce s názvem `log` informace o autorech a časech jednotlivých úprav, a také jejich identifikátory (tzv. `logId`). Všechny tyto informace jsou uloženy ve vícerozměrném poli s názvem `changeLog`. Každý prvek tohoto pole tedy reprezentuje jednu editaci záznamu. Následuje iterace pro každý prvek pole. V rámci iterací se provádí většina níže popsaných úkonů. Pomocí databázového pohledu, zobrazujícího změny provedené nad některou z hlavních tabulek, metoda získá původní data upravených atributů před danou úpravou. Tyto informace následně převede na objekt třídy odpovídající danému záznamu. Poté získá informace o autorovi úpravy a přidá jej objektu jako autora.

5.2.1 Získání provedených změn informací o sňatcích v rámci rodných záznamu

Pro rodné záznamy se pokračuje získáním provedených změn záznamů o manželstvích narozeného. Každý rodný záznam může disponovat více záznamy o manželství. Metoda si uloží identifikátory záznamů z tabulky `birthMarriage` a poté pro každý záznam vybírá původní data z databázových pohledů. Nejdříve tedy získá informace pro tabulku `birthMarriage`. Následně vyhledá jména partnerů. Jak již bylo popsáno v předchozích sekcích, propojovací tabulky nedisponují databázovými pohledy, a proto metoda zpracovává zálohu stavu před úpravou záznamu získanou z pomocné tabulky `birthMarriage_name_h`. Data jsou z formátu *JSON* dekódována a uložena do struktury vícerozměrného pole. Pokud záloha obsahuje hodnotu `null`, znamená to, že záznam před úpravou nedisponoval jménem, a proto je místo jména záznamu přiřazen prázdný textový řetězec. V opačném případě metoda získá údaje pro každé jméno z tabulky `name` a přiřadí ho záznamu o manželství. Následují povolání partnerů. Stejně jako u jmen je z pomocné tabulky vybrána odpovídající záloha záznamů propojovací tabulky. Pro každé povolání jsou následně získána data z tabulky `occupation`, která jsou poté přiřazena záznamu o manželství. Pokračuje se získáním informací o příjmení z tabulky `surname`, a také bydliště z tabulky `domicile`. Tyto dva atributy jsou v tabulce `birthMarrige` zastoupeny atributy s cizími klíči. Úprava příjmení či bydliště je tedy reflektována změnou těchto atributů v tabulce. Pokud záznam před úpravou nedisponoval jedním z těchto atributů a po úpravě mu byl přidán, je mu v rámci této změny přiřazen prázdný textový řetězec.

5.2.2 Získání provedených změn informací o osobách

Metody všech tří typů záznamů pokračují získáním informací o úpravách záznamů osob. Jak již bylo popsáno v kapitole 3.2.1, hlavní tabulkou databázové části věnované osobám je tabulka s názvem `person`. Z té metoda vybere identifikátory osob a jejich vztah, kterými záznam aktuálně disponuje. Pro každou osobu jsou následně získána původní data z databázového pohledu pro zobrazení úprav tabulky `person`. Poté jsou dohledány původní jména osoby z pomocné tabulky obsahující zálohy z propojovací tabulky `person_name`. Záloha je ve formátu *JSON*, a proto jsou nejdříve tato data dekódována. Následně jsou pro každé jméno získány informace z tabulky `name`, které jsou poté přiřazeny dané osobě. Nachází-li se v záloze na pozici `name_id` hodnota `null`, znamená to, že osoba nabyla jména až danou úpravou. V tomto případě se místo jména osobě přiřadí prázdný textový řetězec. Podobným způsobem jsou dohledány informace o povoláních osob. Atributy příjmení, bydliště a vztah mezi osobami jsou podobně jako u záznamů o manželství, popsaných výše, propojeny se záznamy osob v tabulce person pomocí cizích klíčů. K získání verze dat před úpravou atributů tedy stačí vyhledat požadované záznamy v odpovídajících tabulkách.

Po získání všech informací o osobách je potřeba tyto data správně přiřadit k objektu s hlavním záznamem. K tomu bylo implementováno vícenásobné větvení, kde je porovnávanou hodnotou atribut z tabulky `person` s názvem `rel`. Ta určuje vztah osoby k danému záznamu. Každý typ záznamu disponuje alespoň jednou takzvanou hlavní osobou. U rodných záznamu je to narozená osoba. U oddacích záznamů jsou to ženich a nevěsta. Úmrtní záznamy mají jako hlavní osobu zemřelého. Pro hlavní osoby zmíněných typů se ukládá více informací než pro ostatní. Tyto nadbývající atributy však nejsou uloženy v tabulkách databázové části pro ukládání osob. Jsou ukládány do hlavních tabulek záznamů. Pokud se tedy ve větvení zjistí, že získaná data patří například novorozenému dítěti v rámci rodného záznamu, je potřeba k těmto informacím připojit zbylé atributy. V případě rodných záznamů jsou k novorozenci přidávány informace o konfirmaci. Ženichovi a nevěstě jsou připojeny atributy obsahující jejich rodné a případně také úmrtní adresy. K tomu rovněž jejich věk. Zemřelému se v rámci úmrtních záznamů přidává rodná a úmrtní adresa. Jednotlivé adresy jsou uloženy v tabulce `address` a získaní původní adresy funguje stejně jako u příjmení či bydliště. Mění se pouze cizí klíče v hlavních tabulkách, nikoliv však samotné záznamy v tabulce `address`. Jakmile byly hlavním aktérům záznamů přiřazeny všechny potřebné informace, jsou přiřazeny k hlavnímu objektu daného záznamu stejně jako všechny ostatní osoby.

Přiřazením všech osob objektu je daný objekt kompletní. Následuje kontrola, jestli objekt není prázdný, což by mohlo nastat, pokud by uživatel omylem klikl na tlačítko uložit, aniž by provedl jakoukoliv úpravu. Pokud objekt obsahuje alespoň jednu informaci o provedené úpravě, je uložen do pole společně s identifikátorem změny a také časem změny. Takto naplněné pole metoda `getHistory` vrací a tím ukončuje svou činnost.

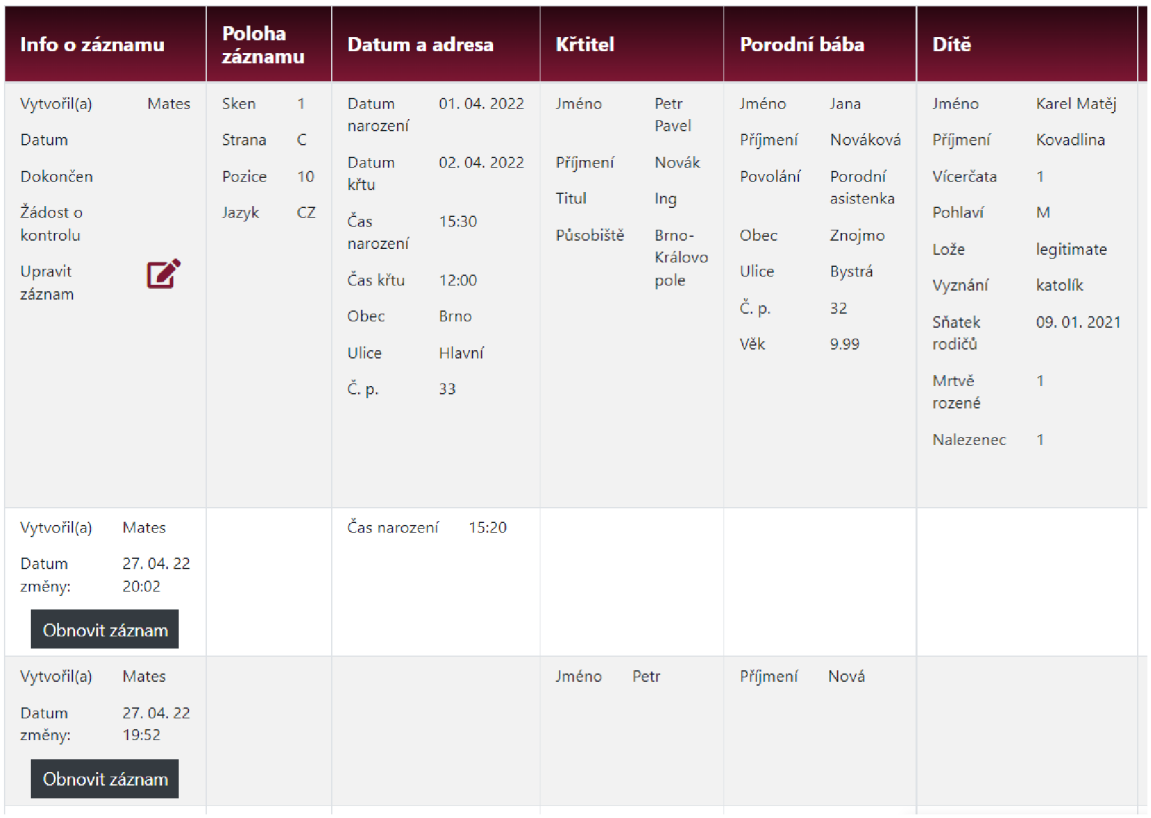
5.3 Zobrazení historie provedených změn

Jak již bylo popsáno v návrhu zobrazení v kapitole 3.3.3, jedním z vedlejších úkolů této práce bylo přepracovat zobrazení záznamů v dané matrice. Z tohoto důvodu byla implementována nová šablona. V této šabloně se nachází tabulka, jejíž sloupce označují určité logické části záznamu. Mohou to být například jednotlivé osoby reprezentující příbuzné narozeného a další. Pro každý typ záznamu byla implementována jedna taková šablona. Nejprve je vykresleno záhlaví tabulky, které obsahuje názvy zmíněných logických částí. Následuje vykreslování jednotlivých řádků. Toho je dosaženo pomocí iterování nad poli objektů reprezentujících dané záznamy. Každá buňka těchto řádků obsahuje vnořenou tabulku. Tato tabulka sestává ze dvou sloupců. Počet řádků je závislý na počtu atributů v dané části. V levém sloupci se nachází názvy jednotlivých atributů. V pravé části jsou doplňovány hodnoty záznamů získané z jejich objektů. V prvním sloupci hlavní tabulky jsou vždy zobrazeny základní informace o záznamu, jako například jeho autor. Dále jsou zde vykreslena tlačítka pro editaci záznamu a pro zobrazení historie záznamu. Pokud uživatel nemá dostatečná práva pro editaci, tlačítko je zašedlé, nedá se na něj kliknout, a ani ve zdrojovém kódu nemá přidělen odkaz pro editaci. Po najetí kurzorem na tlačítko se vypíše nápověda oznamující uživateli, že nemá pro danou operaci dostatečná práva. Zobrazit historii záznamu je povoleno všem uživatelům nezávisle na jejich zkušenostní úrovni. Tlačítko je proto vždy přístupné.

Jakmile uživatel klikne na tlačítko pro zobrazení historie daného záznamu, je přesměrován na odkaz složený z prezentéru, jeho metody, která se má využít, a následně identifikátoru záznamu. Pro tuto práci je využíván pouze prezentér s názvem `RegistryPresenter`, nebo českým překladem `matriky`. V něm se nachází metody pro vykreslení různých zob-

razení. Názvy těchto metod vždy začínají klíčovým slovem `render`. Metoda pro zobrazení historie rodného záznamu má například název `renderBirthRecordHistory`. Pro ostatní typy záznamu je pouze zaměněno slovo `Birth` za `Marriage` pro oddací, nebo `Death` pro úmrtní záznamy. Tyto metody získají informace o matrice pro zobrazení navigace v horní části stránky. Dále získá údaje o aktuálním záznamu a také informace o historii úprav záznamu pomocí výše zmíněné funkce `getHistory`. Jakmile jsou tato data uložena v paměti, framework Nette je předá šablonám.

Zobrazení historie úprav vychází ze zobrazení jednotlivých záznamů v matrice. Opět se tedy jedná o tabulku, jejíž sloupce reprezentují logické části záznamu. Jak je možné vidět na obrázku 5.1, řádky tabulky nereprezentují jednotlivé záznamy matriky, ale jednotlivé úpravy provedené daným autorem v daném čase. V prvním řádku tabulky je zobrazen aktuální stav všech atributů záznamu. V dalších jsou zobrazeny pouze atributy, které byly upraveny. Konkrétně vždy stav dat atributů před vykonáním úpravy. Úpravy jsou řazeny chronologicky od nejnovější po nejstarší. Uživatel má tedy přehled, jaké informace nabýval daný argument v daném čase. V prvním sloupci řádku s úpravami se opět nachází základní informace. V tomto případě se jedná o autora úprav a čas jejich provedení. Dále je zde zobrazeno tlačítko pro obnovení záznamu do stavu, který daný řádek reprezentuje. Pro zobrazení historie byla vytvořena šablona, která generuje pouze jednotlivé řádky s úpravami záznamu. Tato šablona je vložena do dříve popsané šablony pro zobrazení jednotlivých záznamů a aktivuje se pouze tehdy, když má k dispozici data o úpravách.



Obrázek 5.1: **Ukázka zobrazení historie provedených úprav.** Snímek obrazovky zachycuje část zobrazení historie úprav rodného záznamu.

5.4 Obnovení záznamu do některé z předchozích verzí

V zobrazení historie úprav záznamu, které je propsáno v předchozí sekci, disponuje každý řádek tlačítkem s názvem obnovit záznam. Jakmile chce uživatel obnovit záznam do stavu před danou editací, klikne na odpovídající tlačítko. Každé z nich má přiřazený odkaz, který je složen z prezentéru, jeho metody, identifikátoru úpravy a identifikátoru záznamu. Stejně jako u zobrazení historie úprav je zde využit prezentér s názvem `RegistryPresenter`, česky `matriky`. Metody v české verzi odkazů mají název `obnovit-verzi-zaznamu-`, který je doplněn typem záznamu. V kódu začínají názvy těchto metod klíčovým slovem `action`, což ve frameworku Nette znamená, že daná metoda pouze provádí nějakou akci, ale nevykresluje žádný obsah a nevyužívá tedy šablony. Tyto metody volají funkci `restoreRecordState` odpovídajícího manažeru podle typu záznamu.

Metoda `restoreRecordState` se stará o obnovení záznamu do zvoleného stavu. Aby mohla proběhnout kontrola oprávnění uživatele pro obnovení záznamu, získá aktuální informace z hlavní tabulky daného záznamu. Pokud uživatel disponuje oprávněním pro obnovení, aktualizuje se hodnota požadované zkušenostní úrovně pro úpravu záznamu podle uživatele, který provádí obnovení. Poté se v tabulce `log` uloží informace o nastávající úpravě záznamu. Obnovení dat v jednotlivých tabulkách je prováděno pomocí procedur uložených v databázi, jejichž popis se nachází v kapitole 4.5. Metoda tedy pouze postupně volá tyto procedury a předává jim požadované parametry.

Po obnovení dat ve všech tabulkách ukládajících informace o daném záznamu metoda `restoreRecordState` skončí. Prezentér pak uživatele přesměruje na zobrazení historie úprav obnoveného záznamu. Samotné obnovení je v historii úprav znázorněno stejně jako by uživatel přepsal data daných atributů ručně.

Kapitola 6

Testování

Po dokončení implementace této práce bylo potřeba otestovat její funkčnost. Testování probíhalo na dvou úrovních. První bylo prováděno automaticky pomocí napsaných testů a druhé bylo prováděno ručně podle stanovených scénářů.

6.1 Testování s pomocí nástroje Tester

Nástroj *Tester* je, jak již bylo zmíněno v kapitole 2.3.2, součástí frameworku Nette. Umožňuje jednoduché vytváření testů, které lze následně hromadně nebo jednotlivě spouštět. Každý test si před samotným vykonáním testovaných operací připraví systém do stavu potřebného pro testování.

Testy, které byly implementovány v rámci této práce, jsou převážně integrační a pracují také s databází. Nebylo však žádoucí, aby testy pracovaly s originální databází obsahující informace reálných matričních záznamů. Tato data by mohla být chybou v testu či v systému ztracena. Z toho důvodu testy pracují s testovací databází, jejíž struktura je kopií originální databáze. Obsahuje ovšem pouze malou část uložených dat potřebných pro testování.

6.1.1 Přípravné rutiny testů

Pro připojení systému k testovací databázi byl implementován prvek typu *trait* s názvem `ConnectionHelper`. Jazyk PHP třídám umožňuje dědit pouze z jednoho rodiče, což nebylo pro třídy s testy dostačující. Aby se v každé testovací třídě nemusely implementovat metody pro připojení k databázi zvlášť, byl využit právě tento prvek. *Trait* je možné definovat jako sloučeninu metod, kterými lze rozšířit funkčnost dané třídy bez nutnosti vícenásobného dědění. Ve zmíněném traitu byla implementována metoda s názvem `getConnection`, která zajistí napojení na testovací databázi a nahraje do ní předpřipravená testovací data ze souboru `testing_data.sql`. Další metoda s názvem `getContext` vytvoří databázový kontext, který systému umožní pracovat s databází.

Kromě připojení k databázi jsou zde také implementovány metody `getBirthManager`, `getMarriageManager` a `getDeathManager`. Ty zajišťují vytvoření instancí manažerů pro správu záznamů. Konstruktory manažerů vyžadují předání objektů tříd `EntityManager` a také `User` obsahující informace o přihlášeném uživateli. Jelikož jsou však tyto manažery vytvářeny v rámci testování, není v sytému přihlášen žádný uživatel a celkově není funkční vkládání závislostí (*dependency injection*) frameworku Nette. Z toho důvodu bylo potřeba vytvořit takzvané *simulované objekty* (*Mock objects*). Těmto objektům lze nastavit různé vlastnosti, jako například návratové hodnoty pro různé metody objektu a další. K tomu

byl využit nástroj s názvem *Mockery*[1]. Po vytvoření všech potřebných objektů již lze získat instance tříd manažerů, které jsou využívány v testech.

Pro přípravu dat, simulujících zadaní informací o záznamu do formuláře, byly vytvořeny metody v dalším prvku trait s názvem `DataHelper`. Zde jsou uložena data v poli ve formátu jako kdyby byla právě získána z formuláře pro ukládání či editaci záznamu. Samotná data byla opravdu při implementaci získána z formuláře, takže pole disponuje všemi prvky jako v reálném provozu. Pro účely testování však samotné formulářové prvky využity nejsou. Tyto atributy jsou následně v odpovídajících metodách převáděny na objekty reprezentující dané záznamy, či jinak upravovány pro účely testování.

6.1.2 Implementace testů

Tato práce byla implementována pro tři typy matričních záznamů, jejichž data jsou zpracovávána třemi manažery, z toho důvodu byly také implementovány tři třídy s testy. Každý typ záznamu má tedy svou testovací třídu. V každé třídě jsou implementovány metody, které simulují jednotlivé případy užití práce s matričními daty v systému DEMoS.

Jako první se testuje uložení nového záznamu. Metoda získá odpovídající manažer a volá jeho funkci `save`. V parametru mu předává objekt daného záznamu získaný z výše popsaného traitu. V případě, že uložení proběhlo bez problému, vrací metoda `save` daný záznam v podobě řádku tabulky v databázi. Z něj následně metoda `toBirthModel` (pro jiný typ záznamu je slovo `Birth` nahrazené daným typem) opět vytvoří objekt reprezentující záznam. Ten je poté převeden metodou `getFormValues` do pole atributů, které by bylo použito pro zobrazení záznamu ve formuláři s editací. Data uložená v získaném poli jsou následně pomocí funkce *Testeru* `ASSERT::equal` porovnávána s původními daty. Toto porovnávání probíhá v rámci iterování nad získaným polem. Atributy uchovávající identifikátory záznamu v jednotlivých tabulkách však testovány nejsou, jelikož nabyly hodnot až během ukládání.

Další test simuluje editaci záznamu. Opět se využívají data k uložení ve stavu jako by byly získány z formuláře. Následně jsou data některých atributů upravena. Poté se opět volá metoda `save` z manažeru a následuje porovnání atributů záznamu v databázi s upraveným záznamem k uložení. Následující test simuluje zobrazení historie provedených úprav záznamu. Nejprve je provedeno několik úprav záznamu pomocí funkce `save`. Všechny stavy záznamu jsou ukládány do proměnné pro následující kontrolu. Pro získání historie je volána funkce `getHistory` z manažeru. Ta vrací pole verzí záznamů. Každá verze je poté opět porovnávána s původními daty.

Poslední testovanou funkcionalitou je obnovení záznamu do některého z předchozích stavů. Jako první je vytvořena historie úprav záznamu stejným způsobem jako při testování editace. Následně je volána metoda `restoreRecordState`, která záznam obnoví do předchozího stavu. Poté se pomocí metody `get` získá aktuální stav záznamu, jenž je uložen pro následující kontrolu. Takto je obnovení záznamu provedeno několikrát. Nakonec proběhne porovnávání atributů záznamů v jednotlivých stavech.

Takovýmto způsobem byly implementovány testy pro všechny tři typy matričních záznamů. K porovnávání je také přidán komentář označující testovanou funkci, případnou verzi záznamu a název atributu, jehož data jsou kontrolována. To napomáhá přehlednosti výpisu testu a k urychlení identifikace případného problému.

[1] Dokumentace k nástroji Mockery: http://docs.mockery.io

6.2 Ruční testování

Kromě implementace automatických testů byl vytvořený systém testován také ručně. Toto testování bylo převážně zaměřeno na jednotlivá zobrazení, jelikož automatické testování vygenerovaných HTML prvků webové stránky by bylo značně zdlouhavé. Testovány tedy byly formuláře pro ukládání a editaci matričních záznamů, zobrazení seznamu záznamů dané matriky a také zobrazení historie daného záznamu.

Formuláře pro vkládání a editaci záznamů již byly implementovány v rámci původního systému DEMoS. Během analýzy a testování systému před započetím samotné implementace této práce však byly zjištěny závažné nedostatky, jak je blíže popsáno v kapitole 3.3.2. Z toho důvodu byly i tyto formuláře v rámci zprovoznění systému upraveny a bylo tedy nutné otestovat také jejich funkčnost. Samotné testování spočívalo v postupném vyplňování jednotlivých políček formuláře, následováno kontrolou, zdali byla daná informace uložena v odpovídajícím formátu na správném místě v databázi. Kontrolovány byly také jednotlivé typy formulářových políček, jako například typ `date` pro zadání data.

Při testování zobrazení seznamu matričních záznamů či zobrazení historie daného záznamu byl kladen důraz na správné navázání dat jednotlivých atributů získaných z databáze k odpovídajícím názvům ve webové tabulce zobrazení. Kontrolován byl také správný formát vypsaných dat. Kromě toho byla také testována správnost přiřazených odkazů tlačítkům, či zdali správně funguje kontrola oprávnění uživatelů a jí odpovídající chování ovládacích prvků.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo upravit systém pro správu matričních dat (DEMoS) tak, aby uživatelům umožňoval spravovat historii revizí záznamů. Výsledkem je rozšíření systému DEMoS o systém správy verzí, který automaticky ukládá informace o provedených úpravách záznamů, umožňuje uživateli zobrazit historii úprav daného záznamu a také záznam obnovit do některého z jeho předchozích stavů. Návrh úprav původního systému vznikl na základě nastudovaných informací o genealogii, tvorbě rodokmenů a také analýzou a otestováním současného stavu systému DEMoS. Jelikož byly při analýze a testování nalezeny chyby, které by znemožnily funkčnost úprav v rámci této práce, bylo potřeba přepracovat část původního systému a tím ho zprovoznit. Samotný systém pro správu verzí ukládá kromě provedených změn také jejich autory a čas provedení. Kontroluje také oprávnění uživatelů, čímž znemožní obnovu záznamu neoprávněných osobám. Pro zobrazení historie záznamu byl navržen a implementován nový pohled webového uživatelského rozhraní. Po dokončení implementace byl celý systém otestován automatickými testy a také ručně s využitím dat v databázi DEMoS.

Implementace této práce se vztahuje na část systému DEMoS, která zajišťuje přepisování a správu matričních záznamů. K dispozici jsou však také další historické prameny, ze kterých by se daly čerpat informace. V budoucnu by se tedy mohl celý systém rozšířit o podporu více druhů záznamů, což by vyžadovalo implementaci nových tabulek v databázi. Zároveň by měly být vytvořeny obslužné databázové prvky pro systém správy verzí po vzoru této práce, k čemuž mohou budoucí autoři využít vytvořený skript pro automatické generování MySQL kódu těchto prvků. Celý systém v současné době nepodporuje nejnovější verzi PHP. Bylo by tedy vhodné aktualizovat framework Nette, a tím zajistit kompatibilitu s nejnovějšími technologiemi.

Literatura

[1] Chacon, S. a Straub, B. *Pro Git*. 2. vyd. Apress, 2014. ISBN 14-8420-077-2.

[2] Chetia, A. *Different types of MySQL Triggers (with examples)* [online]. GeeksforGeeks, 4. července 2019 [cit. 2022-05-05]. Dostupné z: https://www.geeksforgeeks.org/different-types-of-mysql-triggers-with-examples/.

[3] Čápka, D. *MVC architektura* [online]. itnetwork.cz, 2. září 2020 [cit. 2022-05-05]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.

[4] *Digitalizace v archivech* [online]. Česká genealogická a heraldická společnost v Praze, květen 2022 [cit. 2022-05-05]. Dostupné z: http://www.genealogie.cz/aktivity/digitalizace/.

[5] *Genealogy* [online]. Wikipedia, The Free Encyclopedia., 7. května 2022 [cit. 2022-05-09]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Genealogy&oldid=1086666992.

[6] *Introducing JSON* [online]. json.org, květen 2022 [cit. 2022-05-05]. Dostupné z: https://www.json.org/json-en.html.

[7] *Matrika* [online]. Wikipedia, The Free Encyclopedia., 27. března 2022 [cit. 2022-05-05]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Matrika&oldid=21080458.

[8] *Nette – Pohodlný a bezpečný vývoj webových aplikací v PHP* [online]. Nette Foundation, květen 2022 [cit. 2022-05-05]. Dostupné z: https://nette.org/cs/.

[9] *Nette Tester – pohodové testování v PHP* [online]. Nette Foundation, květen 2022 [cit. 2022-05-05]. Dostupné z: https://tester.nette.org/.

[10] *O Nette* [online]. Nette Foundation, květen 2022 [cit. 2022-05-05]. Dostupné z: https://nette.org/cs/about.

[11] *Rozrod* [online]. Wikipedia, The Free Encyclopedia., 10. září 2021 [cit. 2022-05-05]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Rozrod&oldid=20455166.

[12] *Tracy – ladicí nástroj se kterým je radost chybovat* [online]. Nette Foundation, květen 2022 [cit. 2022-05-05]. Dostupné z: https://tracy.nette.org/.

[13] *Úvodní stránka blogu* [online]. Nette Foundation, květen 2022 [cit. 2022-05-05]. Dostupné z: https://doc.nette.org/cs/quickstart/home-page.

[14] *Vývod (genealogie)* [online]. Wikipedia, The Free Encyclopedia., 10. září 2021 [cit. 2022-05-05]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=V%C3%BDvod_(genealogie)&oldid=20454955.

[15] Wankhade, A. *What is Stored Procedures in SQL?* [online]. GeeksforGeeks, 20. května 2021 [cit. 2022-05-05]. Dostupné z: https://www.geeksforgeeks.org/what-is-stored-procedures-in-sql/.

[16] Wenzel, K. *What is a Database Table?* [online]. Easy Computer Academy, LLC, 15. března 2022 [cit. 2022-05-05]. Dostupné z: https://www.essentialsql.com/what-is-a-database-table/.

[17] Wenzel, K. *What is a Relational Database View?* [online]. Easy Computer Academy, LLC, 15. března 2022 [cit. 2022-05-05]. Dostupné z: https://www.essentialsql.com/what-is-a-relational-database-view/.

[18] *What Is a Database?* [online]. Oracle, květen 2022 [cit. 2022-05-05]. Dostupné z: https://www.oracle.com/database/what-is-database/.

[19] *Začínáme s Latte* [online]. Nette Foundation, květen 2022 [cit. 2022-05-05]. Dostupné z: https://latte.nette.org/cs/guide.