FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Systém přepisu matričních knih formou crowd-sourcing
Student:	Bc. Filip Nezbeda
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Cílem práce je analýza, návrh a 'proof-of-concept' systému, který by formou crowd-sourcingu umožňoval zájemcům přepisovat a sdílet záznamy z matričních knih. Systém bude sloužit primárně uživatelům, kteří se zajímají o genalogii. Hlavním výsledkem práce budou kompletní podklady pro implementaci systému, na které budou moci navázat bakalářské práce.
Postupujte v těchto krocích:
- proveďte stručnou rešerši existujících systémů, jsou-li k dispozici
- poveďte analýzu potřeb, uživatelských rolí a procesů
- zpracujte návrh systému včetně návrhu způsobu provozu a škálování
- na základě návrhu realizujte proof-of-concept řešení.

Elektronicky schválil/a Ing. David Buchtela, Ph.D. dne 23. ledna 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Systém přepisu matričních knih formou crowd-sourcing

Bc. Filip Nezbeda

Katedra softwarového inženýrství
Vedoucí práce: Ing. Michal Valenta, Ph.D.

29. dubna 2021

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce Ing. Michalu Valentovi, Ph.D. za cenné rady a připomínky při psaní práce a zadavatelce Štěpánce Chládkové za zajímavé téma a odborné konzultace. Dále bych chtěl poděkovat rodině a přítelkyni za podporu během psaní práce a během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 29. dubna 2021 . . . . . .

Odkaz na tuto práci

Nezbeda, Filip. *Systém přepisu matričních knih formou crowd-sourcing*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Cílem této diplomové práce je analýza, návrh a proof-of-concept systému, který bude formou crowd-sourcing umožňovat zájemcům o genealogii přepisovat matriční knihy. Dalším cílem je pak rešerše již existujících řešení. Hlavním výsledkem práce je analýza domény a podklady pro implementaci systému. Dále je vytvořeno proof-of-concept řešení, které ukazuje, že je návrh systému validní.

Klíčová slova genealogie, matriční kniha, návrh, informační systém, crowd-sourcing, proof-of-concept

Abstract

The goal of this diploma thesis is the analysis, design, and proof-of-concept of a crowd-sourcing system that allows people interested in genealogy to transcript registry books. The next goal is to review existing alternatives. The main result of this thesis is the analysis of the domain and the basis for implementation. Furthermore, a proof-of-concept solution, which shows that the system design is valid, is created.

Keywords genealogy, registry book, design, information system, crowd-sourcing, proof-of-concept

Obsah

Úvod **1**

1 Cíl práce **3**

2 Literární rešerše **5**
2.1 Genealogie 5
2.2 Matriky 6
2.2.1 Historický vývoj 6
2.2.2 Současnost 6
2.3 Požadavky 7
2.4 Případy užití 8
2.5 Doménový model 9
2.6 Procesy 10
2.6.1 BPMN 10
2.7 Návrh systému 12
2.8 Databáze 13
2.8.1 Nerelační 13
2.8.2 Relační 14
2.9 Proof-of-concept 15
2.10 Crowd-sourcing 15
2.11 Existující řešení 15
2.11.1 Ancestry 15
2.11.2 MyHeritage 16
2.11.3 FamilySearch 17
2.11.4 Gramps 17
2.11.5 Webtrees 17

3 Praktická část **19**
3.1 Analýza 19

3.1.1 Analýza požadavků . . . 19
3.1.2 Případy užití . . . 21
3.1.3 Analýza domény . . . 35
3.1.3.1 Doménový model . . . 36
3.1.3.2 Procesy . . . 39
3.1.3.3 Možnosti do budoucna . . . 44
3.2 Návrh . . . 44
3.2.1 Návrh databáze . . . 44
3.2.1.1 Volba databázové technologie . . . 44
3.2.1.2 Volba databázového systému . . . 45
3.2.1.3 Databázový model . . . 45
3.2.2 Model tříd . . . 48
3.2.3 Návrh architektury . . . 50
3.2.4 Provoz a škálování . . . 52
3.2.5 Návrhy obrazovek . . . 52
3.2.6 Manažerské zhodnocení . . . 55
3.3 Proof-of-concept řešení . . . 57
3.3.1 Popis implementace . . . 57
3.3.2 Implementované části . . . 59
3.3.3 Zhodnocení . . . 60

Závěr **61**

Bibliografie **63**

A Seznam použitých zkratek **67**

B Detailní diagramy systému **69**

C Instalační příručka POC řešení **83**
C.1 Lokální spuštění . . . 83
C.1.1 Databáze . . . 83
C.1.2 Aplikace . . . 83

D Obrazovky POC řešení **85**

E Obsah přiloženého CD **93**

Seznam obrázků

2.1 Druhy brán 11
2.2 Typy úloh 11
2.3 Typy aktivit 11
2.4 Označení aktivit 11
2.5 Kontrola toku 12
2.6 Swimlanes 12

3.1 Diagram případů užití 22
3.2 Doménový model 38
3.3 Vytvoření záznamu 40
3.4 Hledání záznamu 40
3.5 Kontrola záznamu 41
3.6 Revize záznamu 41
3.7 Identifikace osoby 42
3.8 Identifikace osoby 43
3.9 Správa uživatele 43
3.10 Databázový model 47
3.11 Model tříd 49
3.12 Architektura informačního systému 50
3.13 DAO rozhraní 51
3.14 Model nasazení systému 52
3.15 Úvodní obrazovka 53
3.16 Hledání matriky 53
3.17 Výsledek hledání matriky 54
3.18 Detail matriky 54
3.19 Rozcestník hledání záznamu 54
3.20 Hledání záznamu 55
3.21 Výsledek hledání záznamu 55
3.22 Adresářová struktura systému 58
3.23 Generický repozitář 59

B.1 Model nasazení systému 69
B.2 Architektura systému 70
B.3 Datová vrstva 71
B.4 DAO 71
B.5 DAO – interface 72
B.6 DAO – implementace 73
B.7 Vrstva obchodní logiky 74
B.8 Modely obchodní logiky 74
B.9 Interface obchodní logiky 75
B.10 Implementace obchodní logiky 76
B.11 Implementace obchodní logiky 76
B.12 Prezentační vrstva 77
B.13 Controllery prezentační vrstvy 78
B.14 Obrazovky prezentační vrstvy 78
B.15 Modely prezentační vrstvy 79
B.16 Modely přihlašování 79
B.17 Modely matriky 80
B.18 Modely osoby 80
B.19 Modely záznamů 81

D.1 Domovská obrazovka 85
D.2 Přihlašovací obrazovka 86
D.3 Vytvořit záznam 86
D.4 Vytvořit záznam úmrtí 87
D.5 Hledání záznamu 87
D.6 Hledat narození 88
D.7 Hledat sňatek 88
D.8 Hledat úmrtí 89
D.9 Hledat matriku 89
D.10 Výsledek hledání matriky 90
D.11 Hledat osobu 90
D.12 Detail osoby 91
D.13 Rodokmen osoby 91
D.14 Přehled uživatelů 92
D.15 Detail uživatele 92

Seznam tabulek

3.1 Funkční požadavky 20
3.2 Nefunkční požadavky 21
3.3 U1 Registrace 23
3.4 U2 Vytvoření záznamu 24
3.5 U3 Hledání matriky 25
3.6 U4 Hledání záznamu 25
3.7 U5 Kontrola záznamu 26
3.8 U6 Revize záznamu 26
3.9 U7 Identifikace osoby 27
3.10 U8 Hledání osoby 28
3.11 U9 Generování rodokmenu 28
3.12 U10 Úprava záznamu 29
3.13 U11 Smazání záznamu 29
3.14 U12 Založení matriky 30
3.15 U13 Úprava matriky 30
3.16 U14 Smazání matriky 31
3.17 U15 Zobrazení uživatele 32
3.18 U16 Smazání záznamů uživatele 33
3.19 U17 Zablokování uživatele 33
3.20 U18 Úprava uživatele 34
3.21 Mapování požadavků na případy užití 34
3.22 Propočet nákladů 56
3.23 Propočet výnosů 57

Úvod

V současné době jsou veřejně dostupné naskenované stránky matrik z celé České republiky, avšak neexistuje k nim žádný elektronický přepis. Tyto matriky jsou navíc roztříštěné na několika různých webových stránkách v různých formátech. Pokud si někdo přeje vyhledat informace o např. svém předkovi a zná pouze jeho jméno, nezbývá mu nic jiného než najít konkrétní matriku a stránku po stránce procházet jednotlivé záznamy. Jelikož matrik jsou desetitisíce a každá obsahuje desetitisíce záznamů existuje velmi malá šance, že svého předka najde. Zároveň však existuje řada lidí, kteří se zajímají o genealogii a některé matriky již začali přepisovat do elektronické podoby (nejčastěji tabulky), a ty poskytly k dispozici ostatním.

Zadavatelka Štěpánka Chládková tak přišla s nápadem vytvořit systém, který by zveřejnil seznam matrik, které jsou k dispozici a dále umožnil, aby dobrovolníci mohli přepisovat ručně psané záznamy do digitální podoby. Systém by tak nejen sjednotil formáty a místo, kde se matriky nacházejí, ale poskytl by veřejnosti nástroj pro tvorbu elektronického přepisu matričních záznamů. Veřejnost by se tak mohla podílet na jejich digitalizaci a mohla by také nahlížet a vyhledávat v dostupných matrikách a přepisech.

Mým úkolem v této diplomové práci je seznámit krátce čtenáře s doménou matrik a matričních záznamů a provést rešerši existujících systémů, jsou-li k dispozici. Dále provedu analýzu potřeb, uživatelských rolí a procesů které se v systému budou vyskytovat. Následně vytvořím návrh systému, ke kterému navrhnu i způsob provozu a možnosti budoucího škálování. Výsledkem toho pak budou kompletní podklady pro budoucí implementaci systému. Na závěr pak realizuji proof-of-concept (POC) řešení, které prokáže celkovou proveditelnost.

Kapitola 1

Cíl práce

Cílem této diplomové práce je analýza, návrh a POC systému, který bude umožnovat uživatelům formou crowd-sourcing přepisovat záznamy z matričních knih. Systém bude sloužit převážně uživatelům, kteří se zajímají o genalogii. Hlavním výsledkem práce budou kompletní podklady pro implementaci systému, na které poté budou moci navázat bakalářské práce.

Prvním cílem práce je literární rešerše, ta seznámí čtenáře s problematikou, která se bude probírat v navazujících částech práce. Dalším cílem je analýza současných řešení crowd-sourcing systémů a systémů, které umožňují přepisovat matriční záznamy. Následující část se poté bude zabývat analýzou domény a požadavků na systém, na které poté naváží případy užití. Dále budou zmapovány procesy vyskytující se v systému, ze kterých poté bude vycházet doménový model. Na základě doménového modelu pak bude vypracován návrh databáze odpovídající požadavkům na systém. V práci bude také popsán návrhový model tříd a celkový návrh architektury systému. V poslední části práce bude POC řešení ověřující, zda je návrh systému validní.

Kapitola 2

Literární rešerše

2.1 Genealogie

Genealogie je historická věda zabývající se studií vztahů mezi lidskými jedinci vyplývající z jejich rodového původu. Genealogie je univerzálním fenoménem vyskytující se v různých formátech od jednoduchých po velmi složité napříč všemi národy a obdobími. Vývoj probíhal ve 3 fázích

- mluvenená tradice,
- evidence některých rodů,
- evidence všech osob.

Poslední ze zmíněných fází nastala zhruba v 16. století v západní Evropě, kdy se genealogie rozvinula do takové podoby, že jsme nyní schopni dohledat své předky [1].

V současné době je zájem o genealogii zejména ze strany široké veřejnosti hlavně kvůli sledování svého rodinného původu. Profesionální genealog se nezabývá jednou rodinou, ale mnoha a dále zásadami genealogického výzkumu, které vyplývají z rozsáhlého studia. Vzhledem k tomu, že existuje malé množství univerzit, které toto téma vyučují, jsou genealogové z velké části samouci. Vztahy mezi osobami lze zaznamenat pomocí rodokmenu [1].

Rodokmen v lidské genetice slouží ke grafickému zachycení vztahů mezi jedinci. Muž se zachycuje pomocí čtverce, nebo symbolu „♂“ a žena je pak znázorněna pomocí kolečka, nebo symbolu „♀“. Svazek osob je pak reprezentován pomocí horizontální čáry, která osoby propojuje. Potomci jsou zakresleni kolmou čarou na řaru svazku a jsou řazeni zleva do prava, dle data narození [2].

2.2 Matriky

2.2.1 Historický vývoj

Pro genealogické bádání jsou hlavním zdrojem matriky, které obsahují matriční záznamy. První zmínka o povinnosti vést matriky je z roku 1212, kde měly být evidovány údaje o narození. Není však známo, že by se podle tohoto nařízení skutečně vedly. Nejstarší matriky na našem území se vyskytují od roku 1531 a vedli je němečtí luteráni v Jáchymově. Další matriky z tohoto období pochází ze severozápadních Čech. Od roku 1563 připadala povinnost vést matriky na fary a kněze nicméně velká část z nich se ztratila či shořela [3, 4].

Matriky byly poprvé uznány veřejnými listinami v roce 1784, ale roku 1949 byly předány národním výborům a současně s tím zbaveny veškerých údajů souvisejících s náboženstvím. V této době se neevidoval ani nemanželský původ. Církve si mohly vést matriky pouze pro svou potřebu [3].

V matrikách převážne staršího data se také vyskytuje spousta chyb, duplicit či chybějících záznamů. Důvodů pro to je hned několik. Prvním důvodem je, že se zápisy z počátku psaly na útržky papírů, které měly být poté zaznamenány do matrik, ovšem se buď poztrácely, nebo se osoba, která je zapsala, přestěhovala a lístečky bez zapsání do matriky vzala s sebou. To zapříčinilo jednak absenci záznamů v dané matrice, ale občas také zapsání záznamů do matriky jiné či duplikaci do obou matrik. Další příčinou chybějících zápisů byly poplatky, které se z počátku za evidenci musely platit. Tento poplatek si obyvatelé z chudších vrstev mnohdy nemohly dovolit uhradit, a proto zamlčovaly např. úmrtí či svatby [3].

2.2.2 Současnost

V současné době je matrika státní evidence narození, uzavření manželství, vzniku registrovaného partnerství a úmrtí fyzických osob, která se řídí Zákonem č. 301/2000 Sb. Matrika se dělí na:

- matriku narození,
- matriku manželství,
- matriku partnerství,
- matriku úmrtí.

Působnost na úseku matrik stanovené výše zmíněným zákonem vykonávají:

- matriční úřady,
- obecní úřady obcí s rozšířenou působností,
- krajské úřady,

- Ministerstvo vnitra.

Zápisy se do matriční knihy provádějí rukopisně do předem svázaných knih a zároveň se provádí souběžně pomocí výpočetní techniky. Pokud se zápisy v matriční knize a záznamu, který byl veden pomocí výpočetní techniky neshodují, považují se za správné údaje vedené rukopisně [5].

Do knihy narození se zápis provede na základě písemného hlášení o narození živého nebo mrtvého dítěte, nebo na základě ústního oznámení o narození dítěte mimo zdravotnické zařízení. Záznam obsahuje jméno, popřípadě jména a příjmení dítěte, den, měsíc a rok narození, rodné číslo, místo narození a pohlaví dítěte, jméno, popřípadě jména, příjmení, popřípadě rodná příjmení, data a místa narození, rodná čísla, státní občanství a místo trvalého pobytu rodičů, datum zápisu a podpis matrikáře [5].

Do knihy o manželství se zápis provádí na základě protokolu u uzavření manželství. Záznam obsahuje jména, příjmení, popřípadě rodná příjmení, den, měsíc, rok a místo narození, rodná čísla, osobní stav a státní občanství muže a ženy, kteří uzavřeli manželství. Dále den, měsíc, rok a místo uzavření manželství, jména a příjmení, popřípadě rodná příjmení, den, měsíc, rok a místo narození rodičů manželů, dohoda manželů o příjmení a v případě, že si manželé ponechají dosavadní příjmení, i dohoda o příjmení dětí v mužském a ženském tvaru. Poté jména, příjmení a rodná čísla svědků; jde-li o cizince, který nemá rodné číslo, datum a místo jeho narození, datum zápisu a podpis matrikáře. Podobně vypadá i zápis do knihy partnerství, který se místo manželů týká partnerů [5].

Zápis do knihy úmrtí se provede na základě listu o prohlídce zemřelého, nebo pravomocného rozhodnutí soudu o prohlášení fyzické osoby za mrtvou. Záznam obsahuje den, měsíc, rok a místo úmrtí, jméno, popřípadě jména, příjmení, popřípadě rodné příjmení, den, měsíc, rok a místo narození, rodné číslo, osobní stav, pohlaví, státní občanství a místo trvalého pobytu zemřelého. Dále také jméno, popřípadě jména, příjmení, popřípadě rodné příjmení, a rodné číslo žijícího manžela resp. partnera, datum zápisu a podpis matrikáře [5].

2.3 Požadavky

Jedním z prvních kroků v analýze je analýza požadavků, která slouží ke zjištění, jak a k čemu uživatelé daný systém potřebují. Dle [6] je nedostatečné specifikování požadavků a nedostatečné zapojení uživatelů hlavním důvodem, proč projekt neuspěje.

Požadavek lze dle [7] definovat jako „*specifikaci toho, co by mělo být implementováno*“ a rozlišujeme 2 druhy požadavků:

- funkční požadavky,

- nefunkční požadavky.

Funkční požadavky popisují chování systému a specifikují co by měl dělat. Nefunkční požadavky pak upřesňují omezující vlastnosti nebo podmínky kladené na systém.

Informace, které u požadavků evidujeme jsou dle [7]:

- název požadavku,
- zkratka, která umožňuje snadné odkazování na požadavek,
- popis, který je nejdůležitější částí,
- typ,
- priorita,
- složitost.

Požadavek musí být také jednoznačný, splnitelný a ověřitelný.

Poté, co jsou požadavky zmapovány, je zapotřebí je konzultovat se zúčastněnými aktéry a společně si je vyjasnit a potvrdit. Následně je možné vytvořit případy užití a namapovat je na analyzované požadavky.

2.4 Případy užití

Případ užití zachycuje co musí systém umět a splňovat na základě toho, jak jej uživatelé budou používat. Popisuje detailní interakci uživatele se systémem případně jinými uživateli a systém se tomu musí uzpůsobit. Model případů užití se skládá ze 2 částí:

- aktéři,
- případy užití.

Aktéři reprezentují lidi, kteří budou nějakým způsobem pracovat se systémem. Při tvorbě diagramu je nutné se zaměřovat právě na aktory, aby bylo zajištěno, že systém bude užitečný. Aktor má jméno, případně krátký popis a je asociován s případem užití, se kterým interaguje [8].

Případy užití reprezentují akce, které systém provádí pro aktéry. Případ užití není funkce nebo vlastnost a nemůže být více dekomponován. Obsahují název a krátký popis a dále detailní kroky, které popisují jak aktéři využívají systém k tomu, aby bylo vytvořeno něco podstatného, co uspokojí jejich potřeby. [8].

Množina všech aktérů a případů užití se pak nazývá modelem případů užití.

Případy užití snadno zachycují a zobrazují funkční požadavky tím, že popisují chování systému ve smyslu, jak interaguje s uživateli. Ukazují také, jak funkční požadavky pomáhají uživatům docílit vytvořit něco užitečného. Pomocí případů užití lze také zachytit všechny požadavky na systém [8].

Prvním krokem k sestavení diagramu případů užití je dle [9] analýza, ve které je třeba načerpat informace o projektu a všech faktorech, které jej mohou ovlivnit a to jak dobré, tak špatné. Z dostupných informací a konzultací se zákazníkem lze poté sestavit funkční a nefunkční požadavky. Z provedené analýzy tak lze získat celkový přehled o doméně a projektu a je možné identifikovat hranice systému. Poté, co jsou hranice identifikovány může dojít ke zmapování aktorů, kteří se v systému vyskytují a interagují s ním. Poté nastává poslední krok, ve kterém je třeba identifikovat případy užití. Identifikované případy užití pak mohou být přiřazeny k aktorům a odprezentovány zákazníkovi.

2.5 Doménový model

Doménový model slouží k popsání modelu reálného světa pomocí objektů a vztahů mezi nimi. Pokud již máme zmapované požadavky na systém, tak můžeme identifikovat doménové entity a jejich vztahy. Tato identifikace již pomůže k získání celkového přehledu a porozumění domény a pomůže v dalším návrhu. Model se také vytváří k tomu, aby vyplnil mezeru mezi lidmi napříč různými odděleními ve společnosti a zabránil tak různé interpretaci.

Doménový model je znázorněním objektů reálného světa a neměl by tak obsahovat modely, které se vztahují ke konkrétnímu programovacímu jazyku. Jedná se čistě o konceptuální model, který je od technických detailů odstíněn a obsahuje konceptuální objekty resp. třídy [10].

Navíc dle [11] modelování domény také pomáhá agilním společnostem, jelikož jim poskytuje příležitosti pro použití návrhových vzorů určených pro agilní vývoj, a které z dlouhodobého hlediska vývoj urychlí. S měnícím se návrhem systému a refaktorováním kódu je velmi důležité aktualizovat i doménový model, tak aby odpovídal skutečnosti a pomáhal tak stále k porozumění problémové domény.

Dle [10] se tvorba doménového modelu dělí na 4 kroky:

1. identifikace konceptuálních tříd například pomocí techniky identifikace podstatných jmen,
2. zakreslení identifikovaných tříd do doménového modelu,
3. přidání asociací, které zachycují vztahy mezi nimi,
4. přidání atributů, které jsou pro popsání domény podstatné.

2.6 Procesy

Procesy jsou různých druhů a typů, avšak v této kapitole se budu věnovat obchodním procesům, které jsou pro praktickou část této práce podstatné.

„*Podnikový proces je tok práce nebo činností. Každá organizace je v podstatě organizovaná soustava procesů a činností, které na sebe vzájemně navazují, vzájemně interagují probíhají napříč organizačními jednotkami, reagují na různé podněty z vnitřního i vnějšího prostředí. V procesech se transformují vstupy a zdroje na výstupy, které zhodnocuje zákazník procesu.*“ [12]

Modelování obchodních procesů při analyzování domény má spoustu přínosů, jedním z nich je například lepší pochopení činnosti zákazníka. Další výhodou je, že získáme přesnější specifikaci požadavků a předem analyzované požadavky si tak umístíme do kontextu. Další výhodou také je, že získáme lepší podporu procesů v navržené aplikaci, protože při její implementaci je máme zmapované. Zároveň při samotném analyzování procesů můžeme identifikovat slabá místa společnosti a navrhnout tak zákazníkovi jejich zlepšení [7].

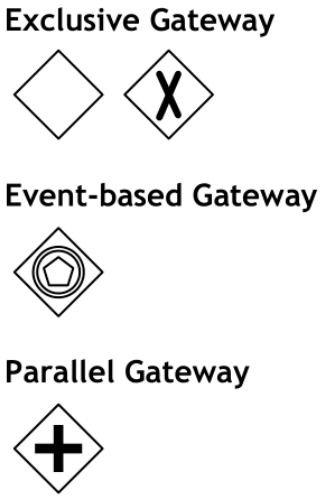
Při modelování procesů je také doporučené modelovat dva typy modelů – současný stav (AS IS) a stav po realizaci (TO BE). V modelu, který popisuje současný stav modelujeme aktuální procesy bez ohledu, jak budou v budoucím systému vypadat. Ve stavu po realizaci modelujeme procesy tak, jak budou vypadat po implementaci systému. Tyto dva modely slouží k tomu, abychom byli schopni vyhodnotit efektivnost a přínos implementovaných procesů [13].

K získání informací potřebných k modelaci procesů můžeme využít např. komunikaci se zákazníkem, který o procesech v jeho společnosti ví typicky nejvíce. Další možností mohou být standardy nebo směrnice. Jejich modelování je iterativní proces a po vytvoření první verze je doporučené jej konzultovat se zákazníkem, který se k němu může vyjádřit. Zároveň konzultace také pomůže k vzájemnému pochopení. Pro popsání procesů můžeme využít různé notace nebo textový popis. Vzhledem k tomu, že v práci využívám notaci Business Process Modeling Notation (BPMN), budu dále popisovat pouze tuto notaci.

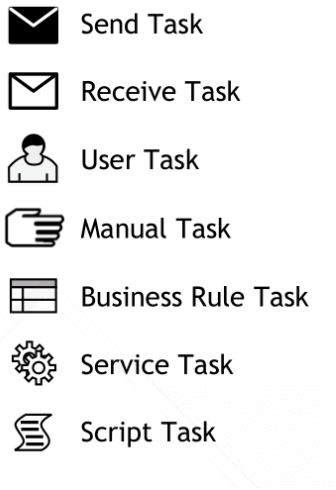
2.6.1 BPMN

BPMN je grafický modelovací jazyk pro analýzu a modelování podnikových procesů. Cílem je poskytnout notaci, která je snadno pochopitelná pro všechny podnikové uživatele od obchodních analytiků, kteří vytvářejí počáteční koncept procesů, až po vývojáře, kteří budou procesy implementovat. BPMN je tedy standardem, který vytváří most mezi návrhem a implementací procesu [14].

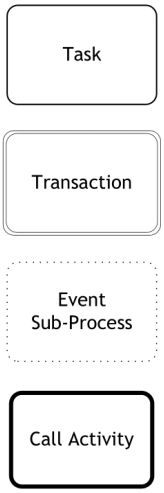
Notace specifikuje základní typy objektů, pomocí kterých lze diagramy procesů vytvářet. Na obrázcích 2.1–2.4, jsou vybrané typy těchto objektů. Na obrázku 2.1 jsou uvedeny brány, pomocí kterých lze řídit tok procesu např. na základě události nebo podmínky, případně jeho rozdělení a sloučení. Dále na obrázku 2.2 se nacházejí typy úloh, které umožňují specifikovat o jakou úlohu



Obrázek 2.1: Druhy brán



Obrázek 2.2: Typy úloh



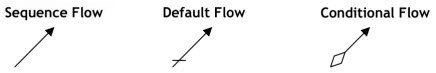
Obrázek 2.3: Typy aktivit

Obrázek 2.4: Označení aktivit

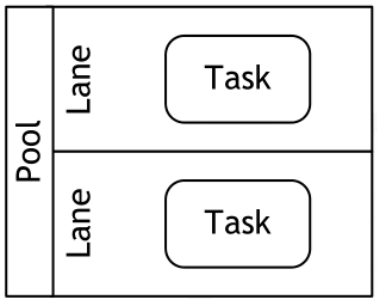
Zdroj: http://www.bpmb.de/index.php/BPMNPoster

se jedná. Poté na obrázku 2.3 jsou typy aktivit, pomocí kterých je možné například upřesnit, jestli se jedná o běžnou aktivitu, nebo transakční aktivitu. Aktivity lze označovat symboly z obrázku 2.4 a ukázat tak, zda je aktivita prováděna v cyklu, paralelně nebo zda se jedná o subproces. Tok procesu lze pak řídit pomocí symbolů na obrázku 2.5, a pomocí swimlanes na obrázku 2.6 lze přehledně zobrazit, kdo má danou aktivitu provádět.

Výše popsané objekty nejsou kompletním výčtem BPMN notace, avšak pro pochopení procesů popsaných v této práci vybraná podmnožina objektů postačuje.



Obrázek 2.5: Kontrola toku



Obrázek 2.6: Swimlanes

Zdroj: http://www.bpmb.de/index.php/BPMNPoster

2.7 Návrh systému

Návrh systému je poslední aktivita, která se provádí před samotnou implementací systému. Smyslem návrhu je přesná specifikace způsobu, jak implementovat analyzované požadavky a případy užití. Vycházet lze z analyzované problémové domény, požadavků a případů užití. Návrh spočívá ve sloučení technických řešení z domény řešení za účelem vytvoření modelu systému, který lze skutečně implementovat [7].

Dle [7] se návrhový model tříd skládá z:

- návrhových podsystému,
- návrhových tříd,
- rozhraní,
- návrhových realizací případů užití,
- diagramu nasazení.

Návrhovým podsystémem jsou v tomto kontextu myšleny části, které bude výsledný systém obsahovat. Konkrétně se jedná například o datovou vrstvu, či vrstvu obchodní logiky.

Návrhovou třídou se pak rozumí taková třída, která je na takovém stupni, že lze implementovat. Zdrojem pro tvorbu je problémová doména, doménový model či případy užití. Z doménového modelu lze dojít k modelu návrhových tříd pomocí upřesňování a doplňování detailů k analytickým třídám. Během tohoto doplňování je možné dekomponovat analytické třídy na několik návrhových tříd, ty pak obsahují kompletní sadu atributů včetně názvu a viditelnosti. Správně formulovaná návrhová třída je pak:

- úplná a dostačující,

- jednoduchá,
- vysoce soudržná,
- bez těsných vazeb [7].

Rozhraní pak předepisují množinu veřejných funkcí. Podstata spočívá v oddělení rozhraní od fyzické implementace prostředníctvím klasifikátoru, jakým je třeba třída nebo podsystém. Rozhraní pouze deklaruje kontrakt, jenž může být realizován jedním nebo několika klasifikátory. To, co je realizováno pomocí rozhraní, musí přijímat a respektovat kontrakt definovaný tímto rozhraním [7].

Diagram nasazení pak specifikuje mapování významných komponent na fyzický hardware a ukazuje nejen fyzický hardware, ale i způsob, jímž je software na tomto hardwaru nasazen. Mapuje architekturu softwaru vytvořenou v předchozích fázích návrhu na fyzickou architekturu systému, na němž bude software spuštěn [7].

2.8 Databáze

„*Databáze jsou jakékoli data nebo informace, které jsou speciálně uspořádány pro rychlé vyhledávání a získávání informací počítačem. Databáze jsou strukturovány tak, aby usnadňovaly ukládání, načítání, úpravy a mazání dat ve spojení s různými operacemi zpracování dat.*“ [15] Základní rozdělení databází je na relační a nerelační.

2.8.1 Nerelační

Nerelační databáze lze pak rozdělit do 5 kategorií:

- databáze klíč – hodnota,
- sloupcově orientované,
- dokumentově orientované,
- grafové,
- objektově orientované [16].

První ze zmíněných typů jak název napovídá, používá k ukládání dat uložiště typu klíč – hodnota. Pod položkou klíče se tak uchovávají uložená data. Typicky se pro to využívají hashovací tabulky. Tento typ ukládání je rychlejší oproti ostatním, protože se přistupuje přímo k datům, jelikož víme, kde jsou podle klíče uložena. Případ užití pro zmíněný druh databáze může být například uložení relace databáze nebo uložení nákupního košíku. Příklad takové databáze je v současné době třeba Amazon DynamoDB.

Sloupcově orientované databáze ukládají záznamy do sloupců místo řádků. Důvod pro to je takový, že jsou pak lépe škálovatelné, mají rychlejší dotazy, agregační dotazy jsou také rychlejší a v neposlední řadě jsou data efektivně uložena. Tato databáze je vhodná například pro data mining nebo analytické aplikace. Příklad takovýchto databází je Cassandra nebo Bigtable [16].

Dalším druhem je dokumentová databáze, která data ukládá ve formě dokumentů. Proto nabízí velkou výkonost a relativně snadnou horizontální škálovatelnost. Dokumenty odpovídají řádkům v relační databázi s tím rozdílem, že dokumenty jsou uzpůsobené tak, aby se záznamy nemusely spojovat. Při dotazu se tak v ideálním případě vrátí pouze dokument, který obsahuje již všechna potřebná data. Vzhledem k tomu, že dokumentové databáze nemají schéma, lze mít dokumenty obsahující položky, které v ostatních dokumentech neexistují. Dokumentové databáze jsou vhodné pro použití v aplikacích, které mají málo relací, nebo se může doménový model rozdělit do logických celků tak, aby jednotlivý dokument obsahoval jeden z celků. Příkladem dokumentové databáze je MongoDB nebo CouchDB [16].

Grafové databáze ukládají data ve formě grafů a nabízí flexibilní datový model. Vrcholy grafu reprezentují data a hrany mezi vrcholy reprezentují vztahy mezi daty. Tento druh databáze se využívá především pro OLTP systémy. Jsou optimalizovány pro transakce a transakční integritu spolu s dostupností. Příkladem grafové databáze je Neo4j a OrientDB [17].

Objektově orientované databáze ukládají data jako objekty ve smyslu objektově orientovaného programování a může na ni být pohlíženo jako na kombinaci databáze a OOP principů. Objektově orientované úložiště nabízí všechny vlastnosti OOP jako zapouzdření polymorfismus a dědičnost. Každý z uložených objektů má identifikátor, který jej jednoznačně identifikuje. Přístup k datům je rychlejší protože k objektům může být přistupováno napřímo díky ukazatelům. Vhodné použití pro tento druh databáze je v aplikacích, které mají spousty objektových relací a často se mění. Jejich nevýhodou je, že jsou hůře škálovatelné a často spjaté s konkrétním programovacím jazykem. Příkladem objektové databáze je db4o [16].

2.8.2 Relační

„*Relační databáze je typ databáze, který ukládá vzájemně propojené datové body a poskytuje k nim přístup. Relační databáze jsou založeny na relačním modelu, intuitivním, přímočarém způsobu vyjádření dat v tabulkách. V relační databázi je každý řádek v tabulce záznam s jedinečným ID nazvaným klíč. Sloupce tabulky obsahují atributy dat a každý záznam má obvykle hodnotu pro každý atribut, což usnadňuje navázání vztahů mezi datovými body.*“ [18]

V porovnání s nerelačními databázemi jsou relační databáze vhodné na uložení dat, které mají hodně relací a není jich příliš mnoho. Dalším rozdílem je, že relační databáze mají standardizovaný dotazovací jazyk SQL, oproti nerelačním, kde má téměř každá databáze jazyk pro dotazování svůj. Výhodou

relačních databázi je také silná konzistence uložených dat a podpora ACID operací. Nerelační databáze zvládají snadno velké objemy dat na úkor konzistence, kterou musí zajišťovat middleware nebo obchodní logika aplikace [19].

2.9 Proof-of-concept

Proof-of-concept je menší část systému, která má za cíl otestovat návrh a prokázat, že navržený koncept je proveditelný. Dalším cílem je prokázat, zda má smysl implementovat systém nebo aplikaci dle návrhu, nebo je návrh neproveditelný a nemá smysl jej realizovat.

Prototyp je narozdíl od POC vytvořen téměř celý a má za cíl ukázat, jak bude systém vyvíjen a posléze používán. Obsahuje téměř všechny funkcionality výsledného produktu, ale nemusí být efektivní nebo graficky zpracovaný [20].

POC je zamýšlen čistě k ověření funkcionality jednoho nebo více konceptů a jejich začlenění do systému. Použitelnost v reálném prostředí v tomto případě ani nemusí být brána v úvahu, jelikož interakce technologií je nejen časově náročná, ale také může zkreslit výsledky při výsledném vyhodnocení. Hlavním cílem je identifikování reálného produktu předtím, než dojde k vývoji. Prototyp je prvním krokem při vytváření fungujícího modelu, který poté bude použitelný v reálném světě [20].

2.10 Crowd-sourcing

Crowd-sourcing je dle [21] definován jako online, distribuovaně problém řešící a produkční model, který zužitkovává kolektivní znalosti online komunit a slouží ke splnění stanovených cílů. Online komunitám, které se nazývají anglickým slovíčkem *crowd*, je poskytnuta možnost se zapojit do aktivit, které jsou nabídnuty organizací. Organizace pak určitým způsobem motivuje participanty, kteří se do aktivit zapojují. Definice se pak upravuje na základě toho v jakém kontextu nebo oboru je použita.

2.11 Existující řešení

V této kapitole analyzuji systémy a programy, které se zabývají genealogií a umožňují sestavovat rodokmeny. Porovnám také do jaké míry odpovídají vytvořené záznamy skutečnosti resp. zda mají reálný podklad, avšak to není jediné kritérium. Další kritéria jsou cena, počet uživatelů, otevřenost systému a technické provedení, pokud je uvedeno.

2.11.1 Ancestry

Prvním z analyzovaných systémů je Ancestry, který se zaměřuje na hledání předků především z USA. Portál čerpá data z několika zdrojů. Prvním z úvá-

děných zdrojů je vlastní databáze systému, dále pak United States Federal Census a snímky, které poskytla společnost FamilySearch. Pokud se hledání předků provádí mimo USA, využívá systém ještě veřejné databáze dané země, pokud jsou dostupné [22].

United States Federal Census je americká verze sčítání obyvatel, které prvně proběhlo v roce 1790 a je prováděno každých 10 let. Legislativa dovoluje tato sčítání zveřejnit nejdříve 72 let po provedeném sečtení, což znamená, že v současné době jsou dostupné záznamy od roku 1940 [23].

Ancestry dovoluje generovat a vytvářet rodokmeny bez nutnosti doložit, zda osoba, kterou do rodokmenu přidáváme skutečně existuje. To umožňuje vytvářet potencionálně neexistující osoby.

Portál nabízí k používání několik druhů předplatných, které jsou nutné i pro běžné úkony jako je například dohledávání záznamů. Společnost nabízí 14 denní zkušební verzi a poté má uživatel možnost zvolit z následujících měsíčních předplatných:

- 24,99 USD – nabízí přístup k záznamům v USA,
- 39,99 USD – nabízí přístup k mezinárodním záznamům,
- 49,99 USD – nabízí navíc vyhledávat i ve strojově zpracovaných novinových článcích [24].

Dle [25] dosáhl systém v první polovině roku 2019 15 milionů členů.

2.11.2 MyHeritage

Systém MyHeritage nespecifikuje geografickou polohu záznamů, na které se zaměřuje a uvádí, že má sbírku záznamů z celého světa. Vyzdvihuje také, že v systému lze vyhledávat ve více než 13 miliardách historických záznamů. Data, ze kterých čerpá jsou sčítání lidu, volební seznamy, rodokmeny, narození, sňatky, úmrtí a armádní záznamy [26].

Podobně jako předchozí systém nabízí MyHeritage zdarma vytvářet rodokmeny, které nemusejí odpovídat skutečnosti. Společnost uvádí, že po vytvoření rodokmenu se přidané osoby pokusí dohledat. Rodokmeny, které uživatelé vytvoří jsou anonymizovaně dostupné k vyhledání jiným uživatelům a je tak možné je spojovat. K osobám, které jsou vytvořené lze nahrávat fotografie a poznámky či jiné detaily [27].

Vyhledávání záznamů, rodokmenů a ostatních dat je ale možné až s placeným předplatným. Systém nabízí celkem 3 možnosti subskripce s různými pokročilými funkcemi. V následujícím výčtu jsou uvedeny měsíční ceny spolu s jejich funkcemi:

- 7,42 USD – možnost dohledávat osoby z různých rodokmenů,
- 11,58 USD – přidávání osob z historických záznamů do rodokmenu,

- 16,58 USD – automatické dohledávání osob uvedených v rodokmenu a prohledávání všech historických záznamů [28].

Dle [29] měl systém počátkem roku 2021 74 milionů uživatelů.

2.11.3 FamilySearch

FamilySearch je webovým portálem, který spravuje Církev Ježíše Krista Svatých posledních dnů zdarma pro širokou veřejnost. V současné době obsahuje více než 1 miliardu historických záznamů. Záznamy existují ve formě snímků, mikrofilmů a digitální podobě. Uživatelé mohou záznamy a snímky vyhledávat a prohlížet rodokmeny, které již byly vytvořeny jinými uživateli [30].

Dobrovolníci se mohou zapojit do indexace záznamů a jejich překladů do jiných jazyků. Portál také umožňuje napojení z externích aplikací pomocí aplikačního rozhraní. V současné době portál využívá průměrně 200 000 uživatelů [30].

2.11.4 Gramps

Gramps je na rozdíl od předchozích systémů desktopová aplikace, která je zdarma a dostupná pro Linux, Windows a macOS. Je šířena pod licencí GPL 2.0 a je možné pro ni navrhovat a vyvíjet změny prostřednictvím Githubu, který vývojáři poskytují. Aplikace je implementována v pythonu a je k ní poskytnuta rozsáhlá Wiki, která obsahuje spoustu informací od instalace až po ovládání programu. Vzhledem k tomu, že se jedná o desktopovou aplikaci, kterou si uživatel musí napojit na vlastní databázi, tak neexistuje databáze společná, která by obsahovala již vytvořené záznamy [31].

2.11.5 Webtrees

Podobně jako předchozí systém je Webtrees webová aplikace, kterou si mohou uživatelé stáhnout a nainstalovat. Neobsahuje tedy globální databázi, která by již byla daty naplněna. Stažení a používání aplikace je zcela zdarma a aplikace je šířena pod licencí GPL 3.0. Zdrojové kódy, které jsou umístěné veřejně na Githubu ukazují, že je systém napsán v jazyce PHP a používá se jako webová aplikace [32].

Webtrees je založena okolo osob a rodokmenů, což znamená, že lze vytvářet osoby a vztahy mezi nimi. Na základě nastavených vztahů lze pak vygenerovat rodokmen. K jednotlivým osobám je možné uvést, z jakého zdroje byla osoba vytvořena, ale není to pro vytvoření osoby požadovaný údaj. Systém poté dále umožňuje generovat statistiky a diagramy jako například vývoj úmrtí napříč stoletími nebo rozložení můžu a žen [32].

Kapitola 3

Praktická část

3.1 Analýza

3.1.1 Analýza požadavků

Na základě jednání se zadavatelkou, jsem identifikoval potřeby uživatelů a požadavky na systém. Jelikož systém bude sloužit primárně k přepisování a hledání matričních záznamů, musí umožnovat vyhledávat matriky. Dále musí systém umožnovat přepis záznamů včetně jejich hledání. Jelikož některé záznamy jsou staršího data musí systém vyhledávat i německá synonyma českých jmen, pokud tedy uživatel vyhledává jméno Jan, musí systém zobrazit výsledky i pro jméno Johan a obráceně. Jelikož se jedná o crowd-sourcing systém, musí systém umožnovat kontroly přepisů záznamů, které provedli jiní uživatelé. Tato funkcionalita je vyžadována, aby záznam získal kredibilitu a mohli mu ostatní uživatelé důvěřovat. Uživatel, který bude zápis kontrolovat, by měl mít možnost označit, zda záznam odpovídá skutečnosti či se jedná o chybný přepis. Tyto uživatelské kontroly pak musí být dostupné u každého záznamu, kde bude vidět celkový počet schválení a zamítnutí.

Matriční záznamy jsou 3 typů – záznamy o narození, záznamy o úmrtí a záznamy o sňatku. Systém musí poskytovat všechny tyto typy přepisů. Dále systém musí umožnit uživatelům identifikovat osoby v záznamech. Jelikož uživatel v době přepisu neví, zda osoba, pro kterou přepis vytváří, již v systému existuje bude tak v systému duplicitně. Systém musí poskytovat možnost tyto osoby identifikovat a tím je tedy sloučit do jedné identifikované osoby. V matričních záznamech se také objevují primární a sekundární osoby, které systém musí umožňovat identifikovat a také zaznamenat do přepisu matričního záznamu. Primární osobou je v tomto kontextu myšlen člověk, kterého se záznam týká např. v záznamu o narození se jedná o narozeného. Sekundární osoba je pak jakákoli další osoba v záznamu, která není primární.

Uživatelé v roli admin potřebují být schopni spravovat již vytvořené matriky a vytvářet nové. Aby se předešlo poškození již vytvořených záznamů a

záměrnému vytváření nevalidních zápisů, je nutné, aby systém evidoval historii editací, které byly provedeny. Uživatel v roli admin pak musí mít možnost smazat všechny úpravy a záznamy, které uživatel provedl. Uživatel v roli admin pak musí mít možnost smazat provedené úpravy a záznamy, ale také zablokovat škodícího uživatele, čímž mu znemožní přístup do systému a současně s touto interakcí dojde ke smazání jeho úprav.

Důležité je, aby systém byl pro uživatele dostupný jako webová aplikace, mobilní aplikace v tomto případě není požadována. Jelikož do systému budou přepisovány matriky od zhruba 16. století, musí systém zvládat pracovat se zhruba 800 miliony záznamy.

Všechny identifikované funkční požadavky jsou uvedeny v tabulce 3.1, všechny nefunkční požadavky jsou pak v tabulce 3.2

Tabulka 3.1: Funkční požadavky

Požadavek	**Popis**
F01 Hledání matriky	Systém bude umožnovat registrovaným uživatelům vyhledávání konkrétní matriky.
F02 Hledání záznamu	Systém bude umožňovat registrovaným uživatelům vyhledávat konkrétní zápis o narození, úmrtí či svatbě. Dále musí vyhledávat i německá synonyma českých jmen např. pokud uživatel bude vyhledávt jméno Jan, musí systém zobrazit výsledky i pro jméno Johan.
F03 Přepis záznamu	Systém bude umožňovat registrovaným uživatelům přepisovat matriční záznamy o narození, úmrtí a sňatku.
F04 Kontrola přepisu	Umožnit registrovaným uživatelům zkontrolovat přepisy jiných uživatelů a potvrdit tak jejich správnost, či zamítnout z důvodu chybného přepsání. Uživatelé v roli admin budou schopni zamítnout či potvrdit záznam bez potřeby validace ze strany běžných uživatelů.
F05 Evidence údajů	Systém musí evidovat všechny typy údajů - narození, sňatky, úmrtí.
F06 Identifikace osoby	Systém bude umět identifikovat konkrétní osobu v různých záznamech. Například označit, že tato konkrétní osoba v záznamu o narození je stejná osoba ze záznamu o úmrtí.
F07 Správa matrik	Umožnit adminům spravovat matriky (vytvoření, úprava, smazání).
F08 Správa přepisů	Umožnit adminům spravovat přepisy (úprava, smazání).
F09 Uchování historie	Systém musí uchovávat historii editací a schválení.
F10 Správa uživatelů	Systém musí umožňovat uživatelům v roli admin zablokovat a smazat záznamy vybraného uživatele.

Tabulka 3.2: Nefunkční požadavky

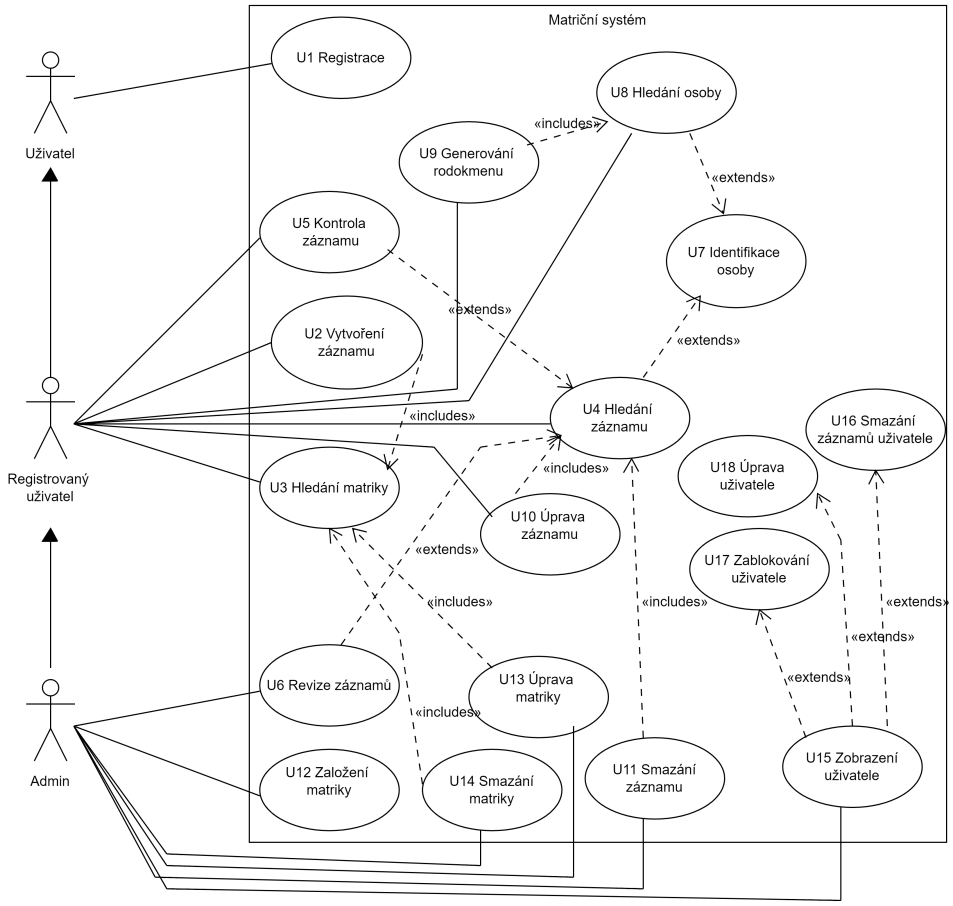
Požadavek	Popis
N01 Počet záznamů	Systém musí bez obtíží zvládnout pracovat se zhruba 800 miliony záznamy přepisů.
N02 Dostupnost přes web	Systém musí být uživatelům dostupný jako webová aplikace. Mobilní aplikace naopak není požadavkem.

3.1.2 Případy užití

Na základě požadavků a konzultací se zadavatelkou jsem vytvořil celkem 18 případů užití, z nichž jediný dostupný pro neregistrované uživatele je U1, který je popsán v tabulce 3.3. Případ popisuje standardní postup, jak se do systému registrovat. Proces přihlášení není vytvořen jako případ užití, jelikož přihlášení probíhá standardním způsobem a není třeba jej detailně popisovat.

V současné verzi systému není zahrnut případ užití pro vytváření matričních úřadů a obcí. Důvod pro to je takový, že seznam úřadů a obcí je neměnný, a proto při vytvoření databáze se matriční úřady a obce jednorázově vyplní a dále vytvářet již nebudou potřeba.

Ostatní případy užití popisují interakci uživatelů a administrátorů se systémem. Všechny případy užití byly konzultovány a následně odsouhlaseny zadavatelkou. Zároveň bylo potvrzeno, že splňují všechny požadavky na matriční systém.



Obrázek 3.1: Diagram případů užití

U1 Registrace

Předpoklad	Uživatel není přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko registrovat 2. Systém zobrazí registrační formulář 3. Uživatel vyplní potřebné údaje a odešle formulář 4. Systém zvaliduje údaje a odešle potvrzovací e-mail na uvedenou e-mailovou adresu 5. Uživatel potvrdí svou registraci kliknutím na odkaz, který byl zaslán na uvedený e-mail 6. Systém potvrdí registraci a uloží výsledek
Alternativní scénář	4.a. Systém detekuje chybné údaje a vyzve uživatele k opravě 5. Uživatel opraví údaje, odešle formulář a tok pokračuje v bodě 4
Podmínka pro dokončení	Uživatel je zaregistrován

Tabulka 3.3: U1 Registrace

U2 Vytvoření záznamu

Předpoklad	Uživatel je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Include(Hledání matriky) 2. Uživatel si vybere konkrétní matriku 3. Systém zobrazí formulář vybrané matriky 4. Uživatel klikne na tlačítko pro vytvoření matričního záznamu 5. Systém zobrazí formulář, pro vyplnění matričního záznamu 6. Uživatel vyplní všechny dostupné údaje a odešle formulář 7. Systém provede základní validaci a uloží záznam
Alternativní scénář	7.a. Systém detekuje chybné údaje a vyzve uživatele k opravě 8. Uživatel opraví údaje, odešle formulář a tok pokračuje v bodě 7
Podmínka pro dokončení	Validní záznam je uložen

Tabulka 3.4: U2 Vytvoření záznamu

U3 Hledání matriky

Předpoklad	Uživatel je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko pro hledání matriky 2. Systém zobrazí formulář pro vyhledávání matriky 3. Uživatel vyplní povinné údaje o matrice a klikne na tlačítko pro hledání 4. Systém na základě vyplněných údajů zobrazí uživateli matriky, které se shodují s uživatelským hledáním
Alternativní scénář	není
Podmínka pro dokončení	Seznam matrik odpovídající kritériím hledání je zobrazen

Tabulka 3.5: U3 Hledání matriky

U4 Hledání záznamu

Předpoklad	Uživatel je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko pro hledání záznamu 2. Systém zobrazí vyhledávací formulář 3. Uživatel vyplní potřebné údaje 4. Systém zobrazí seznam záznamů, které odpovídají vyhledávání <U5 Kontrola záznamu> <U6 Revize záznamu> <U7 Identifikace osoby>
Alternativní scénář	není
Podmínka pro dokončení	Seznam záznamů odpovídající kritériím hledání je zobrazen

Tabulka 3.6: U4 Hledání záznamu

U5 Kontrola záznamu

Předpoklad	Uživatel je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel si ze seznamu vybere záznam ke kontrole 2. Systém zobrazí vyplněný formulář záznamu 3. Uživatel zkontroluje, zda přepis ve formuláři odpovídá skutečnosti a na základě toho jej schválí či zamítne 4. Systém uloží výsledek a uchová informaci o tom, který uživatel jej kontroloval
Alternativní scénář	není
Podmínka pro dokončení	U kontrolovaného záznamu je evidováno rozhodnutí uživatele

Tabulka 3.7: U5 Kontrola záznamu

U6 Revize záznamu

Předpoklad	Uživatel v roli admin je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Admin si ze seznamu vybere záznam ke kontrole 2. Systém zobrazí formulář záznamu 3. Admin schválí či zamítne daný přepis 4. Systém uloží výsledek
Alternativní scénář	není
Podmínka pro dokončení	Záznam je zrevidován

Tabulka 3.8: U6 Revize záznamu

U7 Identifikace osoby

Předpoklad	Uživatel je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko pro identifikaci osoby 2. Systém zobrazí formulář pro dohledání shodné osoby s předvyplněnými údaji osoby z kroku 1 3. Uživatel vyplní údaje osoby a klikne na tlačítko pro hledání 4. Systém zobrazí seznam nalezených osob 5. Uživatel vybere osobu, která se shoduje s osobou v prvním kroku 6. Zobrazí vybrané osoby 7. Uživatel potvrdí shodu 8. Systém uchová historii osoby před sloučením, sloučí osoby a uloží výsledek
Alternativní scénář	7.a. Uživatel odmítne změny a tok pokračuje v bodě 4
Podmínka pro dokončení	Osoba je sloučena

Tabulka 3.9: U7 Identifikace osoby

U8 Hledání osoby

Předpoklad	Uživatel je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko pro hledání osoby 2. Systém zobrazí formulář pro hledání osoby 3. Uživatel vyplní údaje a klikne na tlačítko hledat 4. Systém zobrazí seznam osob, které odpovídají kritériím hledání 5. Uživatel si ze zobrazeného seznamu vybere požadovanou osobu 6. Systém zobrazí formulář osoby spolu se záznamy, které se jí týkají <U7 Identifikace osoby>
Alternativní scénář	není
Podmínka pro dokončení	Seznam nalezených osob je zobrazen

Tabulka 3.10: U8 Hledání osoby

U9 Generování rodokmenu

Předpoklad	Uživatel je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Include (U8 Hledání osoby) 2. Uživatel si vybere osobu 3. Systém zobrazí formulář vybrané osoby 4. Uživatel klikne na tlačítko pro generování rodokmenu 5. Systém z dostupných údajů o dané osobě vygeneruje rodokmen
Alternativní scénář	není
Podmínka pro dokončení	Pro vybranou osobu je vygenerován rodokmen

Tabulka 3.11: U9 Generování rodokmenu

U10 Úprava záznamu

Předpoklad	Uživatel je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Include (U4 Hledání záznamu) 2. Uživatel si vybere záznam 3. Systém zobrazí formulář vybraného záznamu 4. Uživatel klikne na tlačítko pro úpravu 5. Systém zobrazí editovatelný formulář záznamu 6. Uživatel upraví údaje 7. Systém uchová historii záznamu před změnou, resetuje počet kontrol záznamu a uloží výsledek
Alternativní scénář	není
Podmínka pro dokončení	Přepis je upraven

Tabulka 3.12: U10 Úprava záznamu

U11 Smazání záznamu

Předpoklad	Uživatel v roli admin je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Include (U4 Hledání přepisu) 2. Uživatel si vybere přepis 3. Systém zobrazí formulář vybraného přepisu 4. Uživatel klikne na tlačítko smazat 5. Systém smaže přepis
Alternativní scénář	není
Podmínka pro dokončení	Přepis je smazán

Tabulka 3.13: U11 Smazání záznamu

U12 Založení matriky

Předpoklad	Uživatel v roli admin je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko pro založení matriky 2. Systém zobrazí formulář pro vytvoření matriky 3. Uživatel vyplní údaje a odešle formulář 4. Systém zkontroluje údaje a vytvoří matriku
Alternativní scénář	4.a. Systém detekuje chybné údaje a vyzve uživatele k opravě 5. Uživatel opraví údaje, odešle formulář a tok pokračuje v bodě 4
Podmínka pro dokončení	Matrika je vytvořena

Tabulka 3.14: U12 Založení matriky

U13 Úprava matriky

Předpoklad	Uživatel v roli admin je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Include (U3 Hledání matriky) 2. Uživatel si vybere matriku 3. Systém zobrazí formulář vybrané matriky 4. Uživatel upraví údaje 5. Systém zkontroluje údaje a uloží výsledek
Alternativní scénář	5.a. Systém detekuje chybné údaje a vyzve uživatele k opravě 6. Uživatel opraví údaje, odešle formulář a tok pokračuje v bodě 5
Podmínka pro dokončení	Matrika je upravena

Tabulka 3.15: U13 Úprava matriky

U14 Smazání matriky

Předpoklad	Uživatel v roli admin je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Include (U3 Hledání matriky) 2. Uživatel si vybere matriku 3. Systém zobrazí formulář vybrané matriky 4. Uživatel klikne na tlačítko smazat 5. Systém se dotáže na potvrzení 6. Uživatel potvrdí smazání 7. Systém smaže matriku
Alternativní scénář	6.a. Uživatel odmítne smazání 7. Systém zavře formulář
Podmínka pro dokončení	Matrika je smazána

Tabulka 3.16: U14 Smazání matriky

U15 Zobrazení uživatele

Předpoklad	Uživatel v roli admin je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko pro hledání uživatele 2. Systém zobrazí vyhledávací formulář 3. Uživatel vyplní údaje a klikne na tlačítko pro hledání 4. Systém zobrazí seznam uživatelů odpovídající kritériím vyhledávání 5. Uživatel si ze seznamu vybere konkrétního uživatele 6. Systém zobrazí formulář s detaily uživatele, včetně všech akcí, které udělal <U16 Smazání záznamů uživatele> <U17 Zablokování uživatele> <U18 Úprava uživatele>
Alternativní scénář	není
Podmínka pro dokončení	Uživatel je zobrazen

Tabulka 3.17: U15 Zobrazení uživatele

U16 Smazání záznamů uživatele

Předpoklad	Uživatel v roli admin je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko pro smazání záznamů vybraného uživatele 2. Systém se dotáže na potvrzení 3. Uživatel potvrdí smazání 4. Systém smaže všechny uživatelské kontroly záznamů a jím vytvořené záznamy
Alternativní scénář	3.a. Uživatel odmítne změny 4. Systém zavře formulář
Podmínka pro dokončení	Záznamy uživatele jsou smazány

Tabulka 3.18: U16 Smazání záznamů uživatele

U17 Zablokování uživatele

Předpoklad	Uživatel v roli admin je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko pro zablokování vybraného uživatele 2. Systém se dotáže na potvrzení 3. Uživatel potvrdí zablokování 4. Systém zablokuje vybraného uživatele, smaže jeho kontroly záznamů a jím vytvořené záznamy
Alternativní scénář	3.a. Uživatel odmítne změny 4. Systém zavře formulář
Podmínka pro dokončení	Uživatel je zablokován

Tabulka 3.19: U17 Zablokování uživatele

U18 Úprava uživatele

Předpoklad	Uživatel v roli admin je přihlášený
Spuštění	Uživatel klikne na tlačítko
Hlavní scénář	1. Uživatel klikne na tlačítko pro úpravu vybraného uživatele 2. Systém zobrazí editační formulář 3. Uživatel provede požadované změny 4. Systém uloží výsledek
Alternativní scénář	není
Podmínka pro dokončení	Uživatel je upraven

Tabulka 3.20: U18 Úprava uživatele

V tabulce 3.21, jsem analyzoval, že všechny uživatelské požadavky jsou pokryty případy užití a je tak možné se posunout na celkovou analýzu domény a následný návrh systému.

Tabulka 3.21: Mapování požadavků na případy užití

Požadavek / Use case	F01	F02	F03	F04	F05	F06	F07	F08	F09	F10
U1									X	X
U2	X		X		X				X	
U3	X						X			
U4		X						X		
U5				X					X	
U6				X					X	
U7						X				
U8		X				X				
U9						X				
U10								X	X	
U11								X		
U12							X			
U13							X			
U14							X			
U15										X
U16										X
U17										X
U18										X

3.1.3 Analýza domény

Doména matrik v současné době obsahuje záznamy z počátku 16. století až do roku 1921, jelikož dle Zákona o matrikách lze matriku zveřejnit nejdříve 100 let od posledního záznamu v knize o narození a 75 let od posledního zápisu v knize manželství či úmrtí [5]. Tyto záznamy jsou nejčastěji zveřejněny na webových stránkách úřadů, které matriky spravují. V současné době jsou tyto zmiňované matriky nafoceny do digitální podoby a zveřejněny, avšak existují pouze jako digitální snímky bez elektronického přepisu, který se dle dostupných informací ani neplánuje. Zároveň však dle zadavatelky existuje řada dobrovolníků, kteří se zajímají o genealogii a jejich koníčkem je bádání v historických pramenech a jsou ochotni elektronické snímky přepisovat do digitální podoby. Navíc dle zadavatelky existují již neoficiální přepisy od těchto dobrovolníků, kteří je poskytli veřejnosti ve formě tabulek. Pokud by tedy existoval systém, který dobrovolníkům umožní záznamy snadno přepisovat a vyhledávat v nich, je téměř jisté, že jej využijí.

Matriční záznamy jsou 3 typů:

- záznamy o narození,
- záznamy o úmrtí,
- záznamy o sňatku.

Matriky mohou mít tyto záznamy oddělené, nebo mohou obsahovat více druhů záznamů

V matrice narození obsahující záznamy o narození jsou údaje o narozeném a to – datum narození, datum křtu, jméno narozeného, pohlaví, náboženství, zda je manželským dítětem, zda bylo narozeno mrtvé a místo narození. Dále poté jméno křtícího a bábu včetně jejího obydlí. Záznam také obsahuje jméno, příjmení a stav otce dítěte včetně jeho povolání, místa a data narození spolu s jeho rodiči. Téměř shodné údaje jsou uvedené u matky dítěte, u které se neeviduje její řemeslo. V neposlední řadě obsahuje záznam seznam kmotrů a svědků spolu s jejich bydlištěm [33].

V matrice úmrtí, která obsahuje záznamy o úmrtí jsou údaje o zemřelém a to – datum a místo úmrtí, datum pohřbu, jméno, příjmení a bydliště. Je-li je zemřelý dítě, uvádí se ještě stav rodičů, pokud je zemřelým žena, uvádí se navíc jméno, příjmení a stav muže. U zemřelého se dále eviduje jeho náboženství, pohlaví, věk a zda byl svobodný. Dále se zde nachází, místo, kde byl zemřelý pohřben, způsob úmrtí, jméno pochovávajícího kněze či svědka pohřbu a zda a od koho byl svátostmi zaopatřen [34].

V matrice oddaných, která obsahuje záznamy o sňatku se evidují údaje o ženichovi a nevěstě, kde je uvedeno jejich jméno, příjmení, věk, náboženství, místo a datum narození včetně jména, příjmení a stavu rodičů oddávaných. Dále se v zápise nachází datum a místo oddavek spolu s oddávajícím knězem a svědkové, u kterých je uvedeno i jejich bydliště spolu s jejich náboženstvím. V

záznamu o sňatku hraje i dominantní roli poznámka, kam se uvádí např. důkaz plnoletosti pomocí křestného listu, nebo povolení k sňatku při neplnoletých osobách [35].

Na základě požadavků a případů užití jsem v doméně identifikoval následující role:

- uživatel,
- registrovaný uživatel,
- administrátor.

Uživateli je umožňeno se pouze registrovat, nesmí mit přístup ani k prohlížení. Toto vychází z požadavku zadavatelky, kde je potřeba mít uživatelské účty pod kontrolou a případně zablokovat uživatele, aby se do systému nemohl ani přihlásit a důvodů pro to je několik. Jelikož se jedná o veřejně dostupný systém, může se objevit několik uživatelů, kteří mohou mít zájem poškodit jeho reputaci a začnou tak nesmyslně vyplňovat záznamy, nebo je chybně upravovat. Dalším důvodem může být, že v současné době existuje řada společností, které poskytují služby na sestavení rodokmenu a dohledání předků. Tyto společnosti by tak mohly zneužívat systém pro svoje účely či vytěžovat systémovou databázi.

Uživatel se po registraci stane registrovaným uživatelem a je mu umožněn vstup do systému, kde může využívat jemu dostupné funkce, které jsou uvedeny v diagramu případů užití na obrázku 3.1.

Registrovaný uživatel se pak může stát administrátorem systému tak, že mu jiný administrátor tuto roli přidelí. Pro přidělení administrátorské role může administrátor využít use case U18, který je uveden v tabulce 3.20. Logický problém nastává v momentě vytvoření prvního administrátorského účtu. Tento problém se vyřeší při prvotním naplnění databáze, a to ručním přidělením oprávnění v databázi, poté je již možné bežně rozdávat role administrátora.

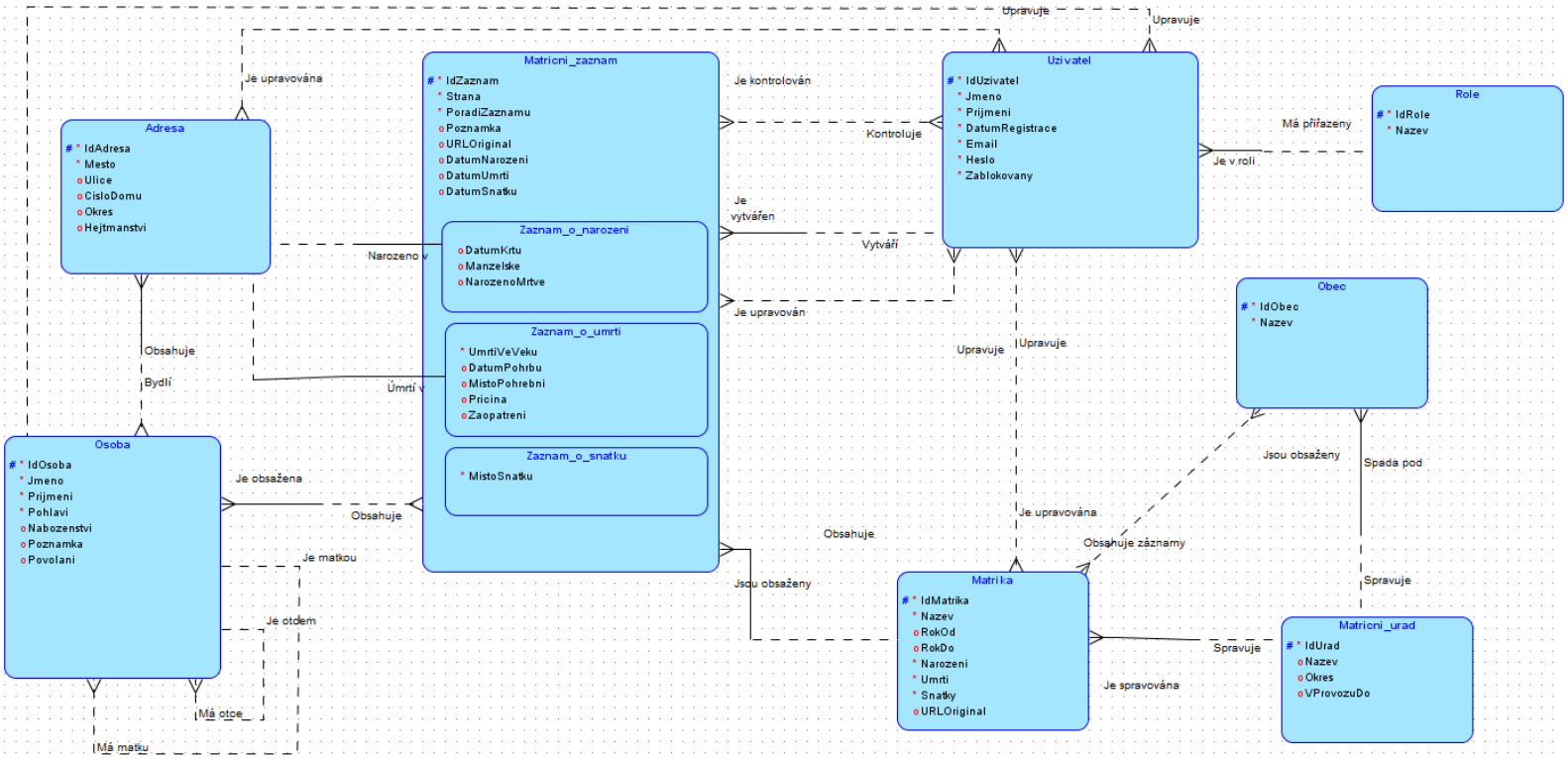
3.1.3.1 Doménový model

Na základě analýzy a konzultace se zadavatelkou jsem vytvořil doménový model, který je na obrázku 3.2.

Dominantním prvkem doménového modelu je entita matričního záznamu, která pomocí ISA hiearchie vyobrazuje všechny typy matričních záznamů. Údaje, které matriční záznamy všech 3 typů obsahují jsou analyzované v sekci 3.1.3. Všechny tyto údaje jsou v entitě zachyceny přímo či skrze vazby do ostatních entit. Za zmínku zde stojí data narození, úmrtí a sňatku, která jsou vyjmuta z konkrétních typů záznamů na společnou entitu matričního záznamu. Důvodem vyjmutí je častý výskyt ručně vepsaných poznámek do matričních záznamů, které se zmíněných dat týkají. Záznamy o narození a o

úmrtí mají vazbu na entitu adresy, jelikož je u těchto záznamů uvedena konkrétní adresa. Naproti tomu záznam o sňatku vazbu na adresu nemá neboť konkrétní adresa v matričním záznamu uvedena není.

Další entitou, kterou bych rád zmínil je entita matriky, která obsahuje atributy popisující o jaký typ matriky se jedná. Tyto atributy jsou jednak kvůli kontrole uživatelů, aby uživatelé nemohli vytvářet záznamy o narození do matriky, která obsahuje pouze záznamy o úmrtí. Dalším důvodem je také snažší budoucí filtrování a vyhledávání záznamů a matrik.



Obrázek 3.2: Doménový model

3.1.3.2 Procesy

Proces uvedený na obrázku 3.3 popisuje standardní postup pro vytvoření přepisu matričního záznamu. Uživatel musí pro přepis vyhledat matriku a zahájit ho z konrétní matriky. Důvod pro to je takový, že uživatelé budou typicky přepisovat více než 1 záznam najednou a bude uživatelsky přívětivější když je budou přepisovat přímo do otevřené matriky místo dohledávání, do které matriky záznam patří. Poté, co je záznam vytvořen je připraven ke kontrole jiným uživatelům.

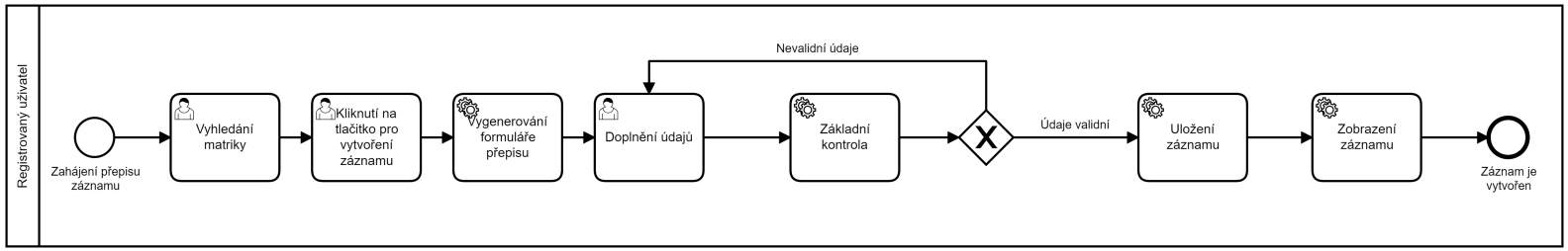
Na obrázku 3.4 je vyobrazen jednoduchý subproces hledání záznamu, který je použit v dalších procesech.

Proces, který je na obrázku 3.5 je paralelní a ukazuje jakým způsobem mohou uživatelé systému kontrolovat záznamy ostatních uživatelů. Proces začíná hledáním záznamu z obrázku 3.4, ze kterého si poté uživatel vybere konkrétní záznam. Uživatel poté vyhodnotí, zda přepis odpovídá originálnímu matričnímu záznamu a poté ho označí za validní či nevalidní.

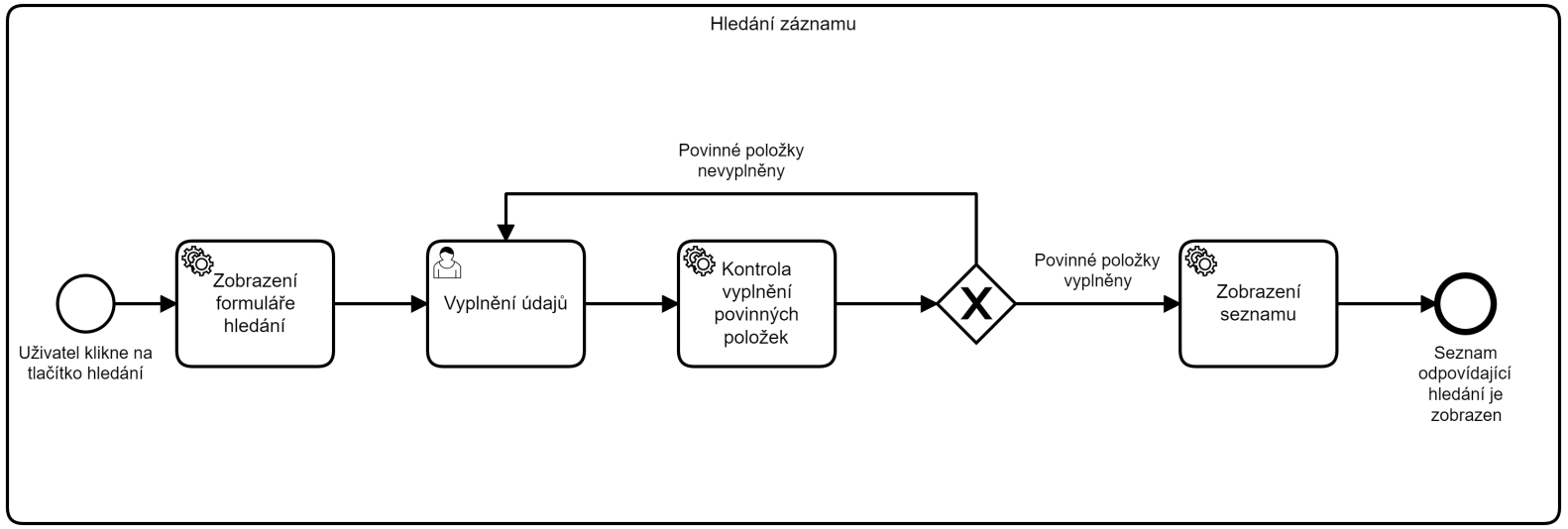
Podobný proces zachycený na obrázku 3.6 ukazuje, jakým způsobem může administrátor manuálně revidovat záznamy. Rozdíl je v tom, že pokud záznam zreviduje administrátor, získá tím záznam na kredibilitě. Dále také, pokud administrátor usoudí, že je přepis nevalidní a označí jej tak, systém ho smaže.

Proces na obrázku 3.7 popisuje jakým způsobem může registrovaný uživatel identifikovat osobu a generovat rodokmen. Proces začne vyhledáním konkrétního záznamu a po jeho zobrazení má uživatel možnost dohledat druhou osobu, která je podle něj shodná. Po dotazu systému a potvrzení uživatele, že se jedná o shodné osoby je systém sloučí.

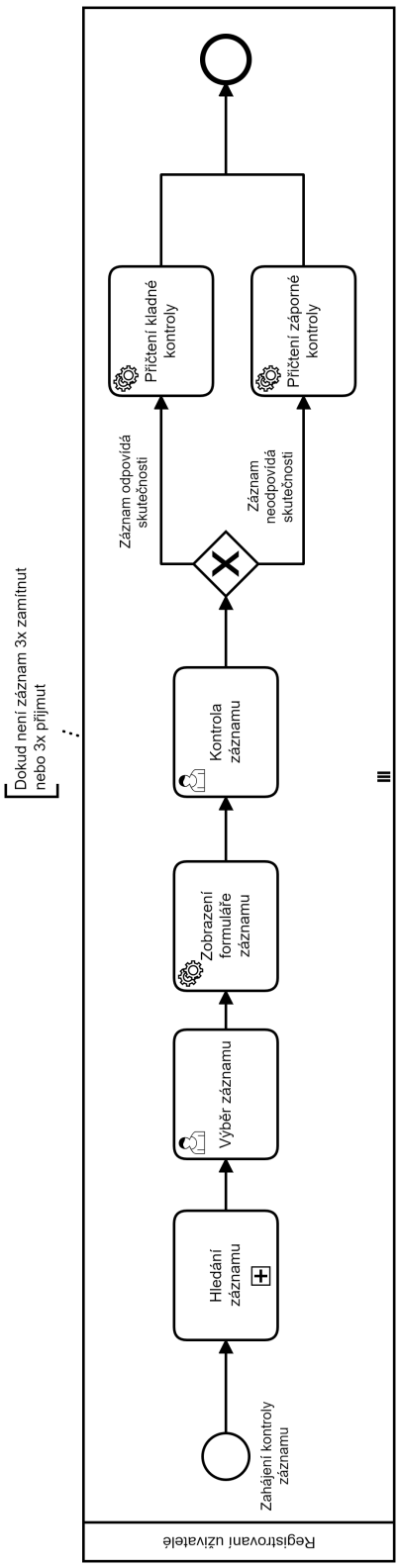
Administrátor systému má pak prostředníctvím procesu na obrázku 3.9 možnost spravovat uživatelské účty. Má možnost smazat všechny uživatelské záznamy v případě, že se jedná o uživatele, který záměrně poškozuje systém. Další možností je uživatele zablokovat, čímž dojde automaticky ke smazání všech záznamů, které daný uživatel vytvořil. Poslední možností je pak povýšení dalšího uživatele do role administrátora.



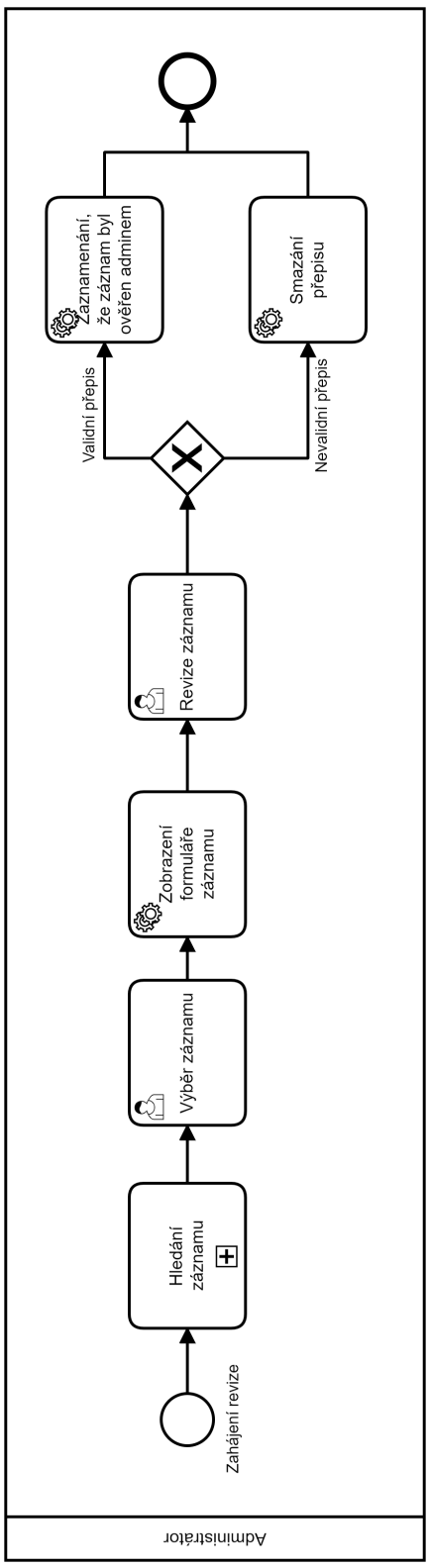
Obrázek 3.3: Vytvoření záznamu



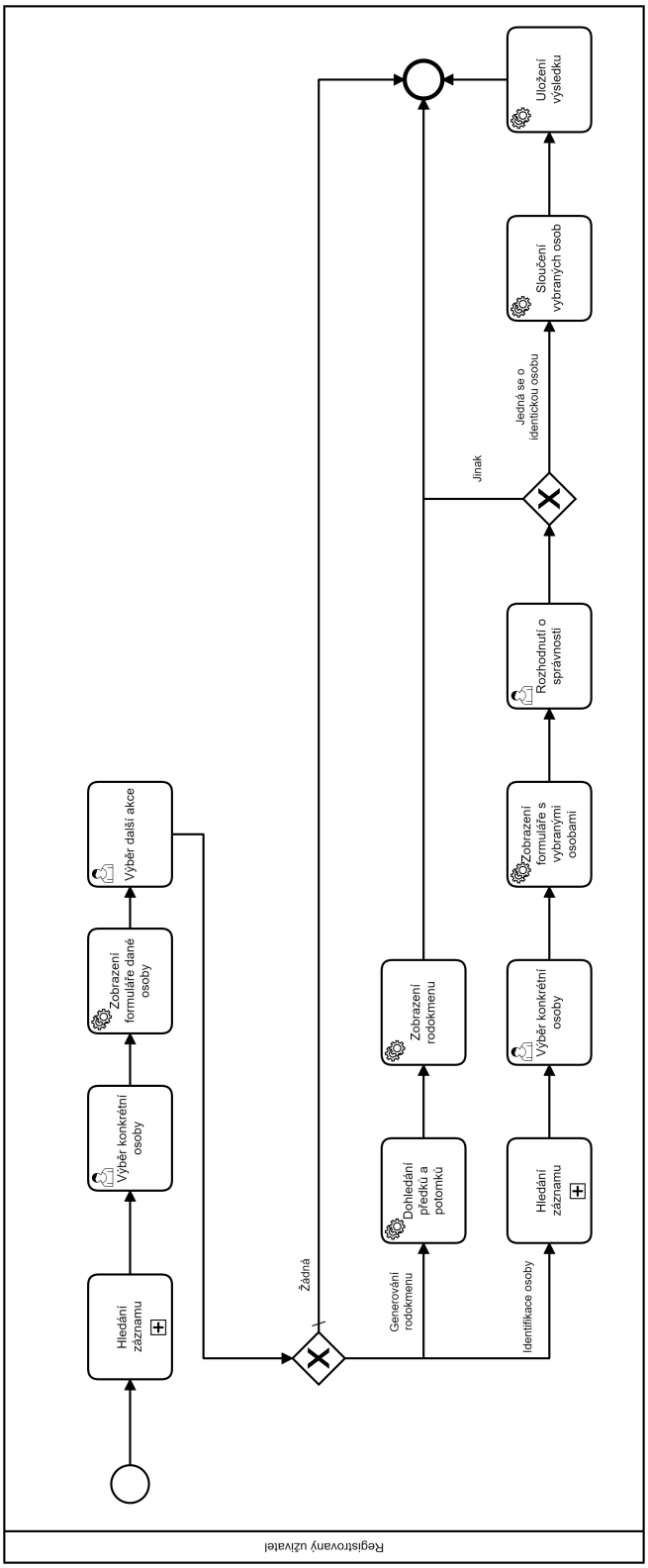
Obrázek 3.4: Hledání záznamu



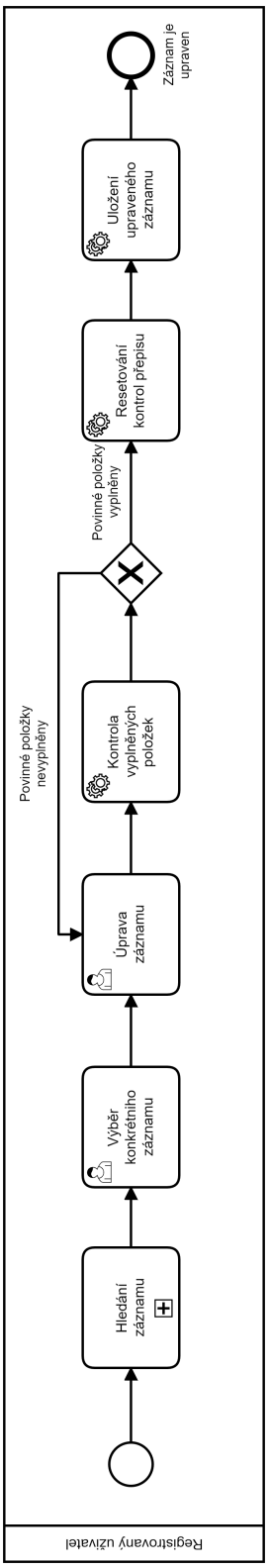
Obrázek 3.5: Kontrola záznamu



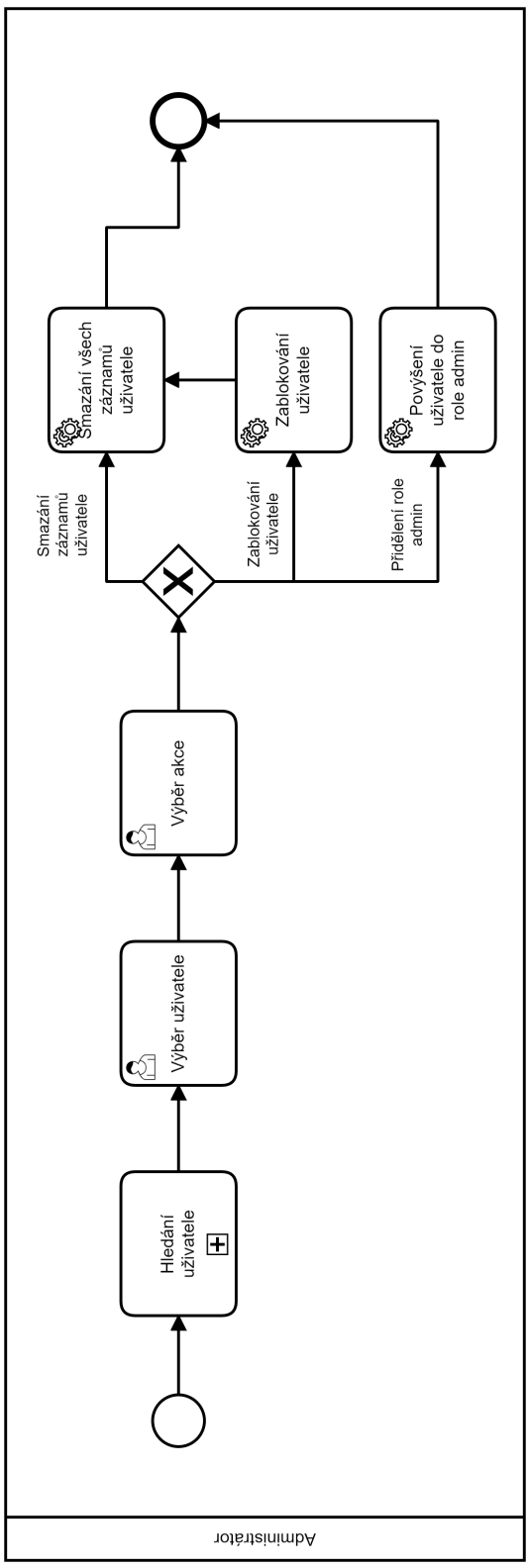
Obrázek 3.6: Revize záznamu



Obrázek 3.7: Identifikace osoby



Obrázek 3.8: Identifikace osoby



Obrázek 3.9: Správa uživatele

3.1.3.3 Možnosti do budoucna

Současný návrh systému počítá dle požadavku zadavatelky s provozem v neziskovém sektoru a všechny funkce jsou dostupné bezplatně, i přesto zde uvedu možnosti monetizace v budoucnu. V současné chvíli existuje několik společností, jejich předmětem podnikání je sestavování rodokmenů na zakázku a částky se pohybují v desítkách tisíc korun [36, 37, 38]. Proto se zde nabízí monetizace funkce, která rodokmen vygeneruje na základě existujících přepisů. Další možností je pak vytvoření nové uživatelské role např. *Placený uživatel*, který by měl tuto funkcionalitu zpřístupněnou. Placený uživatel by také mohl mít možnost přidávat své osobní poznámky k různým přepisům či si svoje generované rodokmeny v aplikaci ukládat a následně upravovat. Detailnímu propočtu se věnuji v kapitole 3.2.6.

3.2 Návrh

3.2.1 Návrh databáze

Největší výzvou, se kterou se bude databáze potýkat jsou požadavky F09 a N01, které mají na databázi vysoké nároky. Logickou otázkou, se kterou je potřeba se tedy vypořádat, je volba databázové technologie.

3.2.1.1 Volba databázové technologie

Jelikož jsou relační databáze v současné době nejpoužívanější a zároveň doménový model obsahuje spousty relací, připadá jako první v úvahu relační databáze [39]. Pokud však uvážím, že požadavek N01 hovoří o 800 milionech záznamů, kde by bylo třeba joinovat několik různých tabulek v rozumném čase, tak má relační databáze poněkud velkou nevýhodu. Dalším omezetím také je, že relační databáze nelze jednoduše škálovat horizontálně. Z výše uvedených důvodů proto není relační databáze pro tento systém vhodná.

Další logickou možností je grafová databáze, která umožňuje rychlé vyhledávání referenčních záznamů. To by byla velká výhoda při generování rodokmenu, kde bude potřeba rekurzivně procházet záznamy a hledat tak předky osob. Primární účel systému je ale přepis matričních záznamů a generování rodokmenů nebude frekventovaně využovaná funkcionalita. Navíc pro využití této funkcionality bude potřeba prvně matriční záznamy přepsat a následně v nich identifikovat shodné osoby. Proto si myslím, že pro toto použití není grafová databáze ta nejvhodnější.

Posledním druhem databáze, nad kterým jsem uvažoval je dokumentová databáze. Ta v tomto případě dává nějvětší smysl. Matriční záznamy jsou v podstatě dokumenty, které obsahují různé typy údajů. Výhodou dokumentové databáze je v tomto případě vnořování objektů, matriční záznam tak může například obsahovat různý počet osob, bez potřeby joinu na jinou tabulku.

Podobně lze přistupovat i k uživatelským kontrolám záznamů a revizím administrátorů. Pomocí návrhových vzorů lze uchovávat revize dokumentů, což splňuje požadavek F09 a lze tak také snadno revertovat změny, které uživatel provedl. Z těchto důvodů, jsem proto zvolil pro systém nerelační dokumentovou databázi.

3.2.1.2 Volba databázového systému

Jelikož je systém navrhován a bude provozován v neziskovém sektoru, hraje v rozhodování podstatou část cena. Dalším významným faktorem volby databázového systému je, zda je systém uzavřený a proprietární. Důležité také je, zda existuje cenově dostupné cloudové řešení a v neposlední řadě se budu rozhodovat na základě dostupné podpory a aktualizací systému spolu s tím, zda kolem systému existuje podpora či komunita vývojářů, kteří jsou schopni pomoci při složitějších problémech.

Všechna zmiňovaná kritéria splňují MongoDB a Apache Cassandra, proto porovnám tyto 2 systémy. Výhodou MongoDB oproti Apache Cassandra jsou sekundární indexy, které mohou být nastaveny na více než 1 sloupec [40, 41]. Toto se bude hodit v případě, že uživatelé budou vyhledávat podle jména a příjmení. Navíc dle [42] je MongoDB oproti Apache Cassandra výkonější téměř ve všech aspektech. Dále také po konzultaci s vedoucím práce a s přihlédnutím k těmto okolnostem jsem se proto rozhodl zvolit databázový systém MongoDB.

3.2.1.3 Databázový model

Na obrázku 3.10 je vytvořený databázový model, který vychází z požadavků, případů užití a doménového modelu.

Největší a nejpodstatnější kolekcí je kolekce *Zaznam*, která reprezentuje matriční záznamy. Dokument také mimo jiné obsahuje ID uživatele, který jej vytvořil a pole obsahující ID osob, které se záznamu týkají. U dokumentů pocházejících z této kolekce je potřeba evidovat úpravy, které uživatelé provedli, proto má složený primární klíč z ID a verze, ve které se záznam nachází. Pokud uživatel provede úpravu záznamu, přesune se z kolekce *Zaznam* do kolekce *ZaznamPredUpravou* a v kolekci *Zaznam* tak vznikne nový dokument. Kolekce záznamů tak bude obsahovat pouze aktuální verze dokumentů. Dokument také obsahuje pole objektů, které uchovávají uživatelské kontroly záznamů. Aby se součet negativních a kladných hodnocení nemusel při každém zobrazení dokumentu přepočítávat, tak je použit návrhový vzor bucket, který uchovává součet z tohoto pole přímo na dokumentu [43].

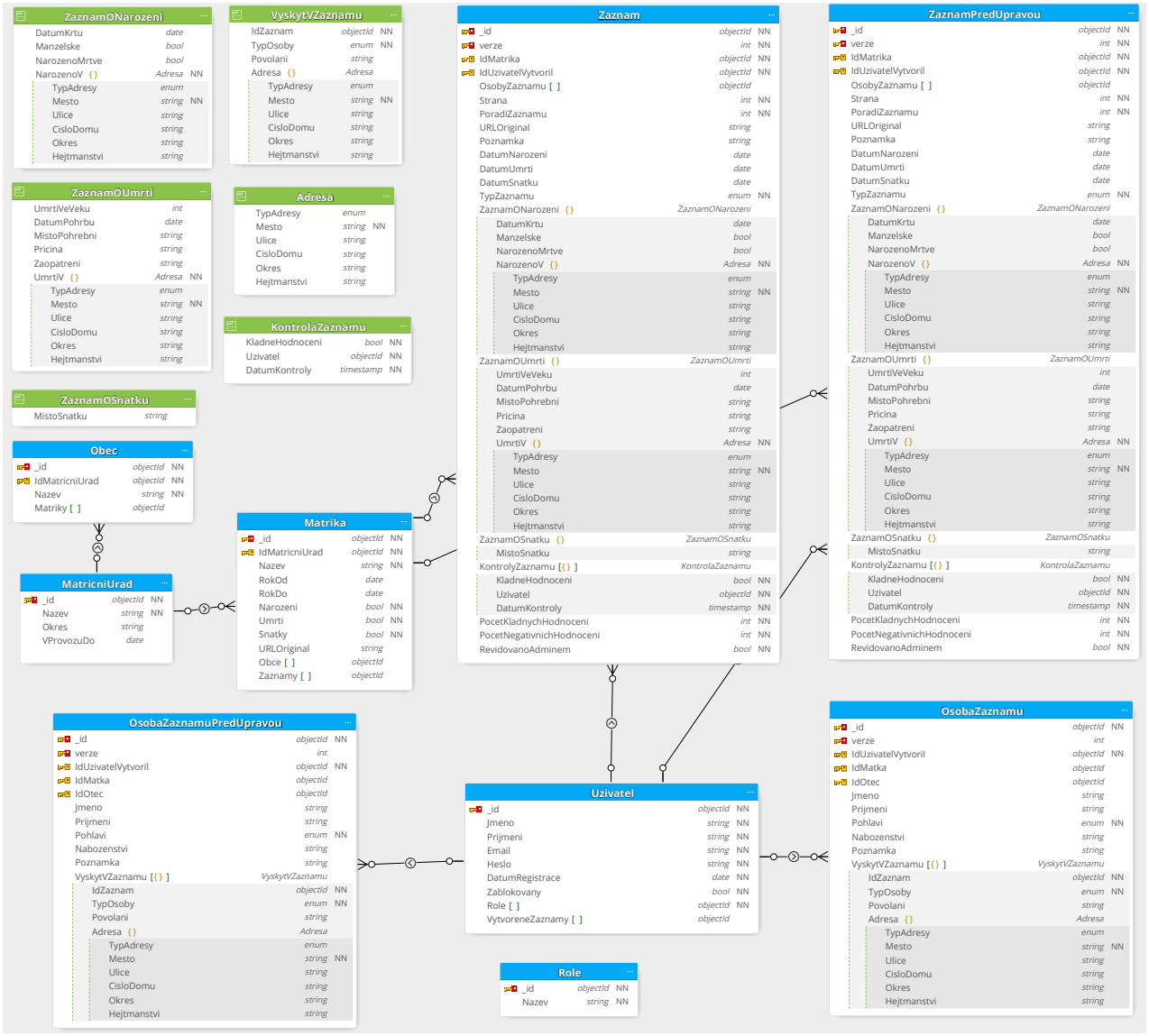
Další významnou kolekcí je kolekce *OsobaZaznamu*, obsahující osoby, které se vyskytují v záznamech. Stejně jako kolekce matričních záznamů má složený primární klíč z ID a verze. Dále také obdobně probíhá proces úprav a zachování originálního dokumentu. Dokument obsahuje pole objektů, které ukazují, ve kterých záznamech je daná osoba vedena. Z počátku bude toto pole obsahovat

pouze 1 položku, protože osoba nebude identifikována. Poté však, co uživatelé osobu identifikují, dojde ke sloučení dokumentů, kde jeden z nich zanikne a do druhého se vloží položka do pole *VyskytVZaznamu*. Identifikovaná osoba tak bude mít více výskytů v záznamech. Číselník *TypOsoby* pak určuje o jakou osobu se v záznamu jedná, zda jde například o novorozeně, otce, matku či svědka. Osoba také během své existence mohla bydlet na různých adresách, proto lze adresu vyplnit jinou pro každý výskyt v záznamu. Dokument osoby záznamu také obsahuje odkaz na otce a matku daného jedince, aby bylo možné snadno splnit případ užití U9, který se týká generování rodokmenu.

Dále je důležitá kolekce uživatelů zahrnující dokumenty, které reprezentují jednotlivé uživatele systému. Mimo běžných údajů o uživatelích obsahuje i číselník role, který určuje v jaké roli uživatel je. V dokumentu se také vyskytuje pole, ve kterém se nachází ID všech záznamů, které uživatel vytvořil, aby je bylo možné v případě potřeby jednoduše dohledat.

Ostatní kolekce reprezentující obce, matriky a matriční úřady, obsahují běžné údaje, které lze najít na zmiňovaném obrázku 3.10 a není potřeba je tedy explicitně popisovat.

POC rešení bude využívat tento návrh databáze. Pokud se však ukáže, že prototyp systému nebude dostatečně rychlý nebo bude zabírat příliš velké množství paměti, je možné návrh databáze upravit. V navazujících pracích, které se budou věnovat kompletní realizaci systému se může ukázat efektivnější místo 2 kolekcí, které uchovávají historii, použít verzování pouze položek, které se změnili. Dojde tak tím k částečné úspoře místa avšak obchodní logika řešící ukládání a verzování může být složitější.



Obrázek 3.10: Databázový model

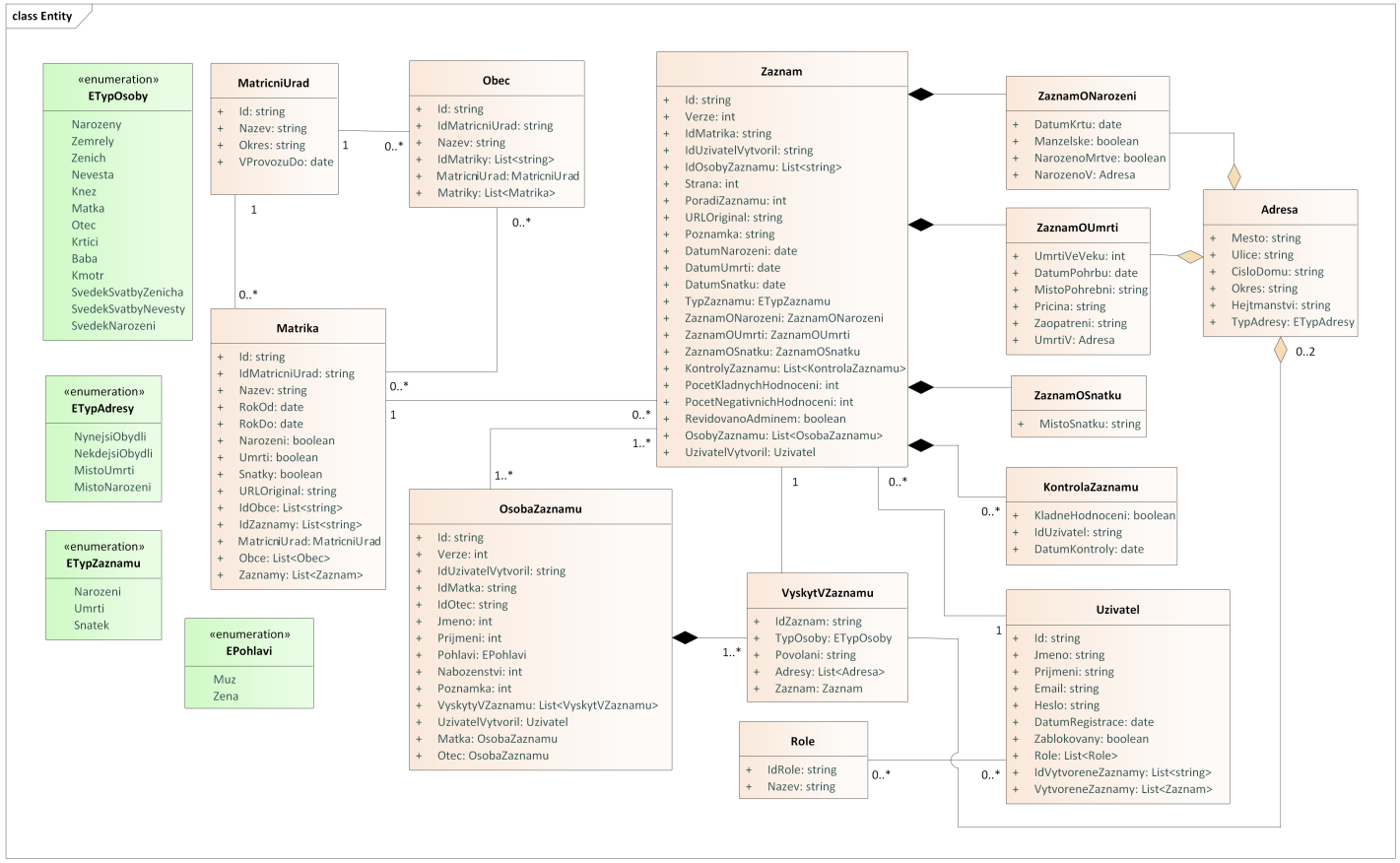
3.2.2 Model tříd

Na základě doménového modelu a návrhu databáze jsem vytvořil model tříd. Jednotlivé třídy reprezentují dokumenty v databázi a v modelu architektury odpovídají entitám v datové vrstvě. Diagram modelu tříd je uveden na obrázku 3.11.

Hlavní třídou je třída *Zaznam*, která reprezentuje matriční záznam a skládá se ze tříd *ZaznamONarozeni*, *ZaznamOUmrti* a *ZaznamOSnatku*. Třída dále obsahuje kolekci osob, které se v záznamu vyskytují. Stejně jako v databázovém modelu jsou osoby vyčleněny do samostatné kolekce, aby nad nimi šlo snadno vyhledávat. Dále třída obsahuje kontroly uživatelů a revize adminů, které potvrzují či vyvrací validitu přepisu. O přidání přepisu se stará služba ve vrstvě obchodní logiky, která se nachází na obrázku B.10 a B.11.

Důležitá je také třída *OsobaZaznamu*, která reprezentuje osoby vyskytující se v záznamu. Každý jedinec má odkaz na svoje rodiče, aby tak šlo snadno vygenerovat rodokmen pro vybraného člověka. Osoba se také může vyskytovat v několika záznamech v různých rolích, a proto má kolekci obsahující objekty třídy *VyskytVZaznamu*. Z počátku bude tato kolekce obsahovat pouze 1 položku, protože v době přepisu se osoba vytvoří duplicitně, avšak po identifikaci dojde ke sloučení vybraných osob a přidání další položky do kolekce.

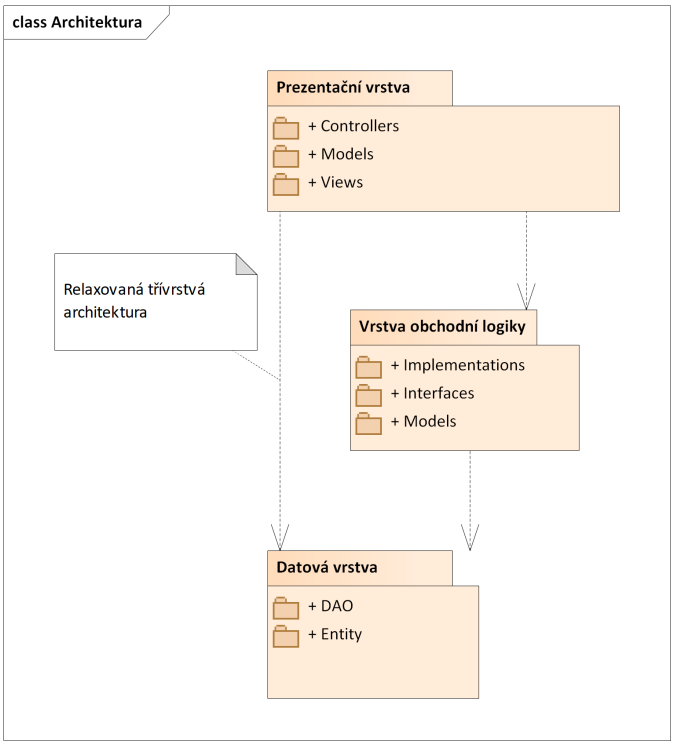
Třídy reprezentující kolekce v databázi, u kterých je potřeba uchovávat historii editací mají vlastnosti Id a Verze. O aktualizaci a vytváření korektní verze je zodpovědná vrstva obchodní logiky, a proto pokud se vytváří nebo upravuje osoba či záznam, je nutné využít metodu, kterou obchodní logika poskytuje.



Obrázek 3.11: Model tříd

3.2.3 Návrh architektury

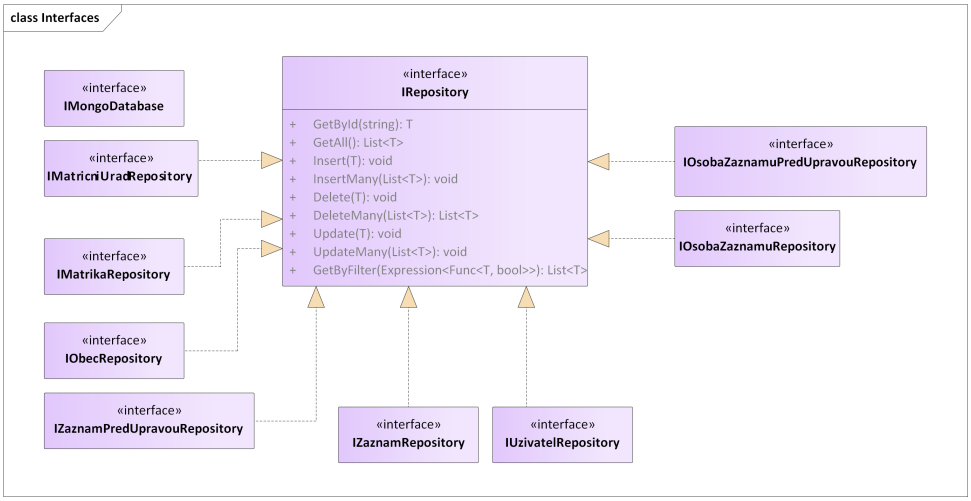
Na základě přechozí analýzy domény, požadavků a vytvořených případů užití jsem vytvořil návrh architektury informačního systému. Rozhodl jsem se pro relaxovanou třívrstvou architekruru, která je vidět na obrázku 3.12. Jednotlivé vrstvy jsou nezávislé na konkrétní implementaci a lze je tak snadno zaměnit při dodržení vystaveného rozhraní, je tedy snadné v případě zájmu vytvořit aplikaci pro jinou platformu implementací jiné prezentační vrstvy. Níže jsou rozepsané detaily o jednotlivých vrstvách a detailní diagramy jsou uvedeny v příloze B.



Obrázek 3.12: Architektura informačního systému

První vrstvou architektury je datová vrstva, která zajišťuje komunikaci s databází. Vrstva obsahuje 2 moduly – Entity a DAO. Entity odpovídají dokumentům v databázi a DAO obsahuje třídy, které odpovídají kolekcím v databázi. DAO je rozdělena na implementaci a rozhraní, aby došlo k oddělení zodpovědností. Obchodní logika, případně prezentační vrstva, používají ke komunikaci výhradně rozhraní a konkrétní implementace je vložena pomocí dependency injection. Na obrázku 3.13 je diagram rozhraní, které DAO poskytuje, a které poté vrstva obchodní logiky a případně prezentační vrstva využívá. V diagramu lze vidět, že je použit generický návrhový vzor repository poskytující základní operace nad kolekcemi v databázi. Výhoda použití vzoru

je v tom, že zapouzdřuje logiku databázových operací a standardizuje přístup k databázi [44].



Obrázek 3.13: DAO rozhraní

Vrstva obchodní logiky, která implementuje veškerou logiku a procesy systému. Obsahuje 3 moduly – rozhraní, implementace a modely. Podobně jako v datové vrstvě používá prezentační vrstva pouze rozhraní a implementace mu bude vložena skrze dependency injection. Každý kontroler z prezentační vrstvy používá 1 rozhraní z vrstvy obchodní logiky, které mu poskytuje veškerou logiku. Modul poskytující prezentační vrstvě modely z obchodní logiky existuje z toho důvodu, že neexistuje entita, která by efektivně reprezentovala rodokmen. V databázi má osoba odkaz na své rodiče, ale pro rodokmen je potřeba znát choť osoby a jejich potomky, proto vrstva obchodní logiky naplní objekt *OsobaRodokmenu*, který obsahuje objekt *Manželství* a kolekci *Potomci* a prezentační vrstva pak tento objekt snadno vykreslí.

Vrstva obchodní logiky používá návrhový vzor unit of work, který zaobaluje používání repozitářů do 1 kontextu a pomáhá sledovat provedené změny. Další výhodou vzoru je snadná testovatelnost funkcionalit, protože je pro testování možné nahradit skutečný objekt za mockovaný objekt. V neposlední řadě je také využit pro zajištění konzistence při vkládání či upravování objektů z různých kolekcí [45].

Poslední vrstvou je prezentační vrstva, která využívá vrstvu obchodní logiky a datovou vrstvu. Prezentační vrstva je zamýšlena jako webová aplikace, která využívá návrhový vzor MVC a skládá se ze 3 částí.

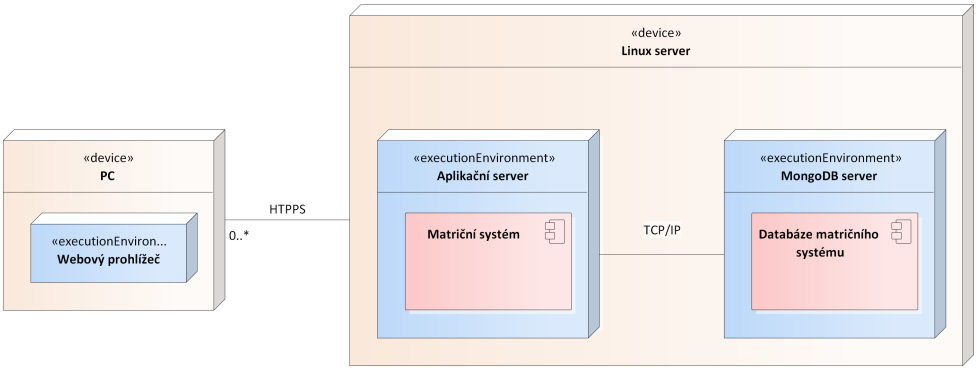
První částí prezentační vrstvy jsou kontrolery, které celou vrstvu ovládají. Kontrolery volají služby z obchodní logiky a v případě, že je to zapotřebí se namapují na získaný objekt z obchodní logiky resp. datové vrstvy na model, který je poté předán na zobrazení do view. Podobně pak kontro-

lery překládají modely z view na objekt, který se předá do obchodní logiky ke zpracování. Každý z kontrolerů je zodpovědný za obsluhu endpointu odpovídající jeho názvu např. OsobaController je zodpovědný za obsluhování https://www.example.cz/osoba.

3.2.4 Provoz a škálování

Vzhledem k tomu, že se jedná o neziskový projekt je nejlepší variantou využít pro provoz cloudové řešení. Cloudová řešení nabízí automatické škálování jak nahoru, tak dolů dle aktuální vytíženosti, což je pro tento případ ideální. Navíc je také dostupné automatické zálohování databáze, a není proto potřeba databázový specialista, který by se tomu věnoval. Detailní ceny a rozvahu popisuji v kapitole 3.2.6.

Na obrázku 3.14 je zobrazen diagram nasazení, kde je viditelné, že aplikace je nasazena jako webová aplikace na aplikačním serveru, ke kterému se uživatele připojují pomocí protokolu HTTPS. Aplikace na aplikačním serveru pak komunikuje s databázovým serverem, na kterém je umístěna databáze systému.

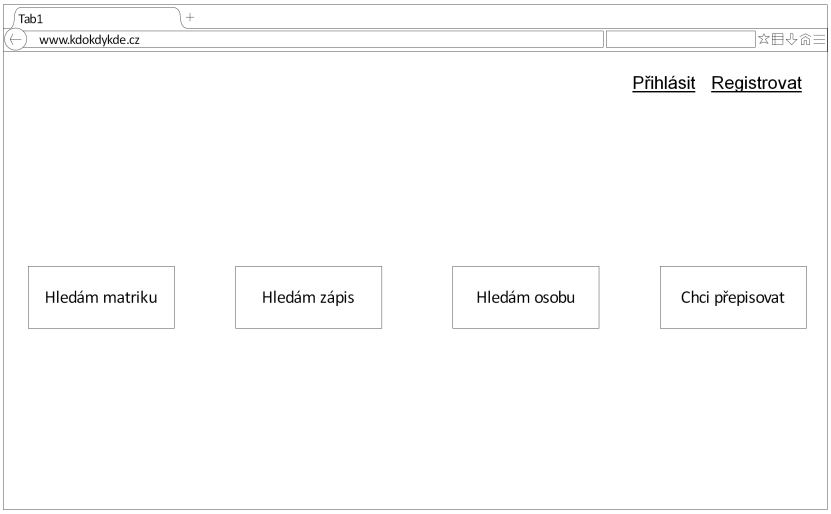


Obrázek 3.14: Model nasazení systému

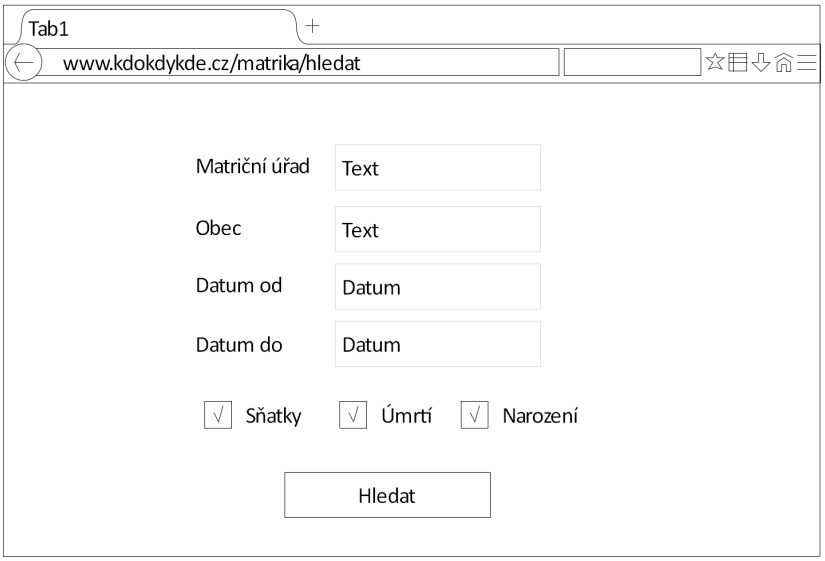
3.2.5 Návrhy obrazovek

Na základě poskytnutých materiálů a konzultací se zadavatelkou jsem vytvořil následující návrhy obrazovek. Uvedené návrhy nepředstavují všechny obrazovky vyskytující se v systému, ale pouze ty, jenž jsou pro běžné používání podstatné a demonstrují nejběžnější úkony.

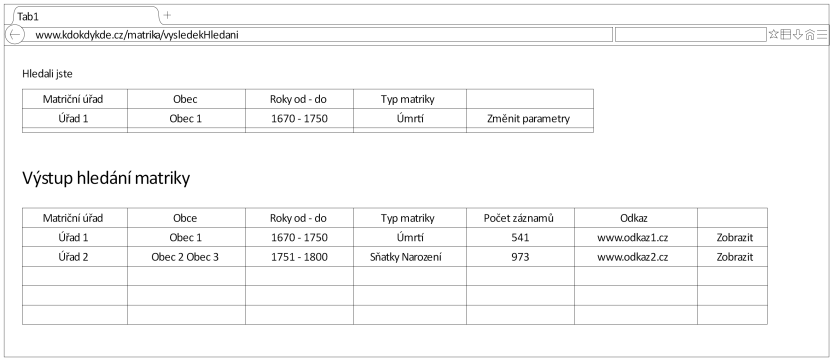
Obrázek 3.20 zobrazuje obrazovku pro hledání záznamu narození, hledání ostatních typů záznamů by probíhalo obdobně. Stejně tak hledání osoby a výsledek hledání osoby zde není vytvořen, protože je téměř identický k hledání matriky.



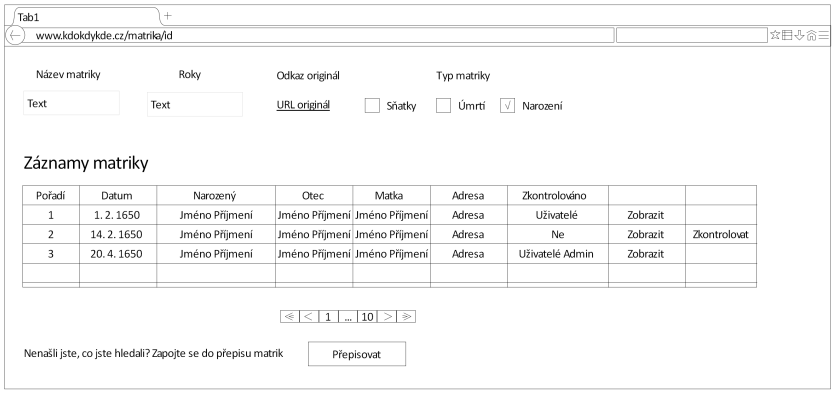
Obrázek 3.15: Úvodní obrazovka



Obrázek 3.16: Hledání matriky

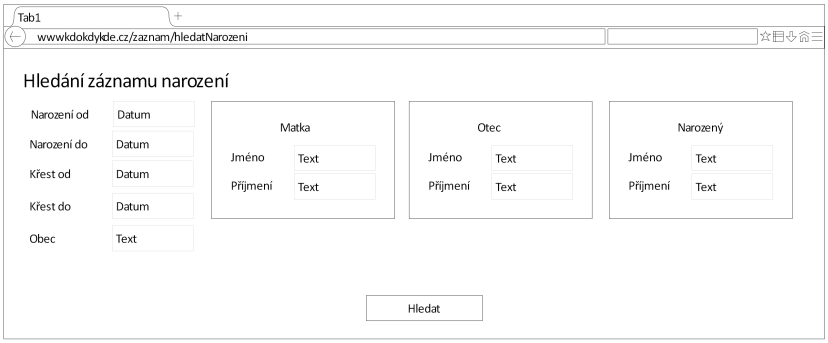


Obrázek 3.17: Výsledek hledání matriky

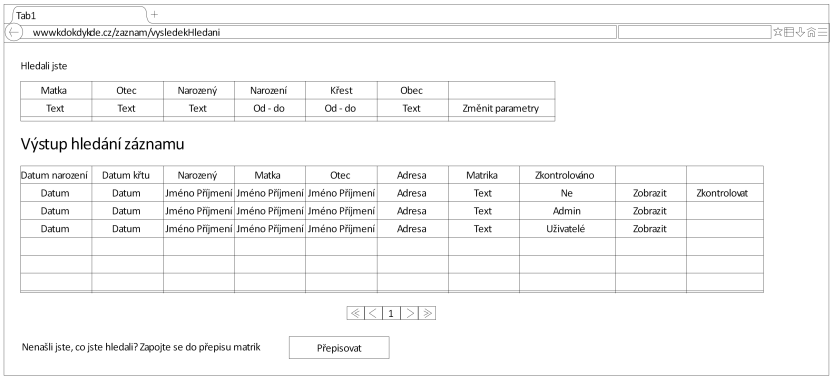


Obrázek 3.18: Detail matriky

Obrázek 3.19: Rozcestník hledání záznamu



Obrázek 3.20: Hledání záznamu



Obrázek 3.21: Výsledek hledání záznamu

3.2.6 Manažerské zhodnocení

V současné verzi návrhu a POC řešení je systém uvažován jako neziskový a vyvýjen pro neziskový sektor. Zadavatelka však zmiňovala, že do budoucna plánuje systém rozšířit a pozměnit tak, aby se některé funkce systému daly zpeněžit. Proto v této kapitole vytvořím model zpeněžení, který by systém uvedl do zisku.

Prvním krokem v tvorbě modelu zpeněžení je propočet kolik stojí provoz a údržba systému. Vzhledem k tomu, že součástí této diplomové práce je analýza, návrh a POC řešení, které má implementované téměř všechny klíčové procesy a lze očekávat, že bude v navazujících bakalářských pracích systém dokončen, nebudu s náklady na vývoj systému ve finanční kalkulaci počítat.

Podstatný náklad na provoz systému bude tvořit databáze, jelikož požadavek N01 hovoří o 800 milionech záznamů. S přihlédnutím k tomu, že systém musí uchovávat historii editací jednotlivých uživatelů bude toto číslo ještě větší. Systém však nebude obsahovat všechny záznamy od začátku, ale záznamy se budou postupně plnit. I proto volím pro kalkulaci cloudové řešení umožňující škálování dle aktuální potřeby. V kalkulaci vycházím z toho, že

velikost 1 záznamu včetně osob, které se v záznamu vyskytují je zhruba 2 kB. Za předpokladu přepsání všech záznamu a dosažení tak 800 milionů zápisů bude zapotřebí zhruba 1,5 TB místa. Současný ceník MongoDB začíná na $0,08/hodina za 8 GB dedikovaného clusteru, který z počátku bude dostatečný. Dále také nabízí automatické škálování, které bude po dosažení horního limitu potřebné, až do odhadované velikosti 1,5 TB. Nejbližší vyšší kapacita databáze, kterou nabízí je velikost 2 TB za cenu $3/hodina [46].

Dalším nákladem na provoz je server, na kterém je systém umístěn. I zde volím cloudové řešení, které je v tomto případě výhodné kvůli jednoduché škálovatelnosti a nízkým nákladům na údržbu. Pro modelový případ jsem zvolil Microsoft (MS) Azure. Jelikož jsem POC řešení vytvořil jako .NET CORE 3.1 aplikaci, lze ji distribuovat jak na Windows server, tak linuxové servery. V kalkulaci v tabulce 3.22 tak počítám s levnějším nasazením na linuxový server. Azure nabízí několik provozních plánů, pro tento systém však z počátku postačí instance B1, která aktuálně stojí 11 € měsíčně [47].

Posledním z uvažovaných nákladů je alokace pro údržbu a rozvoj systému. Pro systém této velikosti budu uvažovat 1 Manday (MD) na měsíc, který je v případě nevyužití přesouván do měsíců dalších. Detailnější propočet je vidět taktéž v tabulce 3.22.

Tabulka 3.22: Propočet nákladů

Měsíc	*Počet uživatelů*	*Velikost DB (GB)*	*Cena DB (Kč)*[1]	*Cena hosting (Kč)*[2]	*Cena MD (Kč)*	*Kumulativní náklady (Kč)*
1	1000	0,04	1265	290	4000	5555
3	3000	0,23	1265	290	4000	16665
6	6000	0,80	1265	290	4000	33330
9	9000	1,72	1265	290	4000	49995
12	12000	2,98	1265	290	4000	66660
15	15000	4,58	1265	290	4000	83325
18	18000	6,52	1265	290	4000	99990
21	21000	8,81	1440	290	4000	117005
24	24000	11,44	1440	290	4000	134195

Dalším krokem v tvorbě modelu zpeněžení je výběr funkcí, které budou zpoplatněné. Jelikož se jedná o crowd-sourcing systém, není možné zpoplatnit funkce, které jsou pro systém prospěšné. Takové funknce jsou například přepisování, upravování a kontroly záznamů. Pokud by se tyto klíčové funkce umístili za platební bránu, došlo by k výraznému poklesu přepsaných a schválených záznamů. Funkce s přidanou hodnotou, které je možné dle mého názoru zpoplatnit je hledání konkrétní osoby a především generování rodokmenu.

[1] cena přepočtena z USD dle aktuálního kurzu
[2] cena přepočtena z EUR dle aktuálního kurzu

V navazujících verzích systému je možné přidat další placené funkce, pro jednoduchost je však nebudu uvažovat. V tabulce 3.23 je propočet modelu zpeněžení, kde počítám s tím, že monetizace je formou subskripce za cenu 49 Kč měsíčně za předpokladu, že subskripci bude využívat alespoň 20 uživatelů z 1000 registrovaných. V tabulce je vidět, že bod zvratu nastává po 10. měsíci kdy se systém dostává do černých čísel a sám si na sebe vydělá.

Tabulka 3.23: Propočet výnosů

Měsíc	*Platící uživatelé*	*Měsíční výnos (Kč)*	*Kumulativní výnos (Kč)*	*Kumulativní zisk (Kč)*
1	20	980	980	-4575
3	60	2940	5880	-10785
6	120	5880	20580	-12750
9	180	8820	44100	-5895
12	240	11760	76440	9780
15	300	14700	117600	34275
18	360	17640	167580	67590
21	420	20580	226380	109375
24	480	23520	294000	159805

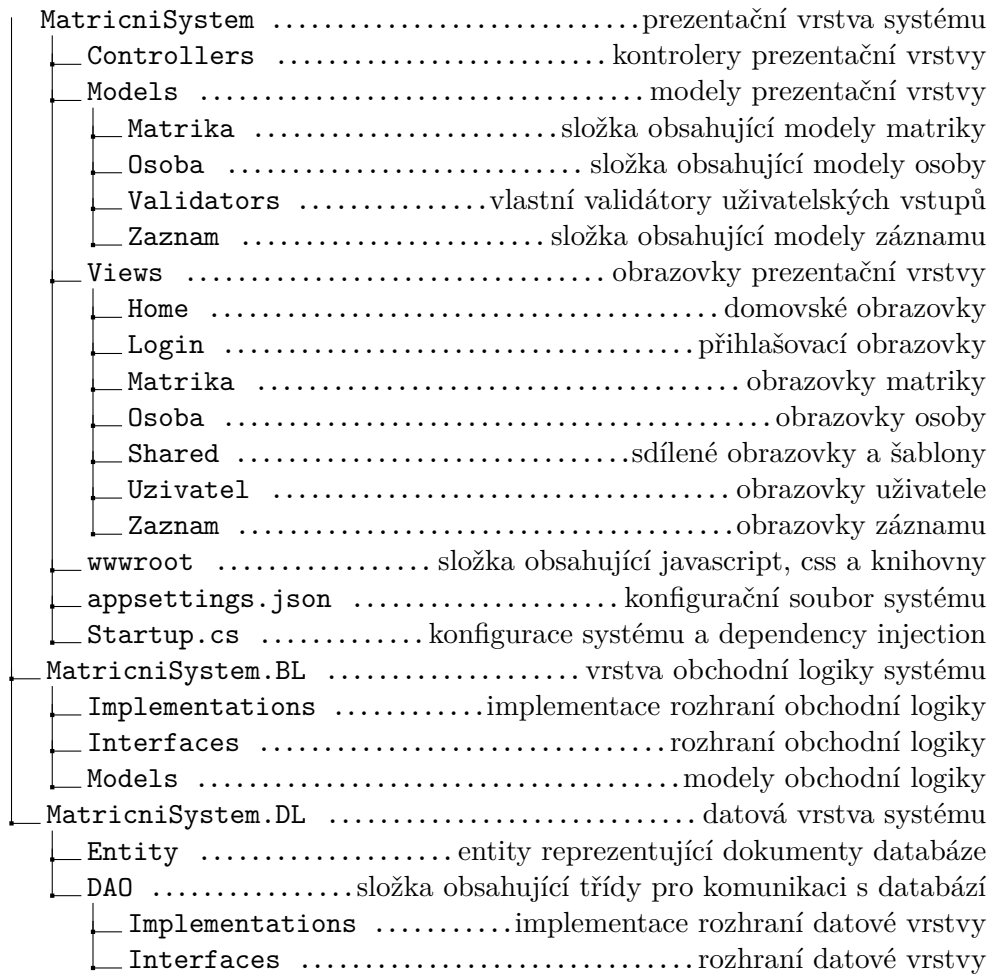
3.3 Proof-of-concept řešení

3.3.1 Popis implementace

Pro implementaci POC řešení jsem se rozhodl použít .NET Core 3.1 a programovací jazyk C#. Pro volbu této technologie jsem se rozhodl proto, že v C# již programovat umím a je to můj oblíbený programovací jazyk. Dalším důvodem je, že .NET Core 3.1, je multiplatformí s dlouhodobou podporou. Vzhledem k tomu, že v porovnání např. s Java EE jsou technologie srovnatelné a nelze posoudit, která z nich je z hlediska výkonu lepší, neexistuje důvod, proč .NET Core oproti Javě nezvolit. Zvolená technologie taktéž bez obtíží zvládne obsloužit odhadovaný počet uživatelů [48].

Systém, který jsem implemetoval vychází z návrhu a analýzy v předchozích kapitolách, proto zde nebudu přidávat diagramy, které jsou duplicitní. Na obrázku 3.22 je uvedena adresářová struktura implementovaného systému spolu s vysvětlivkami.

Pro prezentační vrstvu jsem použil projekt ASP.NET Core MVC, který využívá klasického návrhového vzoru MVC. Pro zobrazení tabulek resp. gridů používám knihovnu NonFactors.Grid.Mvc5, která je šířena pod licencí MIT. Knihovna slouží ke grafickému vykreslení seznamů, ve kterých lze poté vyhledávat a filtrovat, čímž uživateli usnadní vyhledávání požadovaného záznamu. Dále pro vykreslení rodokmenu používám knihovnu dTree, která je taktéž šířena pod licencí MIT. V prezentační vrstvě využivám šablony umožňující



Obrázek 3.22: Adresářová struktura systému

nastavit jak se má daný objekt vykreslit. Pro každý objekt, který se v prezentační vrstvě zobrazuje na více místech mám tak vytvořenou šablonu, aby se předešlo duplikaci kódu. Do šablony jsem přidal parametr, který specifikuje, zda je zobrazený objekt pouze pro čtení, nebo lze i upravovat.

V souboru *Startup.cs* je nakonfigurována dependency injection což znamená, že pokud některá část systému potřebuje ke svému fungování jinou službu, tak mu bude vložena pomocí konstruktoru při vytvoření daného objektu. Ve stejném souboru je pak konfigurace identity, což je interní mechanismus pro používání autorizace na základě rolí. Ta je pak využita v kontrolerech resp. metodách kontrolerů, které jsou určeny pouze pro uživatele v roli admin. Akce, které jsou dostupné pouze administrátorům jsou zmapovány v předešlé kapitole 3.1.2.

Pro vrstvu obchodní logiky jsem zvolil obyčejný projekt, který je kompi-

lován jako dll knihovna a prezentační vrstva jej tak využívá. Obchodní logika poskytuje prezentační vrstvě rozhraní, která používá a konkrétní implementace je dodána pomocí dříve zmiňovaného dependency injection. Vzhledem k tomu, že zvolená databázová technologie MongoDB nepodporuje kaskádní operace, slouží obchodní logika mimo jiné i k zajištění konzistence databáze. Pokud prezentační vrstva potřebuje upravit nebo vytvořit dokument do databáze, musí k tomu využít služby, které vrstva obchodní logiky poskytuje. Ve vrstvě obchodní logiky je implementována logika vytváření a úprav dokumentů, tak aby databáze zůstala konzistentní a po úpravě či vytvoření byly nastaveny potřebné vazby.

Vrstva obchodní logiky využivá při práci s databází návrhového vzoru unit of work, který zajišťuje, že pro přístup do databáze bude využito 1 fyzické spojení. Také je pomocí vzoru možné vytvářet transakce napříč kolekcemi.

Dále vrstva obsahuje modely nereprezentující dokument v databázi, ale jsou potřebné pro logiku některých operací. Příkladem je generování rodokmenu, kdy osoba v databázi má odkaz na své rodiče, ale pro tvorbu rodokmenu je potřeba znát choť osoby a jejich potomky. Proto existují modely obchodní logiky, které rodokmen reprezentují a prezentační vrstva je pak může snadno zobrazit.

Poslední vrstvou je datová vrstva, ve které se nachází entity reprezentující dokumenty v databázi a DAO, které slouží ke komunikaci s databází. V datové vrstvě je použít generický návrhový vzor repository, ke kterému jsem přidal další generický parametr pro specifikaci, jakého typu je id dané entity. Entity, u kterých je potřeba uchovávat historii editací mají místo běžného id, objekt, ve kterém se navíc nachází informace o verzi dané entity. Rozhraní repozitáře je zobrazeno na obrázku 3.23.

Obrazovky z POC řešení jsou součástí této práce v příloze D a implementovaný systém je přiložen na médiu připojeném k této práci.

```
public interface IRepository<T,K> where T : class
{
    Task Insert(T entity);
    Task InsertMany(IEnumerable<T> entity);
    Task Update(T entity);
    Task UpdateMany(IEnumerable<T> entity);
    Task Delete(T entity);
    Task DeleteMany(IEnumerable<T> entity);
    Task<T> GetById(K id);
    Task<IEnumerable<T>> GetByFilter(Expression<Func<T, bool>> filter);
    Task<IEnumerable<T>> GetAll();
    IMongoCollection<T> GetCollection();
}
```

Obrázek 3.23: Generický repozitář

3.3.2 Implementované části

V systému jsem implementoval všechny podstatné částí, které jsou potřebné pro testovací provoz a prokázání, že návrh je validní. V současné verzi není

implementována řádná úprava uživatele, která je součástí U18, a proto nelze změnit roli uživatele jinak, než zásahem do databáze. Pro nynější potřebu je to dostatečné, avšak pro plnohodnotné fungování systému je potřeba tuto funkcionalitu zavést. Další částečná implementace se týká U1 – Registrace, kdy je uživatel zaregistrován ihned, bez potřeby potvrdit e-mailovou adresu. Podobně jako v předchozím případě to nemá vliv na otestování návrhu a je snadné funkcionalitu doprogramovat do plnohodnotné verze. Posledním zjednodušením je to, že po stisku tlačítka se systém nezeptá uživatele na potvrzení, tak jak bylo popsáno v jednotlivých případech užití.

Mimo popsaná zjednodušení jsou implementovány všechny případy užití a systém je možné testovat.

3.3.3 Zhodnocení

Po technické stránce je systém funkční a splňuje všechny případy užití. Výše zmiňovaná podoba systému byla odprezentována zadavatelce, která byla s výsledkem po technické stránce spokojena. Co se vzhledu týče, zmiňovala některé grafické nedostatky, se kterými plně souhlasím. Vzhled v tomto případě nebyl cílem a přívětivé uživatelské rozhraní bude implementováno v navazujících bakalářských pracích, včetně výše popsaných nedodělků.

V navazujících pracích by bylo vhodné také optimalizovat některé databázové dotazy a algoritmy, které jsou v systému použity. Pro POC řešení jsem použil pouze triviální a neoptimalizované dotazy, které by při velkém vytížení mohly způsobovat zbytečnou zátěž jak aplikačního serveru, tak databáze.

Závěr

Hlavním cílem této práce byla analýza, návrh a POC systému umožňující uživatelům přepisovat záznamy z matričních knih. V první části práce jsem provedl literární rešerši pojmů seznamující čtenáře s problematikou probíranou v navazujících částech. Dále jsem se zabýval analýzou současných řešení systémů, kde lze hledat a vytvářet matriční záznamy a také generovat rodokmeny.

Posléze navázala analýza domény matrik a na základě jednání se zadavatelkou byly sestaveny požadavky na systém a tvorba případu užití. Poté jsem vytvořil doménový model a hlavní procesy, které se v systému vyskytují. Pro databázi jsem zvolil databázovou technologii MongoDB a vytvořil její návrh a na základě předchozí analýzy navrhl model tříd popisující doménu.

Následoval návrh architektury systému včetně způsobu provozu a škálování, který formou crowd-sourcingu umožňuje uživatelům přepisovat matriční záznamy. U přepsaných záznamů pak mohou uživatelé kontrolovat validitu, čímž záznam získá na kredibilitě. Jelikož během přepisování záznamu uživatel neví, zda osoba, kterou se právě zabývá, již v databázi existuje mohou vznikat duplicitní záznamy. Z tohoto důvodu existuje způsob, kterým je možné tyto osoby identifikovat a následně sloučit. U identifikovaných osob nabízí systém možnost generovat rodokmeny vycházející z matričních záznamů.

Mimo jiné jsem v práci také navrhl model zpeněžení, kde jsem zmínil funkce vhodné k monetizaci. Cenovou politiku jsem vytvořil formou měsíční subskripce, kde nastává bod zvratu, při zachování odhadovaného růstu počtu uživatelů, po 10 měsících.

Součástí práce je i POC řešení implementované na platformě .NET ukazující, že je návrh systému validní a lze jej zkonstruovat.

V průběhu tvorby byly jednotlivé části analýzy a návrhu konzultovány se zadavatelkou a následně upraveny dle jejích požadavků a připomínek. POC řešení pak bylo odprezentováno a přijato bez faktických výhrad. Jediné výhrady se týkaly vzhledu systému, který ovšem nebyl cílem práce a tento nedostatek mohou případně vyřešit navazující odborné práce.

Stěžejním výsledkem práce jsou poklady pro budoucí implementaci systému, na které mohou navázat bakalářské práce.

Bibliografie

1. PINE, Leslie Gilbert. *Genealogy: anthropology* [online] [cit. 2021-04-09]. Dostupné z: `https://www.britannica.com/topic/genealogy`.
2. *Pedigree: genetics* [online] [cit. 2021-04-21]. Dostupné z: `https://www.britannica.com/science/pedigree-genetics`.
3. PETERKA, Josef. *Cesta k rodinným kořenům, aneb, Praktická příručka občanské genealogie*. Praha: Libri, 2006. ISBN 80-727-7307-0.
4. BARTŮNĚK, Václav. Historický vývoj matrik. *Časopis Rodopisné společnosti v Praze*. 1940, roč. 12, č. 1, s. 6–17.
5. *Zákon č. 380/2008 Sb. Zákon o matrikách, jménu a příjmení a o změně některých souvisejících zákonů*. [B.r.]. ISSN 1211-1244.
6. THE STANDISH GROUP. *The CHAOS Report*. 1994. The Standish Group. Dostupné také z: `https://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf`.
7. ARLOW, Jim; NEUSTADT, Ila. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.
8. BITTNER, Kurt; SPENCE, Ian; JACOBSON, Ivar. *Use Case Modeling*. Addison Wesley, 2003. The Addison-Wesley object technology series. ISBN 978-0-201-70913-1.
9. SCHNEIDER, Geri; WINTERS, Jason P. *Applying Use Cases: A Practical Guide* [online]. 2. vyd. Pearson Education, 2001 [cit. 2021-04-11]. ISBN 9780789745453.
10. LARMAN, C.; KRUCHTEN, P. *Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and the Unified Process*. Prentice Hall PTR, 2002. Safari electronic books. ISBN 9780130925695.
11. *Domain Modeling* [online] [cit. 2021-04-13]. Dostupné z: `https://www.scaledagileframework.com/domain-modeling/`.

12. *Podnikový proces: Co je Podnikový proces* [online] [cit. 2021-04-14]. Dostupné z: https://managementmania.com/cs/business-process-podnikovy-proces.

13. COUSINS, Michael. *As Is -To Be: The Essential Business Model for Process Improvement* [online] [cit. 2021-04-15]. Dostupné z: https://blog.triaster.co.uk/blog/as-is-to-be-essential-business-model-process-improvement.

14. *Information technology: Object Management Group Business Process Model and Notation*. 2013. ISO/IEC, 19510:2013.

15. *Encyclopedia Britannica* [online] [cit. 2021-04-10]. Dostupné z: https://www.britannica.com/technology/database.

16. NAYAK, Ameya; PORIYA, Anil; POOJARY, Dikshay. Type of NOSQL Databases and its Comparison with Relational Databases. *International Journal of Applied Information Systems* [online]. 2013, roč. 5, č. 4 [cit. 2021-04-10]. ISSN 2249-0868. Dostupné z: https://www.researchgate.net/publication/302557703_Article_Type_of_nosql_databases_and_its_comparison_with_relational_databases.

17. ROBINSON, Ian; WEBBER, Jim; EIFREM, Emil. *Graph Databases: New Opportunities for Connected Data*. O'Reilly Media, 2015. ISBN 9781491930892.

18. *Database: What Is a Relational Database?* [Online] [cit. 2021-04-10]. Dostupné z: https://www.oracle.com/database/what-is-a-relational-database/.

19. KUNDA, Douglas; PHIRI, Hazael. A Comparative Study of NoSQL and Relational Database. *Zambia ICT Journal* [online]. 2017-12-11, roč. 1, č. 1, s. 1–4 [cit. 2021-04-10]. ISSN 2616-2156. Dostupné z DOI: 10.33260/zictjournal.v1i1.8.

20. SINGARAM, Muthu; JAIN, Prathistha. *Prototypes: What is the Difference between Proof of Concept and Prototype ?* [Online] [cit. 2021-04-11]. Dostupné z: https://www.entrepreneur.com/article/307454.

21. BRABHAM, Daren C. *Crowdsourcing*. MIT Press, 2013. ISBN 9780262314251.

22. *1860 United States Federal Census: Source Information* [online] [cit. 2021-04-16]. Dostupné z: https://www.ancestry.com/search/collections/7667/.

23. *History: Decennial Census* [online] [cit. 2021-04-16]. Dostupné z: https://www.census.gov/history/www/programs/demographic/decennial_census.html.

24. *Ancestry: Subscribe* [online] [cit. 2021-04-16]. Dostupné z: https://www.ancestry.com/cs/offers/subscribe.

25. *Ancestry Surpasses 15 Million Members in its DNA Network, Powering Unparalleled Connections and Insights* [online]. 2019-05-21 [cit. 2021-04-16]. Dostupné z: https://www.ancestry.com/corporate/newsroom/press-releases/ancestry-surpasses-15-million-members-its-dna-network-powering-unparalleled.

26. *Prohledejte historické záznamy* [online] [cit. 2021-04-16]. Dostupné z: https://www.myheritage.cz/research.

27. *Rodokmen* [online] [cit. 2021-04-16]. Dostupné z: https://www.myheritage.cz/family-tree.

28. *Paywall* [online] [cit. 2021-04-16]. Dostupné z: https://www.myheritage.cz/paywall.

29. TAKANO, Tomohiro. *Product reviews: 6 things you should know before buying MyHeritage* [online] [cit. 2021-04-16]. Dostupné z: https://blog.genomelink.io/posts/myheritage-review-by-experts.

30. *FamilySearch: About* [online] [cit. 2021-04-20]. Dostupné z: https://www.familysearch.org/about/.

31. *Welcome to Gramps* [online] [cit. 2021-04-19]. Dostupné z: https://www.gramps-project.org/wiki/index.php/Main_page.

32. *Webtrees: Web based family history software* [online] [cit. 2021-04-20]. Dostupné z: https://www.webtrees.net/index.php.

33. *Kniha narozených: Sepekov 1902–1914* [online] [cit. 2021-02-16]. Dostupné z: https://digi.ceskearchivy.cz/11512/5/3568/1115/39/0.

34. *Kniha zemřelých: Sepekov 1894–1930* [online] [cit. 2021-02-16]. Dostupné z: https://digi.ceskearchivy.cz/7183/06/1619/790/56/0.

35. *Kniha oddaných: Sepekov 1922-1934* [online] [cit. 2021-02-16]. Dostupné z: https://digi.ceskearchivy.cz/11515/11/4997/1556/47/0.

36. *K pramenům: Ceník* [online] [cit. 2021-02-16]. Dostupné z: http://kpramenum.cz/cenik/.

37. *Rodinný příběh: Ceník* [online] [cit. 2021-02-16]. Dostupné z: https://rodinny-pribeh.cz/cenik/.

38. *Hledání předků: Ceník* [online] [cit. 2021-02-16]. Dostupné z: https://www.hledanipredku.cz/rodokmen-cenik/.

39. *Statista: Ranking of the most popular database management systems worldwide, as of December 2020* [online] [cit. 2021-02-28]. Dostupné z: https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/.

40. *MongoDB: Indexes* [online] [cit. 2021-03-02]. Dostupné z: https://docs.mongodb.com/manual/indexes/.

41. *Apache Cassandra: Secondary Indexes* [online] [cit. 2021-03-02]. Dostupné z: https://cassandra.apache.org/doc/latest/cql/indexes.html.

42. CHAKRABORTTII, Chandranil. Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool. 2015. Dostupné také z: https://www.researchgate.net/publication/331287438_Performance_Evaluation_of_NoSQL_Systems_Using_Yahoo_Cloud_Serving_Benchmarking_Tool.

43. *MongoDB | Blog: Building with Patterns: The Bucket Pattern* [online] [cit. 2021-04-08]. Dostupné z: https://www.mongodb.com/blog/post/building-with-patterns-the-bucket-pattern.

44. *Microsoft: Design the infrastructure persistence layer* [online] [cit. 2021-04-05]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design.

45. *Microsoft: Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application* [online] [cit. 2021-04-05]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application.

46. *MongoDB: MongoDB Pricing* [online] [cit. 2021-04-03]. Dostupné z: https://www.mongodb.com/pricing.

47. *Microsoft Azure: App Service pricing* [online] [cit. 2021-04-03]. Dostupné z: https://azure.microsoft.com/en-us/pricing/details/app-service/linux/.

48. KRONIS, Kristiāns; UHANOVA, Marina. Performance Comparison of Java EE and ASP.NET Core Technologies for Web API Development. *Applied Computer Systems* [online]. 2018-05-01, roč. 23, č. 1, s. 37–44 [cit. 2021-04-06]. ISSN 2255-8691. Dostupné z DOI: 10.2478/acss-2018-0005.

Příloha A

Seznam použitých zkratek

BPMN Business Process Modeling Notation.

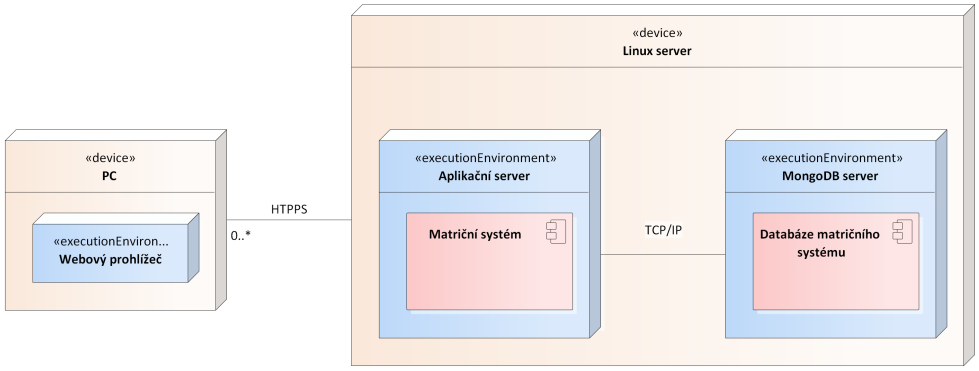
MD Manday.

MS Microsoft.

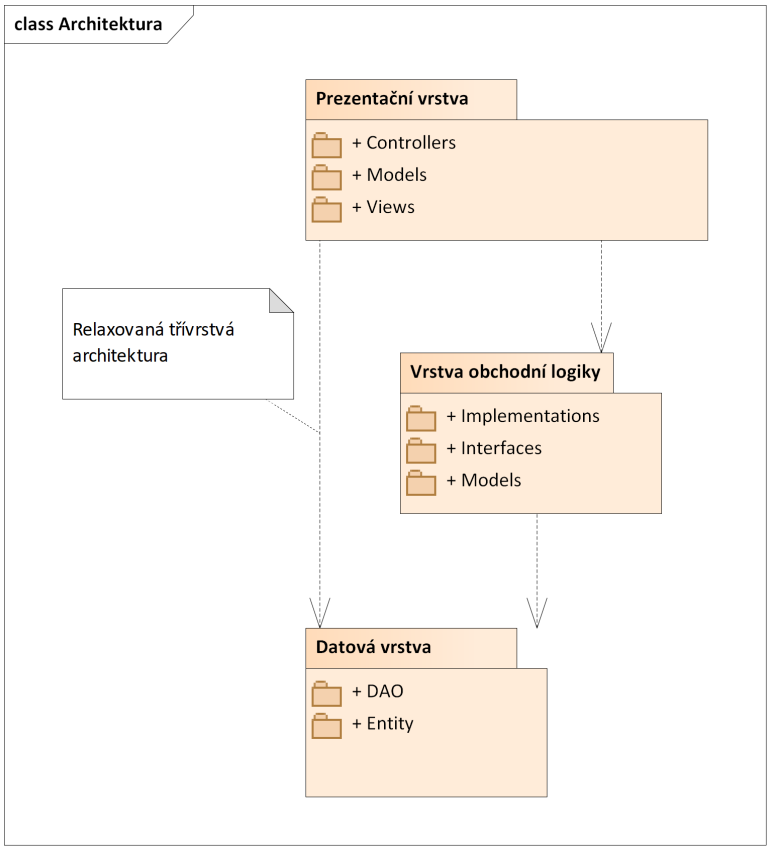
POC proof-of-concept.

Příloha B

Detailní diagramy systému

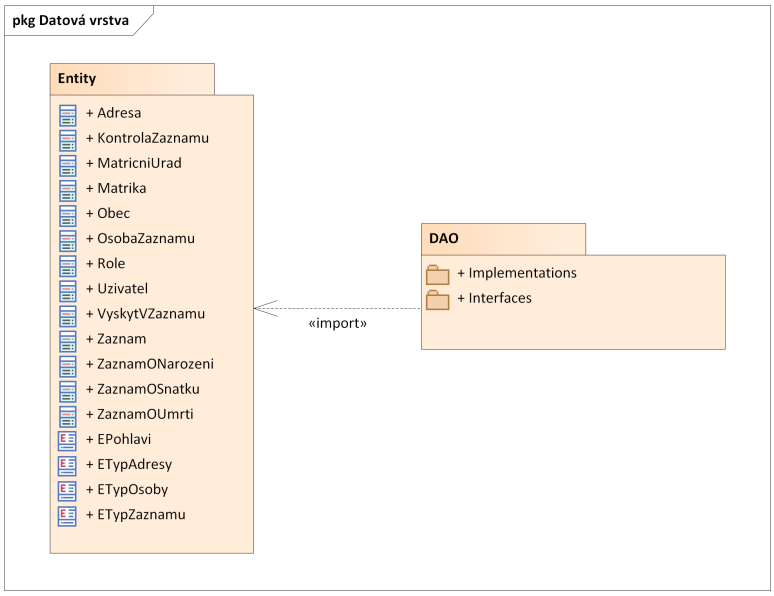


Obrázek B.1: Model nasazení systému

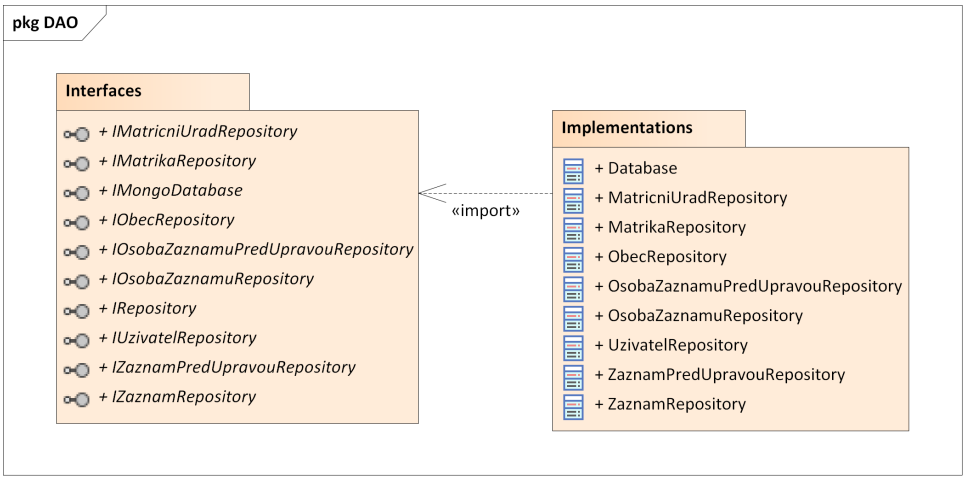


Obrázek B.2: Architektura systému

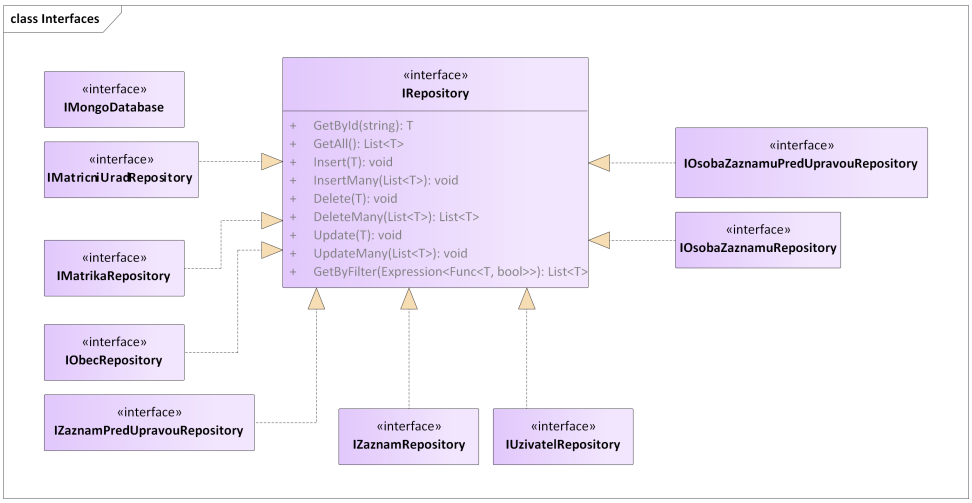
Datová vrstva



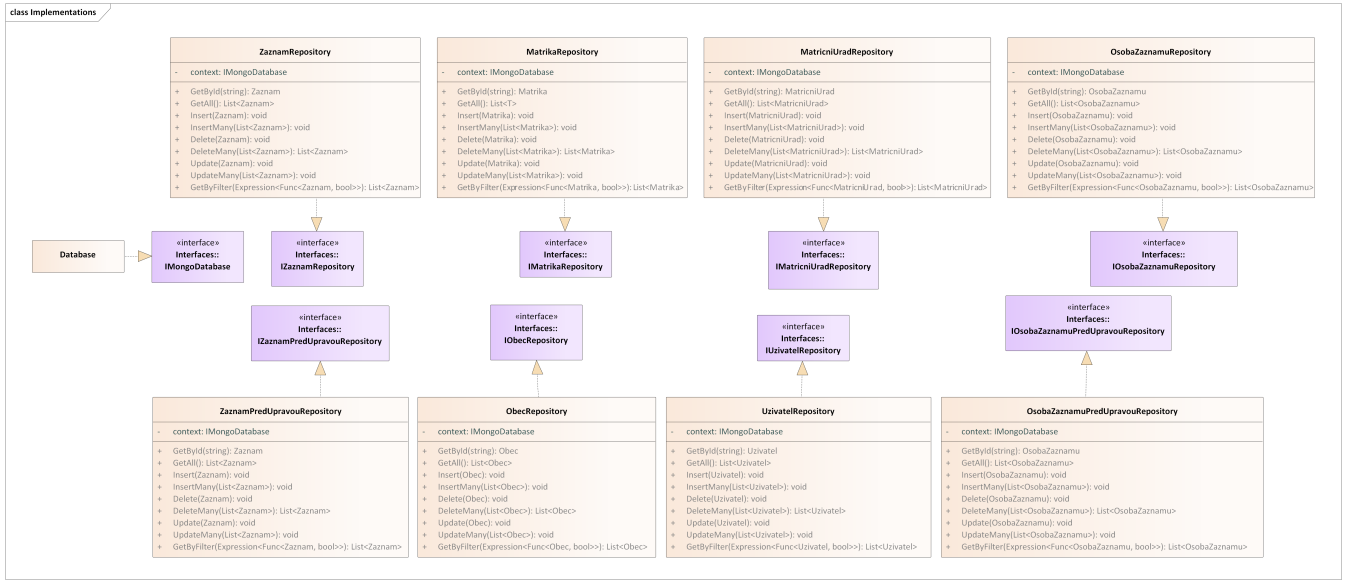
Obrázek B.3: Datová vrstva



Obrázek B.4: DAO

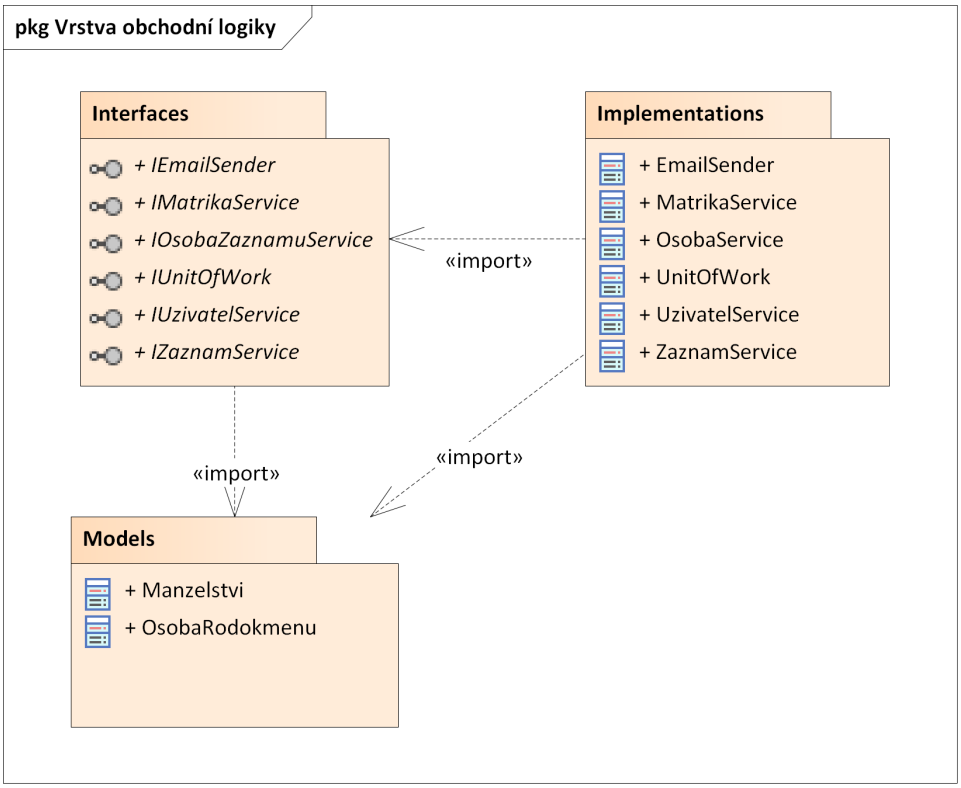


Obrázek B.5: DAO – interface

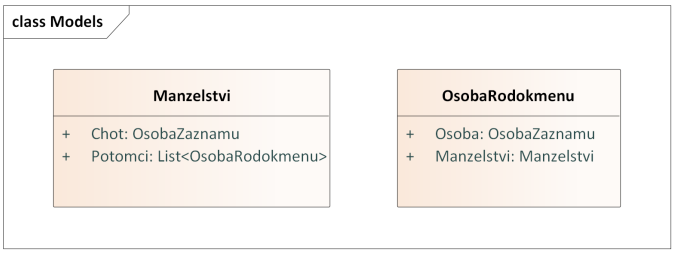


Obrázek B.6: DAO – implementace

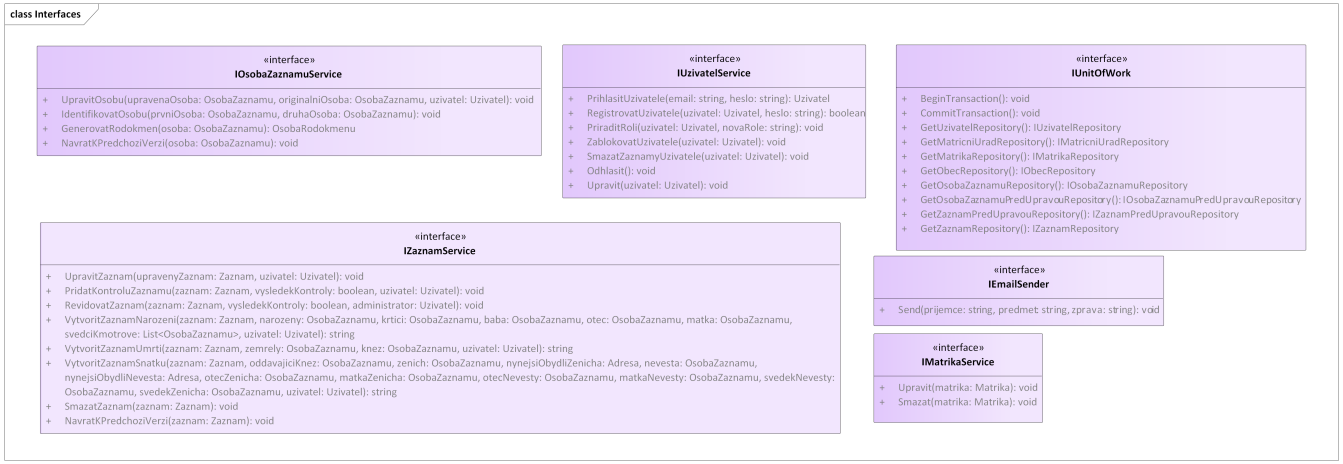
Vrstva obchodní logiky



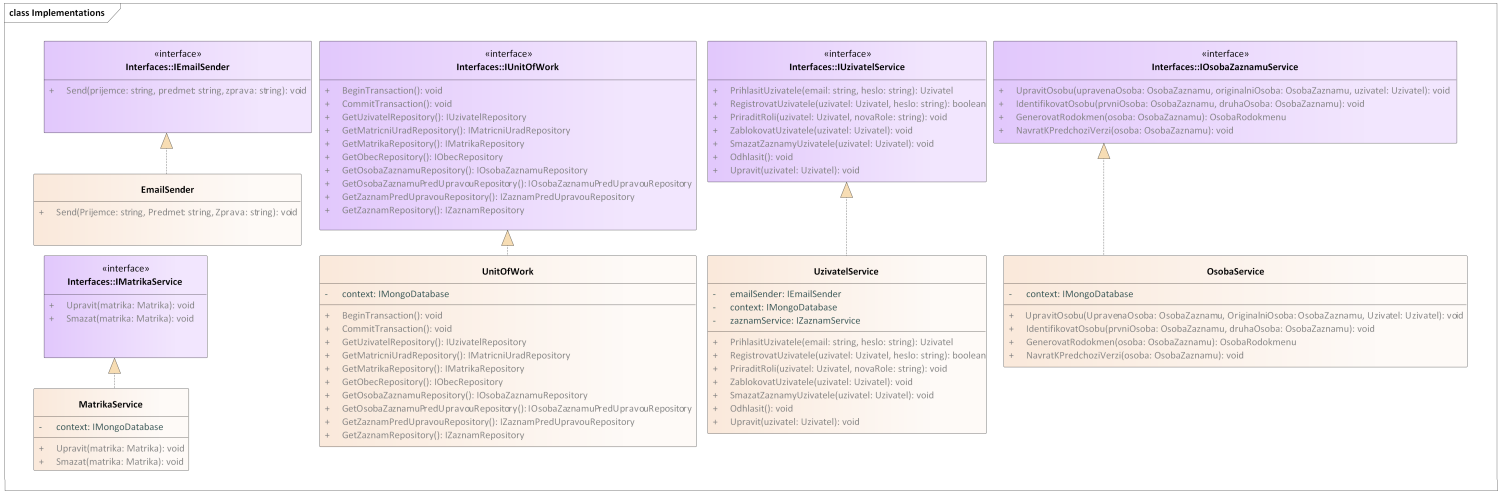
Obrázek B.7: Vrstva obchodní logiky



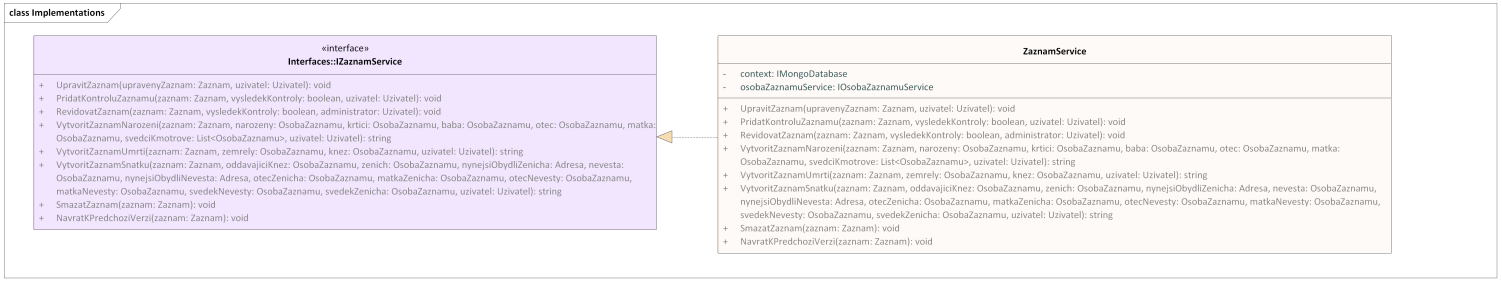
Obrázek B.8: Modely obchodní logiky



Obrázek B.9: Interface obchodní logiky

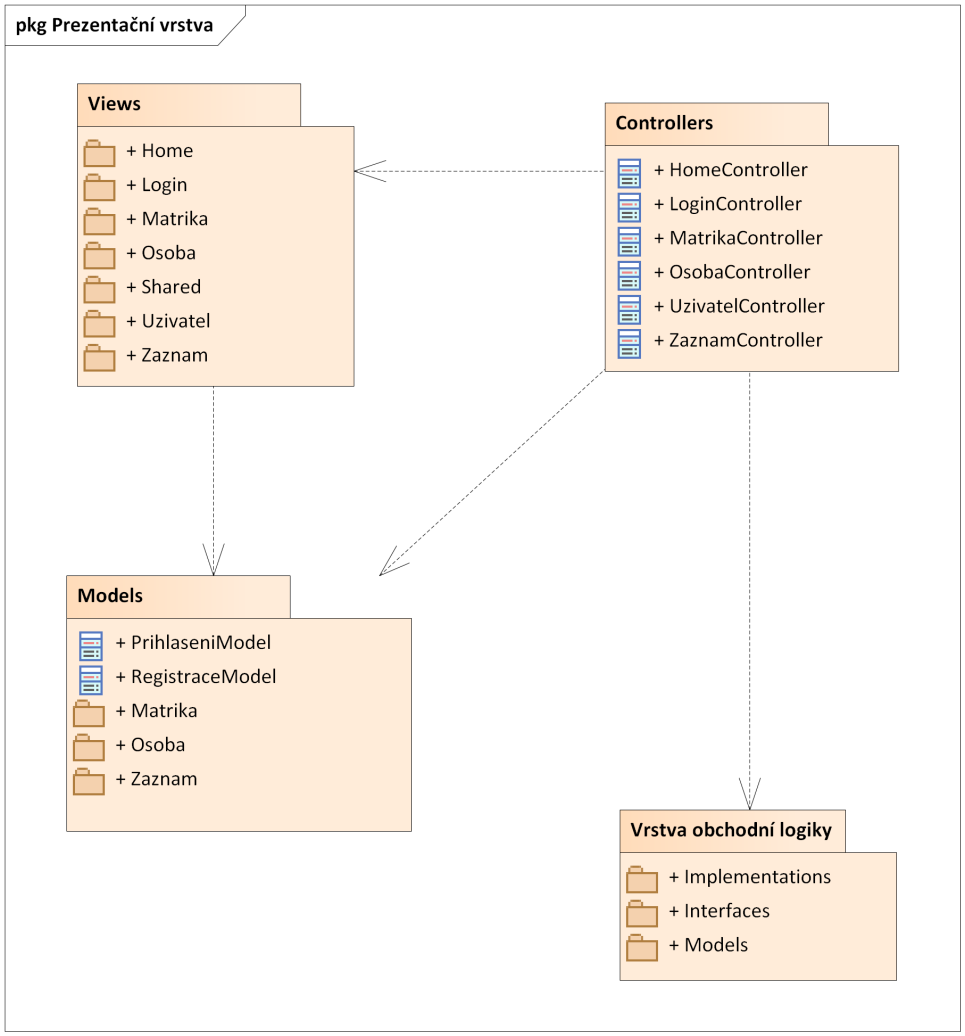


Obrázek B.10: Implementace obchodní logiky

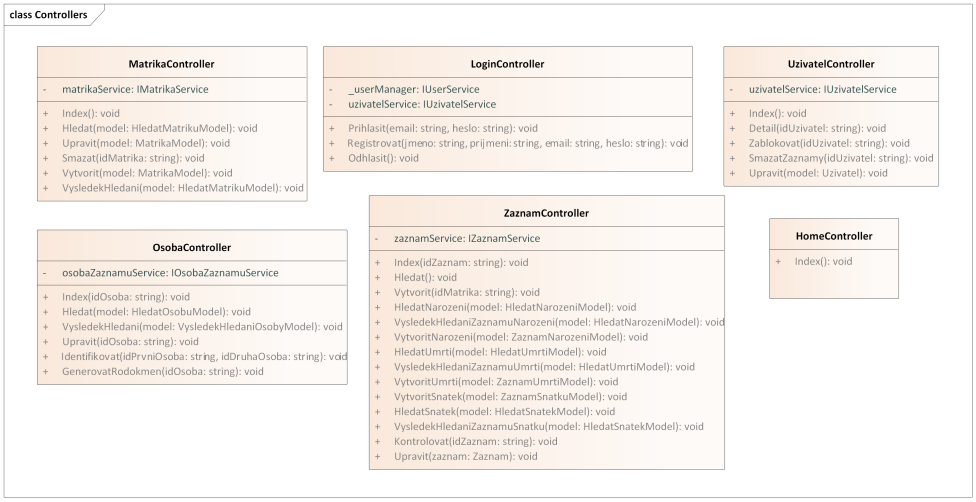


Obrázek B.11: Implementace obchodní logiky

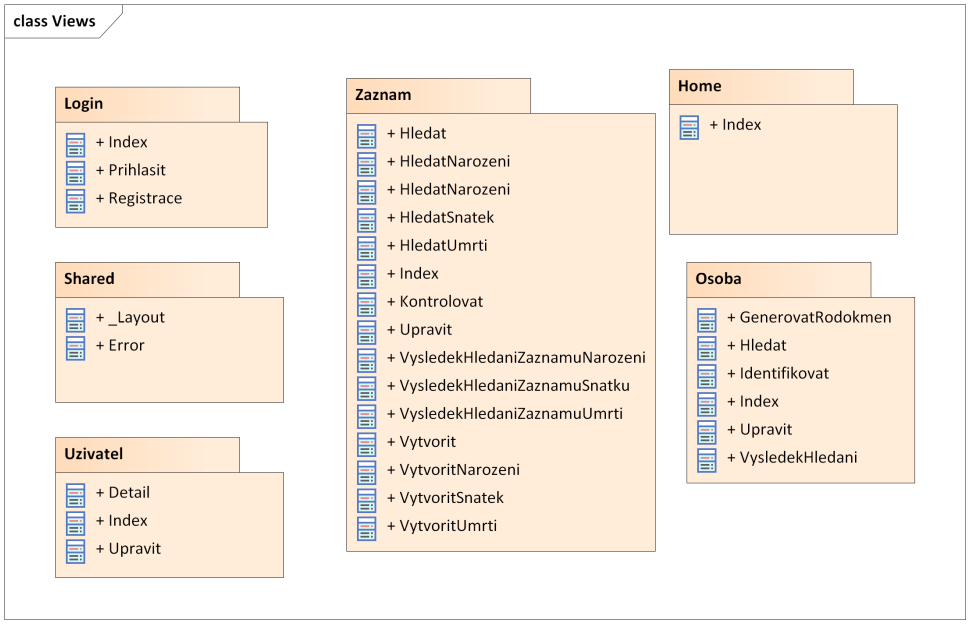
Prezentační vrstva



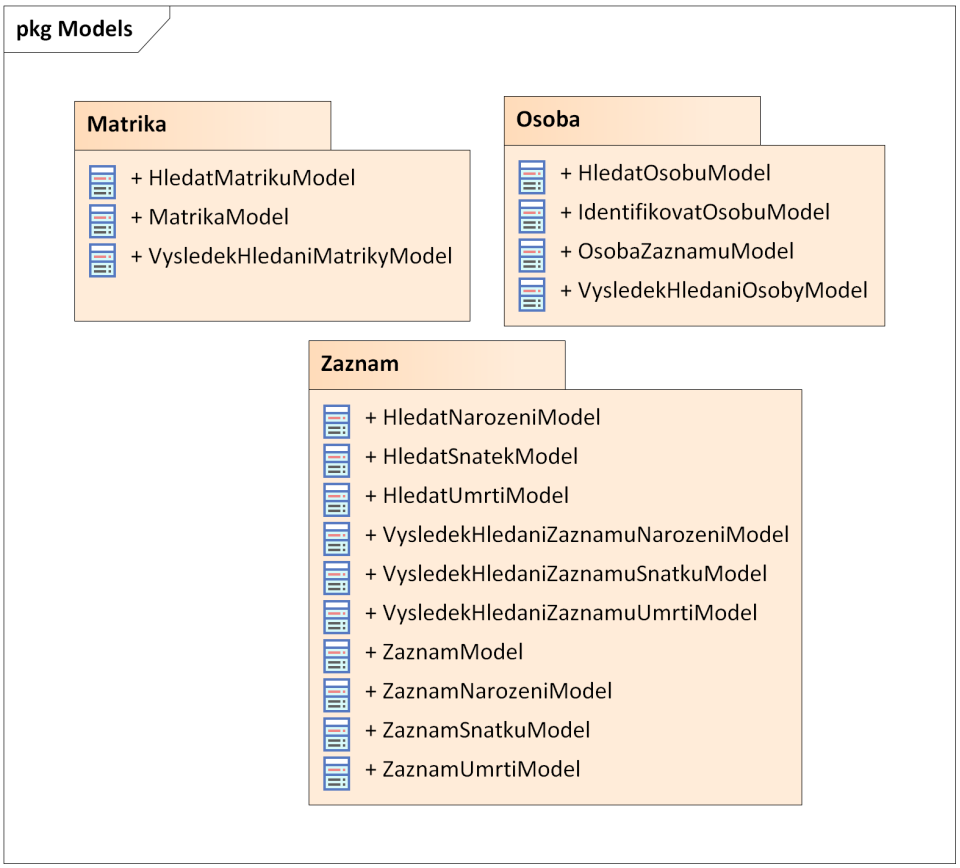
Obrázek B.12: Prezentační vrstva



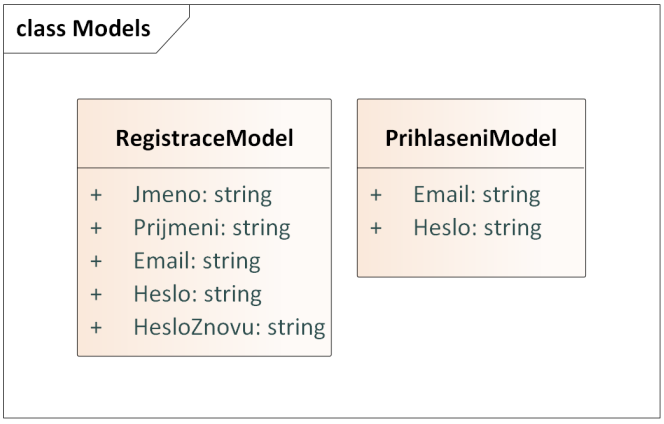
Obrázek B.13: Controllery prezentační vrstvy



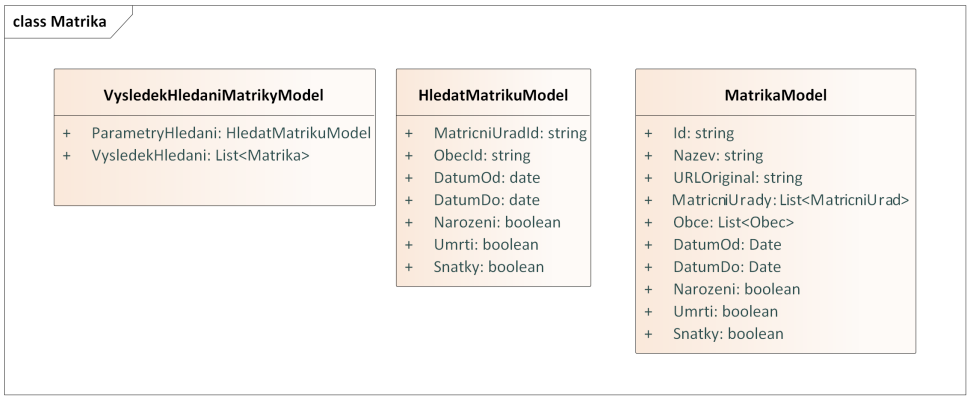
Obrázek B.14: Obrazovky prezentační vrstvy



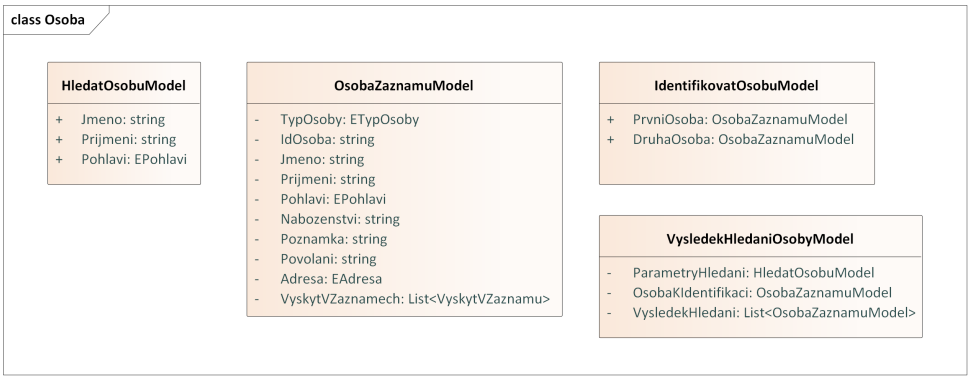
Obrázek B.15: Modely prezentační vrstvy



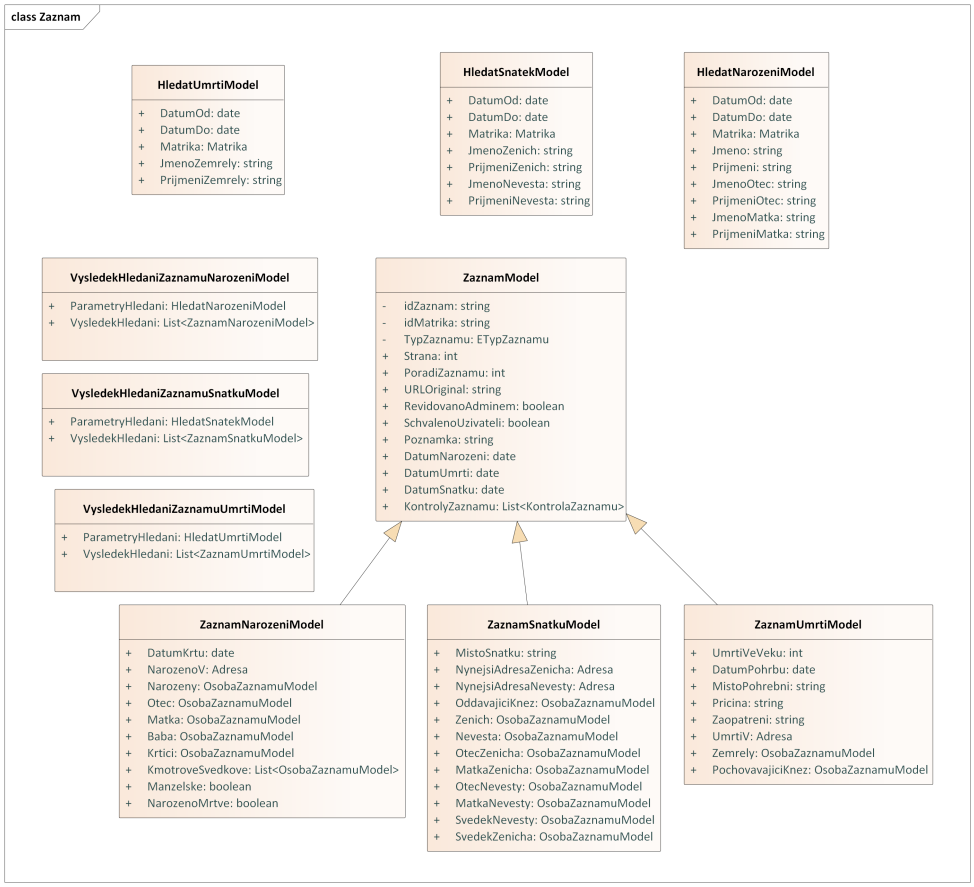
Obrázek B.16: Modely přihlašování



Obrázek B.17: Modely matriky



Obrázek B.18: Modely osoby



Obrázek B.19: Modely záznamů

Příloha C

Instalační příručka POC řešení

C.1 Lokální spuštění

C.1.1 Databáze

Pro zprovoznění je nutné mít buď založenou vlastní databázi v oficiálním cloudu MongoDB, nebo nainstalovanou lokální standalone instanci. V případě lokální instance je nutné instalovat verzi Replica Set. Poté je třeba nainstalovat MongoDB Database Tools, dle oficiálního návodu https://docs.mongodb.com/database-tools/installation/installation/.
Po nainstalování databáze a nástrojů je potřeba spustit příkaz:

```
mongorestore –d nazevNoveDB ../POC/mongodump/dummyDB
```

Příkaz vytvoří dummy databázi obsahující testovací data.

C.1.2 Aplikace

Pro lokální zprovoznění aplikace, je potřeba mít nainstalovaný .NET Core 3.1. Dále je potřeba v souboru *appsettings.json* nastavit connection string do databáze z předchozího kroku. Pro spuštění je pak možné použít Visual Studio, ve kterém lze otevřít soubor *MatricniSystem.sln* a spustit debugování. Další možností, jak systém spustit je zadáním příkazu:

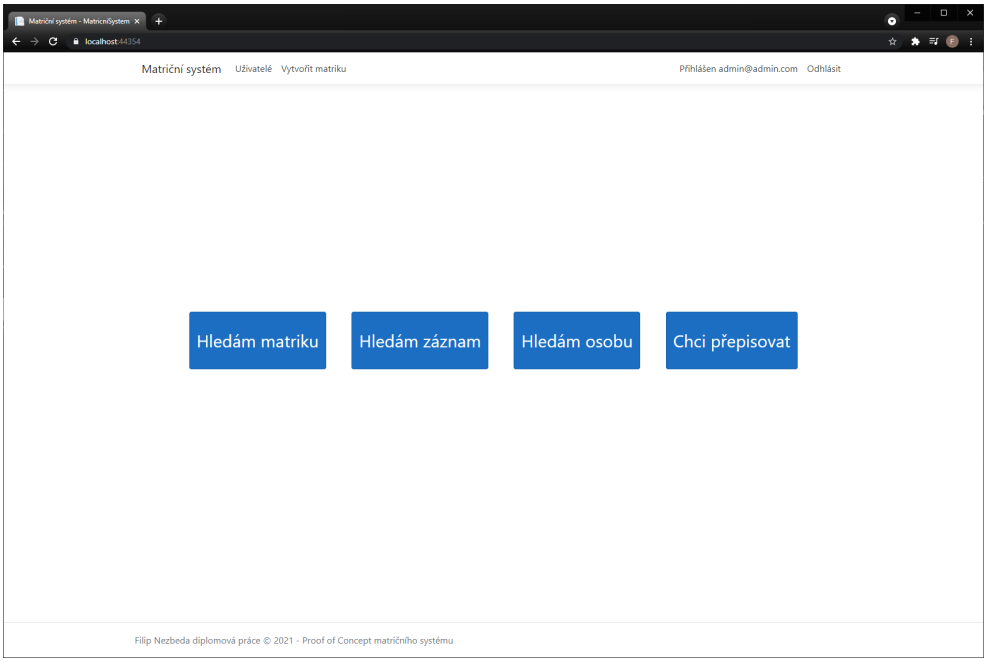
```
dotnet run
```

ve složce *../POC/MatricniSystem/MatricniSystem*. Aplikace pak bude spuštěna na localhostu, nejčastější adresa je *https://localhost:5001*, ale může se lišit v závislosti na nastavení lokálního serveru.

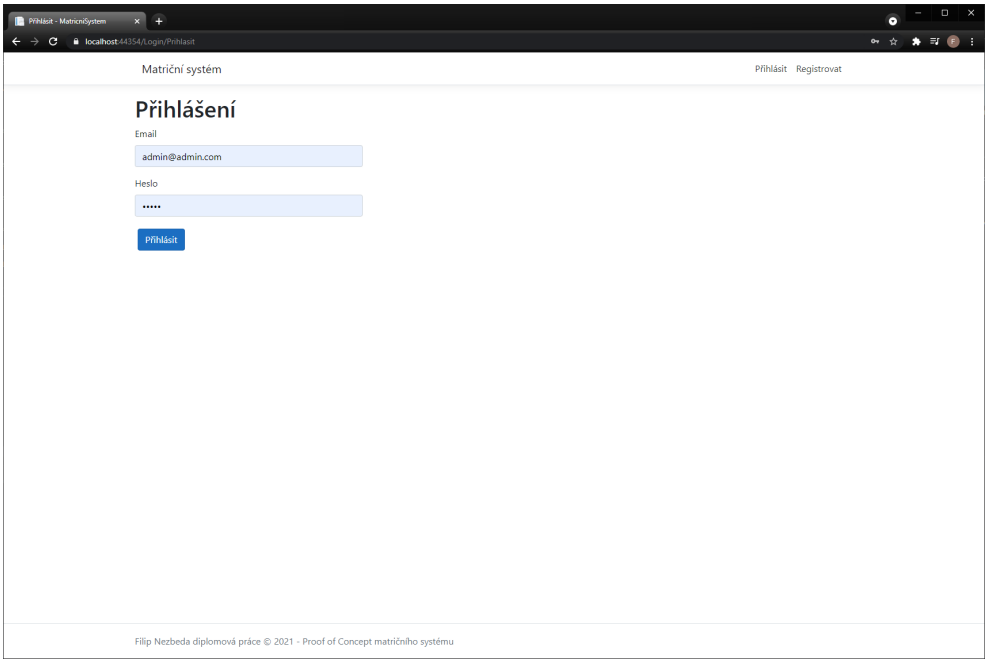
Po spuštění je možné pro přihlášení využít aministrátorský účet s emailem „admin@admin.com“ a heslem „Heslo“.

Příloha D

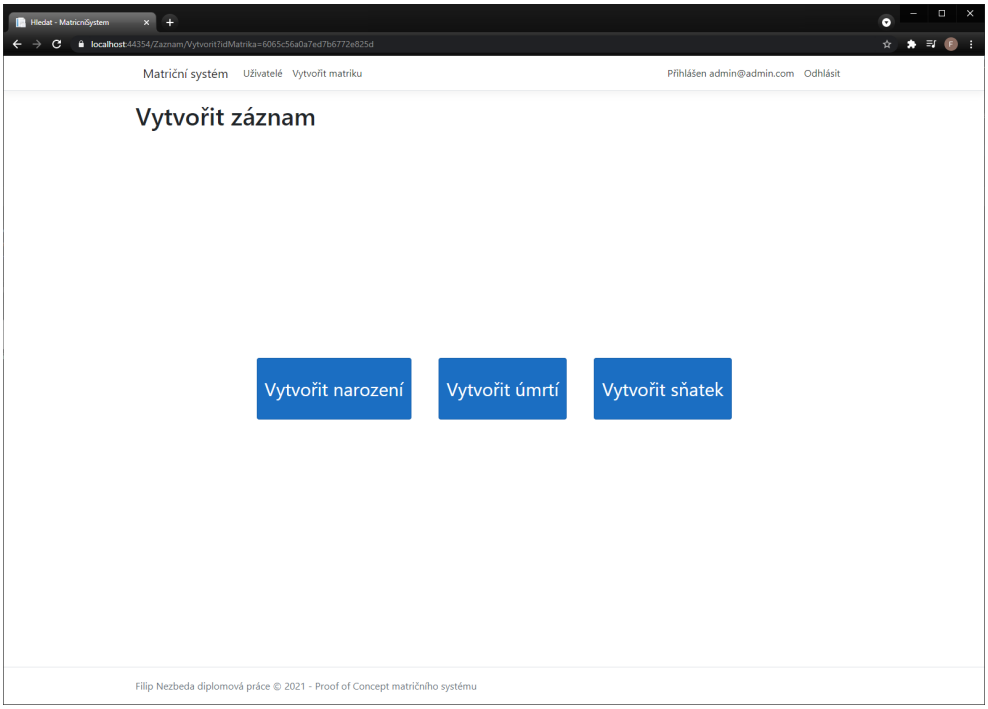
Obrazovky POC řešení



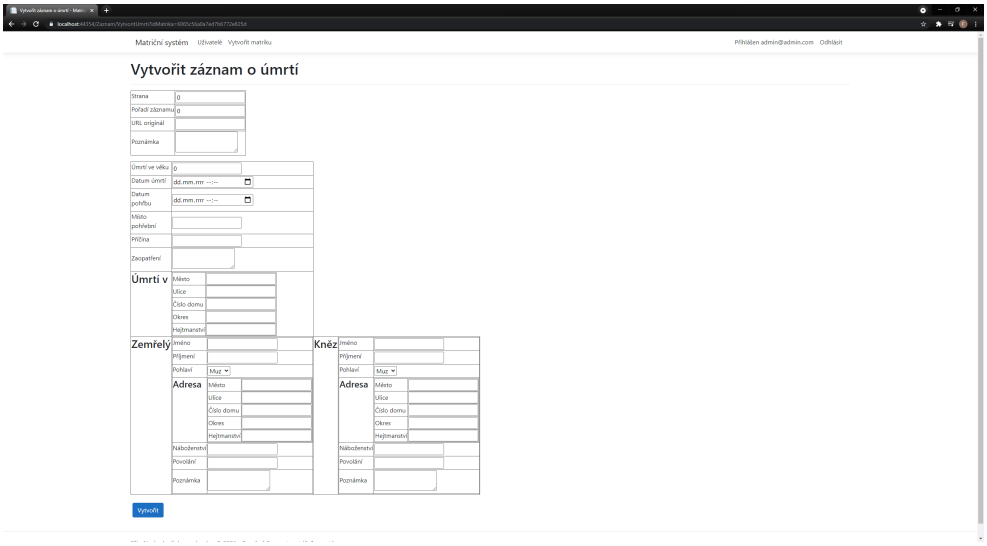
Obrázek D.1: Domovská obrazovka



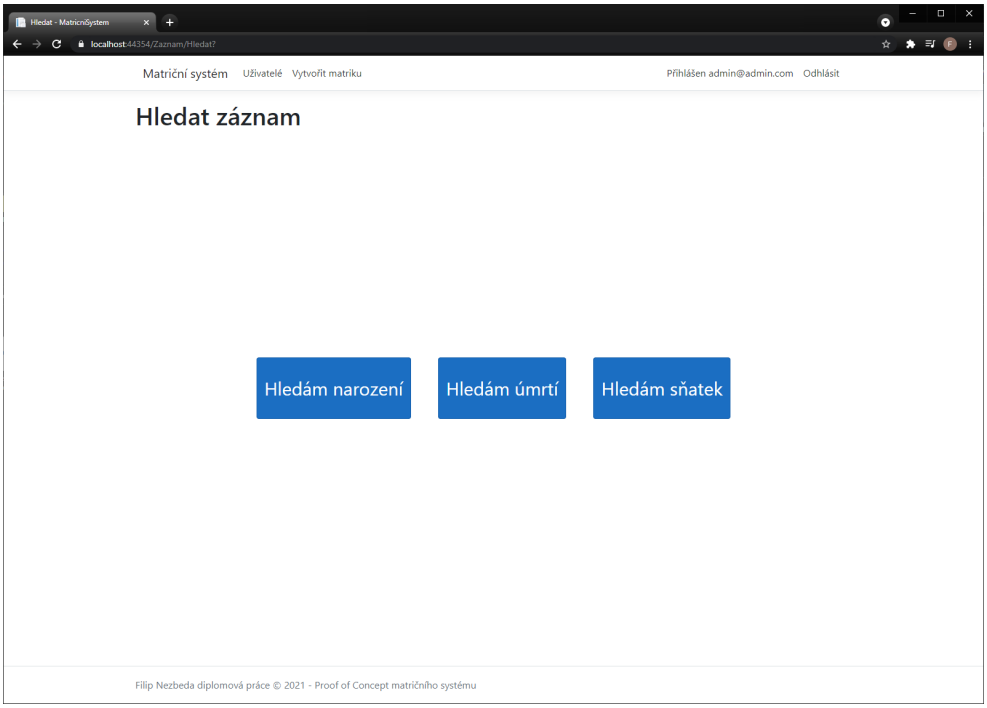
Obrázek D.2: Přihlašovací obrazovka



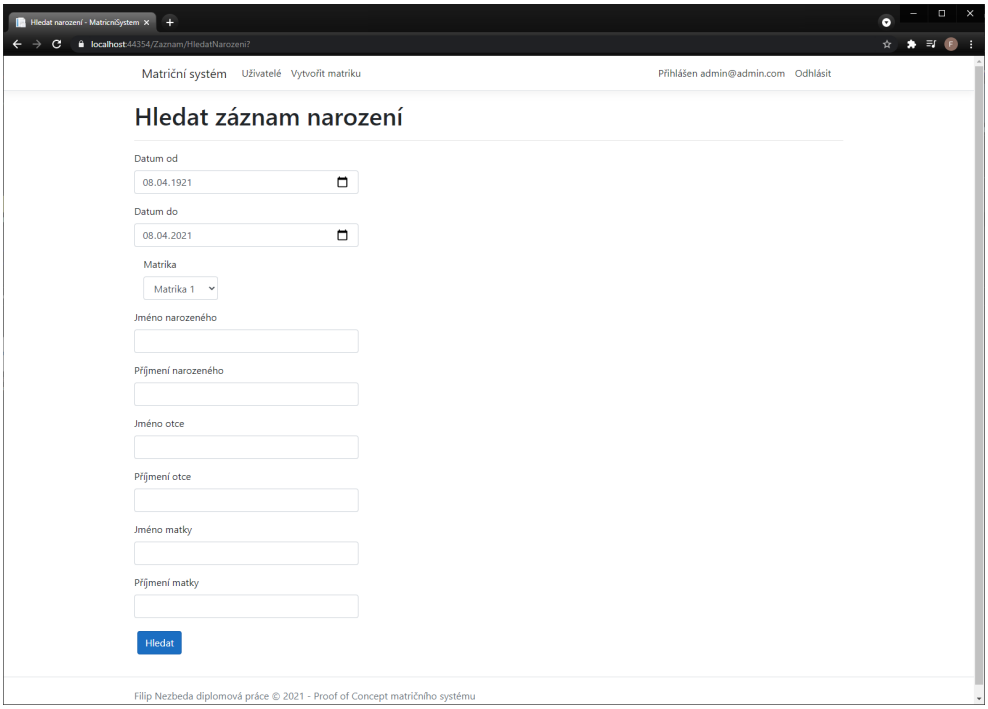
Obrázek D.3: Vytvořit záznam



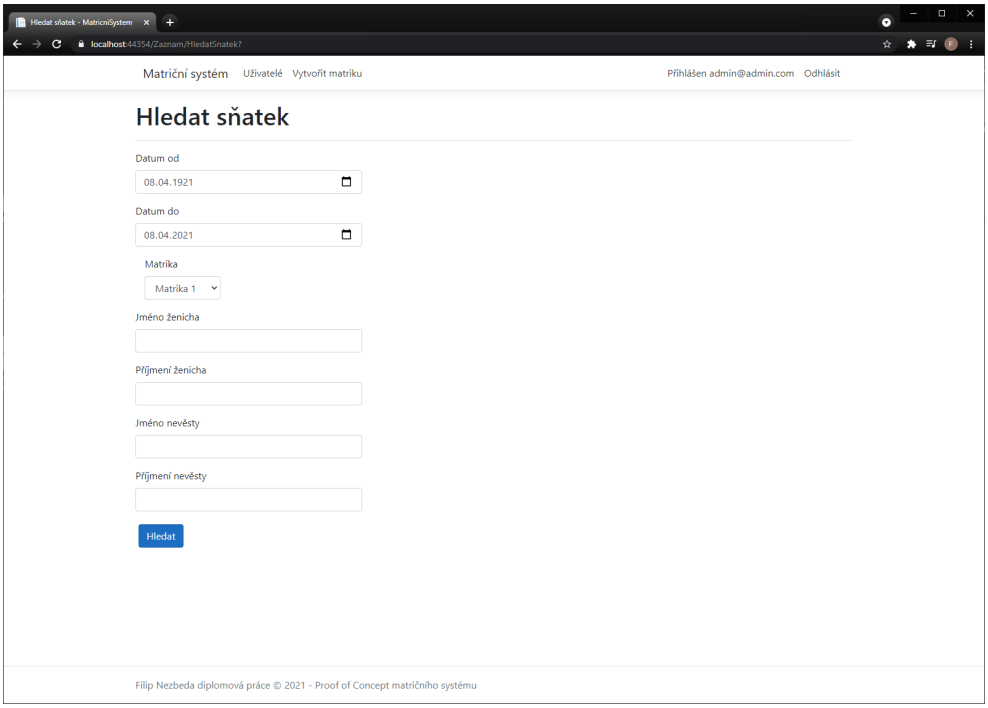
Obrázek D.4: Vytvořit záznam úmrtí



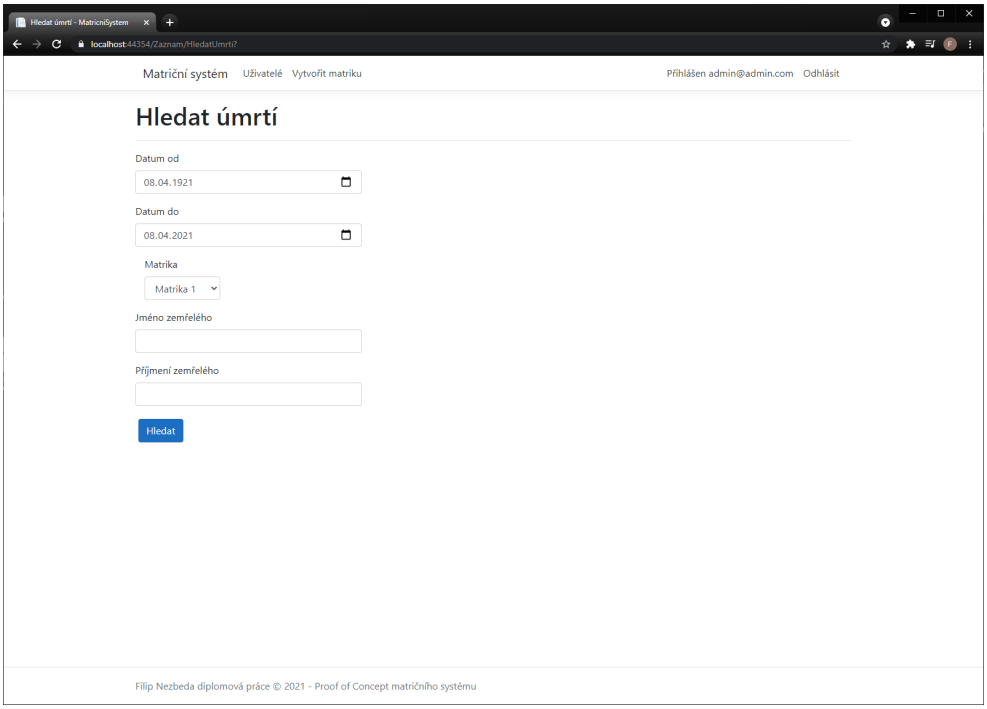
Obrázek D.5: Hledání záznamu



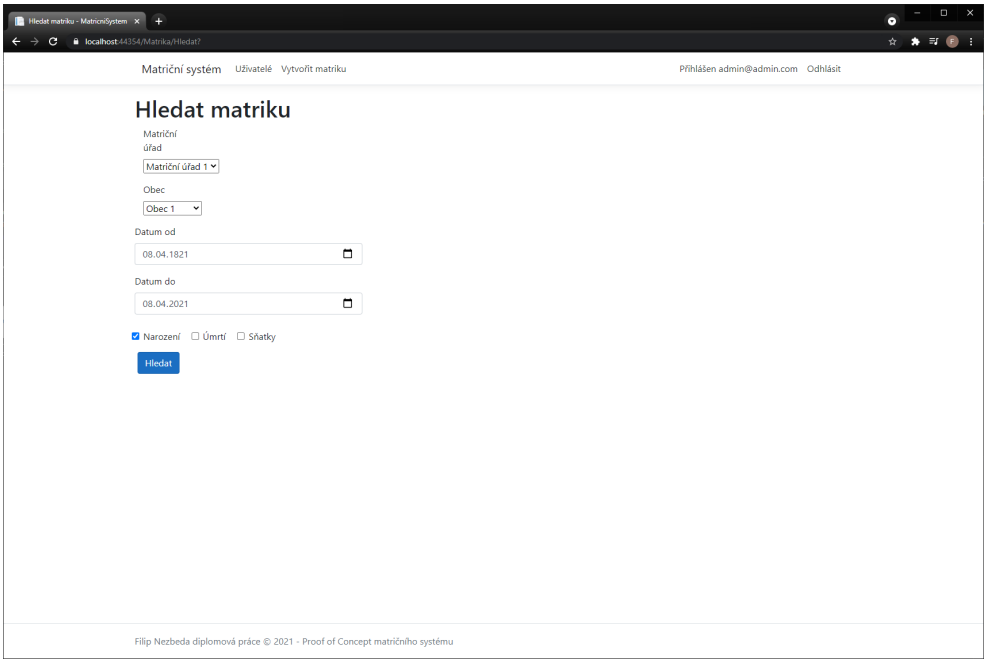
Obrázek D.6: Hledat narození



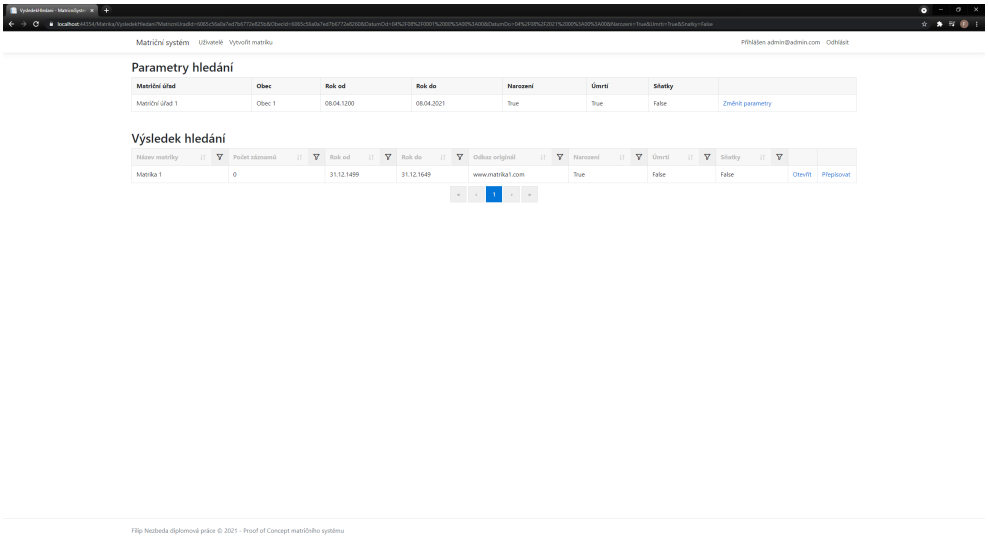
Obrázek D.7: Hledat sňatek



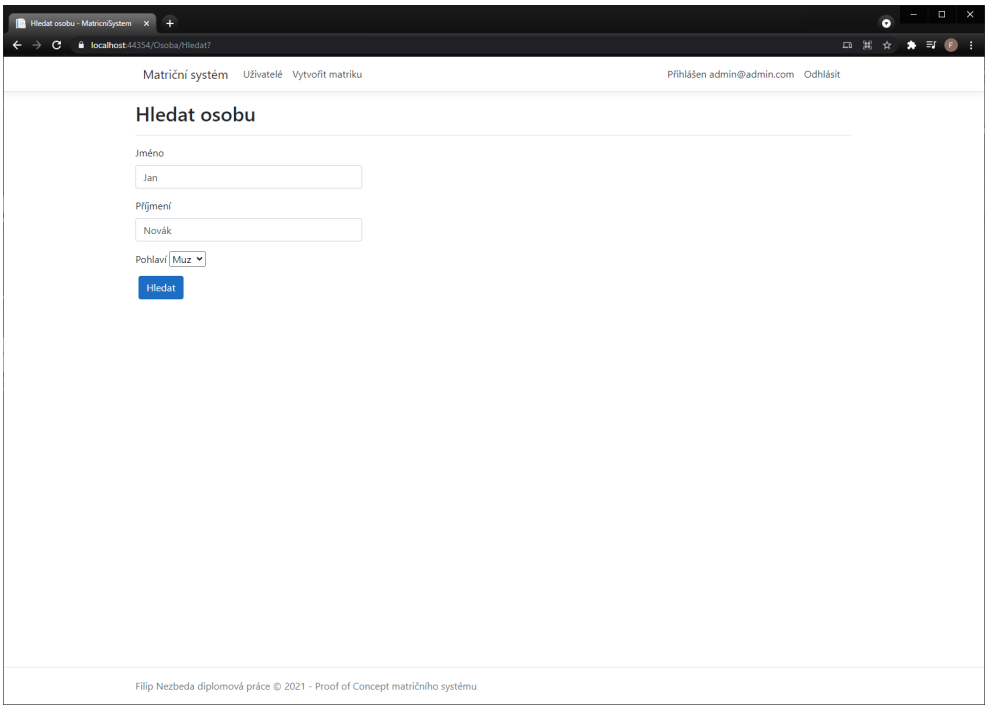
Obrázek D.8: Hledat úmrtí



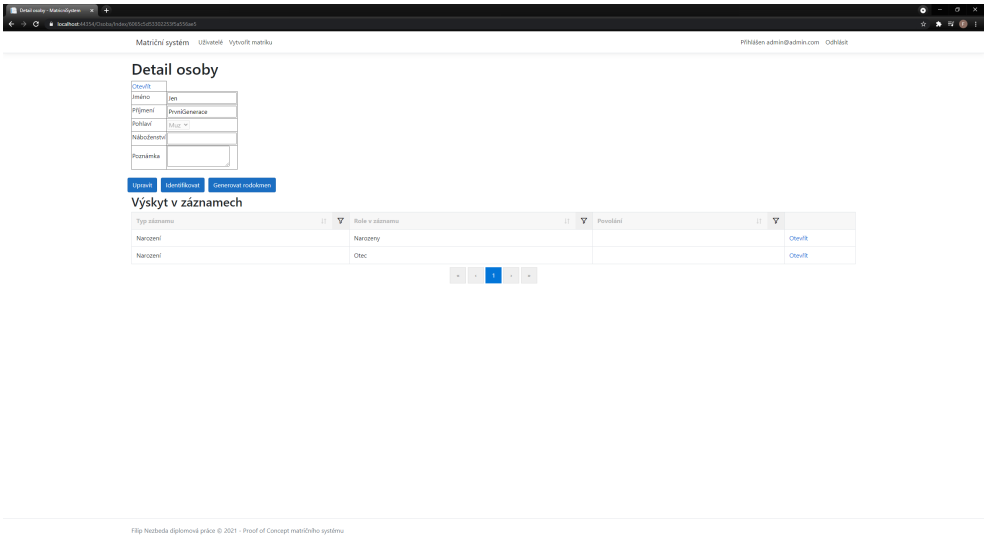
Obrázek D.9: Hledat matriku



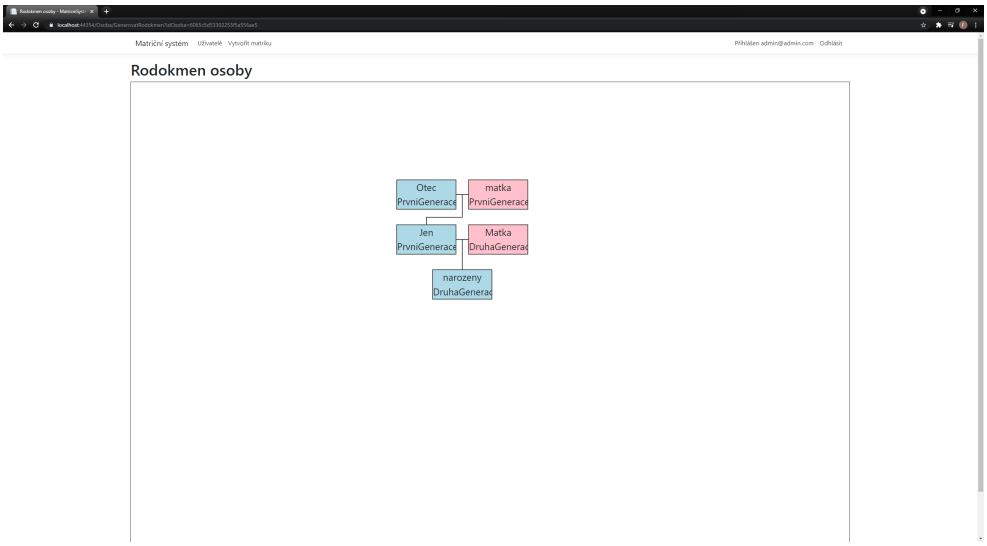
Obrázek D.10: Výsledek hledání matriky



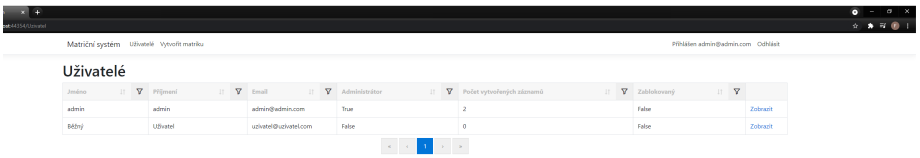
Obrázek D.11: Hledat osobu



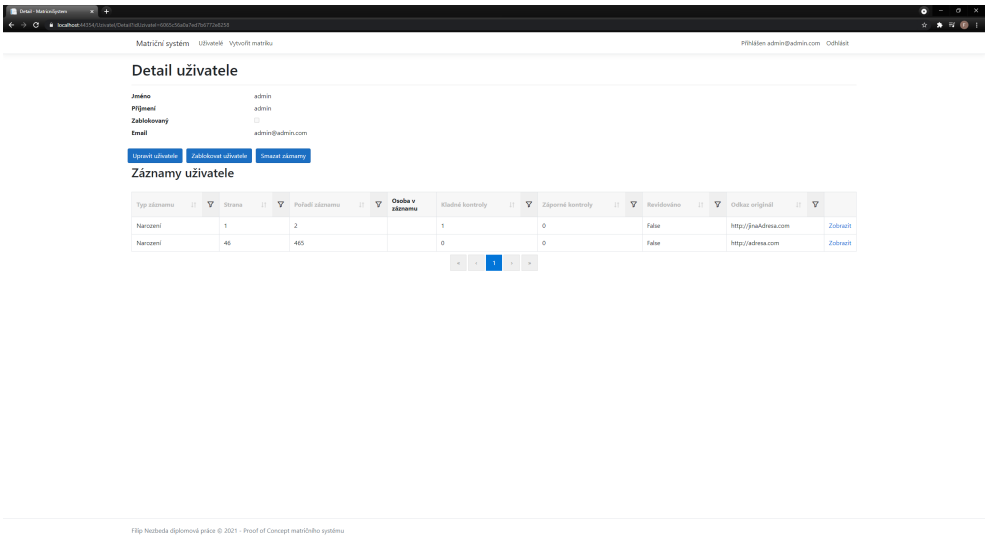
Obrázek D.12: Detail osoby



Obrázek D.13: Rodokmen osoby



Obrázek D.14: Přehled uživatelů



Obrázek D.15: Detail uživatele

Příloha E

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `DP_Nezbeda_Filip_2021.pdf` ..... text práce ve formátu PDF
- `src` ..... zdrojová forma práce ve formátu LaTeX
  - `Databaze` ..... zdrojové soubory modelu databáze
  - `DomenovyModel` ..... zdrojové soubory doménového modelu
  - `ModelTrid` ..... diagramy architektury
  - `POC` ..... obrazovky POC řešení
  - `Procesy` ..... diagramy procesů a jejich zdrojové soubory
  - `TeorieObrazky`
  - `UseCase` ..... diagram případů užití
  - `MatricniSystem.eapx` ..... návrh systému v Enterprise Architect
- `POC`
  - `MatricniSystem` ..... zdrojové kódy POC řešení
  - `mongodump` ..... záloha databáze obsahující dummy data