Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Evidenční systém pro zvířecí útulek

2020

Michal Frýba

Vedoucí práce: RNDr. Arnošt Večerka

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Michal Frýba
Název práce:	Evidenční systém pro zvířecí útulek
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	RNDr. Arnošt Večerka
Počet stran:	54
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Michal Frýba
Title:	Record management software for animal shelter
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, combined form
Supervisor:	RNDr. Arnošt Večerka
Page count:	54
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Aplikace pro evidenci zvířat pro útulky umožňuje evidovat odchycená nebo odebraná zvířata a jejich dokumenty. Umožňuje jejich správu a základní vyhledávání. Dále nabízí možnost uchovávat základní údaje o zvířeti, informace o veterinární péči a zdravotních záznamech, incidentech, vynaložených nákladech, krmných dávkách, době strávené v útulku a různé další záznamy potřebné od příjmu zvířete až po jeho výdej. Aplikace by měla být v praxi schopna poskytovat veškeré údaje jako současná papírová evidence. Celá aplikace byla vytvořena v programovacím jazyce C# jako součást této bakalářské práce.

Synopsis

Animal Record Management Application for Shelters allows to make records of captured or taken animals and their documents. It provides their management and basic search. It also offers the possibility to store basic information about the animal, information about veterinary care and health records, incidents, costs incurred, daily rations, time spent in the shelter and various other records needed for the entirety of animal's life in the shelter. In practice, the application should be able to provide all data as current paper records. The whole application was developed in C# programming language as part of this bachelor thesis.

Klíčová slova: evidence zvířat; evidenční aplikace; zvířecí útulek; adopce

Keywords: animal record-keeping; record management application; animal shelter; adoption

Děkuji svému vedoucímu RNDr. Arnoštu Večerkovi za cenné rady, konstruktivní připomínky a pomoc při tvorbě této práce. Dále bych rád poděkoval útulku Ligy na ochranu zvířat Olomouc za pomoc při testování a za funkční připomínky z praxe.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

1 Úvod

Zvířecí útulky jsou ze zákona povinny vést základní evidenci zvířat. Evidenci poskytují orgánům ochrany zvířat, které zajišťují dozor nad dodržováním právních předpisů týkajících se ochrany zvířat. Většina útulků pro tuto povinnost používá papírovou evidenci doplněnou o množství záznamů v tabulkách z tabulkových editorů (MS Excel, Open Office Calc ...). Práce a vyhledávání v evidenci bývá většinou zdlouhavá, a to zvláště u větších útulků.

Ve spolupráci s olomouckým útulkem Ligy na ochranu zvířat jsem si jako téma své bakalářské práce vybral evidenční systém pro zvířecí útulky. V rámci tohoto tématu jsem v jazyce C# naprogramoval desktopovou aplikaci pro evidenci zvířat přijatých do útulku. Výsledná aplikace by měla poskytovat všechnu funkcionalitu pro fungování evidence útulku. Některé evidenční požadavky ukládá útulku zákon a proto je jejich přítomnost v aplikaci nezbytně nutná. Další údaje, které se dají evidovat, byly vybrány společně s útulkem, aby usnadnily jeho chod, a informace z nich byly pro zaměstnance nějakým způsobem užitečné. Funkcionalita podobné aplikace by mohla být velmi široká a přesahovala by rozsah práce. Aplikaci se budu snažit dále rozvíjet o další nové funkce a nadále ji aktualizovat a zdokonalovat.

1.1 Rozdělení práce

Bakalářská práce je rozdělena do tří hlavních částí. První část je teoretická, druhá programátorská a třetí uživatelská. Hlavní částí je část teoretická.

Teoretická část obsahuje stručné nastínění povinností týkajících se evidence, které jsou útulkům nařízeny zákonem. Dále obsahuje srovnání vlastností systémů podobných tomu, který byl naprogramován jako praktická část této bakalářské práce. Srovnání výhod a nevýhod je provedeno přehledně pomocí tabulky a poté popisem konkrétní aplikace. Teoretická část obsahuje také přehled a analýzu uživatelských požadavků, které byly poskytnuty útulkem. Analýza je provedena UML diagramy případů užití. Poslední částí je návrh databáze s využitím entitně-relačního diagramu.

V úvodu programátorské části je rozebrána struktura naprogramované aplikace a její databáze, včetně základních UML diagramů tříd. Dále je uveden přehled využitých nástrojů. Na závěr je stručně popsán architektonický programovací vzor MVVM, který byl využit při vývoji aplikace. Jelikož se jedná o aplikaci určenou pro nasazení v praxi, bude část věnována uživatelskému rozhraní. K vytvoření uživatelského rozhraní byla použita knihovna Material Design In XAML Toolkit. Material Design je designový jazyk vyvinutý firmou Google.

V poslední uživatelské části jsou stručně shrnuty možnosti práce uživatele s aplikací, popis ovládání programu a pokyny pro práci se vstupními údaji. Tato kapitola může sloužit také jako uživatelská příručka pro ovládání aplikace. Popis ovládání je doplněn o ukázky obrazovek přímo z aplikace.

2 Teoretická část

Tato kapitola je věnována vlastní motivaci pro tvorbu evidenčního systému a popisu nutnosti povinné evidence z pohledu zákona. Dále je provedeno srovnání komerčně dostupných aplikací, které jsou určené pro zvířecí útulky a jejich správu. Vybrány byly aplikace cenově dostupné a nebo ty zdarma. Hlavními body srovnání jsou obsažené funkcionality a cena. Jednotlivé body jsou rozebrány také u aplikace vyvinuté v rámci práce. V této kapitole je obsažena i analýza uživatelských požadavků a jejich znázornění pomocí UML diagramů případů užití. Požadavky poskytnuté útulkem byly doplněny o další případy užití v rámci analýzy a po zhodnocení přínosu pro běh útulku byly zapracovány do návrhu.

2.1 Motivace pro vývoj evidenčního systému

Problematika toulavých a opuštěných zvířat je celosvětový problém. Jejich volný pohyb také představuje nebezpečí pro člověka a ostatní zvířata ať už z důvodu agrese, nebo přenosu nemocí. Je velmi obtížné přesných počet těchto zvířat evidovat. Celosvětově se tento počet odhaduje na 480 milionů. Podle odhadu nevládní organizace CAROdog je v ČR průměrně umístěno v útulcích zhruba 5000 psů [1]. Péče o tato zvířata je také značnou finanční zátěží pro zřizovatele útulků, kterými jsou většinou obce a města. V ČR se o tato zvířata stará přes 300 útulků [2]. Jen Olomoucký útulek Ligy na ochranu zvířat v roce 2019 přijal 535 psů a 19 koček [3]. Toto množství také klade na útulky značné nároky na evidenci těchto zvířat. Evidence je nejen praktická, ale je vyžadována i ze zákona [4]. Od 15. ledna 2020 je také zákonná povinnost nechat psy v domácnosti čipovat [5]. To by mělo útulkům pomoci k identifikaci přijatých zvířat a jejich návratu zpět k majiteli. Oficiální jednotný národní registr ale ještě není k dispozici.

Vhodná aplikace umožňující přehlednou evidenci útulkem přijatých zvířat může značně zjednodušit systém příjmu a výdeje zvířete a ušetřit tak útulku čas i finance. Vývoj aplikace pro útulek v Olomouci pokryl všechny hlavní uživatelské požadavky. Doplňkové požadavky, které se objevily ve výrazně pozdějších fázích vývoje, budou ještě do aplikace implementovány.

2.2 Útulky a zákon

Provoz útulku a povinnosti zřizovatele upravují dva hlavní zákony. Jsou to zákon č. 246/1992 Sb., na ochranu zvířat proti týrání, v platném znění a zákon č. 166/1999 Sb., o veterinární péči a některých souvisejících zákonů, v platném znění. Tyto dva zákony udávají provozovateli nejrůznější povinnosti, patří sem různé kvalifikační, chovatelské, hygienické, konstrukční, skladovací a další zásady a opatření. Mezi tyto povinnosti patří také předepsaná základní evidence. Tato část zákona se přímo týká problematiky v rámci které byla vytvořena evidenční aplikace. Předepsaná základní evidence podle Státní veterinární správy (SVS) znamená, že provozovatel útulku musí vést a mít možnost poskytnout pracovní-

kům orgánů ochrany zvířat[1], následující informace a doklady[6, 7]:

- seznam přijatých zvířat s uvedením počtu, druhu, popisu včetně identifikačních znaků, hmotnosti, data a místa nálezu zvířat nebo uvedení jejich původních chovatelů
- seznam vydaných zvířat a jejich nových chovatelů, včetně data předání, adresy, kde budou zvířata chována, nebo míst, kde byla zvířata opětně vypuštěna do původního prostředí
- evidenci úniků zvířat z útulku
- provozní řád zabezpečující ochranu pohody zvířat a organizaci práce a pracovních postupů stanovených zvláštními právními předpisy
- doklad o odborné způsobilosti

Dále provozovatel musí evidovat ve zvířecích záznamech informace o zdravotním stavu zvířete při příjmu, záznamy o dalších kontrolách zdravotního stavu a o provedených veterinárních zákrocích, případně důvod a datum úhynu nebo utracení zvířete [4, 6].

Tyto povinnosti uvádí SVS přímo na jejích stránkách a při kontrolách se mimo jiných, výše zmíněných věcí, kontroluje právě popsaná evidence. První tři body seznamu, doplněné o záznamy o zdravotním stavu, jsou vhodné pro zpracování do formy evidenční aplikace.

2.3 Evidenční informační systém obecně

Evidenční informační systémy se využívají pro sběr dat a jejich následné zpracování. Mohou se také používat k prezentaci a popřípadě exportu dat do dalších částí systému. Většinou bývá evidenční systém součástí většího celku komplexního informačního systému. Jako příklad můžeme uvést evidenci zaměstnanců firmy, evidenci objednávek a reklamací internetového obchodu nebo skladovou evidenci distribučních firem.

Evidenční informační systém by měl splňovat následující podmínky:

- splnit funkční uživatelské požadavky

  Před návrhem a vlastním programováním informačního systému by měl proběhnout sběr uživatelských požadavků. Jedná se o jednu z nejdůležitějších částí celého vývoje a závisí na ní úspěch projektu. Sběr požadavků může probíhat například konzultací. Analýza požadavků by měla proběhnout z různých úhlů pohledu. Měl by být zhodnocen především přínos pro uživatele a obtížnost implementace. Systém by měl být navržen tak, aby mohl být v budoucnu co nejsnadněji rozšiřován. Uživatelské požadavky

[1] pracovníci Krajské veterinární správy provádějící dozor nad dodržováním předpisů týkajících se ochrany zvířat

můžeme zaznamenat textově nebo graficky. Po grafické znázornění můžeme využít UML Use Case diagramy. Celý tento proces se nazývá analýza požadavků, nebo inženýrství požadavků. Ve velkých firmách jej mají na starost konzultanti a analytici.

- odolnost vůči chybným vstupům od uživatele

  Evidenční systém by měl být odolný vůči nejrůznějším vstupům od uživatele. Chybný nepředpokládaný vstup by neměl způsobit pád nebo nepředvídatelné chování systému. Při práci s databází by také nemělo docházet k ukládání dat, které by po načtení mohly způsobit chybný výstup pro uživatele. Dobrý způsob, jak se těmto chybám vyhnout, je jim předcházet. Systém by měl být schopen vstup ošetřit a popřípadě uživatele informovat, že vstup je chybný a vyzvat ho k opakování zadání.

- jednoduché ovládání pro uživatele

  Jednoduché a intuitivní ovládání systému je v dnešní době samozřejmostí každé aplikace. Uživatelské rozhraní by mělo být navrženo tak, aby uživatele přímočaře vedlo k předpokládanému cíli. Členění celé aplikace a jejích formulářů by mělo být logické a mělo by respektovat potřeby uživatele. Nemělo by uživatele zbytečně rozptylovat a odvádět jeho pozornost od vykonávané práce.

  > "Users are more tolerant of minor usability issues when they find an interface visually appealing. This aesthetic-usability effect can mask UI problems and can prevent issue discovery during usability testing. Identify instances of the aesthetic-usability effect in your user research by watching what your users do, as well as listening to what they say."- Kate Moran [8]

- zjednodušit a urychlit práci uživatelů

  Hlavním důvodem vzniku informačního systému je, aby co nejvíce zjednodušil a urychlil práci jeho uživatelů. Dobře navržený evidenční informační systém podporuje vykonávání jednotlivých dílčích činností uživatele ve správném sledu. Oproti například papírové evidenci šetří čas při zakládání, vyhledávání a třídění záznamů. Systém by neměl být kontraproduktivní a měl by poskytnout určitý komfort při používání.

2.4 Srovnání komerčních aplikací

V této sekci bude provedeno srovnání na trhu dostupného softwaru, který by mohl sloužit útulkům k evidenci zvířat. Většina těchto nástrojů je ale v plné verzi pro útulky finančně nedostupná. Útulky v ČR nejsou většinou natolik dostatečně financovány, aby si mohly dovolit nákup profesionálního softwaru. Vzhledem k jejich omezenému rozpočtu jsou pro ně většinou vhodná jen řešení

dostupná zdarma. Software dostupný zdarma často také není pro jejich potřebu úplně vhodný (různé aplikace pro veterináře). Ke srovnání bylo tedy vybráno několik aplikací, které jsou k dispozici za přijatelnou cenu nebo úplně bez poplatků. Srovnání je zaměřeno zejména na v ČR ze zákona povinné informace k evidenci, finanční dostupnost a některé další kategorie. Srovnání bude provedeno tabulkovou metodou.

Funkce	ASM	Pawlytics	iShelters	Shelterluv
Evidence zvířat	Ano	Ano	Ano	Ano
Evidence adopcí	Ano	Ano	Ano	Ano
Zdravotní záznamy	Ano	Ano	Ano	Ano
Evidence nákladů	Ano	Ano	Ano	Ano
Sponzorské dary	Ano	Ne	Ne	Ano
Export dokumentace	Ano	Ano	Ano	Ano
Správa osob	Ano	Ano	Ano	Ano
Incidenty zvířat	Ne	Ano	Ano	Ano

Zdroj: https://www.capterra.ie/ a webové stránky jednotlivých aplikací

Tabulka 1: Srovnání dostupného softwaru pro zvířecí útulky

Jak můžeme vidět v tabulce, všechny vybrané aplikace jsou velmi dobře vybaveny základními funkcemi. Aplikace budou dále stručně rozebrány a budou popsány jejich přednosti a zajímavé funkční vlastnosti.

2.4.1 Animal Shelter Manager (ASM)

Animal Shelter Manager[2] je jeden z mála specializovaných programů, který je k dispozici zdarma. Je aktivně vyvíjen a udržován od roku 1996. Je distribuován pod licencí open source (GNU-GPL[3]). Útulky si tedy mohou program stáhnout a o jeho běh se starat samy. Výrobce ale také dává možnost zřídit si za měsíční poplatek databázi přímo na jejich serverech. Cena je aktuálně 270 eur za rok a platná je do dubna 2021. V ceně je zahrnuta IT podpora, 10 GB úložného prostoru s možností navýšení, automatická publikace nalezených zvířat na různých adopčních a serverech, registrace čipovaných zvířat do registrů různých organizací a další zajímavé funkce. Uživatelské rozhraní aplikace působí na první pohled nepřehledně, ale je logicky členěno a je možnost pracovat ve více záložkách současně. Aplikace má také českou a slovenskou lokalizaci, překlad je ale pravděpodobně jen strojový. Statistiky na stránkách uvádí, že aplikaci používá téměř 17 000 uživatelů. Současná verze 3.x je napsaná v jazyce Python. Předchozí verze 2.x je napsaná v jazyce Java, dále už však není vyvíjena.

Výhody a zajímavé funkce:

- možnost provozu zdarma

[2]https://www.sheltermanager.com
[3]licence pro svobodný software

- mobilní aplikace
- tvorba vlastních dokumentů
- zabudovaný systém pro posílání e-mailů
- je multiplatformní

2.4.2 Pawlytics

Pawlytics[4] je softwarový startup pod univerzitou v Nebrasce v USA. Jedná se o webovou službu vyvíjenou podle modelu SaaS. SaaS (Software as a service) je model nasazení softwaru, kdy dochází k hostování aplikace provozovatelem služby. Služba je dále nabízena uživatelům přes Internet. Eliminováním potřeb instalace a provozu aplikace na vlastních zařízení se SaaS v poslední době stává oblíbeným způsobem provozu aplikace. SaaS vznikl jako reakce na potřebu snižování nákladů na software a jeho rychlé nasazení [9]. Front-end aplikace je vyvíjen ve frameworku React.Js. Aplikace má moderní a jednoduchý design, vše je velmi přehledné a poskytuje řadu funkcí. Cena je účtována za každou úspěšnou adopci a to 1 USD.

Výhody a zajímavé funkce:

- přijatelná cena
- webová aplikace
- možnost elektronického podpisu dokumentů
- moderní a funkční design
- bohatá vizualizace dat
- automatická generace profilu osob

2.4.3 iShelters

Aplikace iShelter[5] je skupina tří webových aplikací. Jedná se o iShelter, adminShelter a webServiceShelter. Hlavní aplikace iShelter je evidenční částí celé skupiny, umožňuje přidávat záznamy o zvířatech a osobách a spravovat již existující záznamy v systému. AdminShelter slouží ke konfiguraci hlavní aplikace, zakládají se zde účty uživatelům a nastavují se zde jejich přístupová práva. WebServiceShelter slouží jako webová stránka útulku. Aplikace slouží jako klasická stránka organizace a útulek zde může prezentovat například nabídky k adopci, bez nutnosti kopírovat data na jiný webový server. Aplikace je napsaná v jazyce PHP a využívá databázi MySQL. Cena aplikace se mění podle počtu zvířat přijatých za rok. Pro malé útulky, které přijmou do 180 zvířat za rok je cena 20

[4] https://pawlytics.com/
[5] https://www.ishelters.com/

euro. Dále se cena odvíjí do maxima 595 eur a to za přijatých 1000 zvířat na rok. Méně přehledná je karta samotného zvířete, většina akcí se zde provádí pomocí velkého množství malých tlačítek s ikonou, jejich funkce nemusí být uživateli na první pohled jasná.

Výhody a zajímavé funkce:

- výhodná cena pro menší útulky
- webová aplikace
- integrace vlastní webové stránky
- moderní a funkční design
- administrace zaměstnanců a plánování směn

2.4.4 Shelterluv

Shelterluv[6] je další webová aplikace pro evidenci zvířat v útulku. Pro vývoj se používá kombinace javascriptového frameworku Vue, PHP (framework Laravel) a redakčního systému Drupal. Cenová politika je podobná jako u aplikace Pawlytics. Cena je zde od 2 USD za adopci a hromadně je účtována měsíčně. Tato cena platí pro útulky, které přijmou do 500 zvířat za rok. Aplikace je k dispozici také pro mobilní telefony.

Výhody a zajímavé funkce:

- velké množství přednastavených statistických reportů
- mobilní aplikace
- neomezený úložný prostor
- propracovaný systém zdravotních záznamů a vakcinací
- splnění generovaných úkolů aktualizuje záznamy v databázi

2.4.5 Stručný přehled dalšího specializovaného softwaru

Jako u všech druhů softwaru jsou i u aplikací pro útulky k dispozici placená profesionální řešení. Tyto aplikace mají obvykle velké množství rozšiřujících funkcí, podporují automatický export na velké adopční stránky, integrované e-mailové schránky, přijetí několika zvířat najednou, sledování období karantény nebo hárání u fen, registr dlužníků, QR kódy pro jednotlivá zvířata, podrobné statistiky a reporty, čtení čárových kódů například různých druhů krmiva, propracovaný systém diet a mnoho dalších funkcí.

[6] https://www.shelterluv.com/

Problém, na který jsem při srovnávání těchto aplikací narazil, je, že webové stránky nemají často uvedený transparentní ceník. Útulek si nejdříve musí vyžádat nebo stáhnout demo verzi programu a poté si popřípadě o cenu zažádat e-mailem. Aplikace také často fungují na add-on[7] systému a za každý modul se platí zvlášť. Průměrně se ceny pohybují od 100 USD za měsíc až po několik tisíc USD ročně. Pro velké útulky mohou být vhodné aplikace jako Shelter Pro (2395 USD všechny moduly + měsíční poplatky za případný hosting) nebo Rescue-Connection Software (od 200 USD za měsíc ve vybavenější verzi, nejvyšší verze Enterprise je v řádu tisíců USD).

2.4.6 Závěr srovnání

V této kapitole bylo provedeno srovnání různých aplikací pro zvířecí útulky. Byly zde zahrnuty desktopové i webové aplikace a byly popsány jejich hlavní charakteristiky a vyzdviženy zajímavé funkce. Pozornost byla věnována také cenám jednotlivých zástupců. Pro menší útulky, při výběru podobného softwaru, je právě cena rozhodujícím faktorem. Ze všech zástupců mi přijdou nejzajímavější aplikace Animal Shelter Manager, protože je dostupný zdarma a aktivně stále vyvíjen a zdokonalován a Pawlytics, což je poměrně nová webová aplikace a oproti většině ostatních působí moderním a jednoduchým dojmem a přitom zachovává bohatou funkcionalitu. Všechny aplikace zmíněné v této kapitole jsou od zahraničních vývojářů, nejsou tedy většinou přeloženy do češtiny a pokud ano, jedná se o strojový překlad. Aplikace od českých vývojářů jsem v této oblasti nedohledal. Možnosti jejich vývoje bych rád ještě rozebral později v práci.

2.5 Funkce aplikace vyvinuté v rámci bakalářské práce

Aplikace byla navržena tak, aby dokázala evidovat většinu údajů jako podobná komerční řešení. Ještě než budou podrobněji rozebrány uživatelské požadavky útulku, chci zde stručně shrnout funkce a kritéria aplikace, podle kterých byly porovnány již existující podobné aplikace. Podrobnější popis jednotlivých funkcí bude následovat později v práci.

1. Evidence zvířat

   Aplikace odkáže evidovat všechny ze zákona povinné informace týkající se zvířat, jako je místo nálezu, hmotnost, identifikační znaky, plemena, adoptovaná zvířata a jejich nové majitele a úniky z útulku. Zvířecí evidence byla také doplněna o další užitečné informace.

2. Evidence adopcí

   Evidence adopcí je v aplikaci koncipována jako samostatná sekce s možností vytvoření nové adopce, nebo náhledu již existující. Adopce je řešena jednoduše provázáním záznamu zvířete a záznamu osoby (s rolí majitel) do

[7] označení pro přídavné doplňky do nějakého programu

nového vztahu. Ten je doplněn o datum adopce a uložen v databázi. Další částí záznamu je informace o tom zda bylo zvíře z adopce vráceno. Pokud ano je zde uvedeno datum a důvod vrácení.

3. Zdravotní záznamy

   Zdravotní záznamy zvířete obsahují popis záznamu, pokud se jednalo o veterinární zákrok je uveden veterinární lékař (osoba s rolí veterinář), který ho prováděl a při uvedení ceny je založen nový záznam o vzniklém nákladu. Zdravotní záznam se vztahuje vždy k určitému zvířeti.

4. Evidence nákladů

   Záznamy nákladů mohou být buď obecné a nebo se mohou vztahovat k jednotlivým zvířatům. Jako obecné náklady můžeme brát například kupované krmivo, kancelářské potřeby nebo dezinfekční a čistící prostředky. Náklady vázané ke zvířeti mohou být lékařské zákroky, nákup léků, prostředky pro zvířata se speciálními potřebami a další individuální náklady.

5. Sponzorské dary

   Aplikace eviduje také finanční sponzorské dary. V záznamu je uveden název a popis, darovaná částka, datum a jméno sponzora (osoba s rolí sponzor). Celkový přehled je dostupný ve statistikách a na stránce s financemi.

6. Export dokumentace

   Záznamy uložené v databázi by měly být v případě nutnosti dostupné i v papírové podobě. Proto aplikace umožňuje export dat z databáze do formátů PDF a XLSX. Možnost exportu je dostupná u všech hlavních sekcí.

7. Správa osob

   Evidence osob je řešena podobně jako evidence zvířat. U osob se ale eviduje menší množství informací. Podle své role může být osoba navázána na adopci (owner), sponzorský dar (sponsor), procházku (walker), veterinární zdravotní záznam (vet) a nebo může být evidována jako dobrovolník (volunteer). Základní údaje jsou jméno, příjmení, adresa a různé prostředky pro komunikaci s osobou, jako je telefon nebo e-mail.

8. Incidenty zvířat

   Incidenty jsou vázány k jednotlivým zvířatům. V záznamu je informace o popisu, datu a závažnosti incidentu na stupnici od 1 do 10, kde 10 je nejzávažnější. Tyto informace mohou být stěžejní například při umisťování zvířat do rodin s dětmi.

Aplikace je k dispozici zdarma a dle potřeby bude dále upravována. Uživatelské rozhraní jsem se snažil navrhnout jednoduše a moderně. Hlavní navigace probíhá v postranním menu v levé části obrazovky. V kartě zvířete potom ve formě záložek. Více chci uživatelské rozhraní rozebrat v programátorské části.

2.6 Uživatelské požadavky

Jednou z nejdůležitějších částí vývoje nového softwaru je bez pochyby zjištění požadavků budoucích uživatelů. Mezi zdroje požadavků může patřit legislativa (v našem případě veterinární zákon a zákon na ochranu zvířat) nebo požadavky zákazníka získané například konzultacemi (útulek). Pro zachycení požadavků zákazníka je vhodné použít modelovací jazyk UML a jeho *Use Case diagramy*. Špatně definované uživatelské požadavky mohou vést až k neúspěchu projektu nebo finančním ztrátám. Často spočívají v nedostatečném zapojení zákazníka od vývoje.

2.6.1 Modelovací jazyk UML a jeho diagramy

UML (Unified Modeling Language) je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů [10]. Současná standardizovaná verze jazyka 2.0 obsahuje 14 typů diagramů rozdělené do několika skupin. Strukturní diagram (diagram tříd, objektů, balíčků ..) slouží ke znázornění struktury aplikace, diagramy chování (diagram užití, stavový a aktivit) znázorňují dynamické chování a poslední diagramy interakce (diagram sekvenční, časování, komunikace a interakcí) znázorňují interakci mezi jednotlivými částmi systému[10]. Pro identifikaci uživatelských požadavků (requirements) budou použity diagramy případů užití, neboli UseCase diagramy. Ty znázorňují interakci uživatele se systémem. Jedná se o první krok vymezení rozsahu aplikace. Výsledná aplikace by neměla obsahovat funkce, které v těchto diagramech nejsou uvedeny. Pro znázornění základních struktur při popisu aplikace budou použity stručné diagramy tříd (Class diagramy).

2.6.2 Analýza uživatelských požadavků

Uživatelské požadavky, které byly konzultovány s útulkem jsou rozděleny do dvou skupin. První skupina obsahuje požadavky plynoucí ze zákona o ochraně zvířat proti týrání č. 246/1992 Sb.. Druhá skupina byla z části požadována útulkem a z části dohodnuta jako rozšíření funkcionality. Podle důležitosti bude druhá skupina rozdělena na důležité požadavky a doplňkové požadavky. Všechny požadavky budou zpracovány do diagramu případů užití z pohledu pracovníka útulku. Pro přehlednost bude v seznamu požadavků uveden i identifikátor případu užití, který jej pokrývá (např. UC03 Spravovat záznamy zvířete).

Požadavky vyplývající ze zákona:

1. Systém bude umět evidovat přijatá zvířata s uvedením počtu, druhu, identifikačních znaků, hmotnosti, data a místa nálezu zvířat (UC03 Spravovat záznamy zvířete)

2. Systém bude umět evidovat zvířata vydaná z útulku a jejich nové majitele, data adopce a adresy, kde bude zvíře nově chováno (UC21 Spravovat adopce + UC03 Spravovat záznamy zvířete)

3. Systém bude umět evidovat případné úniky z útulku (UC28 Spravovat úniky a úhyny + UC08 Založit nový záznam o útěku)

4. Systém bude umět evidovat informace o zdravotním stavu zvířete a o dalších zdravotních záznamech (UC15 Spravovat zdravotní záznamy)

5. Systém bude umět evidovat případné úhyny/eutanazie (UC28 Spravovat úniky a úhyny + UC08 Založit nový záznam o úhynu)

Důležité požadavky:

1. Systém bude počítat náklady na pobyt zvířete v útulku (UC10 Spravovat náklady)

2. Systém bude umožňovat evidovat osoby pro výpomoc v útulku (UC11 Spravovat osoby)

3. Systém bude umět uchovávat soubory zvířete z externích zdrojů (UC19 Spravovat soubory zvířete)

4. Systém bude umět vyhledávat zvířata podle čísla čipu/tetování (UC02 Vyhledat evidenční kartu zvířete)

5. Systém bude umět evidovat incidenty zvířat (UC16 Spravovat incidenty)

6. Systém bude umět přidávat a evidovat profily osob (UC11 Spravovat osoby)

7. V systému bude možná změna informací o útulku (UC26 Upravit informace o útulku)

Doplňkové požadavky:

1. Systém bude umět zobrazovat poznámky vytvořené k určitému datu (UC27 Přidat upomínku na daný den)

2. Systém bude počítat finanční sponzorské dary (UC29 Sponzorské dary)

3. Systém bude umět zakládat procházky (UC17 Spravovat venčení)

4. Do systému půjdou přidat libovolné vlastnosti zvířete (vzhled, plemeno ...) (UC25 Přidat nové druhy)

2.6.3 Případy užití a UseCase diagram vlastní aplikace

Dále budou rozebrány případy užití aplikace. Grafická podoba případů užití je zachycena na UseCase diagramu (obrázek č. 1).

- UC01 Přijmout zvíře do útulku

  Umožňuje pracovníkům vložit do systému úplně nové zvíře. S novým zvířetem vzniká jeho složka a evidenční karta. Pokud již zvíře v útulku pobývalo je vyhledáno v databázi a je mu v evidenční kartě pouze založen nový pobyt.

- UC02 Vyhledat evidenční kartu zvířete

  Umožňuje pracovníkům vyhledat již existující kartu zvířete a dále spravovat její záznamy. Vyhledání může probíhat podle ID, jména, čísla čipu, plemena a druhu. Pro úpravu je nutné kartu vybrat ve výsledcích a otevřít ji.

- UC03 Spravovat záznamy zvířete

  Během pobytu v útulku mohou být se zvířetem propojeny nejrůznější záznamy. V evidenční kartě mohou být tyto záznamy spravovány. Kromě propojených záznamů je možnost upravovat informace týkající se přímo zvířete jako je jméno, číslo čipu, váha, stáří, druh, plemeno, barva srsti, typ srsti a tak dále. Správa záznamů probíhá v evidenční kartě zvířete a navigace je tady pomocí tlačítek v horní části.

- UC04 Založit nový pobyt v útulku

  Pobyt je jeden ze záznamů, který může být na zvíře navázán. Vzniká při příjmu nového zvířete do útulku, může být založen v již existující kartě zvířete a nebo při příjmu zvířete zpět z neúspěšné adopce. V záznamu se kromě doby kdy zvíře v útulku pobývalo uchovávají i informace o místě a datu nálezu.

- UC05 Založit novou evidenční kartu zvířete

  Při příjmu nového zvířete do útulku je mu automaticky vytvořena nová karta.

- UC06 Ukončit pobyt zvířete v útulku

  Pobyt zvířete může být v určitých situacích ukončen. Zakončení pobytu má 3 základní typy: adopce, útěk nebo úhyn. Podle výběru zakončení je možnost dalšího kroku a vytvoření doplňkového záznamu.

- UC07 Založit nový záznam o útěku

  Při ukončení pobytu s typem útěk může být založen nový záznam o úniku zvířete z útulku. Založení nového záznamu je také možné přímo z obrazovky Útěky a úhyny po uvedení ID zvířete.

- UC08 Založit nový záznam o úhynu

  Při ukončení pobytu s typem úhyn může být založen nový záznam o úhynu zvířete. Založení je možné i z obrazovky Útěky a úhyny po vyplnění ID zvířete.

- UC09 Založit novou adopci

  Umožňuje pracovníkům založit nový záznam o adopci. Jedná se vlastně o provázání zvířete a osoby do nového záznamu adopce. Při této úloze je také generován adopční list. Adopční list je uložen ve složce zvířete a je možné se k němu dostat ze stránky se soubory zvířete. Adoptovat lze jen zvířata, která se nacházejí v útulku.

- UC10 Spravovat náklady

  Umožňuje zakládat nové náklady, které útulek musel vynaložit. Správa nákladů je z části dostupná i z karty zvířete, kde může být založen nový náklad přímo navázaný na konkrétní zvíře. Pokud chce uživatel založit obecný náklad je možné to provést ze samostatné stránky z menu.

- UC11 Spravovat osoby

  Umožňuje pracovníkům spravovat profily osob. Záznamy o osobách mohou být použity v aplikaci při založení adopce, procházky, sponzorského daru nebo veterinárního zákroku. Evidovat adresu nového majitele, tedy nové místo chovu zvířete, je povinné i ze zákona.

- UC12 Založit nový profil osoby

  Umožňuje založení nového profilu osoby, vyplnění informací a výběr rolí. Toto je možné provést přímo ze stránky při vyhledávání.

- UC13 Vyhledat profil osoby a UC14 Editovat záznamy osoby

  Umožňuje pracovníkům vyhledat a upravovat profily osob a jejich rolí.

- UC15 až UC20 Správa záznamů zvířete

  Umožňují v evidenční kartě zobrazit a upravovat záznamy, které jsou propojeny s konkrétním zvířetem.

  - UC15 Spravovat zdravotní záznam

    Umožňuje zakládat a upravovat zdravotní záznamy. Zdravotní záznam se skládá z názvu, popisu, data, ceny a záznamu o veterináři, který šetření prováděl.

  - UC16 Spravovat incidenty

    Umožňuje zakládat a upravovat incidenty zvířete. Záznam o incidentu se skládá z data, popisu a závažnosti incidentu.

  - UC17 Spravovat venčení

    Umožňuje zakládat a upravovat záznamy o procházkách se zvířetem. Záznam o procházce se skládá z data, poznámky a z údaje o venčiteli.

  - UC18 Spravovat navázané osoby

    V evidenční kartě je možnost vybrat osoby s příslušnou rolí a provázat je se záznamem zvířete. Jedná se o veterináře, původního majitele

a nového majitele. Přímo v kartě jsou k nahlédnutí bližší informace (kontakt, adresa) ke každé z osob.

  - UC19 Spravovat soubory zvířete

    Umožňuje nahrávat a mazat soubory do složky zvířete uložené na disku. Je zde také možnost soubor vytisknout

  - UC20 Generovat evidenční kartu

    Umožňuje vygenerovat evidenční kartu zvířete ve formátu PDF. Vychází ze záznamů provázaných se zvířetem. Může sloužit pro jeho fyzickou archivaci.

- UC21 Spravovat adopce

  Umožňuje pracovníkům spravovat záznamy adopcí. Správa adopcí je dostupná z navigačního menu pod sekcí Adopce.

- UC22 Vyhledat adopci

  Umožňuje vyhledat záznam o adopci a nahlédnout na jeho detaily. Vyhledání je možné podle jména majitele, ID adopce nebo jména zvířete.

- UC23 Vrátit zvíře z adopce

  Po vyhledání adopce je možnost v jejím detailu zvíře z adopce vrátit. Je potřeba uvést datum a důvod vrácení. Zvíře je po tomto úkonu vráceno do útulku.

- UC24 Otevřít nastavení aplikace

  Umožňuje pracovníkům provádět uživatelské změny v aplikaci. Je zde možnost přidávat možnosti do výběrových boxů, například různé druhy zvířat, plemena, barvy srstí, typy srstí atd (UC25). Dále je zde možnost upravit základní informace o útulku jako je adresa, telefon, číslo účtu a tak dále (UC26).

- UC27 Přidat upomínku na daný den

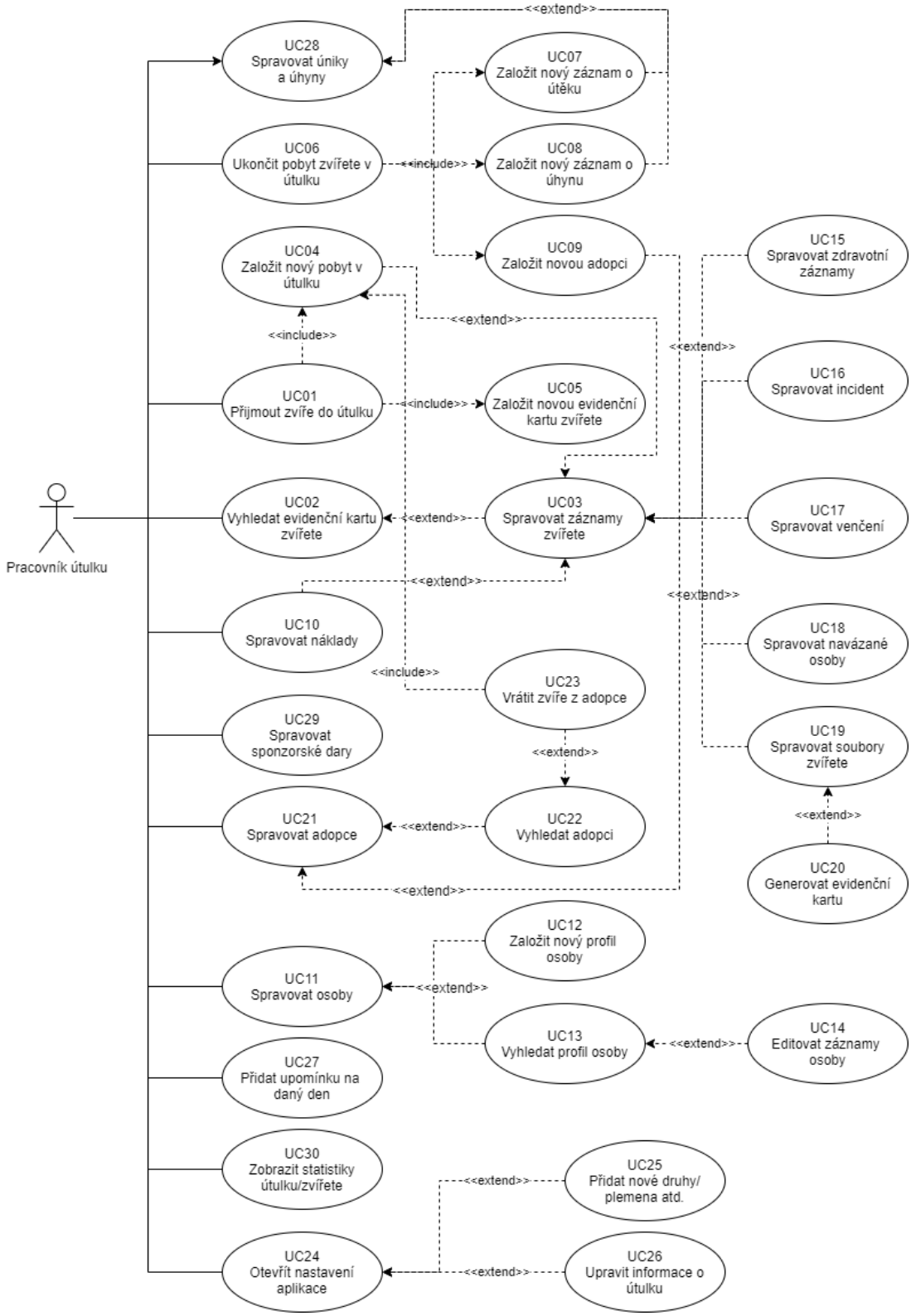
  Umožňuje pracovníkům přidat upomínku nebo úkol na vybraný den a zobrazit ho v daný den na úvodní obrazovce.

- UC28 Spravovat úniky a úhyny

  Úniky a úhyny je možno spravovat i samostatně z určené stránky. Je s nimi tak možné nakládat i bez otevření karty zvířete.

- UC28 Spravovat sponzorské dary

  Umožňuje evidovat finanční sponzorské dary, které útulek dostane. Dar může být navázán na osobu s rolí sponzora.

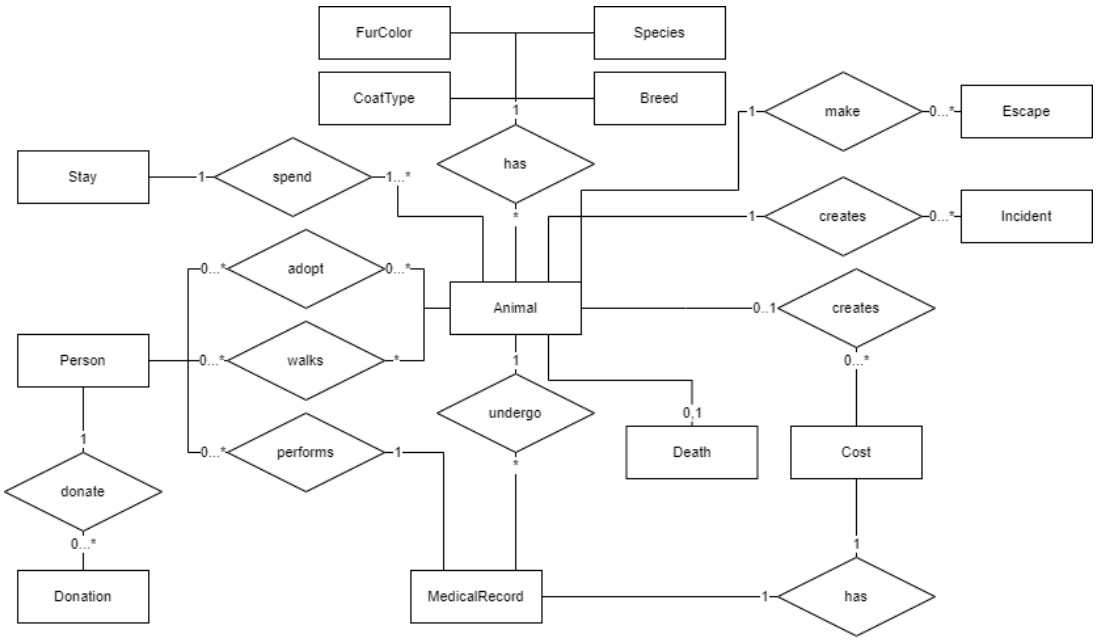


Obrázek 1: UseCase diagram aplikace.

2.7 Návrh databáze na základě požadavků

Z uživatelských požadavků je možné předem odhadnout, jaké entity se budou v systému vyskytovat. Perzistentní data těchto entit bude potřeba uchovávat v databázi. Jednoduchý návrh databáze je možné provést pomocí entitně-relačního diagramu (ERD). Před samotným modelování databáze nám poslouží pro jednodušší znázornění její struktury a poskytne určitou abstrakci vztahů jednotlivých entit. Detailní pohled na databázi včetně atributů entit v tabulkách bude rozebrán v programátorské části práce.

Entitně-relační diagram ukazuje vztahy sad entit uložených v databázi. Entita v tomto kontextu je objekt (řádek tabulky), součást dat. Sada entit je kolekce podobných entit (tabulka). Tyto entity mohou mít atributy, které definují jejich vlastnosti. Původním autorem ERD je Peter Chen z Carnegie-Mellon University v Pittsburghu. Několik drobných vylepšení k původnímu konceptu přidali Charles Bachman a James Martin. Pro vytvoření ERD existuje několik způsobů, které se liší v grafické podobě diagramu. Při znázornění databáze evidenční aplikace (obrázek č. 2) byla využita původní Chenova notace.



Obrázek 2: Entitně relační diagram (ERD) databáze

3 Programátorská část

Tato kapitola popisuje vlastní implementaci aplikace, která je výstupem této práce. Při vývoji aplikace byla využita řada technologií, zvolené technologie zde budou popsány. Celá aplikace je vyvíjena v .NET frameworku firmy Microsoft a prostředí pro vývoj Visual Studio. Jedná se o soubor technologií pro vývoj desktopových, mobilních i webových aplikací. Technologie využité pro vývoj aplikace budou představeny a bude zde stručně nastíněna jejich funkcionalita doplněná o vhodné ukázky z aplikace.

Jako první jsou rozebrány typické situace při práci s aplikací a spolupráce tříd při jejich provádění. Struktura je znázorněna na stručných UML Class diagramech. Dále je uvedena struktura databáze a stručně popsány jednotlivé tabulky. V kapitole je také stručně představen objektově orietnovaný programovací jazyk C#, ve kterém je aplikace napsána, a vývojové prostřední Visual Studio společně s Microsoft .NET frameworkem. Následně je blíže rozebrán aplikační model Windows Presentation Foundation a Material Design In XAML použité pro tvorbu uživatelského rozhraní. Dále jsou představeny technologie Microsoft SQL Server Express a LINQ, použité pro tvorbu databáze a další práci s ní. V závěru jsou informace o softwarovém návrhovém vzoru Model-View-ViewModel(MVVM), který nabízí řešení pro oddělení logiky aplikace a uživatelského rozhraní. Pro jeho snadnou implementaci byl využit framework Caliburn Micro. Návrhový vzor bude popsán na konkrétních ukázkách z aplikace.

Uvedené technologie budou doplněny o vhodnou ukázky kódu z aplikace a popis jednotlivých funkcí. U některých technologií budou pro jednoduchost a přehlednost použity obrázky a diagramy jejich použití ve vlastní aplikaci.

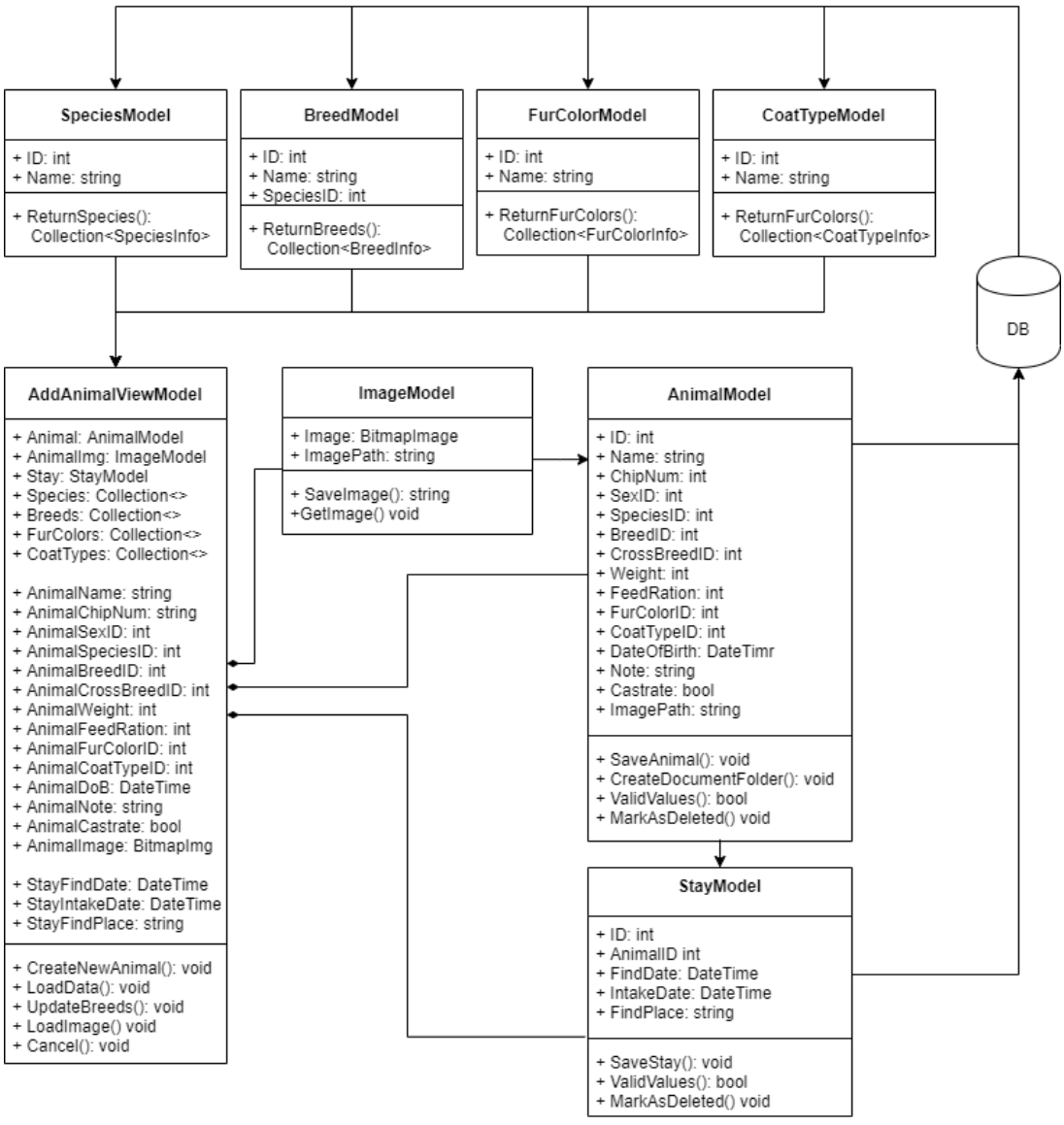
3.1 Analýza případů užití a zjednodušené Class diagramy aplikace

V této podkapitole budou rozebrány vztahy mezi třídami objektů v aplikaci. Rozbor proběhne jen u typických situací, které mohou nastat za běhu aplikace. Vztahy a interakce jsou často mezi třídami podobné, proto budou jednotlivé situace rozebrány na modelovém příkladu a zmíněny v popisu u každého z nich. Pro zjednodušení nebudou mezi třídami uvedeny násobnosti a budou v nich na diagramu znázorněny jen metody a atributy, které se v dané situaci přímo využívají.

Přidání nového zvířete

Základní funkce aplikace je přidání nového zvířete do evidence. Na následujícím diagramu (obrázek č. 2) je znázorněna spolupráce objektů při vytvoření nového zvířete. Při tomto úkonu vzniká v databázi nový záznam o zvířeti se všemi informacemi, které uživatel při založení poskytl. Dále se zakládá složka zvířete na disku. Složka později slouží pro uchovávání dokumentů. Poslední částí je vytvoření nového objektu *Stay* (pobyt) s ID zvířete, který je poté také uložen do databáze. Podobná spolupráce probíhá například i u zakládání nového profilu

osoby. Roli manažera tohoto úkonu tvoří *AddAnimalViewModel*. Ve spolupráci s dalšími třídami poskytuje při založení informace potřebné k navázání správných hodnot do vlastností zvířete, které jsou reprezentovány pomocí ID. Například pomocí metody *ReturnBreeds* třídy *BreedModel* získá z databáze kolekci plemen a jejich ID. Více o ViewModelech a Modelech bude rozepsáno v programátorské části.

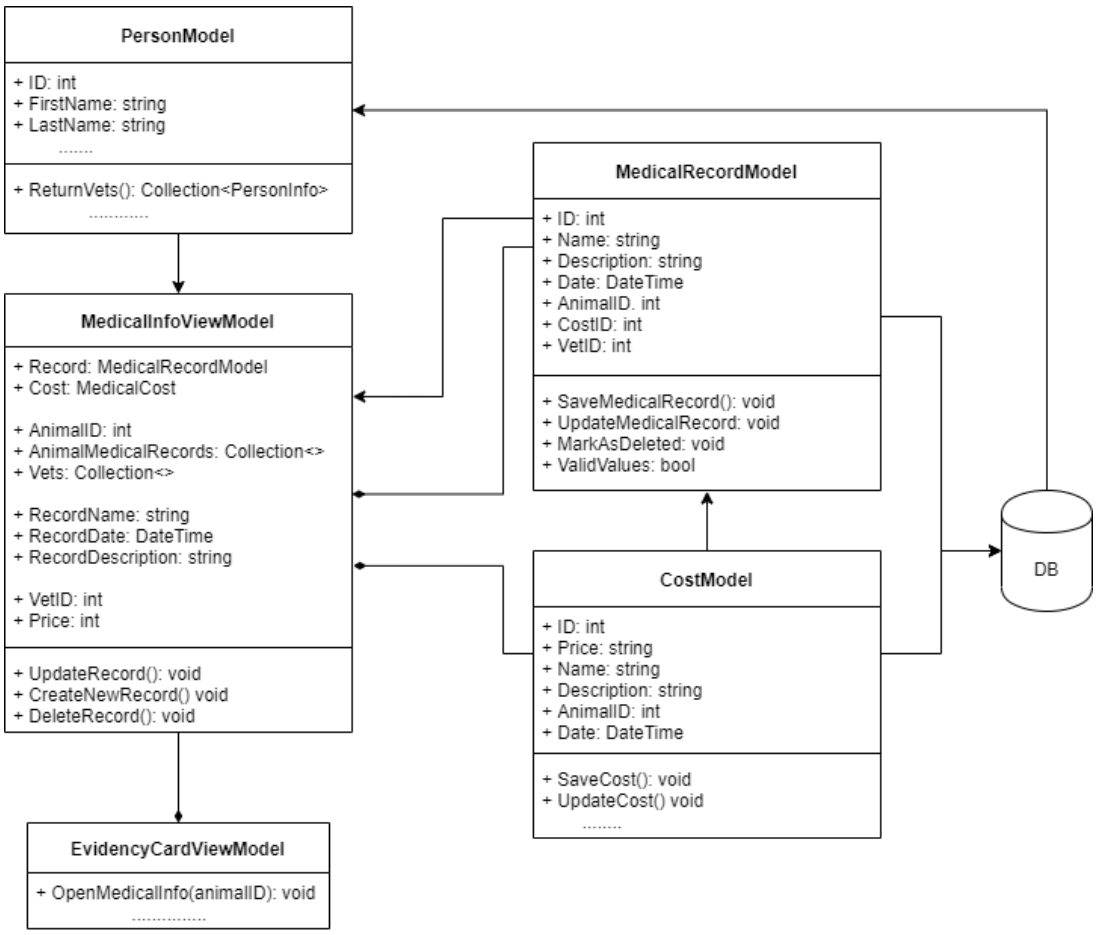


Obrázek 3: Class Diagram přidání nového zvířete.

Přidání záznamu ke zvířeti

V evidenční kartě zvířete je možnost přidání několika druhů záznamů. Patří sem například procházky, incidenty, náklady, zdravotní záznamy nebo nové pobyty v útulku. Tyto a další záznamy jsou se zvířetem provázány v databázi

pomocí jeho ID. Na následujícím diagramu (obrázek č. 3) je znázorněna situace přidání nového zdravotního záznamu. Podobná forma spolupráce probíhá i při vytváření dalších výše zmíněných záznamů. Každý záznam využívá při zakládání svou třídu a někdy dochází i ke spolupráci s dalšími třídami. Pro založení nového zdravotního záznamu je potřeba, aby třída *MedicalRecordModel* spolupracovala ještě s třídou *CostModel* a *PersonModel*, protože zde evidujeme veterináře, který zdravotní úkon prováděl, a náklady na úkon.

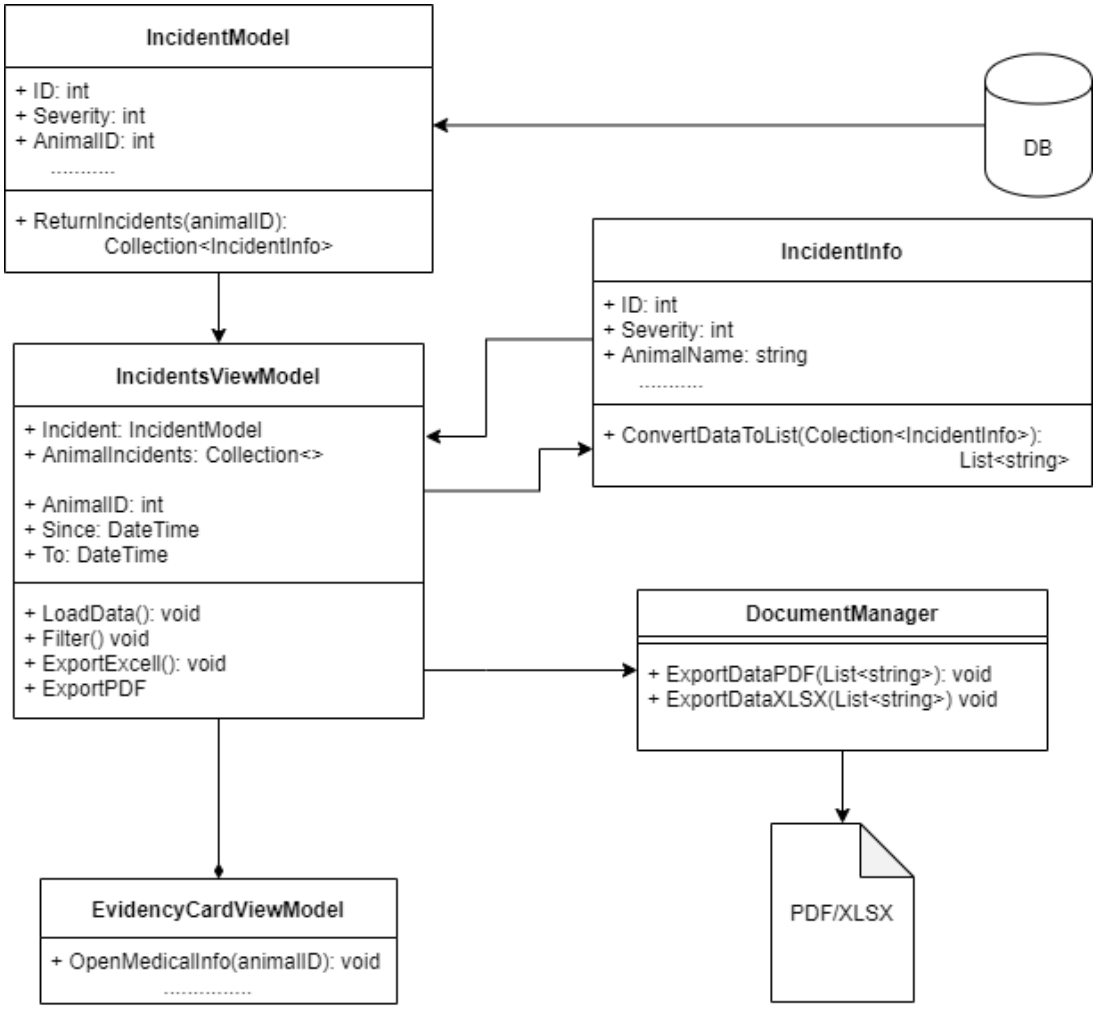


Obrázek 4: Class Diagram přidání nového zdravotního záznamu zvířete.

Export dat z evidence

Poslední situací rozebranou pomocí Class diagramu (obrázek č. 4) je export dat z databáze. Export záznamů je možný do dvou formátů, klasicky do PDF a také do XLSX pro tabulkové editory. Data jsou ve výsledku vyobrazena v tabulce. Do formátu PDF je vhodnější exportovat záznamy, které mají menší počet sloupců. Formát XLSX je vhodný pro všechny druhy záznamů. Export je možný ze všech důležitých sekcí. Rozebrán bude export z evidenční karty zvířete a to konkrétně jeho incidentů. Export záznamů je možný ze stránek vyhledávání zvířat, osob, adopcí, dotací, nákladů a ze všech stránek v evidenční kartě zvířete.

Při exportu využívají jednotlivé ViewModely třídu *DocumentManager* a také pomocné třídy jednotlivých Modelů pro převod dat do vhodného formátu před exportem.



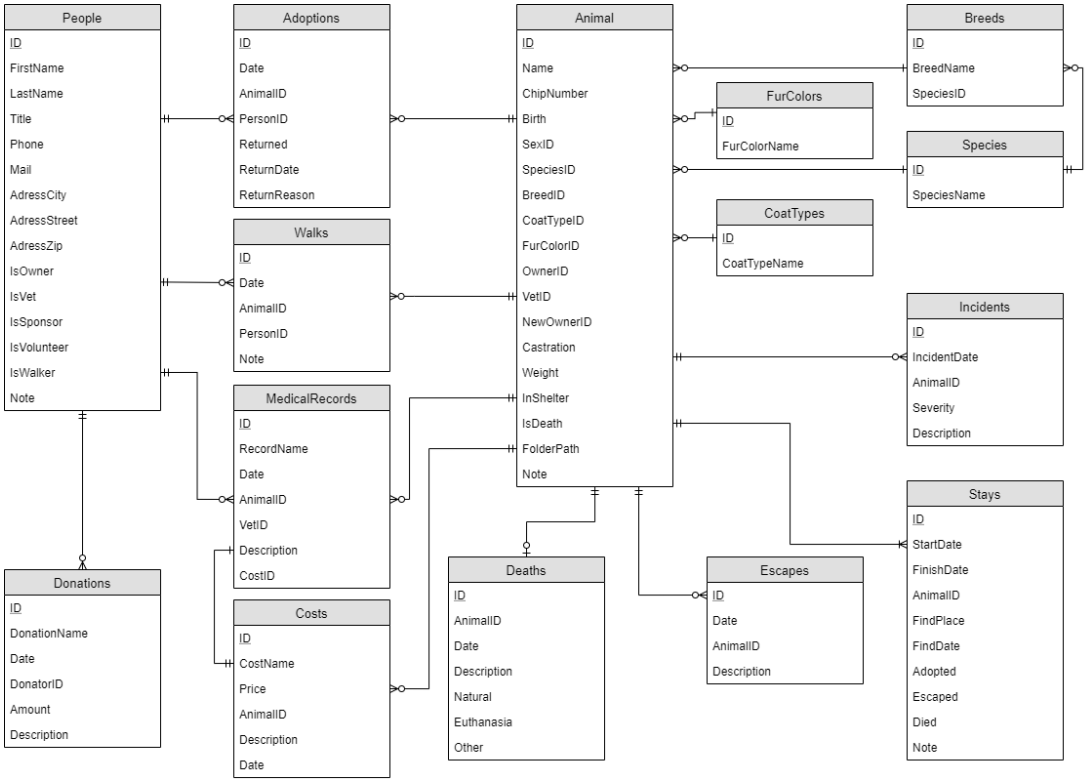
Obrázek 5: Class Diagram přidání nového zdravotního záznamu zvířete.

3.2 Struktura a model databáze

Jednou z nejdůležitějších částí evidenční aplikace je databáze. Databáze se skládá z jednotlivých tabulek se záznamy. Každá tabulka v databázi aplikace obsahuje pole, které jednoznačně identifikuje každý řádek, který se v ní nachází (ID). Tato hodnota se nazývá primární klíč a nesmí být prázdná. Pole primárního klíče můžete zahrnout do jiných tabulek jako odkaz na tabulku, která je zdrojem primárního klíče. V těchto tabulkách se tato pole nazývají cizí klíče. Cizí klíč je tedy vlastně primárním klíčem jiné tabulky [11]. Reference dat z jiné tabulky pomocí cizího klíče nám umožňuje přístup k datům v tabulce, kde je hodnota cizího klíče přítomna jako primární klíč. Navigaci pomocí cizích klíčů rozeberu

ještě později, v části o použitých technologiích. V této podkapitole bude popsána základní struktura celé databáze a vazby mezi tabulkami pomocí cizích klíčů. Model databáze je znázorněn pomocí diagramu, na něm můžeme nejlépe vidět všechny propojení, které se mezi jednotlivými entitami nacházejí.

Ve všech tabulkách se nachází pole *IsDeleted*. Toto pole označuje uživatelem smazané záznamy. Do databáze jsem jej zavedl proto, aby měli zaměstnanci možnost obnovit, například omylem, smazané záznamy. Jedná se o flag[8] a při smazání záznamu se do pole vloží aktuální čas. Záznamy, které mají v tomto poli jakýkoliv časový záznam se nevrací jako výsledky hledání. Přístup se nazývá Soft delete a je vhodný v situacích, kdy správu databáze obstarává sám uživatel bez databázového administrátora [12].



Obrázek 6: Model databáze aplikace

- **Tabulka Animals (Zvířata)**
  Záznamy o zvířatech jsou základem celé struktury. Každý záznam v tabulce má své ID, které každý řádek identifikuje. Další údaje můžeme rozdělit na specifické a odkazy na další tabulky pomocí ID. Specifické údaje jako jsou *Name* (jméno), *Weight* (váha) nebo *FeedRation* (krmná dávka) jsou specifické pro každé zvíře a jsou uchovávány přímo v tabulce *Animals*. Odkazy na další tabulky (cizí klíče) obsahují hodnoty ID záznamů z jiných tabu-

[8] flag je hodnota, která funguje jako signál pro funkci nebo proces

lek. Patří sem *SpeciesID* (odkazuje na záznam v tabulce druhů), *BreedID* (odkazuje na záznam v tabulce plemen) a další.

- **Tabulka People (Lidé)**
  Záznamy osob jsou druhou významnou částí databáze. Kromě základních údajů jako jsou jméno, příjmení nebo adresa mohou mít osoby spojené s útulkem jednu nebo více rolí. Pro role slouží pole *IsOwner* (role majitel), *IsVet* (role veterinář), *IsSponzor* (role sponzora), *IsWalker* (role venčitele), *IsVolunteer* (role dobrovolníka). Podle role se záznamy osob využívají v další evidenci, například osoby s rolí venčitele mohou být přiřazeny k procházkám. Role dobrovolník je důležitá pro evidenci seznamu dobrovolníků, které může útulek kontaktovat při mimořádných situacích. Každá role bude zmíněna u tabulky, ve které se může použít záznam osoby, která danou roli má.

- **Tabulka Stays (Pobyty)**
  Pobyt je první záznam, který je na konkrétní zvíře navázán. Vzniká společně se založením nového zvířete, nebo jeho příjmu zpět do útulku například z neúspěšné adopce. Záznam pobytu obsahuje informace o ID samotného zvířete, datu začátku a konce pobytu, datu a místě nalezení, poznámky a označení jakým způsoben byl pobyt ukončen. Pobyt může končit adopcí, útěkem nebo úhynem zvířete.

- **Tabulky vlastností zvířete**
  Patří sem tabulky se záznamy, jejichž primární klíč se využívá k popisu vlastností zvířete. Většinou obsahují jen ID a název vlastnosti. Patří sem *Species* (druhy), *CoatTypes* (druhy srsti), *FurColors* (barvy srsti) a *Breeds* (plemena), které navíc obsahují odkaz na druh.

- **Tabulka Adoptions (Adopce)**
  Záznamy o provedených adopcích obsahují datum adopce, ID zvířete a nového majitele, informaci o tom jestli bylo zvíře z adopce vráceno a doplňující informace o datu vrácení a jeho důvodu. Přes ID zvířete a majitele se můžeme dostat k bližším informacím o nich. Jde vlastně o typ vazební tabulky mezi zvířetem a osobou, vytvářející určitý vztah mezi nimi. Kardinalita[9] vztahu by měla být 1:N, protože majitel může vlastnit více zvířat z útulku ale zvíře může být v jeden okamžik vlastněno jen jedním majitelem. V roli osoby zde může figurovat jen osoba s rolí majitel.

- **Tabulka Walks (Procházky)**
  Velmi podobné jako u adopcí. Procházka je naplánovaná na určité datum a může být doplněna o poznámku. Do procházky může být přidáno jen ID osoby s rolí *Walker*.

[9] mohutnost

- **Tabulka MedicalRecords (Zdravotní záznamy)**
  Zdravotní záznamy jsou důležitou částí evidence, protože útulek musí evidovat všechny lékařské úkony na zvířatech. Každý záznam obsahuje název a popis. Na každý z nich je navázán záznam v tabulce *Costs* a dá se tak zjistit jeho cena. Další části lékařského záznamu jsou ID zvířete a ID veterináře (osoba s rolí *Vet*).

- **Tabulka Costs (Náklady)**
  V tabulce nákladů se evidují všechny výdaje útulku. Každý záznam má jméno, popis, cenu a datum. Záznam může být také navázán na konkrétní zvíře. Jeho ID je potom vyplněno v příslušném poli.

- **Tabulka Donations (Sponzorské dary)**
  V tabulce se evidují všechny sponzorské dary. Na sponzorský dar může být navázána osoba s rolí sponzora. Obsahuje informace o hodnotě daru, názvu, popisu a datu, kdy byl dar obdržen.

- **Tabulka Incidents (Incidenty)**
  Incident je záznam navázaný na určité zvíře. Každý incident má datum, popis, cenu a číslo ID zvířete. Záznam je doplněn o závažnost od 1 do 10.

- **Tabulka Escapes (Útěky)**
  Útěk je jeden ze záznamů, který se zakládá při ukončení pobytu zvířete v útulku. Občas se stává, že zvíře z útulku unikne a všechna tato zvířata musí útulek evidovat. Záznam obsahuje ID zvířete, datum úniku a popis.

- **Tabulka Deaths (Úhyny)**
  Poslední možnost, jak může skončit pobyt zvířete v útulku je jeho úhyn. Kromě základních informací, jako je datum, ID zvířete, a popis, obsahuje i typ úmrtí. Typy jsou tři - eutanazie, přirozený úhyn, jiný důvod.

- **Tabulka DiaryRecords (Poznámky)**
  Do databáze se také ukládají poznámky a úkoly zaměstnanců. Záznam nemá žádné vazby, obsahuje jen datum a text záznamu. Poznámky jsou v daný den zobrazeny na úvodní obrazovce.

- **Tabulka Shelter (Útulek)**
  Poslední tabulkou databáze je tabulka s informacemi o útulků. Informace z ní se používají například při generování dokumentů a jsou pro snadný přístup zobrazeny na úvodní obrazovce aplikace. Nalezneme zde jméno útulku, telefony, e-mail, adresu a číslo účtu. Zaměstnanci mají možnost informace upravovat v nastavení.

3.3 Programovací jazyk C#

Aplikace byla naspána v jazyce C#. V této podsekci bude rozebrána jeho charakteristika. Budou zde představeny pojmy, které budou používány dále v pro-

gramátorské části práce.

Jazyk C#je objektově orientovaný a typově bezpečný programovací jazyk vyvinutý společností Microsoft. Třídy v C#jsou seskupeny do jmenných prostorů. C# je schválený standardizačními komisemi ECMA[10] a ISO[11][13, 14].

Jazyk podporuje pouze jednoduchou dědičnost, implementace vícenásobné dědičnosti může být provedena pomocí rozhraní. Třída se skládá z členů. Členové třídy reprezentují data a chování třídy. Mezi nejčastěji používané patří *methods* (výpočty a akce prováděné třídou), *properties* (akce spojené s zápisem do proměnných třídy - *getter* a *setter*, slouží k zapouzdření dat) a *constuctors* (obsahuje akce vykonané při vytvoření objektu dané třídy) Jednotlivé členy mohou být specifikovány některým z modifikátorů přístupu. Základní modifikátory přístupu jsou *public* (přístupné jakýmkoli kódem v aktuální nebo externí sestavě), *protected* (přístupnost ve stejné třídě nebo odvozené třídě) a *private* (přístupnost ve stejné třídě)[13, 14].

Dále rozlišuje velká a malá písmena, je tedy *case sensitive*. Správa paměti je v C#automatická, o její uvolňování se stará *Garbage Collector*. Pomocí try-catch bloků je také možné zpracovávat chyby a korektně se při jejich výskytu zachovat.

3.4 Visual Studio (VS)

Aplikace byla vyvíjena v integrovaném vývojovém prostředí od firmy Microsoft. Visual studio je možné používat k vývoji počítačových programů, webových aplikací, webů nebo mobilních aplikací. Vývojové prostředí podporuje 36 programovacích jazyků[15]. Dále je k dispozici v několika verzích a to pro studenty a nezávislé vývojáře ve verzi Community, komerční verzi Proffesional a nadstandardní verzi Enterprise.

3.5 Microsoft .NET Framework

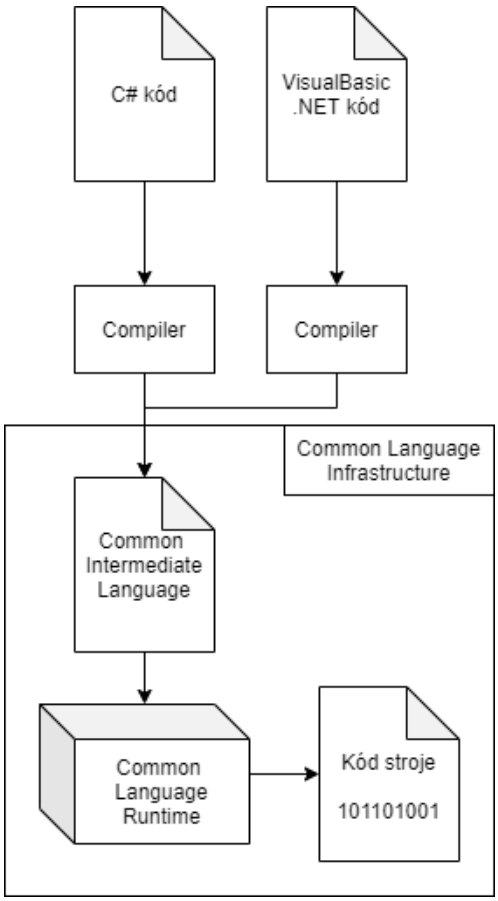
Microsoft .NET Framework je programovací infrastruktura vytvořená společností Microsoft. Společnost Microsoft začala framework vyvíjet koncem 90. let a na konci roku 2000 vydala první beta verzi .NET 1.0. Během roku 2000 také pracovala na standardizaci *Common Language Infrastructure* (CLI) a jazyka C#. Framework v základu nepředepisuje použití žádného konkrétního jazyka, aplikace je vždy přeložena do přechodného jazyka *Common Intermediate Language* (CIL). CIL je nejnižší, člověkem čitelný, objektově orientovaný jazyk. Architektura frameworku se skládá z následujících částí:

- Common Language Infrastructure and Runtime (CLI a CLR)

  Jedná se standardizované technické normy, poskytující jazykově nezávislou platformu pro vývoj a běh aplikací. Umožňuje spolupráci různých jazyků,

[10] ECMA-334
[11] ISO/IEC 23270

které jsou s CLI kompatibilní. Kód napsaný v těchto jazycích se kompiluje do *Common Intermediate Language* (CIL) Po spuštění kódu se CIL kompiluje do jazyka stroje závislého na konkrétním hardwaru. Celý proces provádí *Common Language Runtime* (CLR), což je virtuální stroj, který je součásí .NET frameworku. CLR[16].



Obrázek 7: Překlad kódu pomocí CLI

- Assemblies (sestavy) a knihovna tříd (Class library)

  Kompilovaný CIL kód je uložen v CLI sestavě. *Assemblies* (sestavy) jsou v .NET framewroku logické kolekce souborů, které obsahují nějakou aplikaci. Sestavy se ve Windows vyskytují dvojího typu DLL[12] nebo EXE [13] soubor. Standardní knihovny mohou být tako rozšířeny o nové funkce, příklady takových rozšíření mohou být ADO.NET nebo *Language Integrated Query* (LINQ), které slouží k manipulaci s daty. Patří sem také rozšířený set základních knihoven například Windows Forms, ASP.NET, a *Windows Presentation Foundation* (WPF), které slouží pro tvorbu grafického rozhraní. Souhrně se označují jako .NET *Framework Class Library* (FCL).

[12]Dynamic-link library - knihovny funkcí
[13]bežná přípona spustitelného souboru

- Modely aplikací

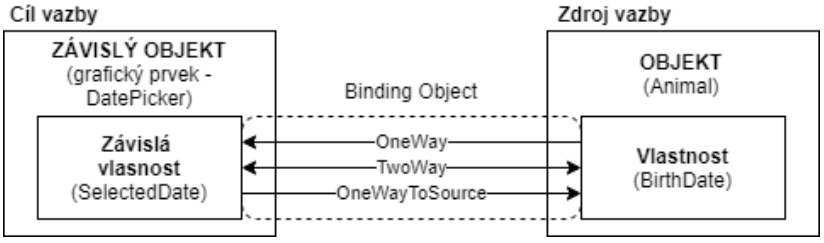
  Pro vytváření aplikací v .NET frameworku se využívá více modelů aplikací. Všechny tyto modely mohou využívat všech funkcí, které nabízí FCL. Ve výchozím nastavení podporuje framework konzolové aplikace, aplikace Windows Forms, *Windows Presentation Foundation* (WPF) a ASP.NET. Zmíněný ASP.NET je webový aplikační rámec pro produkci dynamických webových stránek. Při tvorbě aplikace byl využit model *Windows Presentation Foundation*, který používá značkovací jazyk XAML a bude blíže rozebrán.

3.5.1 Windows Presentation Foundation (WPF)

WPF je grafický subsystém .NET frameworku, který se používá pro psaní běžných aplikací pro operační systém Windows. WPF zahrnuje nový značkovací jazyk založený na XML, který se nazývá XAML (*Extensible Application Markup Language*). [17] Mezi hlavní výhody WPF patří, že prvky nejsou nezávislé na rozlišení obrazovky, podporuje zanořování prvků (control inside control) a je možné pro ovládací prvky definovat znovu použitelné vzory a styly. WPF má také přímo vestavěnou sadu datových služeb, které vývojářům umožňují vytvořit vazbu mezi prvkem a daty (zdroj a cíl) nebo také mezi dvěma prvky.

3.5.1.1 WPF Data binding

Tato technika se využívá k propojení XAML elementů a prvků s daty v kódu, který je obsluhuje. Je možné vytvořit propojení prvku *CheckBox* s logickou proměnnou (bool) nebo také propojit prvek *DataGrid* s kolekcí dat. V kombinaci s možností zanořování prvků nám tyto technologie dávají velmi silný nástroj pro tvorbu uživatelského rozhraní. V aplikaci toho využívám například při zobrazování obrázků zvířat přímo v DataGridu. Prvek *Image* jako zdroj bere cestu k souboru obrázku z kolekce napojené na *DataGrid*. Stačí tedy jen nastavit vazbu mezi vlastností prvku (závislá vlastnost) a vlastností objektu a další práci již obstarává vazba bez obsluhy. [17]



Obrázek 8: Ukázka principu data bindingu pomocí diagramu

Chování vazby můžeme dále ovlivnit doplňujícími atributy. Můžeme nastavit směr toku dat nebo jaká událost spustí aktualizaci zdroje dat. Směr toku se dá nastavit pomocí atributu *Mode* a příslušné hodnoty:

- *OneTime* – synchronizace proběhne pouze při startu a prvek dále aktualizace zdroje ignoruje
- *OneWay* – synchronizace probíhá pouze ze zdroje do vlastnosti navázaného prvku
- *OneWayToSource* – jedná se o obrácený *OneWay* mód, aktualizuje zdroj podle změny vlastnosti prvku
- *TwoWay* – hodnota vlastnosti je synchronizována oboustranně, jedná se o výchozí nastavení

Vazby, které mají typ *TwoWay* nebo *OneWayToSource*, naslouchají změnám v navázané vlastnosti a synchronizují je zpět do zdroje. Spuštěč (*trigger*) aktualizace nastavujeme pomocí atributu *UpdateSourceTrigger* a jeho různými hodnotami (*LostFocus*, *PropertyChanged*, *Explicit*).[18]

```
1 <!-- Vazba mezi závislou proměnnou SelectedDate a~vlastností
     BirthDate zvířete v~evidenční kartě-->
2 <DatePicker
3     SelectedDate="{Binding Path=BirthDate, Mode=TwoWay,
          UpdateSourceTrigger=PropertyChanged}"
4     SelectedDateFormat="Long"/>
```

Zdrojový kód 1: Ukázka data bindingu v XAMLu

3.5.2 LINQ (Language Integrated Query)

LINQ je zabudovanou komponentou .NET frameworku od verze 3.5. Jazyky rozšiřuje o možnost dotazovat se pomocí speciálních výrazů na nejrůznější data. Výrazy mohou připomínat syntaxi dotazovacího jazyka SQL. Pomocí výrazů je možné extrahovat data z kolekcí, XML dokumentů nebo relačních databází. Právě poslední zmíněná funkce je využívána v aplikaci. Rozšíření LINQ to SQL dává možnost tvořit query (dotazy) nad daty uloženými v relačních databázích Microsoft SQL serveru. Query[14] napsaná v LINQ to SQL je přeložena do klasického dotazu SQL.

Microsoft SQL server ale uchovává data ve formě relací a LINQ pracuje s daty jako s objekty, oba tyto způsoby uložení dat tak musí být na sebe namapované. Pro vzájemnou kompatibilitu dat využívá tedy LINQ to SQL ještě mapovací framework, který pro každou tabulku v databázi vytvoří korespondující třídu. Třída, která se v programu při mapování využívá se nazývá DataContext. Při použití této třídy může LINQ to SQL přeložit LINQ query do korespondující T-SQL[15] query.[19] Ukázka LINQ to SQL dotazu nad databází bude uvedena

[14] dotaz nad daty

[15] Transact-SQL - částečné rozšíření SQL od Microsoftu

v sekci o navigaci pomocí LINQ to SQL. Zde je uvedena část třídy *FurColors* v jazyce C#, která je vytvořená podle tabulky *FurColors* v databázi a je součástí třídy *DataContext*. Třída obsahuje mnoho dalších složitých metod, které se při převodu z relační reprezentace do objektové a zpět používají. Ukázka obsahuje i metodu pro navázání konkrétní barvy srsti na záznam zvířete.

```
1   [global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.
       FurColors")]
2   public partial class FurColors : INotifyPropertyChanging,
       INotifyPropertyChanged
3   {
4     private static PropertyChangingEventArgs emptyChangingEventArgs =
          new PropertyChangingEventArgs(String.Empty);
5     private int _Id;
6     private string _FurColorName;
7     private System.Nullable<System.DateTime> _IsDeleted;
8     private EntitySet<Animals> _Animals;
9
10    private void attach_Animals(Animals entity)
11    {
12      this.SendPropertyChanging();
13      entity.FurColors = this;
14    }
15  }
```

Zdrojový kód 2: Ukázka mapování - atributy třídy FurColors vytvořené podle tabulky v databázi

3.5.2.1 LINQ to SQL Navigation

V předchozí sekci bylo popsáno rozšíření LINQ to SQL. V této sekci bude představena jedna z jeho funkcí - navigace. K navigaci se využívá třídy *DataContext*. Třída definuje mapování tříd a vlastností do tabulek a sloupců databáze pomocí příslušných atributů. Uchovává také vztahy mezi jednotlivými třídami. Pomocí klasické tečkové notace můžeme tedy přistupovat k vztahům definovaných v databázi pomocí cizích klíčů. Pokud máme v databázi definovaný vztah mezi dvěma tabulkami pomocí cizího klíče, DataContext tento vztah také uchovává a můžeme ho tedy použít v LINQ to SQL query a není nutné k tomu používat explicitní spojení (*join*) z klasického SQL. Při správném propojení tabulek v databázi tak téměř nemusíme využívat explicitní joiny, které často bývají zdrojem chyb.[20] V následující ukázce zdrojového kódu je funkce pro získání kolekce adopcí. Technika navigace je použita u jména majitele (*OwnerName*) a jména zvířete (*AnimalName*). Obdobně se dá navigace použít i v metodě *where* pro specifikaci výsledku. Pro zkrácení zdrojového kódu je vynechán try-catch blok.

```
1  public static BindableCollection<AdoptionInfo> ReturnAdoptions()
2  {
3     using (ShelterDatabaseLINQDataContext db = new
          ShelterDatabaseLINQDataContext())
4     {
5        var results = (from adoption in db.Adoptions
6                       where (adoption.IsDeleted.Equals(null))
7                       select new AdoptionInfo
8                          {
9                             ID = adoption.ID,
10                            Date = adoption.Date,
11                            OwnerName = adoption.People.FirstName + " "
                                 + adoption.People.LastName,
12                            AnimalName = adoption.Animals.Name,
13                            Returned = adoption.Returned,
14                            ReturnDate = adoption.ReturnDate,
15                            ReturnReason = adoption.ReturnReason
16                         });
17                return new BindableCollection<AdoptionInfo>(results);
18    }
19 }
```

Zdrojový kód 3: Ukázka navigace pomocí LINQ to SQL

3.6 Microsoft SQL Server Express

SQL Server Express je jedna z verzí Microsoft SQL Serveru a je k dispozici zdarma. I přesto, že se jedná o verzi zdarma, poskytuje SQL Server Express většinu funkcí jako placená plná verze. Omezení, které má, ji ale dělají nevhodnou pro velké projekty. Maximální velikost databáze je 10 GB a je také omezená výkonem. Pro svou práci využívá pouze jeden procesor a 1 GB paměti RAM. K práci s databází se využívá dotazovací jazyk T-SQL. Jedná se o procedurální jazykové rozšíření jazyka SQL od společnosti Microsoft. V aplikaci je využita SQL Server Express LocalDB. Jedná se o odlehčenou verzi se všemi funkcemi Express. Na straně klienta není potřeba žádné konfigurace. Tato verze bude rozebrána v následující podkapitole.

3.6.1 SQL Server Express LocalDB

Tato verze je vlastně jednou z funkcí poskytovanou verzí Express. Při její instalaci zkopíruje jen soubory nezbytné ke spuštění SQL Server Databázového stroje. Při instalaci se vytvoří automatická instance databáze, ta je veřejná a mohou ji využívat všechny aplikace. Automatické instance LocalDB mají speciální název, ten patří do vyhrazeného oboru názvů. Název automatické instance je *MSSQLLocalDB*. LocalDB běží jako proces ve vlastnictví aktuálního uživatele. Díky tomu není potřeba žádná konfigurace a je velmi snadné s ní pracovat. Bohužel to také znamená, že je v celku obtížné instanci LocalDB sdílet s jinými uživateli. Pro

sdílení je potřeba instanci vybrat jedinečný sdílený název (alias).

Pro aplikaci jako je popisovaná evidence pro zvířecí útulek je verze Express LocalDB plně postačující. O evidenci se většinou stará jen jeden zaměstnanec. Omezení výkonu databáze by u velikosti evidence útulku také neměl být problém. Obrázky a soubory jsou ukládány přímo na disku a v databázi je uložena většinou jen cesta k nim, nebo cesta ke složce zvířete. Omezení velikosti databáze na 10 GB by mělo být také na dlouhou dobu dostačující.

3.7 Softwarový architektonický vzor MVVM

Návrhový vzor MVVM (*Model-View-ViewModel*) vychází ze vzoru *Presentation Model* Britského softwarového vývojáře Martina Fowler. Oba přístupy mají společné to, že se snaží co nejvíce oddělit uživatelské rozhraní a logiku celé aplikace a udělat na sobě obě části nezávislé. To umožňuje snadnější údržbu kódu, měnit GUI[16] podle přání zákazníka až v pozdějších fázích vývoje a při vývoji se na každou část aplikace soustředit zvlášť. MVVM vytvořili vývojáři Microsoftu Ken Cooper a Ted Peters v roce 2005. Do WPF jej začlenil jeho architekt John Gossman, který MVVM v roce 2005 oficiálně představil na svém blogu [21]. Vývojáři celý vzor navrhli tak, aby mohl plně využívat výhod Data Bindingu ve WPF. Grafické rozhraní aplikace se tedy dá vytvořit čistě jen pomocí značkovacího jazyka (XAML) a od ostatního kódu je úplně odděleno. Zkratka MVVM značí jednotlivé komponenty vzoru, tedy *Model*, *View*, a *ViewModel*.[22]

3.7.1 Komponenty vzoru MVVM

Návrhový vzor se skládá z několika vrstev. V této podkapitole bude stručně vysvětlena funkce každé vrstvy a popsány její vlastnosti. K popisu budou použity konkrétní případy použitá v aplikaci pro evidenci zvířat.

- Model

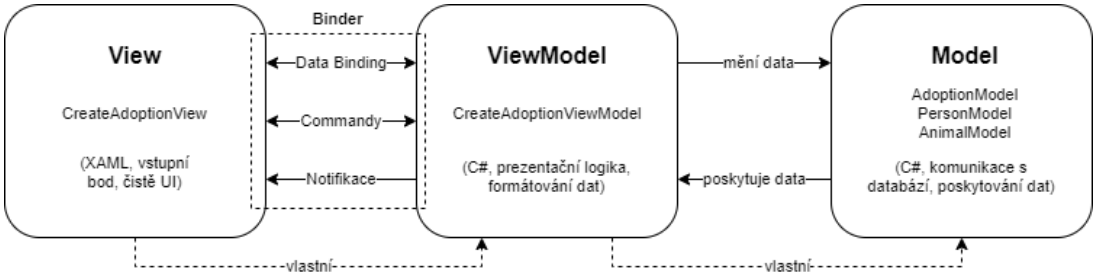
  Model je vrstva, která zajišťuje přístup k datům a komunikuje s databází. Popisuje data se kterými aplikace pracuje a struktura třídy modelu odpovídá tabulkám v databázi. Model vůbec neví o existenci ovládacích prvků v uživatelském rozhraní.

- ViewModel

  ViewModel je nejdůležitější vrstva návrhu. Spojuje Model a View a poskytuje všechna data pro View. ViewModel má dvě základní součásti. První jsou kolekce, které hlásí změny jejích prvků a druhá je rozhraní, které popisuje změnu vlastností ViewModelu. Konkrétní podoba těchto základních kamenů bude popsána v další podkapitole.

- View

[16] grafické uživatelské rozhraní



Obrázek 9: Znázornění použití MVVM na příkladu z aplikace

Tato vrstva komunikuje s uživatelem. Poskytuje uživateli grafickou reprezentaci Modelu a přijímá od něj akce a vstupy. Stejně jako Model neví o existenci View ani View neví, že jsou data poskytována Modelu. V aplikaci pro evidenci každá obrazovka s ovládacími prvky odpovídá jednomu View.

- Binder

  Automatizuje komunikaci mezi View a ViewModelem a synchronizuje jejich data. Jedná se o příkazovou vazbu a ve WPF ji obstarává *Data Binding*.

Při implementaci MVVM v aplikaci pro útulek byl použit framework Caliburn Micro. Framework poskytuje příjemnější práci při provázání View a ViewModelu a bude stručně popsán na příkladech v další podkapitole. Pro jednoduchá uživatelská rozhraní je implementace jednotlivých komponent MVVM oproti jiným návrhovým vzorům pracnější. A u velkých aplikací může při udržování aktuálního stavu aplikace ve ViewModelu dojít ke značné spotřebě paměti. Sám Gossman toto kritizuje.

> "Disadvantages? For simple UI, M-V-VM can be overkill. In bigger cases, it can be hard to design the ViewModel up front in order to get the right amount of generality. Data-binding for all its wonders is declarative and harder to debug than nice imperative stuff where you just set breakpoints."- John Gossman [23]

3.7.2 Caliburn Micro Framework

Framework Caliburn Micro vytvořili Nigel Sampson, Rob Eisenberg a Thomas Ibel. Jedná se o malý ale šikovný framework pro *View* a *ViewModel* vrstvy používané v MVVM. V této sekci budou na příkladech rozebrány všechny jeho funkce použité v aplikaci.

Při popisu komponenty ViewModel jsme se zmínili o jeho dvou základních částech, kolekcích a rozhraním popisujícím změnu vlastností. Caliburn Micro má pro tyto dvě věci vlastní implementace. Jako kolekce se využívají *BindableCollection<Type>* a pro oznamování změn *NotifyOfPropertyChange()*. *BindableCollection* dědí z třídy *ObservableCollection*, která je součástí jazyka C#,

a poskytuje oznámení o tom, když se položky přidají, odeberou nebo když se obnoví celý seznam. *NotifyOfPropertyChange* poskytuje oznámení o změně vlastností, aby mohl být automaticky aktualizován View. Tato metoda je zabudována v kódu a je volána pokaždé, když je vlastnost aktualizována. Metoda je implementována pomocí rozhraní *INotifyPropertyChanged*, které je opět součástí C#. Tyto dvě součásti se starají o hladký průběh synchronizace a oznamování změn při běhu aplikace.[24]

```
1  private BindableCollection<AnimalInfo> _animals;
2  public BindableCollection<AnimalInfo> Animals
3  {
4      get
5      {
6          return _animals;
7      }
8      set
9      {
10         _animals = value;
11         NotifyOfPropertyChange(() => Animals);
12     }
13 }
```

Zdrojový kód 4: Ukázka použití kolekce BindableCollection a metody NotifyOfPropertyChange při její změně ve ViewModelu

Framework také umožňuje propojit vlastnosti View a ViewModelu na základě jmenné konvence. Pojmenování View a ViewModelu je pro fungování frameworku velmi důležité. Nejběžnější je pojemnovat View jako *<název>View* a ViewModel jako *<název>ViewModel*. Tato jednoduchá konvence umožňuje Caliburn Micro nalézt ke každému ViewModelu odpovídající View. V podstatě bere celý řetěz jména a odstraňuje z něj podřetěz „Model“. V aplikaci evidence bude tedy k *ShelterEvidency.ViewModels.EvidencyCardViewModel* hledat *ShelterEvidency.Views.EvidencyCardView*. Toto správné pojmenování nám umožní podobně propojit i ovládací prvky například s metodami nebo proměnnými. Pokud tedy pojmenujeme tlačítko ve View *OpenEvidencyCard* můžeme při kliknutí zavolat stejně pojmenovanou metodu v odpovídajícím ViewModelu. [24]

```
1 <Button x:Name="UpdatePerson">
2     <StackPanel Orientation="Horizontal">
3         <materialDesign:PackIcon Kind="ContentSaveOutline"/>
4         <TextBlock VerticalAlignment="Center">Uložit</TextBlock>
5     </StackPanel>
6 </Button>
```

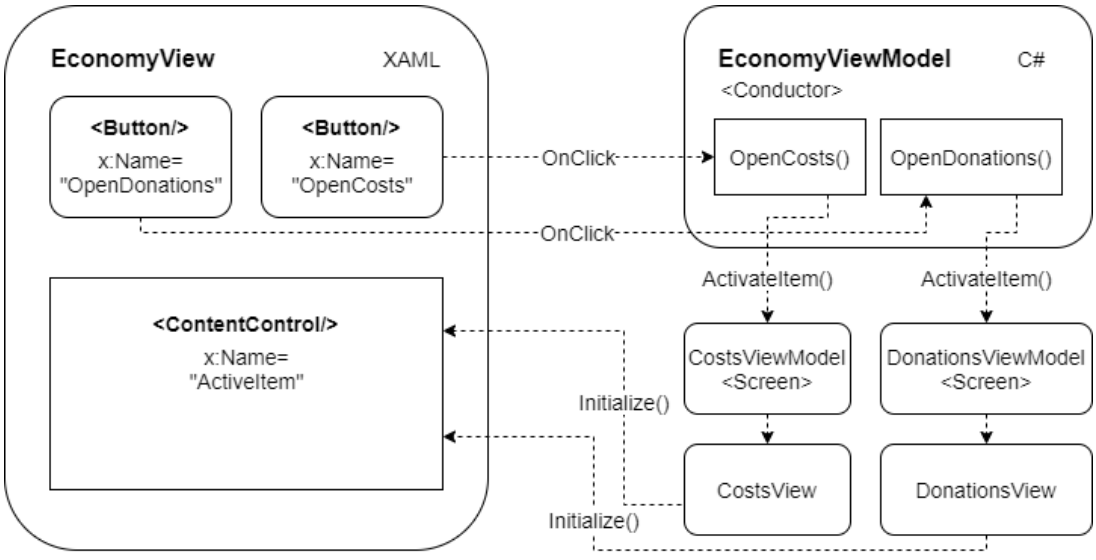
Zdrojový kód 5: Tlačítko pro aktualizaci údajů o osobě ve View (XAML)

```
1  public void UpdatePerson()
2  {
3      if (Person.ValidValues())
4      {
5          Person.UpdatePerson();
6          MessageBox.Show("Aktualizováno.");
7          Prnt.UpdatePeople();
8      }
9      else
10         MessageBox.Show("Vyplňte prosím jméno a~příjmení.");
11 }
```

Zdrojový kód 6: Odpovídající metoda, která akci provede ve ViewModelu (C#)

3.7.2.1 Caliburn Micro Screen a Conductor

Caliburn Micro nabízí několik základních tříd, které je možné pomocí dědění využít ve ViewModelu. Můžeme tak ViewModelu přidělit konkrétnější roli při tvorbě uživatelského rozhraní. Základní dva typy jsou *Screen* a *Conductor*. Screen je jednoduchá obrazovka s krátkým životním cyklem. Slouží k zobrazení požadovaného View, provedení akce a následné deaktivaci. Constructor slouží jako správce pro ViewModely dědící ze třídy Screen. Stará se o jejich správnou aktivaci a uzavření. Conductor daný Screen (jeho View) zobrazí v XAML prvku ContentControl svého View, pomocí metody *ActivateItem(new <název>ViewModel())*.[24] Znázornění celého procesu je vyobrazeno na následujícím diagramu.



Obrázek 10: Zobrazení obsahu v prvku ContentControl ve View aplikace

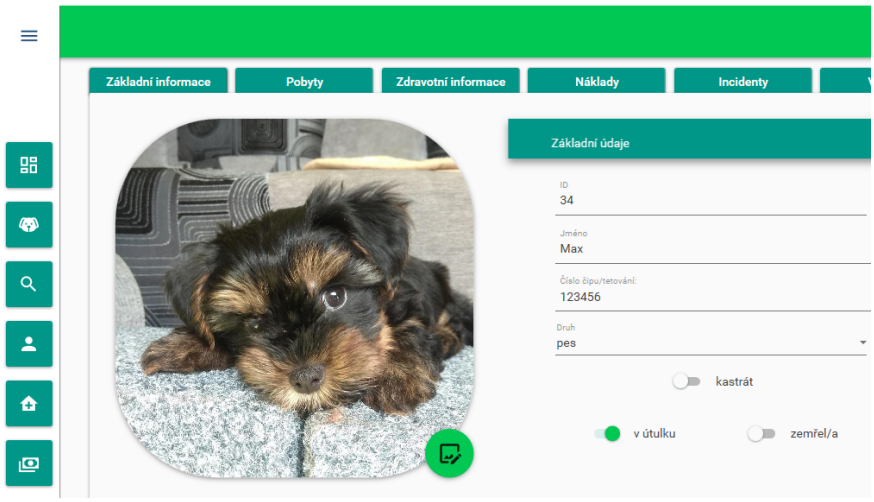
3.8 Material Design

Material Design, vyvíjený pod názvem Quantum Paper, je designový jazyk představený společností Google v roce 2014. Celý koncept je založený na co největší podobnosti uživatelského rozhraní aplikace s papírem a inkoustem v reálném životě. Jednotlivé karty a ostatní elementy pod sebe mohou vrhat stín a tím se dá docílit určitého vrstvení jednotlivých elementů a jejich optického přiblížení uživateli. Mezi hlavní přednosti patří jednoduchost celého návrhu. Uživatelské rozhraní navržené pomocí Material Designu využívá většinou jen dvě výraznější barvy a to *Primary* a *Accent color* . Primary color (primární) je využívána k podbarvení ploch a zvýraznění hojně využívaných elementů a Accent color (zvýrazňující) například u tlačítek vyvolávajících nějakou akci. Accent color se většinou volí výraznější a má vyvolat v uživateli pocit důležitosti. Aplikace navržené v Material Designu používají bezpatkový font Roboto. Material Design je s pomocí frameworků dostupný na všech platformách. Velmi hojně ho využívá právě společnost Google na svých webech a ve webových aplikacích. [25, 26]

> "Unlike real paper, our digital material can expand and reform intelligently. Material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch."- Matías Duarte (Google's Vice President of Design)[27]

3.8.1 Material Design In XAML Toolkit

Jedná se o grafickou knihovnu obsahující styly a variace pro standardní ovládací prvky WPF aplikací. Knihovna je dostupná z NuGet.org pod Open-source licencí a je tedy zdarma. Ukázku vzhledu některých prvků je vidět na následujícím obrázku obrázku (11). V levé části obrázku můžeme vidět Primary color použitou v menu, Accent color tlačítka u výběru obrázku a stín, který vrhá karta s obrázkem psa.



Obrázek 11: Ukázka použití Material Designu v aplikaci - evidenční karta zvířete

4 Uživatelská část

V této kapitole bude stručně rozebrán popis ovládání aplikace a pokyny pro práci se vstupními údaji. Kapitola může sloužit jako základní uživatelská příručka. Popis bude doplněn o vhodné obrázky z aplikace. V úvodu budou popsány obecné pokyny pro práci. Kapitola bude dále rozdělena na menší podkapitoly, které budou pokrývat celý pobyt zvířete v útulku a evidenci, která je s ním spojena.

4.1 Obecné ovládání aplikace

V této podkapitole bude stručně popsáno ovládání, které není přímo spojeno s evidencí konkrétního zvířete ani osoby.

4.1.1 Úvodní obrazovka

Úvodní obrazovka je první stránka, která se zobrazí při spuštění aplikace. Hlavní ovládání aplikace probíhá přes hlavní menu, které se nachází v levé části obrazovky. V pravém horním rohu se nachází tlačítko pro řádné ukončení aplikace. V levé části stránky se nachází kalendář a sekce pro přidávání poznámek.

Poznámku lze přidat na daný den vybráním data v kalendáři. Při vybrání data se také zobrazí již existující poznámky vytvořené dříve na dané datum.

Zbytek obrazovky je vyplněn kartami s užitečnými informacemi jako jsou poslední přijatá zvířata, dnešní procházky, základní údaje o útulku a další.

4.1.2 Nastavení

Stránka *Nastavení* je dostupná pomocí posledního tlačítka v navigačním menu. Slouží pro přidávání nových druhů zvířat, plemen, barev srsti a dalších údajů, které identifikují vlastnosti a vzhled zvířete. Na stránce se dají také editovat údaje o útulku a logo, které je zobrazeno na úvodní obrazovce. Poslední funkcí, která je zatím v nastavení dostupná je obnovení smazaných údajů ve zvoleném časovém rozmezí.

4.1.3 Statistiky

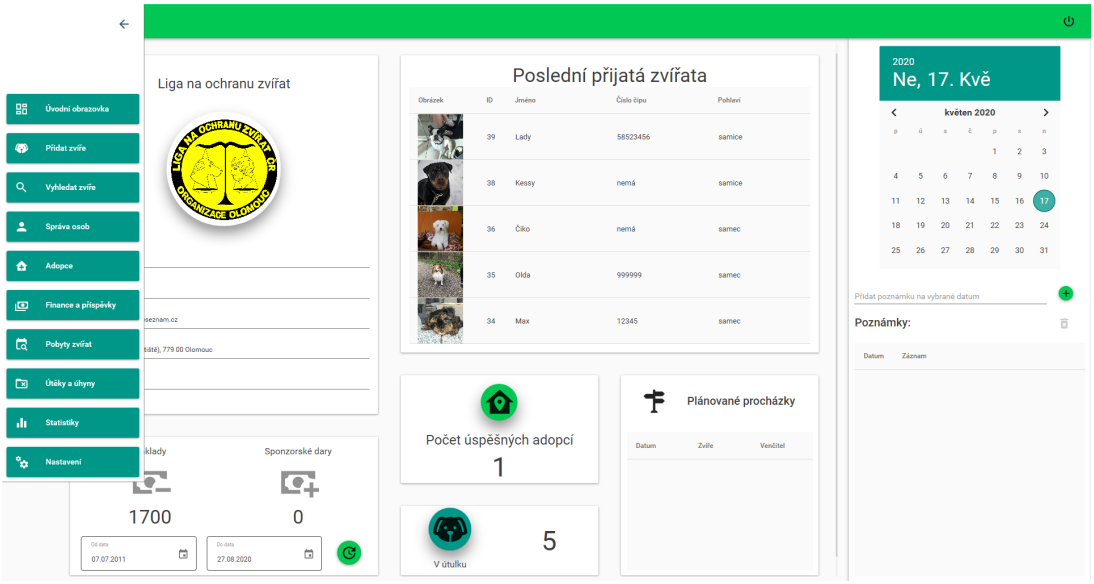
Statistiky slouží pro rychlý náhled informací spojených s určitým zvířetem. Pro zobrazení statistik je nutné zadat ID zvířete. Statistiky zvířete obsahují základní informace z evidenční karty na jednom místě.

Na stránce jsou také dostupné obecné statistiky pro celý útulek v určitém časovém období. Tato sekce slouží pro přehledné sledování financí a aktivit v útulku.

4.1.4 Finance a příspěvky

Stránka *Finance a příspěvky* je opět dostupná z hlavního menu. Obsahuje dvě záložky. První záložka slouží pro sledování nákladů útulku a zakládání nových obec-

ných nákladů. Druhá záložka slouží pro evidenci sponzorský darů. Obě stránky jsou doplněny o možnost filtrace dle data.



Obrázek 12: Úvodní obrazovka aplikace s otevřeným navigačním menu

4.2 Evidence osob

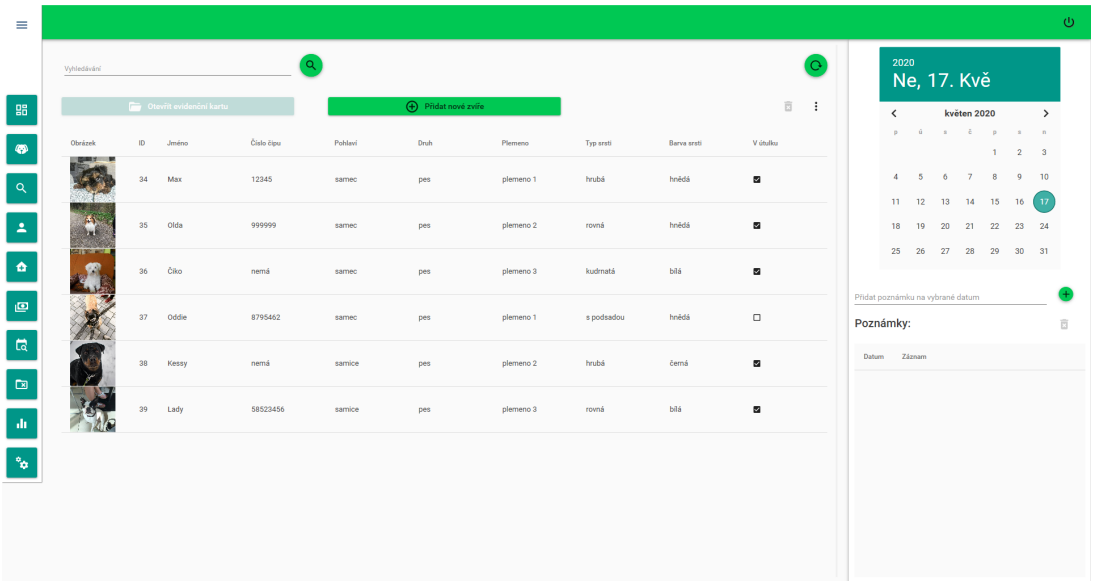
Evidence osob probíhá na obrazovce *Správa osob* dostupné z navigačního menu. V dolní části obrazovky je možné přidat novou osobu. Horní část slouží k vyhledávání a zobrazení již existujících profilů. Po označení osoby v seznamu je možné pomocí tlačítka Detail osoby upravovat její údaje a měnit role. V detailu osoby jsou také zobrazeny navázané adopce, sponzorské dary a procházky.

Role osoby mají důležitou roli při vytváření dalších záznamů. Osoba s rolí majitel může být navázána na adopci, osoba s rolí veterináře může být evidována u zdravotního záznamu zvířete, role sponzora umožňuje osobu navázat na sponzorský dar a s rolí venčitel zase na záznam o procházce.

4.3 Příjem zvířete do útulku

Při příjmu nového zvířete do útulku je nejdříve nutné mu založit evidenční kartu. Stránka *Přidat nové zvíře* umožňuje uživateli vyplnit základní údaje o zvířeti a rovnou mu založit nový pobyt v útulku. Je možné zde vyplnit základní charakteristiku jako je druh nebo plemeno, dále váhu, krmnou dávku, datum a místo nálezu a datum příjmu do útulku. Další informace se vyplňují až v založené kartě zvířete, kde mohou být později změny i již zadané informace.

Pokud zvíře není v útulku poprvé, je nutné jeho evidenční kartu vyhledat na stránce *Vyhledat zvíře* a poté v ní založit nový pobyt. Na stejné stránce je možné nalézt i nově založenou evidenční kartu.



Obrázek 13: Obrazovka s evidenčními kartami zvířat

4.4 Evidence při pobytu zvířete v útulku

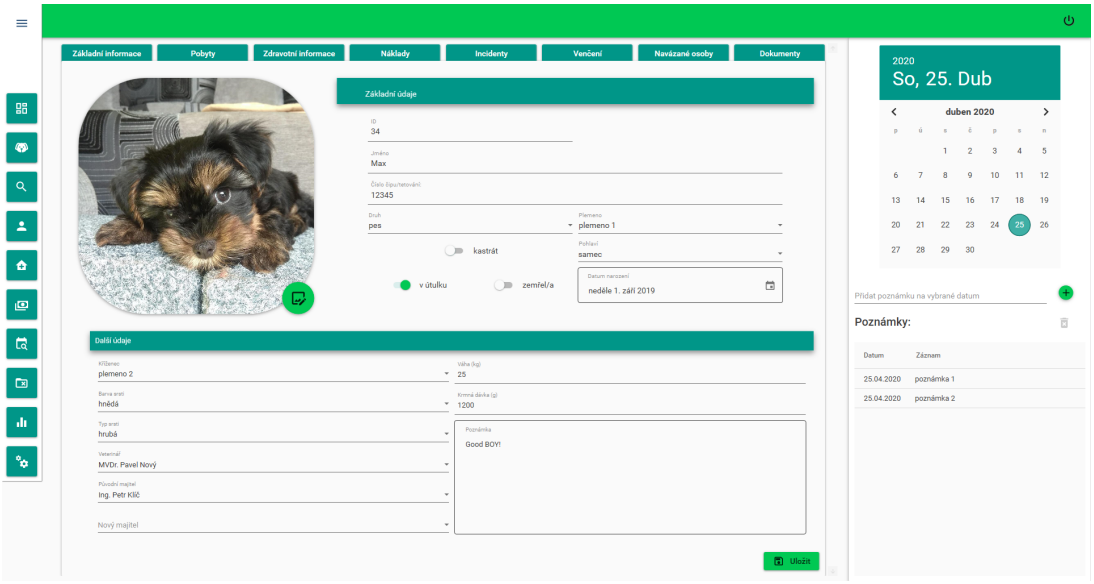
Při pobytu zvířete v útulku je možné s ním propojit nejrůznější záznamy. Zakládání těchto záznamů a přístup k nim probíhá v evidenční kartě zvířete. Dostat se do evidenční karty je možné výběrem zvířete ve vyhledávací obrazovce a kliknutím na tlačítko *Otevřít evidenční kartu*. Navigace v evidenční kartě probíhá přes záložky v její horní části. Je možné zakládat nové pobyty, zdravotní záznamy, nové náklady, incidenty a venčení. Všechny údaje je možné v daných záložkách také upravovat.

Obecné informace je možné upravovat v záložce *Základní informace*. Záložka *Navázané osoby* slouží pro rychlý náhled na detail osob propojených se zvířetem. Poslední záložka *Dokumenty* slouží pro nahrávání, mazání, otevírání a tisk dokumentů ze složky zvířete. Je zde také možné vygenerovat evidenční kartu ve formátu PDF.

Pobyty všech zvířat v určitém období je možné sledovat na stránce *Pobyty* zvířat dostupné z menu. Jsou zde pohromadě zobrazeny základní informace o pobytech doplněné o počet dnů strávených v útulku.

4.5 Ukončení pobytu zvířete v útulku

Ukončení pobytu zvířete probíhá v jeho evidenční kartě v záložce *Pobyty*. Při ukončení pobytu musí být v záznamu doplněn typ ukončení. Pobyt může být ukončen třemi způsoby a to adopcí, útěkem nebo úhynem zvířete. Při ukončení s typem útěk nebo úhyn je možnost přímo vyplnit podrobnosti o dané situaci. To vede k založení nového záznamu o útěku či úhynu. Hromadně lze tyto záznamy sledovat na obrazovce *Útěky a úhyny* dostupné z menu. Při ukončení pobytu adopcí je nutné založit novou adopci.



Obrázek 14: Evidenční karta zvířete se záložkami v horní části

Založení adopce je možné na obrazovce *Adopce* dostupné z menu pomocí tlačítka *Vytvořit novou adopci*. Při založení adopce je vybráno zvíře, které se aktuálně nachází v útulku a osoba s rolí majitel a dojde k jejich provázání v záznamu o adopci. Po vyplnění data adopce a potvrzení formuláře je generována adopční smlouva, která je uložena do složky zvířete a je později dostupná z evidenční karty v záložce *Dokumenty*.

Pokud je adopce neúspěšná a zvíře se vrací do útulku je možné adopci vyhledat na stránce Adopce a pomocí tlačítka *Upravit adopci* v ní vyplnit detaily o vrácení zvířete. Tlačítko také slouží pro náhled založené adopce.

Závěr

Úvod práce je zaměřen na problematiku opuštěných zvířat a zákony upravující činnost útulků pro tato zvířata. Shrnuje motivaci pro tvorbu evidenční aplikace a obecné informace o tom, co by takový systém měl obsahovat. Dále je zde provedeno srovnání komerčních aplikací pro zvířecí útulky a jejich hlavní výhody a nevýhody. U každé z nich jsou zmíněny zajímavé funkce. Funkce, které byly ve srovnání hodnoceny jsou poté zhodnoceny i pro vytvořenou aplikaci. V závěru první části práce jsou analyzovány uživatelské požadavky a je vytvořen diagram případů užití. Poslední částí je jednoduchý entitně-relační diagram, který slouží jako kostra pro návrh databáze evidenční aplikace.

Programátorská část se věnuje vlastní implementaci aplikace a ukázkám technologií použitých při vývoji. Nejdříve je provedena analýza třech ukázkových případů užití aplikace. Analýza je doplněna o zjednodušené UML diagramy tříd, které znázorňují spolupráci objektů v aplikaci, jejich vlastnosti a metody. Další část je zaměřena na tvorbu databáze. Jednotlivé tabulky a vztahy mezi nimi jsou popsány a popis je doplněn o diagram modelu databáze. Diagram modelu databáze zachycuje celkový pohled na jednotlivé entity a vztahy, které mezi nimi mohou postupem času vznikat. Závěr programátorské části je věnován představení použitých technologií spolu s vhodnými ukázkami jejich použití v aplikaci. Představen byl programovací jazyk C# a .NET framework jako programovací technologie, Microsoft SQL Server a LINQ použité pro tvorbu a manipulaci s databází, softwarový návrhový vzor MVVM, podle kterého byla vytvořena struktura aplikace a jako poslední Material Design pro tvorbu uživatelského rozhraní. Při ukázkách z aplikace byly zvoleny vhodné části zdrojového kódu a nebo bylo jejich použití popsáno pomocí diagramů.

Poslední částí práce je uživatelská část. V této kapitole je uveden stručný popis ovládání aplikace a pokyny pro práci se vstupními údaji. Grafické uživatelské rozhraní není nijak složité a proto je popis stručný a je doplněn o ukázky obrazovek z aplikace. Kapitola je rozdělena na obecné funkce a na evidenční část týkající se zvířat. Popis ovládání v evidenční části je strukturován tak, aby pokryl celý cyklus pobytu zvířete v útulku od jeho příjmu až k výdeji.

Do budoucna mám v plánu na aplikaci dále pracovat, rozšiřovat její funkcionalitu a opravovat případné chyby. Chtěl bych rozšířit hlavně funkce, které jsem, z časových důvodů, neměl možnost dostatečně rozpracovat. Jmenovitě například propracovanější vyhledávání a více náhledových fotek u zvířete přímo v aplikaci. Možnosti pro rozšíření funkcionality jsou opravdu široké. Otázkou ale je, zda by pokročilé a složitější funkce menší útulek využil, a to hlavně z důvodu časového a pracovního vytížení zaměstnanců. Menší útulky také často inzerují zvířata pouze na svých webových nebo sociálních stránkách. Velmi zajímavé by bylo podobný projekt zpracovat například jako webovou službu dostupnou jak pro útulky, tak pro lidi, kteří mají zájem si zvíře z útulku adoptovat. To by zvýšilo šanci na udání zvířete i pro menší útulky a lidé by mohli útulek navštívit s jasnou představou o zvířatech právě dostupných k adopci. Vznik podobného

systému by mohl více zpopularizovat adopce a útulku by ušetřil čas s hledáním nových domovů pro zvířata.

Conclusions

The introduction of this paper describes the problematics of abandoned animals and statuary regulations for activity of shelters for said animals. It offers the reader motivation for the creation of records management application and basic information about what should be included in it. In the next part comparison of available applications for animal shelters is made with their main advantages and disadvantages and interesting functions they offer. Functions that have been evaluated are then evaluated for the created application as well. In the final part of the introduction, requirements analysis has been made with and a use case diagram is presented. It is followed by entity-relationship diagram which serves as a framework for the layout of the database.

In the practical part of this paper the reader will learn about the implementation of this application and examples of technologies used in its development. In the first part, three sample cases of the application are analyzed. The analysis includes simplified UML class diagrams, which show attributes and methods of objects and relationships between them. The next part is aimed at the creation of the database, it describes all tables and relations between them, diagram of database model which shows the overview of all entities and relations that may be established throughout the time, is included. In the final part of the practical part, used technologies are introduced with suitable cases of their use in the application. Programming language C# and a programming technology .NET framework are introduced along with, Microsoft SQL Server and LINQ which was used for creation and handling of the database, software design pattern MVVM, which served as a baseline for the structure of the application and Material design which was used for UI development. In examples from the application, suitable parts of codes have been used or their use was described with diagrams.

In the user documentation application controls are described and guidelines for the work with input data are set. GUI is not complicated, therefore its description is brief and screenshots from the application are added to it. User documentation is split into overview of general functions and database part regarding the animals. Description of controls in the database part is structured in a way, so it covers the entire cycle of the animal's life in the shelter from the beginning to the end.

In the future I plan to work further on the application by expanding its functionality and fix bugs that might appear. I would mostly like to expand functions which I could not work as much as I would like because of the time constraints. For example more sophisticated searching and more preview photos in animal card. The possibilities for the functionality expenditure are very high, but it remains a question if a smaller animal shelter would make use of more advanced and complicated functions as their employees have a large workload. As smaller animal shelters often advertise their animals on the internet, it might be an interesting option to recreate this application as a web application which would be available for both animal shelters and people that would like to adopt

an animal. It could lead to higher chances for smaller animal shelters to find new life for their animals and people could visit the shelter with clear idea of which animals are currently available for adoption. Creation of a similar system could make the adoption more popular and would save shelters' time with searching for animal's new home.

Literatura

[1] NĚMEC, Jan. *Řešení problému toulavých psů a regulace zájmového chovu psů ve vybraných státech Evropské unie* [online]. 2015 [cit. 2020-3-28]. 14 s. Dostupný z: ⟨https://www.psp.cz/sqw/text/orig2.sqw?idd=107157⟩.

[2] SPRÁVA, Státní veterinární. *Registrované útulky pro zvířata* [online]. 2020 [cit. 2020-3-28]. Dostupný z: ⟨%5Curl%7Bhttps://www.svscr.cz/registrovane-subjekty-svs/registrovane-utulky-pro-zvirata/%7D⟩.

[3] ÚTULEK, Olomoucký. *2019 v číslech* [online]. 2020 [cit. 2020-3-28]. Dostupný z: ⟨http://www.olomouckyutulek.cz/2020/01/2019-v-cislech/⟩.

[4] *Zákon na ochranu zvířat proti týrání*. [online]. [cit. 2020-3-28]. Dostupný z: ⟨https://www.zakonyprolidi.cz/print/cs/1992-246/zneni-20191209.htm?sil=1⟩.

[5] *Zákon o veterinární péči*. [online]. [cit. 2020-3-28]. Dostupný z: ⟨https://www.zakonyprolidi.cz/print/cs/1999-166/zneni-20200115.htm?sil=19⟩.

[6] *Způsob oznámení o týrání zvířat*. [online]. [cit. 2020-3-28]. Dostupný z: ⟨https://www.svscr.cz/zdravi-zvirat/zpusob-oznameni-o-tyrani-zvirat/⟩.

[7] *PODMÍNKY ZŘÍZENÍ ÚTULKU*. [online]. [cit. 2020-3-28]. Dostupný z: ⟨https://www.jaksestaviutulek.cz/otazky-a-odpovedi/podminky-zrizeni-utulku/⟩.

[8] MORAN, Kate. *The Aesthetic-Usability Effect* [online]. 2017 [cit. 2020-3-28]. Dostupný z: ⟨https://www.nngroup.com/articles/aesthetic-usability-effect/⟩.

[9] WIKIPEDIA. *Software as a service* [online]. 2020 [cit. 2020-3-28]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Software_as_a_service⟩.

[10] WIKIPEDIA. *Unified Modeling Language* [online]. 2018 [cit. 2020-3-28]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Unified_Modeling_Language⟩.

[11] MICROSOFT. *Přidání nebo změna primárního klíče tabulky v Accessu* [online]. [cit. 2020-4-5]. Dostupný z: ⟨https://support.office.com/cs-cz/access⟩.

[12] ZOAR, Brent. *What Are Soft Deletes, and How Are They Implemented?* [online]. 2020 [cit. 2020-4-5]. Dostupný z: ⟨https://www.brentozar.com/archive/2020/02/what-are-soft-deletes-and-how-are-they-implemented/⟩.

[13] BĚHÁLEK, Marek. *Programovací jazyk C#* [online]. 2017 [cit. 2020-4-8]. Dostupný z: ⟨http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/index.html⟩.

[14] MICROSOFT. *Dokumentace k jazyku C#* [online]. [cit. 2020-4-8]. Dostupný z: ⟨https://docs.microsoft.com/cs-cz/dotnet/csharp/⟩.

[15] WIKIPEDIA. *Microsoft Visual Studio* [online]. 2020 [cit. 2020-4-8]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Microsoft_Visual_Studio⟩.

[16] MICROSOFT. *Common Language Runtime (CLR) overview* [online]. 2020 [cit. 2020-4-8]. Dostupný z: ⟨https://docs.microsoft.com/cs-cz/dotnet/standard/clr⟩.

[17] CHARLES, Petzold. *Mistrovství ve Windows Presentation Foundation: [aplikace = kód + markup]*. Second. Brno: Computer Press, 2008. 928 s. ISBN 978-80-251-2141-2.

[18] MICROSOFT. *Data binding overview in WPF* [online]. 2019 [cit. 2020-4-9]. Dostupný z: ⟨https://docs.microsoft.com/en-us/dotnet/desktop-wpf/data/data-binding-overview⟩.

[19] AGARWAL, Vidya Vrat. *Databáze v C# 2008 průvodce programátora*. First. Brno: Computer Press, 2008. 428 s. ISBN 978-80-251-2309-6.

[20] ABEL, Anders. *Don't use Linq's Join. Navigate!* [online]. 2012 [cit. 2020-4-10]. Dostupný z: ⟨https://coding.abel.nu/2012/06/dont-use-linqs-join-navigate/⟩.

[21] GOSSMAN, John. *Patterns - WPF Apps With The Model-View-ViewModel Design Pattern* [online]. 2005 [cit. 2020-4-11]. Dostupný z: ⟨https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps/⟩.

[22] WIKIPEDIE. *Model–view–viewmodel* [online]. 2020 [cit. 2020-4-8]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel⟩.

[23] GOSSMAN, John. *Advantages and disadvantages of M-V-VM* [online]. 2006 [cit. 2020-4-11]. Dostupný z: ⟨https://docs.microsoft.com/en-gb/archive/blogs/johngossman/advantages-and-disadvantages-of-m-v-vm⟩.

[24] *Documentation.* [online]. [cit. 2020-4-11]. Dostupný z: ⟨https://caliburnmicro.com/documentation/⟩.

[25] DESIGN, Google. *Making Material Design* [online]. 2015 [cit. 2020-4-10]. Dostupný z: ⟨https://www.youtube.com/watch?v=rrT6v5sOwJg⟩.

[26] WIKIPEDIE. *Material Design* [online]. 2020 [cit. 2020-3-5]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Material_Design⟩.

[27] SANCTIS, Valerio De. *ASP.NET Core 3 and Angular 9*. Third. 2020. 732 s. ISBN 978-1-78961-216-5.

[28] KANISOVÁ HANA, MÜLLER Miroslav. *UML srozumitelně*. Second. Brno: Computer Press, 2006. 176 s. ISBN 80-251-1083-4.

[29] WIKIPEDIE. *.NET* [online]. 2020 [cit. 2020-4-8]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/.NET⟩.

[30] MICROSOFT. *Dokumentace pro .NET* [online]. [cit. 2020-4-8]. Dostupný z: ⟨https://docs.microsoft.com/cs-cz/dotnet/⟩.

[31] DAJBYCH, Václav. *mvvm: model-view-viewmodel* [online]. 2009 [cit. 2020-4-8]. Dostupný z: ⟨https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel⟩.

[32] MICROSOFT. *The history of C#, What's new in C# 8.0* [online]. 2017 [cit. 2020-4-9]. Dostupný z: ⟨https://docs.microsoft.com/cs-cz/dotnet/csharp/whats-new/csharp-version-history⟩.

[33] SMITH, Josh. *Patterns - WPF Apps With The Model-View-ViewModel Design Pattern* [online]. 2009 [cit. 2020-4-11]. Dostupný z: ⟨https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern⟩.