VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INFORMAČNÍ SYSTÉM KLUBU CHOVATELŮ PSŮ

INFORMATION SYSTEM OF A DOG CLUB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **IVO ČAPOUN**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20175/2016/xcapou00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Čapoun Ivo**
Obor: Informační technologie
Téma: **Informační systém klubu chovatelů psů**
Information System of a Dog Breeding Club
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s principy tvorby webových aplikací, dále se současnými frameworky pro tvorbu informačních systémů.
2. Analyzujte požadavky pro informační systém pro klub chovatelů psů, který bude řešit členské poplatky, rodokmeny psů, kontroly vrhů, správa závodů, výstav a jiných, statistiky různých vlastností psů a chovatelů, správa krycích listů. Konzultujte s reálným klubem chovatelů nějakého plemene psů.
3. Navrhněte informační systém splňující získané požadavky. Využijte k tomu vhodné modelovací techniky.
4. Navržený informační systém implementujte, naplňte testovacími daty a ověřte jeho funkčnost.
5. Zhodnoťte dosažené výsledky a další možné pokračování v tomto projektu.

Literatura:

- Welling, L., Thomsonová, L.: PHP a MySQL: rozvoj webových aplikací. Vyd. 1. Praha: SoftPress, 2003, 910 s. ISBN 80-86497-60-7.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá analýzou a specifikací požadavků, návrhem a implementací informačního systému pro klub chovatelů psů. Systém umožňuje uživatelům spravovat informace o psech, chovatelských stanicích, vrzích a osobách. Součástí je i správa klubových akcí. Nabízí možnost evidovat v systému přijaté platby, na jejichž základě se automaticky provádí další akce. V nabídce je i žádost o krycí list, který poté čeká na schválení. Systém nabízí správu a plánování kontrol vrhů. Výsledek prováděné kontroly vrhu je možné zadat do systému. U psů systém vykresluje jejich rodokmeny po tři generace dozadu. Některé informace jsou statisticky vyhodnoceny a výsledky jsou zobrazeny v grafech.

Abstract

This thesis deals with requirements analysis and specification, design and implementation of information system for dog breeding club. The system allows users to manage informations about dogs, kennels, litters and persons. It includes the club event manager. The ability to edit received payments that automatically trigger additional actions. Allows you to ask a cover sheet that is pending approval after it is created. The system offers management and planning of litter inspections. The result of the litter checks can be entered into the system. For dogs, the system renders their pedigrees for three generations backwards. Some information is statistically evaluated and results are displayed in charts.

Klíčová slova

klub chovatelů psů, Nette, Bootstrap, PHP 7, jQuery

Keywords

dog breeding club, Nette, Bootsrap, PHP 7, jQuery

Citace

ČAPOUN, Ivo. Informační systém klubu chovatelů psů. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém klubu chovatelů psů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D.
Další informace mi poskytla hlavní poradkyně klubu bílého ovčáka paní Věra Pecková.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ivo Čapoun
17. května 2017

Poděkování

Rád bych poděkoval Ing. Vladimíru Bartíkovi, Ph.D. za vedení mé bakalářské práce, čas strávený na konzultacích a kontrolou mé práce. Druhé poděkování patří Klubu bílého ovčáka, především hlavní poradkyni chovu paní Věře Peckové, za důležité informace o klubu, bez kterých by bylo obtížné tento systém vytvořit.

Obsah

1 Úvod

Pes je od pradávna považován za nejlepšího přítele člověka. Jeho úkolem bylo doprovázet svého pána a starat se o jeho bezpečí. Byl nápomocen také při lovu zvěře a pasení dobytka. Pes je údajně největší domestikovanou šelmou. Obecně se domníváme, že jeho předky jsou vlci. Postupem času s pomocí člověka vzniklo mnoho různých druhů, ovšem jen 355 plemen je oficiálně uznáno v plemenné knize celosvětové organizace FCI. V České republice se o plemennou knihu stará organizace s názvem Českomoravská kynologická unie[1] (ČMKU), která byla založena 5. prosince 1992. Celosvětové požadavky byly splněny v roce 1997 a Česká republika se tak stala řádným členem FCI. V současné době (duben 2017) se pod záštitou ČMKU chová 276 plemen uznaných FCI a 18 plemen neuznaných FCI. Česká republika patří mezi významné a kvalitní chovatele a psi se od nás exportují do celého světa.

První němečtí ovčáci, kteří se objevili na výstavách, byli bílí. Tato jejich vlastnost byla v roce 1968 označena za nepřípustnou a vyřadila je z čistokrevného chovu. Díky snaze jejich majitelů o to aby zůstali tito psi i nadále čistokrevní a byli zapsáni v plemenné knize, došlo 26. listopadu 2002 k provizornímu uznání a 4. července roku 2011 k definitivnímu zapsání plemene Bílý švýcarský ovčák (white swiss shepherd) do plemenné knihy FCI. Během dlouhé doby mezi vyřazením bíle zbarvených německých ovčáku a jejich uznáním jako švýcarských ovčáků, proběhlo hodně vývojových změn. Především povaha dnešních švýcarských ovčáků se radikálně změnila. Staly se z nich zvířata pro rodiny s dětmi, spíše společníci než ochranáři.

Klub bílého ovčáka[2] (dále pouze klub nebo KBO), jakožto člen ČMKU je jediný klub zabývající se chovem bílého švýcarského ovčáka v České republice. Cílem klubu je udržet toto plemeno zdravé a vyhovující standardu. Tento klub rozhoduje o tom, který jedinec bude chovný a který bude z chovu vyřazen. Aby bylo možné kvalitně rozhodnout, musí každý pes i fena podstoupit dvě kontroly prováděné tímto klubem. Jednou z nich je tzv. svod mladých, kterého se musí zúčastnit každý jedinec ve věku 6 až 18 měsíců (výjimečně 24 měsíců). Pro starší importované jedince tato podmínka neplatí. Druhou a závěrečnou kontrolou je bonitace. Spodní věková hranice pro účast na této kontrole je 16 měsíců. Pokud jedinec u bonitace uspěje, stává se chovným a po dosažení předepsaného věku (18 měsíců u psa a 20 měsíců u feny) je tento jedinec zařazen do chovu. Před bonitací je ještě potřeba psa dvakrát předvést na výstavě pro posouzení exteriéru a povahy. Další prerekvizitou účasti na bonitaci jsou rentgeny kyčelních a loketních kloubů, prováděné od 16 měsíců. Díky těmto rentgenům jsou z vrhu vyřazeni jedinci s vážnými vývojovými vadami. Méně závažné vady jsou při chovu limitovány. Buď prodloužením intervalů mezi jednotlivými vrhy nebo výběrem jedince opačného pohlaví ke krytí se správně vyvinutými klouby, čímž se omezí šíření vad mezi další generace.

S chovem je spojeno mnoho informací, které je potřeba ukládat a vhodně zobrazovat. Dnes má klub všechny informace uloženy v tabulkách a jsou spravovány pomocí tabulkových editorů. Toto řešení není vhodné. Největší nevýhoda je práce offline a následné nahrávání nebo posílání souborů. Některé informace je poté potřeba zobrazit v mírně odlišné podobě na webu, a tak jsou vytvářeny další a další tabulky. Kvůli předešlým nevýhodám je předávání informací pomalé a členové klubu se dozvídají tyto informace se zpožděním. Proto vznikl tento informační systém. Dojde k urychlení celého postupu. Díky práci online budou informace ihned dostupné pro členy klubu. Zadávání informací má přesně definovaný formát, eliminuje se tak zadání dat do nesprávných buněk. Všechny

[1] http://www.cmku.cz/
[2] http://www.bily-ovcak.cz/

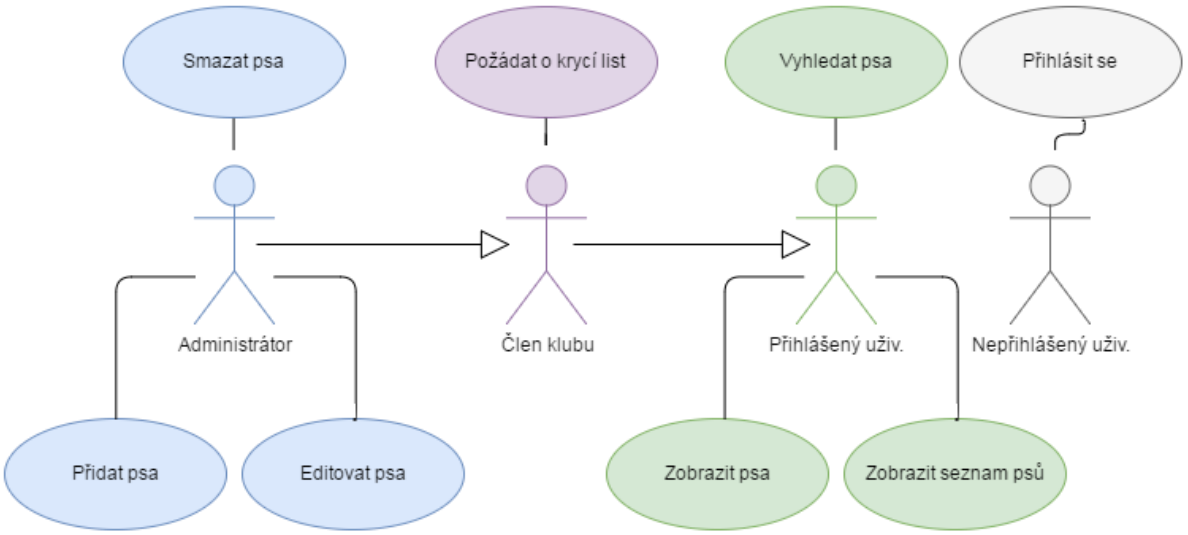
potřebné informace o jednotlivých entitách (pes, ch. stanice, vrh atd.) nalezne uživatel na jednom místě, případně se k nim dostane pomocí odkazu a nemusí procházet několik tabulek.

V kapitole Analýza a specifikace požadavků budou popsány požadavky kladené na aplikaci. Další kapitola Použité technologie informuje o použitých technologiích. Jsou zde vysvětleny především základy, více se dočtete z literatury. Kapitola Návrh řešení je rozdělena do dvou podkapitol. První řeší datový návrh pomocí ER diagramu a následnou transformaci do relační databáze. Druhá podkapitola se zabývá grafickým uživatelských rozhraním, především rozložením prvků na obrazovce. Kapitola Implementace a testování informuje o způsobu implementace jednotlivých částí aplikace. Po přečtení by měla být čtenáři jasná struktura aplikace a měl by mít základní představu o zdrojovém kódu. Předposlední kapitola Testování popisuje scénář testování aplikace a shrnuje jeho výsledky. Kapitola Závěr obsahuje dosazené výsledky, aktuální stav aplikace, hodnocení použitých technologií. Obsahuje také několik nejdůležitějších věcí, které jsou potřeba v nebližší době implementovat. Některé z nich jsou volitelné. Také je zde popsán plán nasazení aplikace do provozu a její následné rozšíření.

2 Analýza a specifikace požadavků

Požadavkem je vytvoření webové aplikace pro klub chovatelů psů, konkrétně pro klub bílého ovčáka. Cílem aplikace je usnadnění práce správcům klubu bílého ovčáka s informacemi ohledně chovu psů. Nejvyšší prioritu mají informace o samotných psech (základní informace, rodokmeny, krycí listy, vrhy, kontroly vrhů). Psi jsou chování v chovatelských stanicích, pro které bude implementována jejich správa. Psi i chovatelské stanice mají své majitele, proto nesmí chybět ani správa osob. Součástí aplikace bude také správa klubových akcí, plateb, krycích listů a kontrol vrhů.

2.1 Správa psů



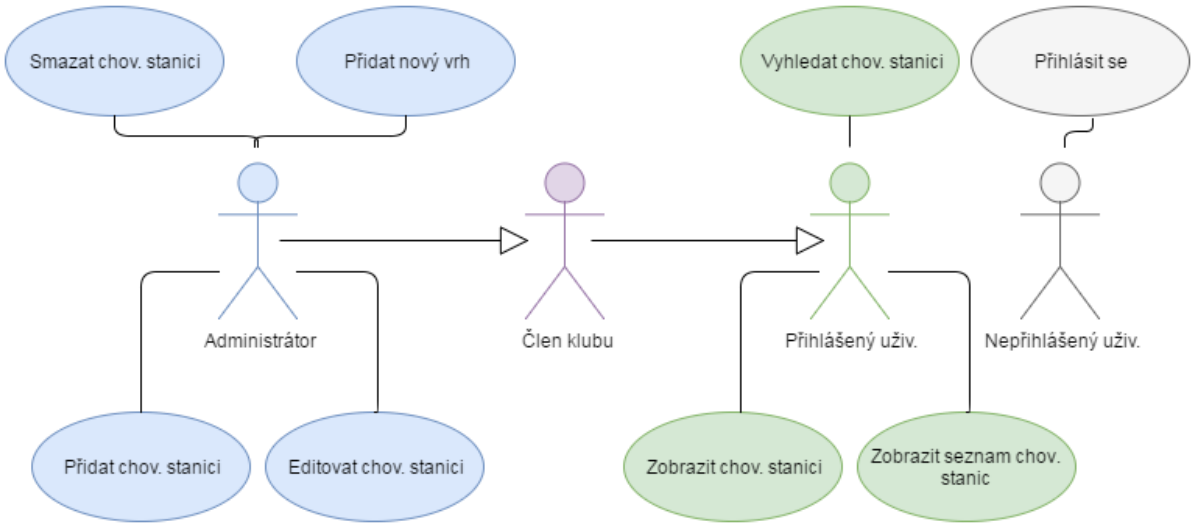
Obrázek 2.1: Use case diagram pro správu psů.

Mezi nejdůležitější požadavky na tento informační systém patří rozhodně správa informací o psech. Správci klubu chtějí přidávat do systému nové psy. Existující psy později upravovat a odstraňovat. Aby bylo možné procházet již vytvořené psy, bude vytvořena stránka se seznamem těchto psů, kde na první pohled uvidíme důležité informace o každém psovi. Zde chceme umět vyhledávat podle jména psa. Každý pes bude moct mít na svém profilu fotografii, která se může časem změnit.

Na profilové stránce psa nalezne uživatel základní informace o psovi (jméno, chovatelská stanice, jména rodičů, datum narození atd.). Informace, které představují existující entitu v systému (chovatelská stanice, matka, otec, atd.), budou vykresleny jako odkazy na příslušné entity. Dále bude zobrazen rodokmen psa, ve kterém uvidíme tři předchozí generace s odkazy na jednotlivé předky.

V případě, že má pes nastaveno ženské pohlaví, umožníme uživateli na profilové stránce psa vytvořit a odeslat žádost o krycí list tohoto psa. V žádosti o krycí list nesmí chybět název chovatelské stanice, předpokládané datum hárání a psi, kterými by chtěl nechat majitel psa tuto fenu nakrýt.

2.2 Správa chovatelských stanic



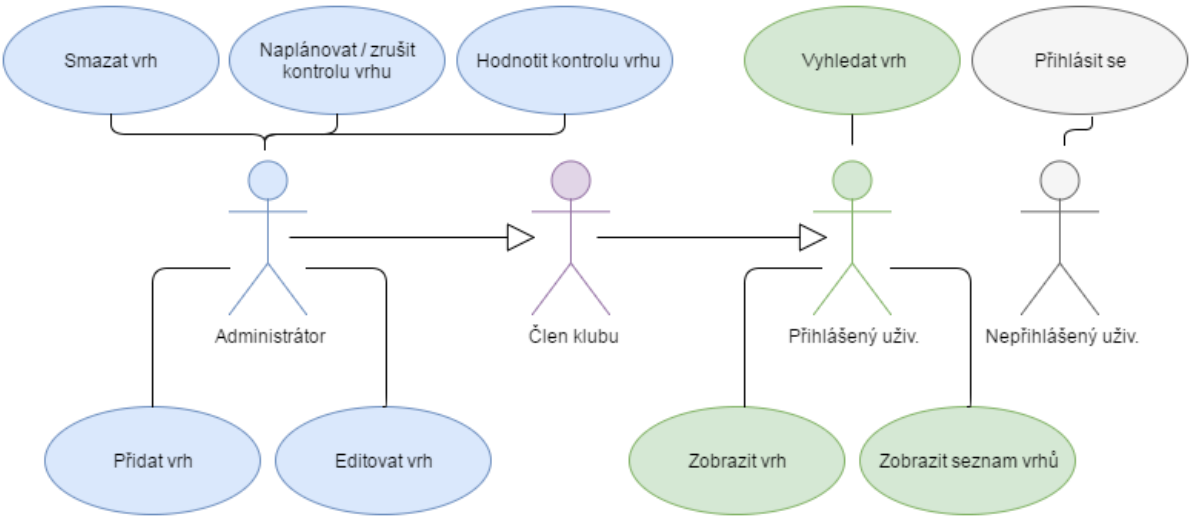
Obrázek 2.2: Use case diagram pro správu chovatelských stanic.

Požadavky na chovatelské stanice jsou v základu podobné s požadavky na správu psů popsané v předchozím odstavci. Chovatelská stanice musí jít vytvořit, upravit a smazat. Pro možnost procházení mezi jednotlivými chov. stanicemi bude existovat stránka se seznamem jednotlivých položek, kde uvidíme základní informace o každé chov. stanici a bude možné se prokliknout na stránku chov. stanice.

Na stránce každé chovatelské stanice budou zobrazeny základní informace o této entitě, kterými jsou název, majitel stanice a označení, zda je chovatelská stanice aktivní nebo nikoli. Pokud je zadán majitel chov. stanice, bude tato informace zobrazena jako odkaz na stránku s informacemi o této osobě. Dále budou na této stránce vhodně zobrazeny jednotlivé vrhy, které patří této chovatelské stanici, a bude možné jednoduchým způsobem získat důležité informace o psech narozených v těchto vrzích. Na první pohled zjistíme i počet psů narozených ve vrzích.

Vrhy se silně pojí na chovatelské stanice, proto je vhodné na stránce se stanicí umožnit uživateli přidat vrh k této stanici. Při vytváření vrhu bude možné zadat všechny základní informace o vrhu [kapitola 2.3] kromě chovatelské stanice. Tato informace bude zadána systémem automaticky.

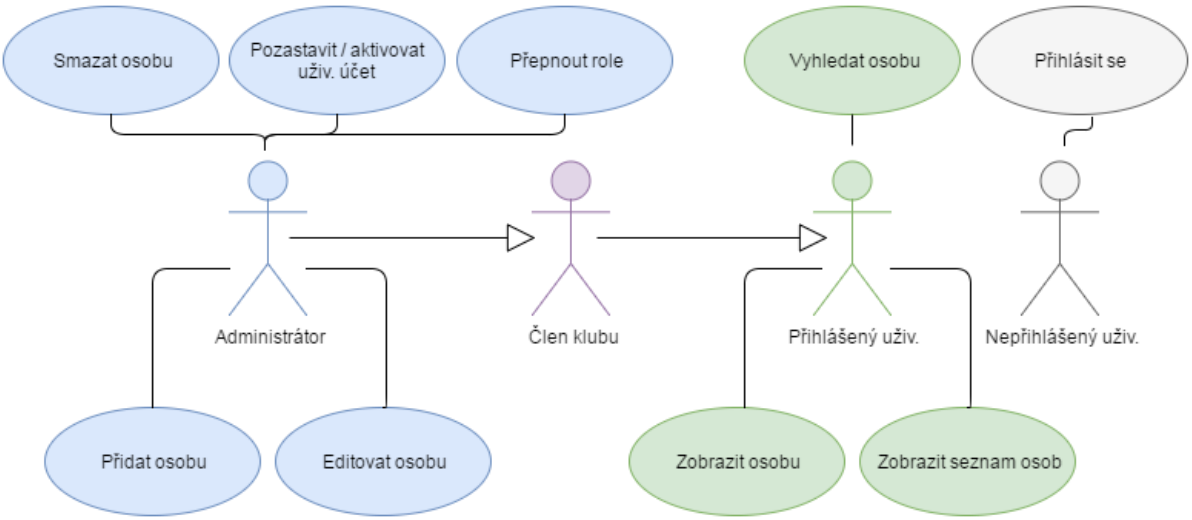
2.3 Správa vrhů



Obrázek 2.3: Use case diagram pro správu vrhů.

Vrhy bude možné prohlížet i samostatně, tj. mimo stránku s chovatelskou stanicí [kapitola 2.2]. Proto je nutné vytvořit stránku se seznamem vrhů, kde budou vypsány důležité informace, které budou odkazovat na stránku s jednotlivými vrhy. Mělo by být možné ve vrzích vyhledávat. Bude možné vytvořit nové vrhy, editovat a odstraňovat existující. Na stránce s jednotlivými vrhy nalezneme mimo základních informací o vrhu také důležité informace o psech narozených v tomto vrhu. Na této stránce bude možné plánovat kontroly vrhů a zadávat jejich výsledky.

2.4 Správa osob



Obrázek 2.4: Use case diagram pro správu osob.

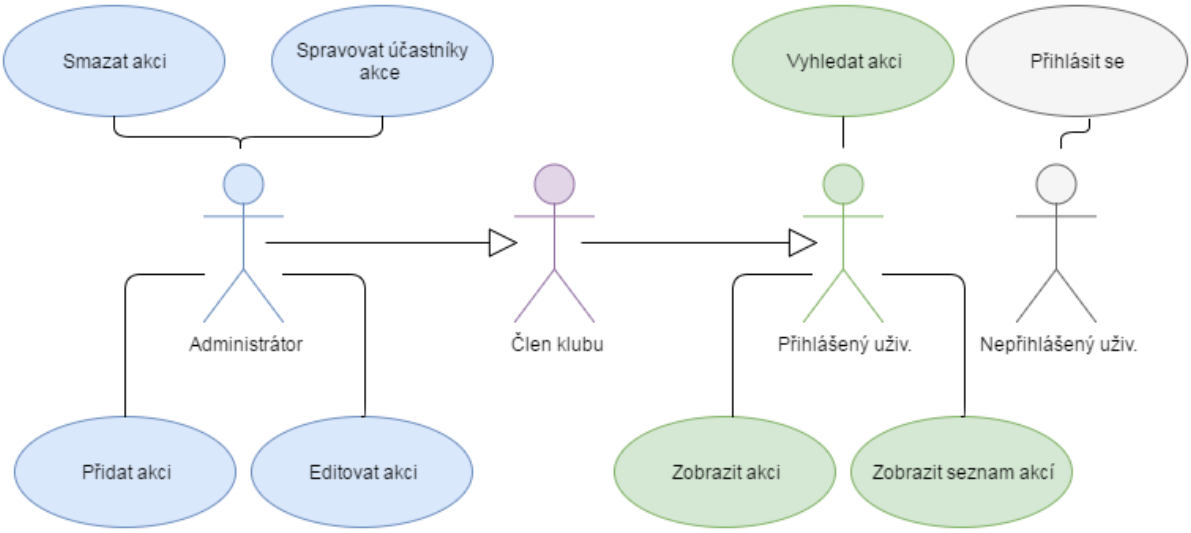
V systému bude potřeba spravovat mnoho různých typů osob, mezi které patří například majitel psa, majitel chovatelské stanice, rozhodčí, organizátor, člen klubu a ostatní osoby. Všechny tyto typy mají

většinu základních informací společných, proto budou mít společnou stránku. Rozdílné informace budou zobrazeny jen v případě, že danému typu osoby přísluší. Je možné přidávat nové, upravovat a mazat již existující osoby.

Na profilové stránce každé osoby budou mimo základních informací zobrazeny důležité informace o psech, jejichž majitelem je daná osoba. Je-li osoba majitelem jedné nebo více chovatelský stanic, budou zde vidět důležité informace o těchto stanicích. Na stránky s vlastními psy i chov. stanicemi se bude možné prokliknout pomocí vhodně umístěného odkazu.

Osobám bude možné nastavit role hlavní administrátor, administrátor, člen klubu, přihlášený uživatel a nepřihlášený uživatel. Role hlavního administrátora umožní uživateli dělat cokoliv. Na druhou stranu role nepřihlášený uživatel nedovolí uživateli udělat nic, kromě zobrazení úvodní stránky a možnost přihlášení. Přihlášený uživatel může zobrazovat a vyhledávat ve všech entitách. Člen klubu může navíc u fen žádat o krycí list. Podrobnější informace o rolích jsou vyobrazeny na jednotlivých use case diagramech v této kapitole.

2.5 Správa klubových akcí

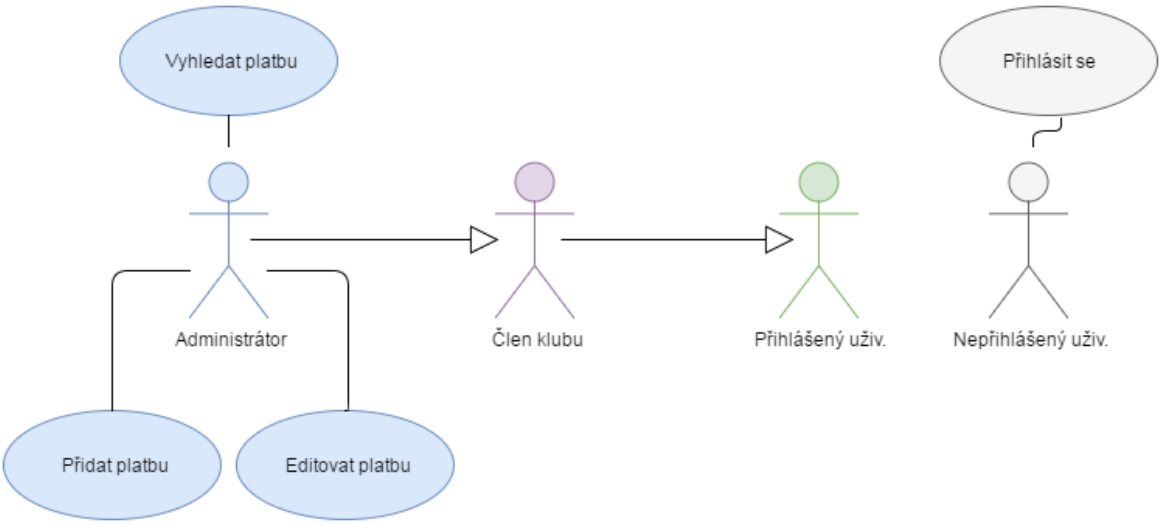


Obrázek 2.5: Use case diagram pro správu klubových akcí.

Klub pořádá pro své členy akce, mezi které patří především svody, bonitace, výstavy a závody. Samozřejmostí je vytvoření nové akce, editace a odstranění již existujících akcí. U akce bude možné zadat název, organizátora akce, místo konání, začátek a konec akce, víceřádkový popis a vybrat volitelně více rozhodčích. Systém musí umět vytvořit více typů akcí. Každý typ akce má volitelně vlastní kategorie, je vhodné podle typu vytvářené akce nabídnout uživateli příslušné kategorie. Organizátor a rozhodčí budou existující osoby v IS, proto je bude možné pouze vyhledat a vybrat.

Na stránce s klubovou akcí budou zobrazeny základní informace s odkazy na existující entity. Mimo to zde budou vidět i účastníci akce, rozděleni po kategoriích. Můžeme přidávat nové účastníky, kde zadáme psovoda, psa a kategorii, ve které se zúčastní. Volitelně můžeme zadat startovní číslo, které zadáváme ideálně na začátku akce a výsledné body, které jsou zadávány na konci akce. Všechny informace jsou možné zadat i najednou, například po ukončení akce. Informace o účastnících bude možné i editovat. Účastníci jsou v jednotlivých kategoriích automaticky řazeni sestupně podle počtu bodů a startovního čísla.

2.6 Správa plateb



Obrázek 2.6: Use case diagram pro správu plateb.

Do systému bude možné manuálně vkládat platby. U platby musí být uveden plátce, částka a typ platby. Systém automaticky doplní zadavatele platby podle přihlášeného uživatele. Přidáme-li platbu s typem *členství*, automaticky se prodlouží členství osobě, která je zadaná jako plátce. Vytvořenou platbu lze v případě potřeby editovat. Provede-li se změna typu platby z členství na jiný typ, zkrátí se členství na původní hodnotu. Pokud při editaci platby změníme plátce, původnímu plátci se zkrátí členství na původní hodnotu a nově nastavenému plátci bude členství prodlouženo.

3 Použité technologie

V této kapitole budou vysvětleny základy technologií, které byly v této práci využity. Na konci kapitoly se nachází stručný popis frameworků pro tvorbu webových aplikací, které mohly být využity místo Nette frameworku.

3.1 Relační databáze – MySQL, MariaDB

Databáze se v aplikacích využívají jako perzistentní uložiště dat[3]. Typů databází je v dnešní době velká spousta. Důležitým faktorem pro výběr databáze je způsob, jakým s ní budeme komunikovat. Lze je rozdělit na databáze, které používají jednotný SQL jazyk (anglicky Structured Query Language). Proti tomu stojí ty, které nepodporují SQL, proto se jim říká NoSQL. Mezi známé NoSQL patří například:

- MongoDB od společnosti MongoDB, Inc.[4]
- Hadoop HBase, Cassandra, Cloudera od společnosti The Apache Software Foundation[5]

V této práci se používá SQL databáze. Z této kategorie zle vybrat opět z velkého množství databází od různých společností. Opět zmíním pouze ty známější:

- MySQL od společnosti Oracle[6]
- MariaDb od společnosti MariaDb Foundation[7]
- PostgreSQL od společnosti The PostgreSQL Global Development Group[8]
- SQLite od společnosti Hipp, Wyrick & Company, Inc.[9]

Z výše uvedených databází se v této práci používá MySQL nebo MariaDb. I přesto, že obě jsou vyvíjeny nezávisle dvěma různými firmami, jsou navzájem kompatibilní[10]. MariaDb vznikla jako odnož původní MySQL, od té doby je vyvíjena odděleně. [2]

Všechny tabulky používají databázový engine, kterým je InnoDB. Je řádkově orientovaný, to znamená, že s daty je manipulováno po řádcích. Tento engine není vhodný tam, kde se pracuje s daty po sloupcích, např. statistiky. Nabízí podporu pro transakce, které jsou využity v této aplikaci. Dřívější engine MyISAM tuto možnost nenabízí. Nepodporuje ani cizí klíče, tedy není vhodný pro systémy, které vyžadují propojovat více tabulek. [1]

3.2 HTML

HTML je zkratkou pro HyperText Markup Language, tedy pro tzv. značkovací jazyk. Byl vytvořen v roce 1991 Timem Berners-Lee ve verzi 0.9 jako součást projektu Word Wide Web[11]. Tato verze umožnila vložit do textu odkazy a obrázky. Postupem času se zvyšovaly nároky a tak byla vydána

[3] trvalá paměť, v níž data přežijí i po vypnutí stroje
[4] https://www.mongodb.com/
[5] http://hadoop.apache.org/
[6] https://www.oracle.com/mysql/index.html
[7] https://mariadb.com/
[8] https://www.postgresql.org/
[9] https://www.sqlite.org/
[10] https://mariadb.com/kb/en/mariadb/mariadb-vs-mysql-compatibility/
[11] https://www.w3.org/

verze 2.0[12], která podporovala navíc i formuláře. V současné době se používá majoritní verze HTML5.

Dokumenty jsou tvořeny s pomocí speciálních značek. Těmto značkách se říká také tagy. Každý HTML tag začíná znakem menšítka '<' a konční znakem většítka '>'. Existují dva typy tagů, párový a samostatný. Párový tag je zpravidla doprovázen ukončovacím tagem, který obyčejně obsahuje za počátečním znakem menšítka znak lomeno '/'. Mezi počáteční a koncový tag je umístěn obsah, na který se vztahují vlastnosti daného tagu.

Každá HTML stránka musí obsahovat přesně definovanou strukturu. Celá stránka je obalena tagem <html> a ukončovacím tagem </html>. Obsahem těchto značek budou také párové značky <head> a <body> doprovázené odpovídající koncovou značkou. Z překladu názvů těchto značek vyplývá, že se jedná o hlavičku a tělo dokumentu. Hlavička dokumentu obsahuje důležité metainformace (kódování, titulek stránky, css styly atd.). Tělo již obsahuje samotný obsah stránky formátovaný pomocí mnoha HTML značek. Chceme-li používat HTML5, musíme na začátek dokumentu vložit DTD[13] značku <!DOCTYPE html>.

3.3 Kaskádové styly

CSS z anglického Cascading Style Sheets je technologie, která pomáhá oddělovat definici vzhledu webové stránky od definice jejího obsahu. Dříve byl vzhled definován přímo v obsahu s pomocí HTML tagů a jejich vlastností. Nevýhody tohoto řešení:

- nepřehledný zdrojový kód
- komplikovaná změna vzhledu stránky
- různé prohlížeče zobrazovaly obsah jinak

CSS umožňují 3 způsoby zápisu stylů:

- přímo do HTML tagu
- definice v hlavičce dokumentu
- v externím souboru (nejpoužívanější, umožňuje použití jednotných stylů na více stránkách)

Styl stránky je tvořen pravidly. Každé pravidlo se skládá ze selektoru[14] a deklarace. Pravidel může být použito i více. Deklarace je tvořena z názvu vlastnosti a její hodnoty. Pro jedno pravidlo může být zadáno i více pravidel, oddělených středníkem. [10]

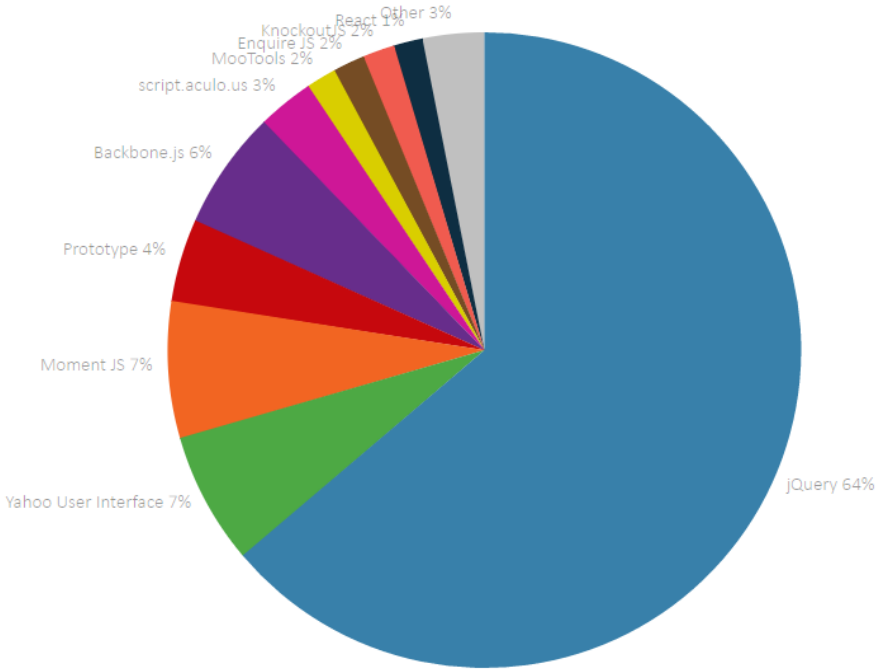
3.4 Bootstrap

Je to populární HTML, CSS a javascriptový framework[15], pro responzivní[16] webový design vhodný především pro mobilní zařízení. Pomáhá k rychlejšímu a snadnějšímu vývoji. Nabízí širokou škálu CSS a JS komponent, které je možné snadno používat. Komponenty jsou navrženy tak, aby byly vhodné pro mobilní zařízení. V této práci je použita nejvyšší stabilní verze Bootstrap 3. [15]

[12] RFC standard https://tools.ietf.org/html/rfc1866
[13] Data Type Definition - definice typu dokumentu. https://www.w3schools.com/xml/xml_dtd_intro.asp
[14] Seznam selektorů naleznete na stránkách. https://www.w3schools.com/cssref/css_selectors.asp
[15] Pomocný software, který usnadňuje programátorovi práci především v často opakujících se případech, ten se tak může více soustředit na své zadání.
[16] Vzhled stránky se přizpůsobuje především šířce displeje.

3.5 jQuery

Je jedna z mnoha knihoven jazyka Javascript. Díky svojí malé velikosti, vyšší rychlosti, mnoha funkcím a skvěle navrženému API se stala tato knihovna jednou z nejpoužívanějších knihoven javascriptu používaných na webových stránkách.



Obrázek 3.1: Statistika použití javascriptových frameworků na webových stránkách ze serveru Builtwith. https://trends.builtwith.com/javascript/javascript-library

Hlavní předností jQuery před samotným javascriptem je práce s DOM[17], především průchod a manipulace s ním. Snadné je také odchytávání událostí a AJAXová[18] komunikace se serverem. Podobně jako CSS i jQuery definuje pravidla, vlastnosti a jejich hodnoty. Nevýhodou javascriptu je nekompatibilita mezi jednotlivými prohlížeči, jQuery tuto nevýhodu potlačuje. [9] [5]

3.5.1 Chart.js

Jedná se o javacsriptovou open source knihovnu sloužící ke snadné tvorbě grafů na straně klienta. Na výběr je až z osmi typu grafů (koláčový, radarový atd.). Veškeré grafy jsou responzivní. Lze je různě konfigurovat a existuje k nim přehledná a obsáhla dokumentace. [14]

3.5.2 Select2

Je jQuery plugin, který nahrazuje klasické html select boxy. Podporuje vyhledávání, načítání dat pomocí AJAXu, stránkování výsledků, překlad do 40 světových jazyků a mnoho dalšího. V této

[17] dokument object model - objektový model dokumentu
[18] Asynchronous JavaScript and XML - asynchronní komunikace se serverem, při níž se nemusí znovu načítat celá stránka.

bakalářské práci je použita verze 4.0.3, která je nyní (duben 2017) nejnovější. Je využívána na veškeré select boxy, které se nachází ve formulářích. [13]

3.6 PHP

Dnešní zkratka z anglického Hypertext Preprocessor. Je populární, všestranně použitelný skriptovací jazyk, vhodný především pro vývoj webových aplikací. Byl vytvořený v roce 1994 Rasmusem Lerdorfem. Zdrojový kód jazyka PHP je vždy vkládán mezi tagy <?php a ?>. Je to interpretovaný, netypovaný jazyk, nepodporující modularitu. Možné používat objektově orientované programovací principy. Tento jazyk můžeme poznat tak, že před názvem každé proměnné se nachází znak dolaru '$' (mají i některé další jazyky). V této práci je použita verze PHP 7, ale brzy chci přejít na verzi 7.1, která má vylepšené typehinty[19], které umožní zadat mimo přesného typu i hodnotu *null*. [12]

3.7 Nette framework

Je český populární webový framework pro jazyk PHP. Automaticky řeší základní bezpečnostní problémy XSS[20], SQL injection[21] a jiné. Součástí je modul Tracy, který pomáhá při ladění aplikace. Při chybě v aplikaci při zapnutém debug módu zobrazí chybovou stránku s tracebackem, doplněnou o části kódu s voláním několika posledních funkcí před chybou a jejich parametry. Nette má open-source licenci. Díky široké komunitě existuje mnoho modulů, které rozšiřují jednoduché použití. Tento framework je využíván například servery Mlada fronta[22], slevomat[23], uloz.to[24] a dalšími.

Součástí Nette frameworku je šablonovací systém Latte. Tento systém je navržený také pro programovací jazyk PHP. Díky němu je vývoj webové aplikace ještě snadnější a příjemnější. Odpadá nám nutnost zadávat složité PHP značky na každý začátek a konec PHP kódu. My vkládáme pouze makra. Bez použití Latte není někdy poznat, jestli se PHP vkládá do HTML nebo opačně. Každé makro začíná otevírací složenou závorkou '{' a končí zavírací složenou závorkou '}'. Je možné vytvářet pojmenované bloky a ty následně využít na více místech. Pokud si nevystačíme se širokou nabídkou vestavěných maker, můžeme si vytvořit vlastní. HTML kód je díky tomu mnohem více přehledný. [8]

Pomocí AJAXu jsou odesílány některé formuláře, které jsou většinou uvnitř modálních oken. Samotný Nette framework tuto možnost nepodporuje, proto je nutné do něj přidat rozšíření nette ajax. To nám také přidává možnost používat tzv. extensions, které nám umožňují provést automaticky nějakou akci v určité fázi odeslání (před, ihned po, v případě úspěchu, neúspěchu nebo vždy po) AJAXového požadavku). [7]

[19] PHP je dynamicky typovaný jazyk, typehinty tuto volnost omezují a dávají nám větší kontrolu nad datovými typy.
[20] Cross-side scripting – vložení škodlivého kódu na stránku
[21] úprava SQL dotazu přímo z webové stránky
[22] http://www.mf.cz/
[23] https://www.slevomat.cz/
[24] https://uloz.to/

3.8 Ostatní webové frameworky

Součástí zadání této práce je seznámení se současnými frameworky pro tvorbu informačních systémů. V této podkapitole si tedy představíme pár známých frameworků, které bylo možné použít jako náhrada za Nette.

Webových frameworků existuje velká spousta. Důležitým faktorem pro výběr vhodného frameworku je pro mě programovací jazyk. Největší zastoupení má jazyk PHP [kapitola 3.6]. Méně používané jazyky jsou například python[25] a ruby[26].

Zend framework je objektově orientovaný framework implementovaný v jazyce PHP, licencovaný jako New BSD. Je navržený podle modulární architektury. Díky tomu je možné používat jen ty komponenty, které programátor potřebuje. Zend framework vyvíjí společnosti Zend Technologies Ltd.[27] a Rogue Wave Software, Inc.[28]. Partnerem jsou společnosti jako Google, Microsoft a StrikeIron, které poskytují rozhraní pro webové služby a další technologie, které mají být vývojářům dostupné. V nabídce jsou i certifikační testy, které ověří znalost těchto technologií. [6]

Pokud někomu nevyhovuje jazyk PHP, může zkusit například python framework Django, který je od společnosti Django Software Foundation[29]. Je to univerzální open source framework, který umožňuje rychlý vývoj webových aplikací. Řeší za programátora běžné bezpečnostní rizika, díky tomu se programátor může soustředit na samotný vývoj. [16]

Pokud ani python někomu nevyhovuje, může sáhnout po frameworku Ruby on Rails[30]. Vyvíjí se v něm v jazyce Ruby[31]. Jazyk i framework jsou open source. Pro vývoj ho využívají například služby GitHub[32], Airbnb[33] nebo Hulu[34]. [17]

Jako poslední si představíme framework od společnosti Microsoft[35], kterým je ASP.NET. Stejně jako Nette i tento spadá do kategorie MVC[36] a je to také open source. Výhodou je integrované uživatelské prostředí (IDE) od stejné společnosti, které poskytuje vývojářům komfort při vývoji. Framework umožňuje TDD[37] přístup k vývoji aplikací. [3]

[25] https://www.python.org/
[26] https://www.ruby-lang.org/en/
[27] http://www.zend.com/
[28] https://www.roguewave.com/
[29] https://www.djangoproject.com/
[30] http://rubyonrails.org/
[31] https://www.ruby-lang.org/en/
[32] https://github.com/
[33] https://airbnb.com/
[34] https://hulu.com/
[35] https://www.microsoft.com/
[36] Model-view-controller - softwarová architektura rozdělující aplikaci na tři nezávislé celky (datový model, uživatelské rozhraní, řídící logika)
[37] Test-driven development - vývoj aplikace probíhá po malých krocích, které začínají vytvořením jednoduchého testu, který neuspěje. Druhým krokem je úprava aplikace tak, aby test uspěl. Tyto kroky se stále opakují.

4 Návrh řešení

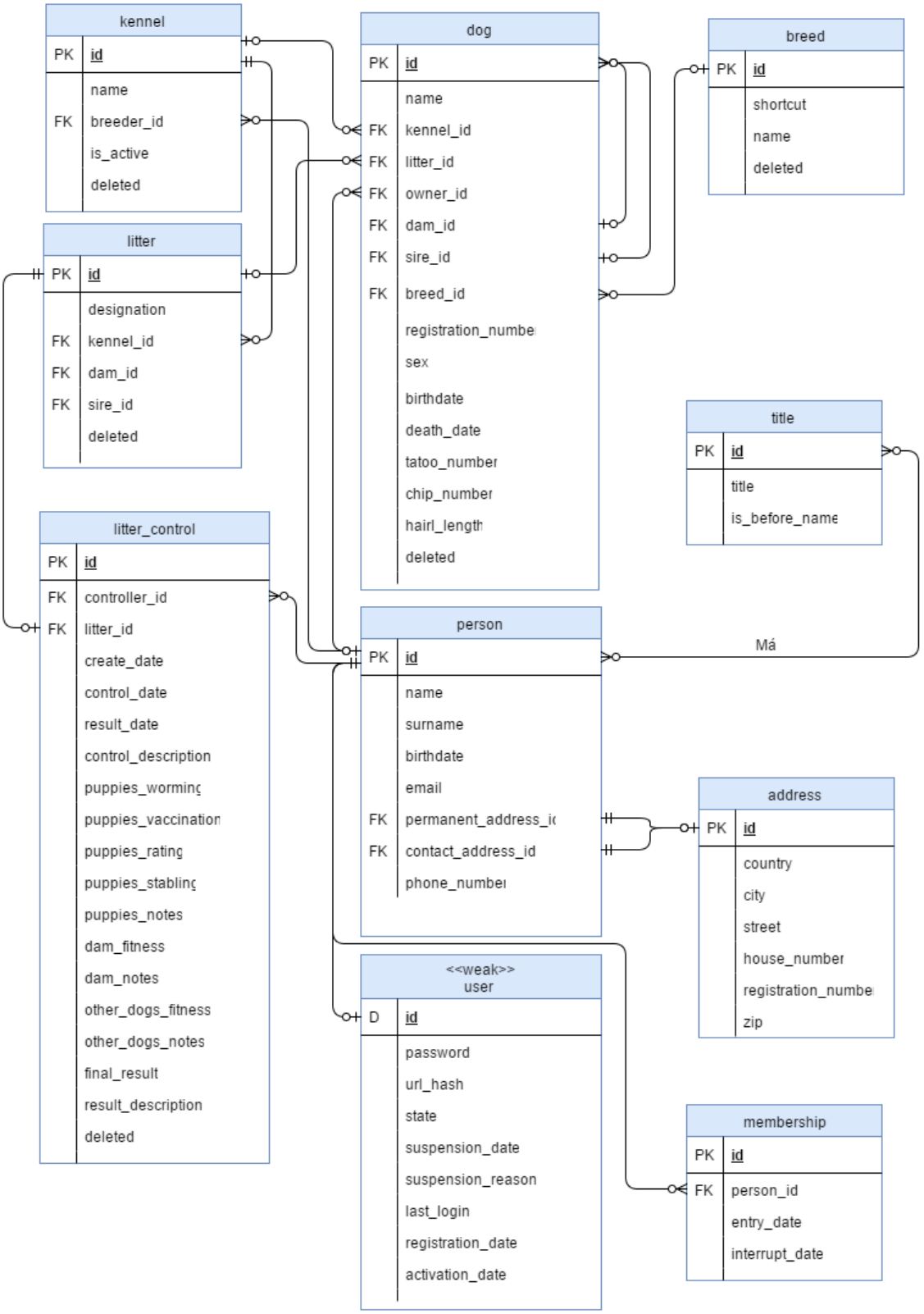
Tato kapitola se detailně zaměřuje na datový návrh, který využívá konceptualního[38] modelování. V této práci využívám ER Diagramu[39]. Popíši zde i návrh uživatelského rozhraní aplikace, kde bude podrobně rozebráno především rozložení prvků na obrazovce, struktura položek v menu a responzivní vlastnosti aplikace.

4.1 Datový návrh

Pro tento informační systém jsem zvolil relační databázi MySQL nebo MariaDB [kapitola 3.1], především díky dostupnosti na většině webových hostingů, znalosti jazyka SQL a její vysoké popularitě.

[38] fáze datové, případně objektové analýzy využívající modelů založených na objektech aplikační domény http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_kmod.pdf
[39] Entity relationship diagram - diagram pro modelování datových entit a vztahů mezi nimi

4.1.1 Návrh ER diagramu



Obrázek 4.1: Hlavní část ER diagramu (1 ze 4).

Z vytvořené analýzy požadavků již můžeme začít vytvářet ER diagram, který poté podle známých pravidel přetransformujeme do podoby relační databáze [kapitola 4.1.2].

Na obrázku výše [Obrázek 4.1] vidíme základní část ER diagramu aplikace. Z analýzy požadavků [kapitola 2] vyplývá, že bude potřeba ukládat informace o psech, osobách, chovatelských stanicích, vrzích, kontrolách vrhů, členství. Proto vytvoříme entitní množiny s těmito názvy. Adresa je složený atribut, proto bude mít také vlastní entitní množinu.

Kvůli typografickým pravidlům pro psaní titulů, z nichž některé se píší před a jiné za jménem, bereme i titul jako složený atribut a vytvoříme mu vlastní entitní množinu. Tituly z této množiny bude možné přiřadit jednotlivým osobám.

Entitní množina pro plemena s názvem *breed* je zde z důvodu budoucího vývoje. Pro klub bílého ovčáka si vystačíme s jedním plemenem, které mohlo být automaticky přiřazováno všem psům přidaným do systému nebo dokonce považováno za defaultní a neuváděno vůbec. Díky tabulce s plemenem je možné psům přiřadit libovolné plemeno, které jednoduše přidáme do entitní množiny.

Nyní si představíme vztahy mezi entitními množinami a kardinality těchto vztahů. Opět se držíme požadavků [kapitola 2]. Protože je tento systém uzavřený a informace v něm jsou spravovány vedoucími členy KBO[40], nenutíme uživatele zadávat všechny informace, ačkoliv by se zdály jako povinné. Kdybychom vyžadovali všechny důležité informace, docílili bychom pouze toho, že by si uživatelé chybějící informace vymýšleli nebo zadávali nesmysly. Z toho vyplývá volitelnost hodnoty u dále uvedených vztahů.

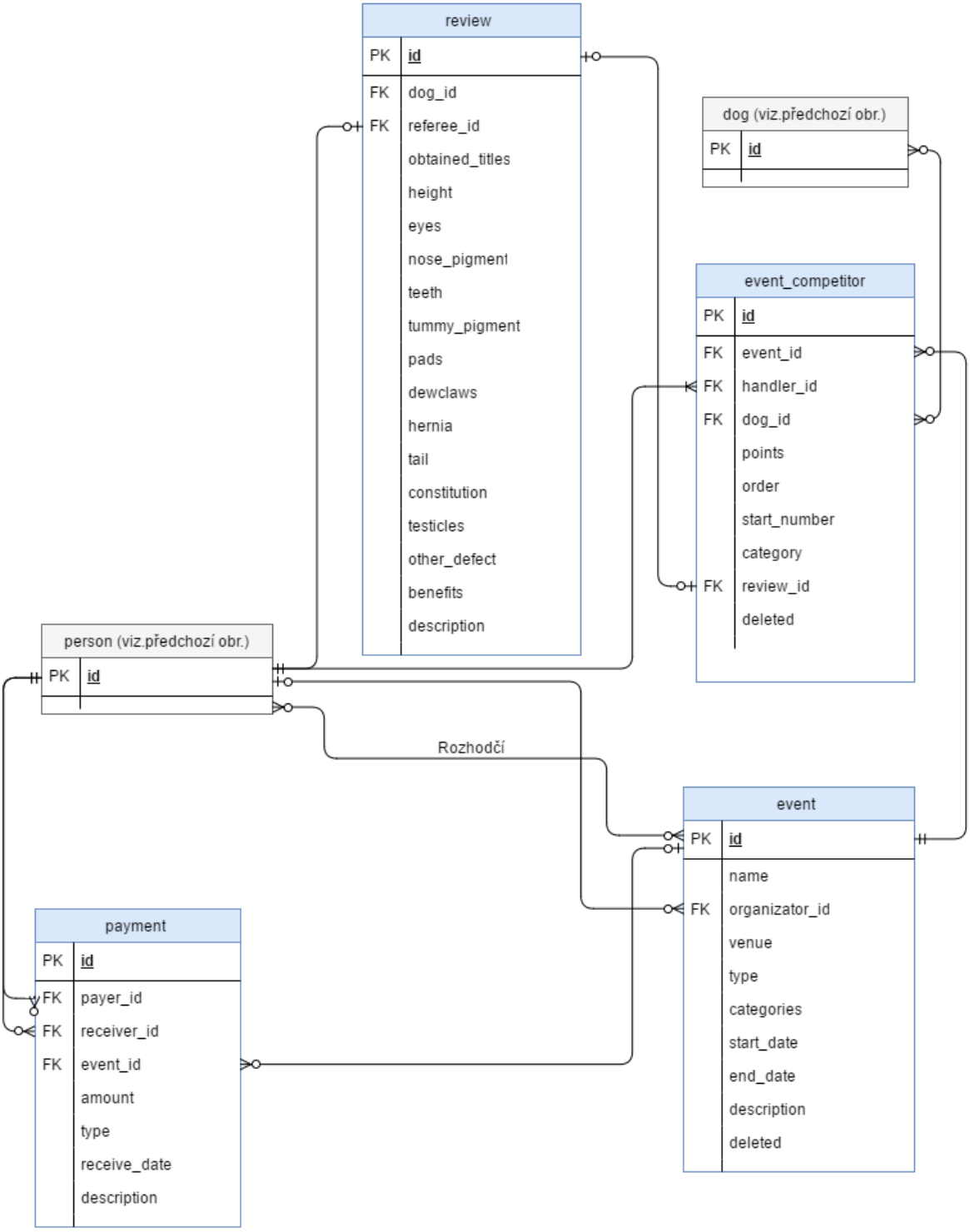
Pes uchovává informaci o jeho předcích. Tyto vztahy se nevidí tak často jako ostatní. Jsou to vztahy unární (reflexivní)[41]. Každý pes má volitelně jednoho otce a volitelně jednu matku. Může být matkou nebo otcem volitelně více psů. Všechny další vztahy jsou binární[42]. Pes je volitelně narozen v jedné chovatelské stanici, volitelně jednom vrhu a vlastní ho volitelně jeden majitel. S ohledem na budoucí vývoj má pes zadané volitelně jedno plemeno.

V chovatelské stanici se může narodit volitelně více psů, v jednom nebo více vrzích. Majitelem stanice je volitelně jedna osoba.

Každý vytvořený vrh musí mít zadanou povinně jednu chovatelskou stanici, ke které tento vrh patří. Povinné jsou také informace o tom, kdo je matka a otec daného vrhu. V jednom vrhu se může pochopitelně narodit více psů. Každý vrh může mít jednu kontrolu vrhu, která patří právě k jednomu vrhu. Kontrolu provádí právě jedna osoba.

Osoby mohou v tomto IS vlastnit volitelně více psů i chovatelských stanic. Jedna osoba může provádět volitelně více kontrol vrhů. Jelikož každá osoba nebude mít v tomto systému možnost přihlášení, nepotřebujeme u všech uchovávat přihlašovací údaje a informace spojené s uživatelským účtem. Z toho důvodu jsou tyto informace ve druhé entitní množině, nazvané *user*. Uživatelské informace patří právě k jedné osobě. Osoba může mít volitelně více členství, z nichž aktivní bude vždy pouze jedno. Neaktivní členství slouží pro přehled historie dané osoby. Jedno členství patří vždy k jedné osobě. Osobě lze zadat volitelně trvalá a kontaktní adresa. Každá adresa patří právě k jedné osobě. Jeden titul může mít více osob a osoba může mít více titulů.

[40] Klub bílého ovčáka
[41] vztah mezi jednou entitní množinou
[42] vztah mezi dvěma entitními množinami

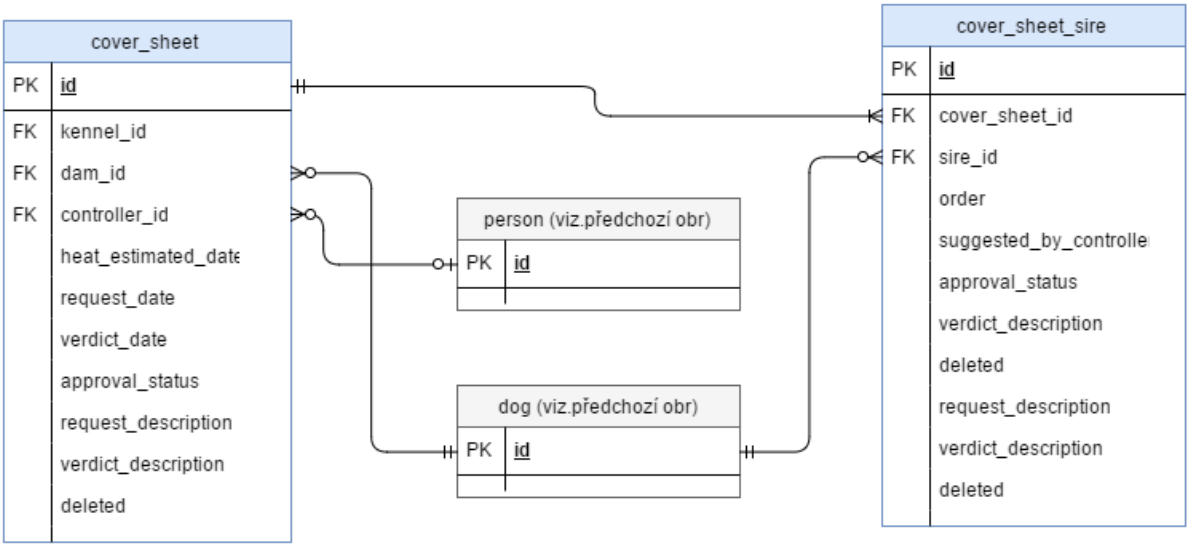


Obrázek 4.2: ER diagram, klubové akce, hodnocení psů a platby (2 ze 4).

V tomto odstavci bude vysvětlen datový návrh klubových akcí, hodnocení psů a platby [Obrázek 4.2]. Stejně jako u předchozího obrázku vycházíme z analýzy požadavků. Entitní množiny *dog* a *person* jsou zde znázorněny jen pro možnost zobrazení všech vazeb a jsou vysvětleny v předchozím diagramu. Dle požadavků vytvoříme entitní množiny pro informace o klubových akcích, účastnících akcí, hodnocení psa a platbách.

Klubové akce mohou mít volitelně více rozhodčích i účastníků a volitelně jednoho organizátora. Účastník akce je platný vždy jen pro jednu akci. Účastník obsahuje právě jednu osobu v roli psovoda a právě jednoho psa. Účastník akce může jako výsledek akce získat hodnocení. Hodnocení se vždy nemusí vázat k nějakému účastníkovi, protože se používá i u kontrol vrhů. Hodnocení má právě jednoho rozhodčího, který toto hodnocení udělil.

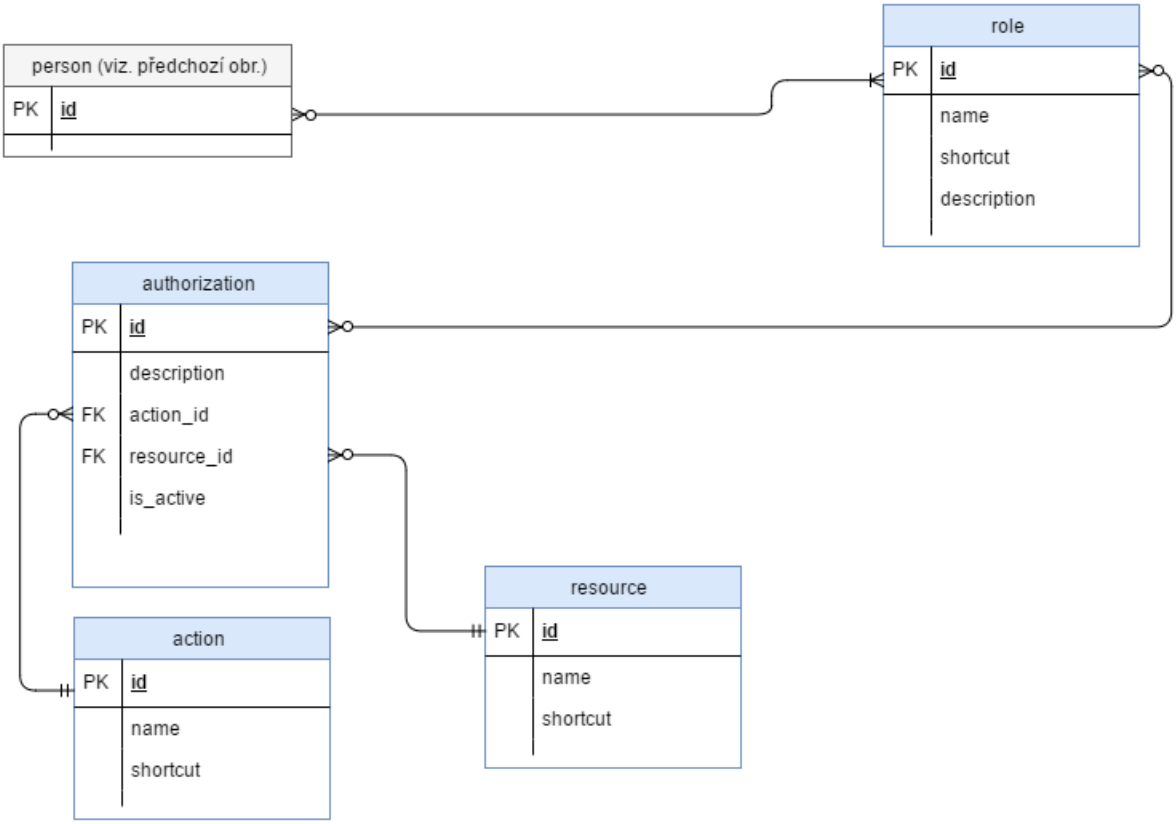
Každá platba uchovává informaci o plátci, kterým je právě jedna osoba a informaci o osobě, která platbu přijala, tou je také právě jedna osoba. V případě, že je typ platby *startovné*, je možné tuto platbu volitelně provázat s jednou klubovou akcí. Akce může mít volitelně více plateb.



Obrázek 4.3: ER Diagram krycích listů (3 ze 4).

Součástí zadání je i možnost požádat o krycí list a následně jej schválit nebo zamítnout. Proto musíme připravit datové struktury i pro tyto informace. K žádosti je nutné přidat jednoho či více krycích psů. Entitní množiny *person* a *dog* jsou opět vykresleny pouze pro možnost zobrazení všech vazeb.

V žádosti o krycí list nesmí chybět právě jeden pes v roli chovné feny. Před schválením či zamítnutím žádosti není zadaný kontrolor, proto v žádosti vidíme volitelnou účast kontrolora. Žádost povinně obsahuje informace o jednom nebo více krycích psech. Pes může být uveden volitelně vícekrát v roli krycího psa. Informace o krycím psovi se váží právě k jedné žádosti.



Obrázek 4.4: ER Diagram oprávnění uživatele (4 ze 4).

Poslední diagram závisí na zadání jen tím, že je požadováno ověřování uživatelů a více rolí. Zbytek je odvozený od požadavků na nastavení ACL[43] [kapitola 5.2], který nabízí Nette framework a je z většiny založený na *Zend_Acl* od společnosti Zend Technologies USA Inc.

Celá datová struktura pro tento ACL se skládá z entitních množin pro role, zdroje, akce a autorizaci, která propojuje všechny předchozí a umožňuje uložit uživatelský popis jednotlivých oprávnění. Entitní množina osoby je zde vyobrazena opět pro možnost zobrazení všech vazeb.

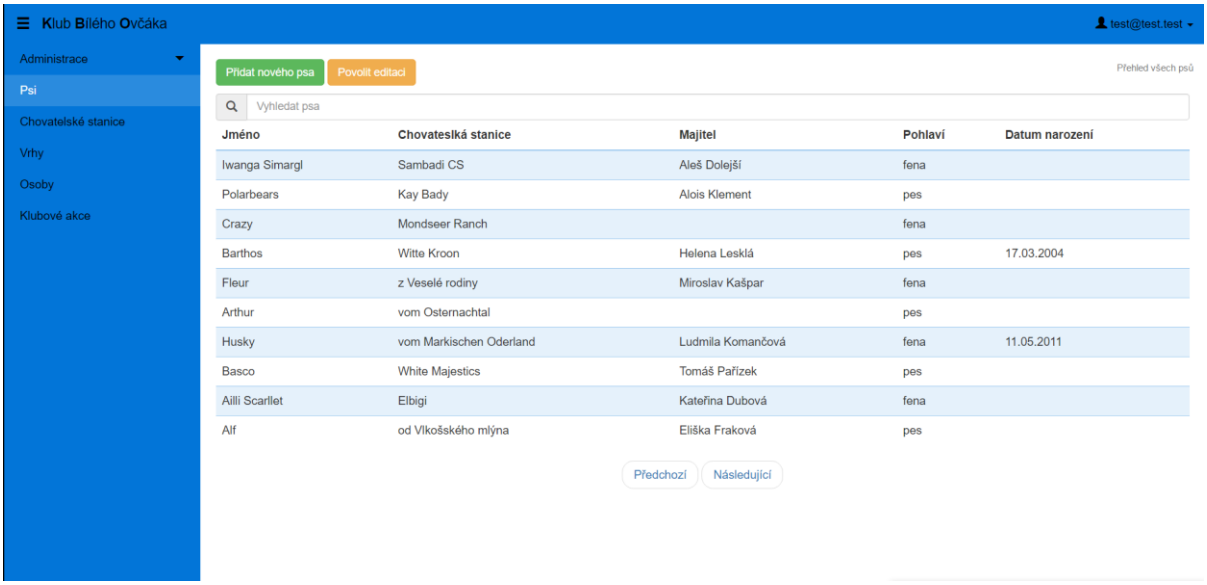
Osoba neboli uživatel může mít více rolí. Jednu roli může mít více uživatelů. Role může být napojena k více autorizacím a autorizace může být přiřazena více rolím. Autorizace musí obsahovat povinně jednu akci a povinně jeden zdroj. Zdroj i akce mohou být volitelně ve více autorizacích. Tímto vytvoříme množinu všech akcí a zdrojů, na které mohou uživatelé s jednotlivými rolemi přistupovat.

4.1.2 Transformace ER diagramu na relační databázi

Každá entitní množina se převede i s primárními klíči na tabulku relační databáze. Vztahy 1:1 a 1:N se převedou stejně, jako jsou v ER diagramu. Atributy s označením FK se v MySQL databázi označí jako cizí klíče, odkazující na tabulku podle diagramu. Vztahy M:N vyžadují vytvoření vazebních tabulek. Ty budou ve všech našich případech obsahovat pouze 2 atributy, které budou dohromady tvořit jeden složený primární klíč, a každý bude sám za sebe cizí klíč s odkazem na jednu z tabulek původního M:N vztahu. Takto vytvořená vazební tabulka nedovolí vytvořit duplicitní kombinace obou atributů.

[43] access control list - zajišťuje přístup pouze pro oprávněné uživatele

4.2 Návrh uživatelského rozhraní



Obrázek 4.5: Náhled stránky se seznamem psů.

4.2.1 Hlavička

Hlavička je standardně umístěna u horního okraje stránky a je zde fixována. Na levé straně hlavičky je umístěn název klubu, který může být později nahrazen logem klubu. Název či logo slouží jako odkaz na hlavní stránku informačního systému. Nalevo od odkazu se nachází tlačítko na zobrazení nebo skrytí levého menu. Na pravé straně hlavičky se nachází odkaz pro zobrazení přihlašovacího modálního okna. V případě, že je uživatel přihlášený, bude text stejného odkazu nahrazen jeho přihlašovacím jménem. Po kliknutí na tento odkaz se zobrazí nabídka, která nabízí uživateli vstup do administrace nebo odhlášení z aplikace.

4.2.2 Menu

Menu je situováno na levé straně webové stránky. Začíná hned pod hlavičkou a rozprostírá se po celé výšce stránky. Pomocí tlačítka v hlavičce je možné toto levé menu zasunout (skrýt) a rozšířit tak místo pro obsah stránky na celou šířku obrazovky. Na menších zařízeních je menu automaticky skryto, na širších zobrazeno. Změníme-li stav menu (skryto nebo zobrazeno), uloží se nový stav a po znovunačtení stránky nebo přechodu na jinou stránku zůstane tento stav zachován. Položky v menu umožňují v případě potřeby přidávat subpoložky a tvořit tak hierarchickou strukturu menu. Samotné zasouvací menu je modul *Simple Sidebar*, jehož autorem je David Miller. [4]

4.2.3 Správa uživatelských práv

Správa uživatelských práv je jednou z mála věcí, které nebyly požadovány, ale i přesto byly implementovány. Je to důležité nastavení systému, které by mělo být editovatelné minimálně hlavním administrátorem. Na stránku se dostaneme přes položku *Administrace* v levém menu a podpoložku

Práva. Stránka je vizuálně rozdělena po uživatelských rolích, mezi kterými lze přepínat pomocí *bootstrap pills*[44].

Práva pro hlavního administrátora jsou na této stránce zobrazeny z informačních důvodů a pro úplnost, ale nelze je měnit (tato role má neomezená práva). Jejich tlačítka jsou deaktivována. Práva jsou na stránce oddělena po kategoriích (pes, chovatelská stanice, osoba, admin atd.) pro lepší orientaci uživatele. Každé právo má vlastní popis, podle kterého lze jednoznačně pochopit, co znamená. Zapínání a vypínání práv se dělá pomocí přepínacího tlačítka[45]. Toto tlačítko odesílá při přepnutí stavu AJAXové požadavky na server.

4.2.4 Listování a vyhledávání entit

Stránka se seznamem všech entit jednoho typu (pes, chov. stanice, osoba atd.) slouží jako vstupní stránka pro tyto entity. Mimo to se na ně dá dostat přes odkazy z jiných entit, které jsou s nimi propojené. Je zde vhodné místo pro zobrazení tlačítka pro vytvoření, editaci a smazání entity téhož typu. Tato tlačítka jsou umístěna na začátku stránky. Pod nimi se nachází vyhledávací vstup. Vyhledávání je realizované pomocí AJAXu, který se spustí po zadání řetězce do vyhledávacího pole a následně se pomocí snippetu překreslí oblast, která zobrazuje vyhledané výsledky. Počet položek na stránce je omezen, aktuálně na 10 položek na stránce.

4.2.5 Vytváření nových entit

Vytváření nových entit je v systému implementováno dvojím způsobem. Pokud je u vytváření entity žádoucí vytvořit na stejné stránce entitu jinou a tu použít jako vstup původní entity (při vytváření psa chceme vytvořit majitele psa, abychom ho mohli použít pro tohoto psa), používáme pro vytváření normální stránku. Potřebnou entitu je pak možné vytvořit pomocí odkazu, který otevře modální okno s formulářem pro její vytvoření. Po vytvoření sekundární entity se tato automaticky vloží jako vstup do primárního formuláře. Pokud není potřebné vytvářet sekundární entity, vytváří se entity přímo z modálních oken.

Formulářové prvky jsou navrženy tak, aby co nejvíce ulehčovaly uživateli zadávaní dat. Vstupy pro datum a čas nabízí datepickery[46]. Na hodně místech je žádoucí zadávat jako vstup již existující entity. Tyto vstupy jsou implementovány pomocí select2 [kapitola 3.5.2] vyhledávacích select boxů, které podporují dynamické hodnoty. Díky tomu se tím jednoduše implementuje AJAXové vyhledávání. Uživatel tak napíše část informací o hledané entitě a pak už jen vybere entitu z nabídky.

4.2.6 Editace a mazání entit

Editace využívá stejné formuláře jako při vytváření. Jediný rozdíl je v tom, že jsou všechny vstupy vyplněny hodnotami z databáze. U editací mimo modální okna se informace získávají při načítání stránky. U modálních okem se informace získají pomocí AJAXového dotazu. Jednotlivé vstupy jsou pak vyplněny pomocí jQuery.

[44] Sada tlačítek, pomocí kterých lze přepnout kontext stránky - http://getbootstrap.com/components/#nav-pills.
[45] bootstrap toogle button - http://www.bootstraptoggle.com/
[46] Vizuální kalendář, pomocí kterého zde jednoduše zvolit datum.

5 Implementace a testování

V této kapitole si objasníme samotnou implementaci jednotlivých části této práce. Kapitola bude více technicky zaměřená, než všechny předchozí. Jelikož jsou všechny entity navrženy v základu podobně a liší se pouze v drobnostech, budou vysvětleny dohromady. Podkapitoly jsou rozděleny podle akcí, které je možné s entitami provádět. Mezi ně patří vytvoření, editace, vyhledání, odstranění, vylistování. Dále jsou vysvětleny další důležité části jako vytvoření nového uživatelského účtu, přihlášení, vlastní mezivrstva mezi modelem a DB.

5.1 Správa entit

Tato podkapitola popisuje všechny možnosti správy entit (vytváření, editace, mazání). Informuje o principu listování, vyhledávání a zobrazení na implementační úrovni.

5.1.1 Vytváření a editace entit

Každá entita má svou vlastní továrnu umístěnou v adresáři *app/forms*, která se využívá na vytváření a editaci. V této továrně jsou nadefinované prvky formuláře a také jejich vlastnosti - omezení maximální délky, omezení vstupních znaků, povinnost zadání hodnoty atp. Vykreslení prvků je implementované ručně, v šabloně entity, většinou v adresáři *app/presenters/templates/components/<nazevEntity>*, buď v šabloně určené pouze na editaci či vytváření nebo v šabloně, kde je formulář jako modální okno. Prostředník mezi šablonou a formulářovou továrničkou je presenter. V případě modálního okna se formulář odesílá pomocí AJAX požadavku, jehož úspěšnost se kontroluje pomocí jQuery. V případě úspěšnosti se modální okno automaticky zavře a vypíše se flash zpráva[47]. V případě neúspěchu modální okno zůstane vyobrazené a do formuláře se doplní vzniklé chyby.

Prvky typu selectbox jsou inicializovány jako *select2* boxy. Selectboxy, které je nutné plnit daty z databáze, jsou plněny pomocí jQuery. Ten pomocí AJAXu získává data z presenteru dané entity, konkrétně pomocí metody `actionSearchEntity`. Tato metoda data předá zpět do šablony v řetězcové podobě a tam si je jQuery zpracuje pomocí metody `stringToJson` a doplní je do select2 boxů.

Jestliže nastane při odeslání formuláře nějaká chyba, je pomocí metody `redrawControl` vykreslena pod input, kterého se týká. Překreslení zajistí `snippet` s názvem `<názevInputu>ErrorSnippet`. Jestliže se jedná o obecnou chybu formuláře, je vykreslena na vrchol formuláře. Pokud se formulář odešle bez chyb, vykreslí se zpráva o úspěšnosti do flash zprávy a stránka se následně přesměruje.

V případě, že se provádí editace entity, tak se proces liší od vytváření pouze v tom, že se všem prvkům ve formuláři načtou hodnoty z databáze. V případech, kdy se jedná o modální okno, plní se formulář pomocí AJAXu a také se nadpis modálního okna změní pomocí jQuery na text, který vypovídá o editaci dané entity. Tento text se změní na původní, pokud se modální okno použije na vytváření entity. Mimo modální okno se formulářové hodnoty vkládají pomocí proměnné, kterou předává presenter do šablony při načtení stránky a ta se do každého prvku doplní pomocí atributu `value`. Při editaci je nutné znát identifikátor editované entity, ten se ukládá do skrytého vstupního

[47] Oznámení, které se zobrazí uživateli po přesměrování na další stránku. Slouží především k oznámení úspěšnosti prováděné akce uživateli.

prvku, který je součástí formuláře. Veškeré formuláře, využívané v modálních oknech, jsou před každým zobrazením pomocí jQuery metody `resetForm` vymazané.

Při vytváření i editaci se z frontendu volají metody `save<názevEntity>`, které jsou umístěny v souborech *app/model/<názevEntity>Manager.php*. Vstupem této metody jsou data přímo z formuláře. Než začneme s těmito daty pracovat, upravíme si je do lepší podoby. To udělá metoda `normalizeFormData` zděděná z objektu `BaseManager`. Všechny nevyplněné formulářové prvky se posílají jako prázdné řetězce, ale pro další použití zejména v databázi nahradíme tyto prvky hodnotou `NULL`. Metoda na normalizaci dat přijímá ukazatel na pole dat, proto upravuje rovnou původní data a nevrací nic. Pokud je potřeba některé hodnoty přejmenovat, udělá se to v tomto okamžiku. Následuje částečná validace vstupních dat, především existence jiných entit v systému.

V tomto okamžiku máme správně pojmenované a částečně validní data, která můžeme poslat datovému objektu `<názevEntity>`. Vytvoříme datový objekt a zavoláme jeho metodu `loadData`, které předáme připravená data. Metoda načte data do instančních proměnných. Tyto proměnné jsou ve skutečnosti property[48]. Dále již nepracujeme se samotnými daty v poli, ale s objektem, který má implementovány vlastní metody pro práci s daty.

Na konci metody `save<názevEntity>` se teprve rozděluje kód pro vytváření a editaci entity. Dostáváme se do privátních metod `_create<názevEntity>` nebo `_edit<názevEntity>`. V nich se vytvoří buď vkládací nebo editovací SQL dotaz a odešle se do databáze. Situace je složitější, pokud spolu s entitou ukládáme i další entity. Například spolu s osobou jsou ukládány tituly do vazební tabulky a adresy do tabulky *address*. Při vytváření jsou tyto dodatečné záznamy pouze vytvořeny. Při editaci je potřeba zjistit, který titul byl přidán a který odebrán, abychom totéž mohli zadat do DB.

5.1.2 Mazání entit

Veškeré mazání entit je řešeno pomocí akcí, které jsou definované v presenteru. Tyto akce vyžadují parametr, kterým je identifikátor mazané entity. Než se entita skutečně smaže, je nutné potvrdit svou volbu v modálním okně. Po kliknutí na tlačítko *Ano* se teprve provede jQuery AJAXové volání, které je napojené na akci v presenteru. Při kliknutí na tlačítko *Ne* se modální okno pouze zavře a žádná jiná akce se neprovede. Akce nejdříve vytvoří požadavek na backend, aby se pokusil smazat danou entitu. Následně vykreslí výsledek požadavku do flash zprávy, čímž informuje uživatele o úspěšnosti.

Na mazání entit slouží jedna společná metoda `delete`, které předáme jako vstup identifikátory mazaných entit, název tabulky a pole s názvem sloupců jako klíče a hodnotami sloupců jako hodnoty. Toto pole slouží k nastavení některých sloupců na požadované hodnoty. Při mazání entity nastavujeme sloupce *deleted* na hodnotu 1, tedy nemažeme, ale označujeme jako smazané. Tyto záznamy poté nejsou za normálních okolností vyčítány.

5.1.3 Listování entit

Listování neboli seznam entit se zobrazuje v šabloně, která se většinou nazývá *default.latte*. Je umístěna v adresáři *app/presenters/templates/<názevEntity>*. Tato šablona je vykreslována pomocí render metody definované v presenteru entity, která si nejprve vyžádá od backendu veškerá potřebná data a následně je předá do šablony, kde se pomocí cyklu foreach jednotlivé entity postupně vypíší do tabulkové podoby.

Na backendu se pro zobrazení seznamu entit využívá metod nazvaných `getList`, umístěných v objektech *<názevEntity>*Manager. Tato metoda pouze vytváří objekt

[48] Proměnná, se kterou se pracuje pomocí getteru a setteru.

`SearchPattern` [kapitola 5.3], který zde přenáší informaci o vyhledávaných slovech [kapitola 5.1.4]. Instance `SearchPattern` je odeslána metodě `get<názevEntity>ByPattern`, která získá veškeré potřebné informace o hledaných entitách. Informace jsou opět načteny do datových objektů. Tato metoda pracuje hromadně nad všemi vyhledaným entitami najednou. Díky tomu se radikálně omezí počet databázových volání. Výsledkem metody je pole datových objektů daných entit, indexované pomocí identifikátorů entit pro jednoduší procházení polem.

5.1.4 Vyhledávání entit

Nad seznamem entit se nachází input prvek na vyhledávání, který funguje jako filtr entit. Tento input prvek je obalený formulářem, který má svou vlastní továrnu opět definovanou mezi ostatními formulářovými továrnami v *app/forms*. Na tento input prvek je napojena jQuery událost, která sleduje změny tohoto prvku. Po vložení minimálně jednoho znaku se hodnota vstupu odešle pomocí AJAXového požadavku do presenteru, který předá požadavek backendu. Jako výsledek získáme nový, zúžený seznam všech entit vyhovujících řetězci zadanému v input prvku. Po obdržení těchto dat se presenter postará o vylistování vyfiltrovaných hodnot pomocí snippetu, kterým je obalena oblast, kde se zobrazuje seznam entit.

V případě vyhledávání se volá stejná backendová metoda *getList*, jako je tomu v předchozí podkapitole [kapitola 5.1.3]. Jediný rozdíl je v tom, že se vyhledávaný řetězec načte do objektu `SearchPattern`. Tato instance vytváří SQL podmínku pro následující funkci, která získává data z DB. Díky tomu máme zajištěn přenos vyhledávacího řetězce. Z databáze se tak vytáhnou jen data, která odpovídají vyhledávání.

5.1.5 Zobrazení entity

Entity, které podporují kompletní přehled o dané entitě, mají definovanou šablonu na tuto možnost v adresáři *app/presenters/templates/<nazevEntity>*. Většinou se tato šablona nazývá *show.latte*. Data se do této šablony předávají pomocí rendrovací metody pojmenované obvykle `renderShow` definované v presenteru dané entity. Tato metoda vyžaduje parametr obsahující identifikátor entity, která má být vyobrazena. Vytváří se zde požadavek na backend, požadující data o zobrazované entitě. Následně se tyto data předají do šablony, kde se vykreslí.

Získání dat z backendu je při zobrazování entity shodné jako při zobrazení seznamu entit [kapitola 5.1.3]. Jediný rozdíl je v tom, že vstupem je identifikátor právě jedné entity. Výsledkem je tedy také pouze jeden objekt.

5.2 Uživatelské účty

V této podkapitole získáte implementační podrobnosti o správě uživatelských účtů (vytváření, aktivace, pozastavení) a detaily o přihlášení.

5.2.1 Vytváření uživatelských účtů

Díky tomu, že je tento informační systém uzavřený, není potřeba mít registrační formulář. Uživatelský účet může být vytvořen každé osobě, která má vyplněný údaj s emailovou adresou. Nové uživatelské účty vytváří administrátoři pomocí tlačítka na profilové stránce osoby.

Po stisknutí tlačítka se odešle AJAXový požadavek na vytvoření uživatelského účtu pro danou osobu odeslaný do presenteru metodě `handleCreateAccount`. Osoba je identifikována pomocí ID, které se předává jako parametr metody. Presenter ihned po ověření uživatelských práv přepošle ID osoby modelu, který se postará o vytvoření účtu.

Důležitá část implementace se nachází v souboru *app/model/UserManager.php* spolu s dalšími metodami pro správu uživatelských účtů. Nyní nás zajímá pouze metoda `add`, která vyžaduje jako parametr ID osoby, které bude nový účet patřit. Podle návrhu databáze [Obrázek 4.1] víme, že databázová tabulka *user*, do které se ukládají uživatelské účty je jen rozšíření tabulky *person* (1:1) a tak mají společné identifikátory. Proto ID osoby, která bude vlastnit tento účet, se použije rovnou jako ID uživatelského účtu. Protože budeme uživateli posílat emailem aktivační odkaz, je vhodné využít nějakého hashe, který přidáme k odkazu, k identifikaci příchozího uživatele pomocí odkazu. Metoda tedy vytvoří md5[49] hash. Vloží do databáze ID účtu spolu s vytvořeným hashem. Pokud již záznam s tímto ID existuje, vrátíme výjimku s informací o vytváření duplicitního účtu. V opačném případě je účet úspěšně vytvořen. Návratovou hodnotou funkce je aktivační hash.

Ke slovu se dostává opět presenter. Použije šablonu pro odeslání aktivačního emailu z *app/presenters/templates/Person/registration_email.latte*, dodá jí parametr, jímž je hash vrácený z modelu. Šablonu s vyplněným hashem odešle pomocí `Nette\Mail\Message` jako emailovou zprávu uživateli, kterému účet patří.

5.2.2 Aktivace účtu

Jakmile uživatel klikne na aktivační odkaz, přejde na stránku *person/accountActivation*, kterou má na starost opět *PersonPresenter.php*. Zavolá se metoda `renderAccountActivation`, která vykreslí uživateli šablonu ze souboru *app/presenters/templates/Person/accountActivation.latte*. Její dominantou je aktivační formulář, ve kterém uživatel uvede nové heslo a pro kontrolu ho ještě zopakuje. Před odesláním formuláře se validuje shoda obou hesel. Po úspěšném odeslání formuláře se pouze přepošle požadavek na aktivaci účtu do modelu spolu s hashem a heslem. Uživatel je poté automaticky přihlášen a přesměrován na hlavní stránku.

Za backend se o aktivaci účtu stará metoda `activateAccount` v souboru *app/model/UserManager.php*, která provádí následující. Nejdříve zjistí, jestli daný hash existuje v DB a ke kterému uživ. účtu patří. Následně zjistí, jestli není již tento účet aktivován. Pokud už byl účet dříve aktivován, ale z důvodů nějaké chyby mu nebyl smazán aktivační hash, smažeme mu ho nyní. Není-li ještě účet aktivován, upravíme záznam s ID uživ. účtu v databázové tabulce *user* takto:

- uložíme uživatelem zadané heslo
- změníme stav účtu z neaktivní na aktivní
- nastavíme datum aktivace účtu a poslední přihlášení na aktuální datum a čas
- nastavíme aktivační hash na *NULL*, aby nebylo možné tento účet znovu aktivovat

5.2.3 Přihlášení

O autentizaci se stará metoda `authenticate` v souboru *app/model/UserManager.php*. Vstupem této metody je uživatelské jméno (email) a heslo v jednom poli. Jako první se pokusí najít v databázi uživatele se shodným emailem a získat jeho ID a heslo. Pokud nenajde žádného uživatele s tímto emailem, vyvolá výjimku s informací o neexistujícím uživateli. V opačném případě (uživatel existuje) přistoupíme ke kontrole hesla. S tím nám pomůže třída `Nette\Security\Passwords`.

[49] Rychlý hashovaní algoritmus, který generuje hash o délce 128 bitů, reprezentované často jako řetězec s o délce 32 znaků. http://www.faqs.org/rfcs/rfc1321

Zavoláme její metodu `verify`, které předáme zadané heslo a hash hesla v DB a ona nám ověří jejich shodu. Už zbývá jen zjistit, jestli není tento účet pozastavený. Na pozastavený účet není možné se přihlásit. Metoda vrací důležité informace o přihlášeném uživateli (id uživatele, přihlašovací jméno, role atd.).

5.2.4 Pozastavení účtu

Proces pozastavení účtu začíná na profilové stránce některé osoby, kde se administrátorům vykresluje tlačítko s možností pozastavení účtu dané osoby (pokud je účet vytvořen a není pozastaven). Po kliknutí na tlačítko se otevře modální okno, ve kterém je vykreslený formulář s jedním vstupem typu *textarea*, který slouží jako důvod pozastavení. Dále obsahuje jedno tlačítko pro odeslání a skrytý parametr `userId`, oznamující o který účet se jedná. Při odeslání tohoto formuláře se odešle AJAXový požadavek do presenteru. Zde se vytáhnou data z formulářové struktury a přepošlou se jako parametry metodě `userSuspension` *v* instanci objektu `UserManager` v souboru *app/model/UserManager.php*. Tato metoda upraví záznam uživatelského účtu tak, že ve sloupci *suspension_date* bude aktuální datum a v *suspension_reason* bude administrátorem zadaný důvod pozastavení.

5.3 Vlastní mezivrstva před DB

V mnoha funkcích napříč celým systémem bylo potřeba předávat informace o entitách, které se nakonec použily pouze v SQL klauzulích `WHERE`. Informace byly různých typů a obtížně se s nimi jednotlivě manipulovalo. Proto jsem vytvořil objekt, kterému na začátku předáme výše zmiňované informace a dále již předáváme pouze tento objekt. Až jsou informace potřeba pro SQL, zavolá se jeho metoda `getSqlCondition`, která vrátí rovnou SQL podmínku. Objekt byl původně využíván pouze na vyhledávání, proto byl pojmenován jako `SqlPattern` a název mu zůstal do dnes. Nachází se v souboru *app/model/DbManager.php*. Je možné mu zadat více podmínek a vybrat si i oddělovací logický operátor pro každou podmínku. Nepodporuje závorkování. Umožňuje zadat i klauzuli `ORDER BY`. Do tohoto objektu se postupně přesouvá obecná část SQL podmínek. Posledním rozšířením byl příznak zobrazení smazaných záznamů (sloupec *deleted*). Původně musel být uveden u všech SQL selektů. Nově stačí instanci `SqlPattern` předat název datové třídy entity, pro kterou je daný selekt určen. Podle konstanty `HAS_COLUMN_DELETED` dané třídy objekt `SqlPattern` zjistí, jestli může tento příznak do SQL dotazu přidat. Není vhodný na složitější dotazy, ale podporuje možnost manuálního zadání části SQL dotazu.

5.4 Kontroly vrhů

Hodnocení vrhu jako celku je implementované jako formulář umístěný v modálním okně, který je definovaný pomocí továrny a vykreslený v souboru *app/presenters/templates/components/createLitterControlModalForm.latte*. Celkově celý formulář je až na pár menších detailů implementačně podobný jako vytváření a editace entit [kapitola 5.1.1]. Při editaci se využívá jQuery, která plní prvky formuláře z proměnné, kterou do šablony odešle presenter v `renderShow` funkci, pro zobrazení celého vrhu. Tato proměnná má hlavní využití při zobrazení celého přehledu o vrhu. Veškeré select prvky formuláře jsou realizovány opět pomocí *select2* rozšíření. Na celkové číselné

hodnocení je využit jQuery *slider*[50], který plní funkci posuvníku. Má nastavený rozsah hodnot od 1 do 10 a výchozí pozici uprostřed tohoto rozsahu.

Hodnocení jednotlivého psa ve vrhu je realizované stejně jako celkové hodnocení, kromě plnění dat. Tento formulář se plní pomocí jQuery, která bere data pomocí AJAX žádosti, kterou obslouží metoda `handleGetDogRating` umístěná *v* souboru *app/model/LitterPresenter.php*. Tato metoda má jako parametr identifikátor hodnoceného psa. JQuery mimo jiné rozhoduje o tom, zda se jedná o hodnocení psa či feny. V případě feny se pomocí atributu *disabled* skryjí políčka na hodnocení varlat.

V backendové části je pro kontrolu vrhů podstatný datový objekt *LitterControl* umístěný v souboru *app/model/LitterManager.php*. Tento objekt obsahuje data pro jednu kontrolu vrhu a metody pro práci s těmito daty. Kontroly vrhů se zobrazují na stránce vrhů. Kvůli tomu je potřeba umět získávat informace o kontrolách podle vrhu, ke kterému patří. K tomu slouží metoda `getControlByLitterId`. Parametrem metody je pole identifikátorů vrhů, ke kterým chceme získat informace o jejich kontrolách. Tato metoda pouze připraví `SearchPattern` a odešle jej privátní metodě `_getControl`. Tato metoda získá veškeré informace o kontrole zadaných vrhů, vloží je do datových objektů a tyto vrátí v poli. Předchozí metoda si toto pole přeindexuje podle identifikátoru vrhu, aby se s tím lépe pracovalo.

Ačkoliv je frontend správy kontrol vrhů rozdělen na více částí (naplánování, zadávání výsledků, editace výsledků), backend je navržen jako jedna část. Této části se dá obecně říkat uložení kontroly. Datová struktura kontroly vrhu je taktéž jedna. Zde záleží čistě na frontendu aplikace, jaké informace si zde v jednotlivých krocích uloží a stejně tak je bude i rozlišovat při zobrazení.

Součástí kontroly vrhů je i kontrola všech jedinců. Tyto informace se vytváří i editují podobně jako ostatní entity [kapitola 5.1.1]. I mazání je shodne [kapitola 5.1.2].

5.5 Správa akcí

Správa akcí je řešena jako každá jiná entita. Vytváření a editace probíhá v modálním okně, které obsahuje formulář definovaný v továrně. Při úpravě či editaci se ve formuláři nachází select box `category`, který je závislý na výběru hodnoty v select boxu *type*. Tato závislost je řešena pomocí jQuery a AJAXového požadavku, který obsluhuje metoda `handleCategoryChange` umístěná v souboru *EventPresenter.php*. Metoda pomocí `redrawControl` a `snippetu` zajistí přepsání select boxu `category` s hodnotami, které odpovídají výběru volby v select boxu `type`.

Výpis veškerých akcí je realizován pomocí cyklu `foreach`, který entity vypíše do tabulkové podoby. Editační formulář se plní pomocí jQuery, která získá data pomocí AJAX požadavku. Ten obslouží metoda `handleGetCompetitor` umístěná v *EventPresenter.php*.

5.6 Krycí listy

Krycí listy se vytváří pomocí modálního okna, které je vložené v šabloně *app/Presenters/templates/Dog/show.latte* a definované v *app/Presenters/templates/components/createCoverSheetModalForm.latte*. Modální okno se vkládá, pouze pokud se jedná o editaci feny, tedy při splnění podmínky v makru `{if $dog->sex != "pes"}`. Formulář je definovaný opět ve své továrně.

[50] Posuvník, který umožňuje zadat hodnoty v daném rozsahu.

Schvalování krycích listů probíhá v administraci, kde jsou veškeré žádosti v šabloně *app/Presenters/templates/CoverSheet/default.latte* pomocí `foreach` cyklu vypsány do tabulkové podoby. Nad těmito daty je vytvořený formulář, pomocí kterého lze vyhledat pouze žádosti odpovídající zadanému řetězci, stejně jako v kapitole 5.1.4.

Při kliknutí na požadovanou žádost se uživatel pomocí jQuery přesměruje na přehled žádosti. Kde jsou do tabulkové podoby vypsané informace o žádosti. Tyto informace dodá presenter `CoverSheetsenterPresenter` metodě `renderShow` s parametrem zobrazené žádosti. Zde lze pomocí modálního okna žádost zamítnout či schválit. Zamítnutí či schválení je řešeno pomocí formuláře. Další možností je editace žádosti, to je také řešeno pomocí modálního okna a formuláře. Toto modální okno se plní hodnotami žádosti již při vykreslení stránky a to pomocí jQuery, která získá data od metody `renderShow`.

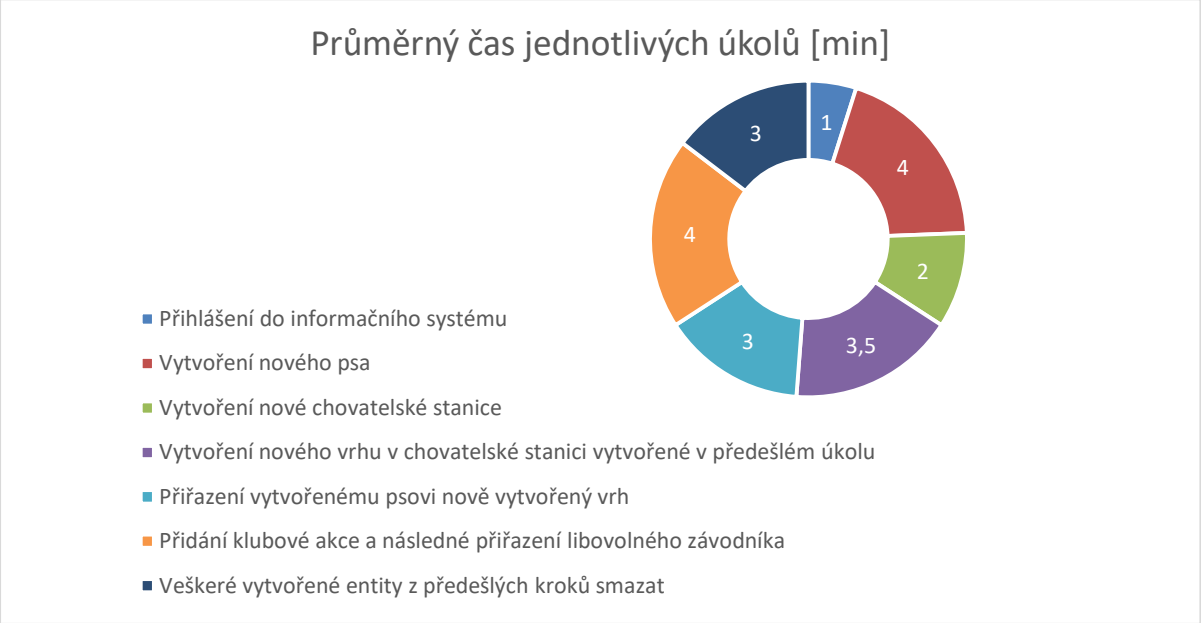
5.7 Statistiky klubu

Statistiky jsou vyobrazené nejprve v tabulkovém zobrazení pomocí HTML tabulky, kde jsou vykreslena data předaná z presenteru `HomepagePresenter` z metody `renderDefault`. Tato metoda také poskytuje data, která využívá jQuery skript na vypsání dat do *Chart.js* grafů. Grafy jsou koláčového typu, ve skriptu definované jako typ `pie`.

Veškeré výpočty získávající data pro grafy jsou prováděny v databázi. SQL dotazy k těmto výpočtům se nachází v souboru *app/model/StatisticManager.php*.

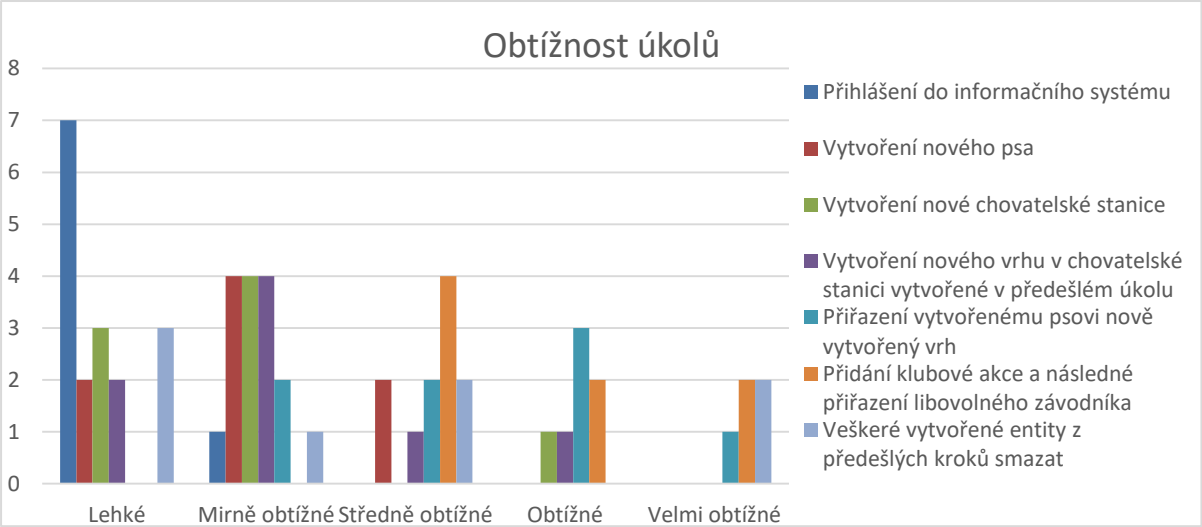
6 Testování

Vytvořil jsem formulář s úkoly a jejich hodnocením, které jsem následně předal celkem osmi lidem, kteří by mohli být reální uživatelé tohoto informačního systému. Věkové rozmezí testovacích uživatelů bylo od 20 do 59 let. Uživatelé dostali adresu webu, kde vypracovali zadané úkoly [Příloha A] a následně své výsledky zaznamenali do formuláře. K testování byl vytvořen uživatelský účet, který měl administrátorská práva. Každý pracoval doma na svém počítači. Neprováděl jsem žádný záznam obrazovek apod., aby se uživatel nemusel stresovat tím, že ho někdo sleduje.



Graf 6.1: Průměrný čas jednotlivých úkolů.

V Graf 6.1 vidíme průměrný čas jednotlivých úkolů. Nejvyšší hodnoty se nachází u úkolů, kde bylo potřeba vytvořit entitu psa, vrhu a klubové akce. Dalo se to předpokládat, protože zde zabere spoustu času vyplnění formulářů. Vytvoření chovatelské stanice zabralo méně času než ostatní vytvářecí úkoly, protože tento formulář obsahuje podstatně méně polí. Smazání celkem 4 různých entit zabralo uživateli průměrně 3 minuty. Zde pravděpodobně hodně zdržuje to, že tlačítka na editaci a smazání jsou defaultně skryty, aby zbytečně nerušily uživatele při procházení a také aby nedocházelo k nechtěným překlikům. Při dalším vývoji bude zvážena úprava tohoto způsobu zobrazení.



Graf 6.2: Graf obtížnosti jednotlivých úkolů.

Graf 6.2 zobrazuje obtížnost jednotlivých úkolů. Vidíme, že přihlášení do systému bylo pro většinu testovacích uživatelů nejlehčím úkolem. Ostatní úkoly odpovídají skutečné složitosti dle jejich zadání. Nejvíce obtížný se uživatelům zdál být úkol na přidání klubové akce a vytvoření jednoho účastníka této akce. Tento dojem si vysvětluji tím, že je tento úkol tvořen ve skutečnosti ze dvou úkolů. Průměrný čas na tomto úkolu z předchozího grafu nevykazuje o mnoho větší hodnoty, než ostatní úkoly, proto by mohlo být moje tvrzení pravdivé. Pří dalším testování tento úkol rozdělím na dva samostatné úkoly. Snad se jejich celkové hodnocení zlepší oproti aktuálnímu výsledku.

Ze sekce připomínky/nápady jsem se dozvěděl pár nedostatků systému, které jsem následně eliminoval. Uvedu zde důležité nedostatky a jejich řešení.

1. **Popis nedostatku:** Položka menu *administrace* měla další podpoložky, ale nebyla nijak zvláštně označena od ostatních položek, které žádné podpoložky neobsahovaly. Jakmile na ni uživatel klikl, byl přesměrován na hlavní administrační stránku a rozbalilo se menu, kde se zobrazily podpoložky administrace. Toto chování dva uživatelé napadli jako velice nepřehledné, že nebylo poznat, že se pod administrací schovávají další položky a chce-li uživatel přejít na nějakou stránku z podpoložek, musí nejprve přejít na hlavní administrační stránku.
   Řešení nedostatku: Toto chování jsem přehodnotil, menu vizuálně upravil tak, aby šlo poznat, že administrace obsahuje podpoložky (přidal k ní šipku). Menu se rozbalí ihned po kliknutí na položku administrace. Stránka, na kterou se původně odkazovalo při kliknutí na administraci, byla z informačního systému vyjmuta. V případě, že bude někdy v budoucnosti potřeba, vytvoří se podpoložka v menu odkazující na tuto stránku.

2. **Popis nedostatku:** Nepřehledná tlačítka na editaci a přidávání entit. Původně byly všechny stejné modré barvy, která podle testovacích uživatelů splývala s menu a vrchní lištou stránky, což způsobovalo, že byly málo výrazné a jednoduše přehlédnutelné.
   Řešení nedostatku: Tuto chybu jsem opravil tím, že jsem tlačítka různě barevně odlišil.

3. **Popis nedostatku:** Veliký nedostatek byl, že se špatně orientuje v řádcích vypsaných dat - přehled všech psů, osob, vrhů…
   Řešení nedostatku: Vyřešil jsem to opět barevným odlišením. Liché řádky těchto výpisů jsou modré barvy a sudé bílé. Bílé řádky se při najetí myší změní na šedé.

7 Závěr

Zadání této práce i základní požadavky reálného systému specifikované v kapitole Analýza a specifikace požadavků byly splněny. Navíc byly implementované nadbytečné možnosti např. správa uživatelských práv, aktivace účtu pomocí aktivačního emailu atd.
Aplikace byla otestována osmi lidmi. Testovací scénář, výsledky a podrobnější informace naleznete v kapitole Testování. Systém byl naplněn testovacími daty pro školní účely. Druhá instance systému byla naplněna skutečnými daty obdrženými od klubu. Tato instance bude v dalším volném čase rozšířena o další data, která si vyžádám od klubu. Následně bude nasazena do provozu.

Před zahájením této práce jsem měl zkušenost s jednotlivých technologiemi (HTML, CSS, PHP a JS). V této práci bylo potřeba použít všechny tyto zkušenosti dohromady, což mi občas způsobilo problémy. V praxi jsem si vyzkoušel práci s Nette frameworkem, který je vhodný především na projekty využívající jeho standardní prvky. Při použití AJAXu a překleslování části stránky pomocí nette snippetů se ukázaly první nedostatky tohoto frameworku. Další problémy nastaly při použití jiných typů inputů ve formulářích, než nabízí Nette. Na druhou stranu nic není neřešitelné, díky těmto problémům jsem se naučil tyto nedostatky obcházet. Ve specifických částech systému je potřeba vynaložit více programovacího úsilí k dosažení požadovaného výsledku. Nette nám toto úsilí vynahradí v klasických částech systému (zobrazení informací, formuláře s vestavěnými typy inputů atd.).

Reálný systému klubu bílého ovčáka nabízí další možnosti rozšíření a nová vylepšení. Při budoucím vývoji je možné implementovat:

- úschovnu rentgenových snímků kyčelních a loketních kloubů psů
- správu vykonaných zkoušek psů s automatických výpočtem bodů v soutěži o nejlepšího pracovního psa roku
- správu umístění psů v závodech a výstavách pořádaných jiným klubem a výpočet bodů v soutěžích
- statistika způsobů úmrtí psů
- notifikace systému (chybějící informace jednotlivých entit, různá upozornění)
- možnost majiteli některé entity navrhnout změnu informací o entitě a umožnit tuto změnu aplikovat administrátorem po ověření správnosti

Po otestování aplikace v ostrém provozu bych chtěl rozšířit systém pro další chovatelské kluby nebo případně i jiné kluby zaměřené na kynologii. Vytvořit tak skupinu informačních systémů sdílejících některé informace nebo jeden systém s odděleným přístupem (dle požadavků organizací).

Literatura

[1] WWW stránky: MySQL : MySQL Documentation. https://dev.mysql.com/doc/.

[2] WWW stránky: Learn - MariaDB.org. https://mariadb.org/learn/.

[3] WWW stránky: ASP.NET overview | Microsoft Docs. https://docs.microsoft.com/cs-cz/aspnet/overview.

[4] Miller, David: Start Bootstrap - Simple Sidebar [online]. https://github.com/BlackrockDigital/startbootstrap-simple-sidebar, [cit. 2017-01-12].

[5] WWW stránky: jQuery. https://jquery.com/.

[6] WWW stránky: Zend Framework. https://framework.zend.com/.

[7] DOBEŠ, Vojtěch: Nette.ajaj.js | For Nette framework [online]. https://github.com/vojtech-dobes/nette.ajax.js. [cit. 2017-02-04].

[8] WWW stránky: Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework. https://nette.org/.

[9] JavaScript Krok za krokem [online]. Brno: Computer Press, 2008 [cit. 2017-04-04]. ISBN 978-80-251-2241-9. Dostupné z: https://knihy.abz.cz/imgs/teaser_pdf/4449788025122419.pdf.

[10] KOSEK, Jiří. Kaskádové styly. Dostupné z: http://www.kosek.cz/vyuka/4iz228/prednasky/css.pdf. [cit. 2017-03-28]

[11] KOSEK, Jiří. PHP: tvorba interaktivních internetových aplikací : podrobný průvodce. Praha: Grada, c1999. ISBN 80-716-9373-1.

[12] WWW stránky: PHP: Hypertext Preprocessor. http://php.net/.

[13] WWW stránky: Select2 - The jQuery replacement for select boxes. https://select2.github.io/.

[14] WWW stránky: Chart.js | Documentation. http://www.chartjs.org/docs/.

[15] WWW stránky: Bootstrap · The world's most popular mobile-first and responsive front-end framework. http://getbootstrap.com/.

[16] WWW stránky: The Web framework for perfectionists with deadlines | Django. https://www.djangoproject.com/.

[17] WWW stránky: Ruby on Rails | A web-application framework that includes everything needed to create database-backend web applications according to the Model-View-Controller (MVC) pattern. http://rubyonrails.org/.

Příloha A

Testovací protokol

Testovací protokol k informačnímu systému klubu chovatelů a psů

Jméno a příjmení:

Přihlašovací údaje do systému: *test@example.com*
Heslo: *test*

Jednotlivé úkoly: (Svou volbu zakroužkujte, veškeré údaje do systému si můžete vymyslet)

Přihlášení do informačního systému:

Lehké Mírně obtížné Středně obtížné Obtížné Velmi obtížné

Časová obtížnost (min):

Vytvoření nového psa:

Lehké Mírně obtížné Středně obtížné Obtížné Velmi obtížné

Časová obtížnost (min):

Vytvoření nové chovatelské stanice:

Lehké Mírně obtížné Středně obtížné Obtížné Velmi obtížné

Časová obtížnost (min):

Vytvoření nového vrhu v chovatelské stanici vytvořené v předešlém úkolu:

Lehké Mírně obtížné Středně obtížné Obtížné Velmi obtížné

Časová obtížnost (min):

Přiřazení nově vytvořeného vrhu některému psovi:

Lehké Mírně obtížné Středně obtížné Obtížné Velmi obtížné

Časová obtížnost (min):

Přidání klubové akce a následné přiřazení libovolného závodníka:

Lehké Mírně obtížné Středně obtížné Obtížné Velmi obtížné

Časová obtížnost (min):

Veškeré vytvořené entity z předešlých kroků smazat:

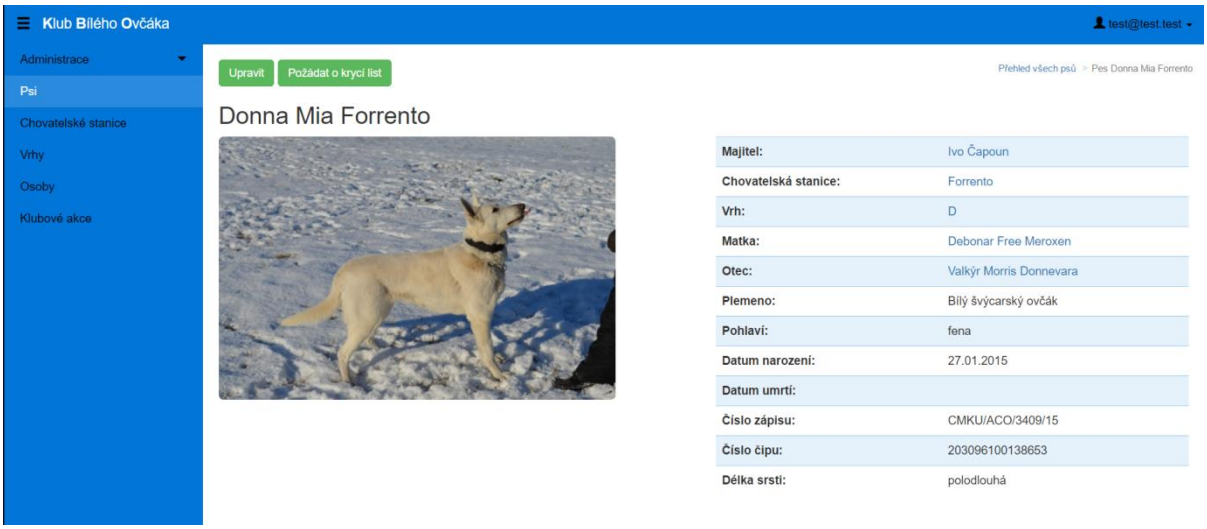
Lehké Mírně obtížné Středně obtížné Obtížné Velmi obtížné

Časová obtížnost (min):

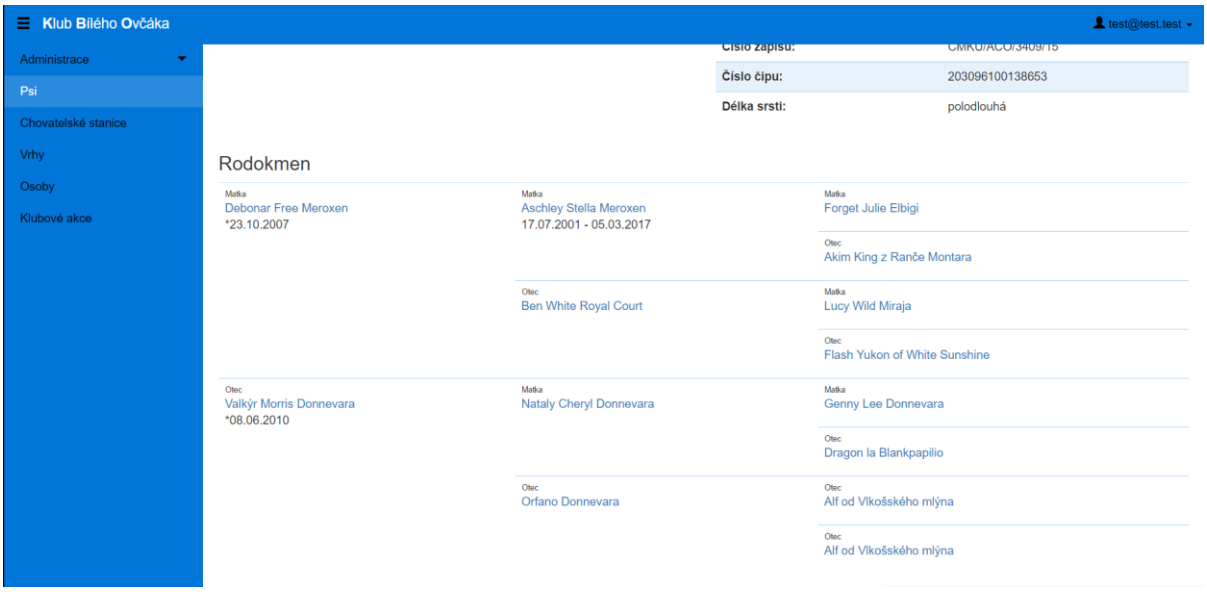
Připomínky/nápady:

Příloha B

Náhledy aplikace



Obrázek B.1: Náhled profilové stránky psa.



Obrázek B.2: Náhled rodokmenu psa.