



## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

<b>Název:</b>	Databáze zvířat pro neziskovou organizaci
<b>Student:</b>	Tomáš Taro
<b>Vedoucí:</b>	Ing. Lukáš Maleček
<b>Studijní program:</b>	Informatika
<b>Studijní obor:</b>	Webové a softwarové inženýrství
<b>Katedra:</b>	Katedra softwarového inženýrství
<b>Platnost zadání:</b>	Do konce zimního semestru 2020/21

### Pokyny pro vypracování

Cílem práce je vytvořit novou verzi webové aplikace Plemenná kniha (databázi zvířat), která slouží k evidenci informací o zvířatech v aktivním chovu, evidenci odchovů, schvalování vrhů, pomáhá členům neziskové organizace v plánování chovu a umožňuje jednoduché generování průkazů původu pro odchovy.

- \* Analyzujte současný stav aplikace, seznamte se s její vnitřní implementací, současnou strukturou databáze a již implementovanými funkcemi.
- \* Na základě analýzy navrhnete optimálnější strukturu databáze (zejména vyřešte současné duplicitní ukládání dat).
- \* Vytvořte zcela novou aplikaci pro správu plemenné knihy v PHP s použitím MySQL databáze. V aplikaci využijte současné návrhové a architektonické vzory a postupy.
- \* Implementujte nové funkce na základě jejich specifikace dodaných vedoucím práce.
- \* Ověřte správnou funkci aplikace pomocí unit a případně i integračních testů.
- \* Migrujte data z původní aplikace do nové a navrhnete postup, jak zajistit kontrolu a ověření správnosti importu.

### Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.  
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.  
děkan

V Praze dne 21. února 2019





**FAKULTA  
INFORMAČNÍCH  
TECHNOLÓGIÍ  
ČVUT V PRAZE**

Bakalárska práca

## **Databáza zvierat pre neziskovú organizáciu**

*Tomáš Taro*

Katedra softwarového inžénýrství  
Vedúci práce: Ing. Lukáš Maleček

1. júna 2020





---

## Pod'akovanie

Touto cestou by som sa chcel poďakovať pánovi Ing. Lukášovi Malečkovi za vedenie a trpezlivosť pri riešení problematiky týkajúcej sa tejto bakalárskej práce. Taktiež by som sa chcel poďakovať rodine za neustálu podporu počas celého štúdia.



---

# Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 1. júna 2020

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2020 Tomáš Taro. Všetky práva vyhradené.

*Táto práca vznikla ako školské dielo na FIT ČVUT v Prahe. Práca je chránená medzinárodnými predpismi a zmluvami o autorskom práve a právach súvisiacich s autorským právom. Na jej využitie, s výnimkou bezplatných zákonných licencií, je nutný súhlas autora.*

### **Odkaz na túto prácu**

Taro, Tomáš. *Databáza zvierat pre neziskovú organizáciu*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

---

# Abstrakt

Náplňou tejto bakalárskej práce je analýza súčasnej webovej aplikácie používanej pre evidenciu zvierat, vrhov a ich odchovov neziskovou organizáciou „Český klub potkanů“. Na základe tejto analýzy je navrhnutá a implementovaná nová webová SPA aplikácia s použitím backendového frameworku Laravel a frontendového frameworku Vue.js. Komunikácia medzi klientom a serverom prebieha pomocou REST API. V rámci práce bola navrhnutá optimálnejšia štruktúra súčasnej databázy, ktorá okrem iného rieši duplicitné ukladanie dát. Správnosť funkcionality výslednej aplikácie je overená integračnými testami.

**Kľúčová slova** Webová aplikácia, responzívny dizajn, SPA, Laravel, Vue, Normalizácia databázy, MySQL, REST API, Integračné testy

---

# Abstract

The aim of this paper is the analysis of the current version of web application used as a database for storing records of animals, litters and their breeding by non-profit organisation „Český klub potkanů“. Based on this analysis, new web SPA application was designed and implemented, with the use of the backend framework Laravel and frontend framework Vue.js. Communication between client and server takes place with the help of REST API. Within the process, more optimal structure was created for current database, which among other things solves the problem of saving duplicities of the same data. Correctness of the functionality for the final application is verified with various integration tests.

**Keywords** Web application, Responsive design, SPA, Laravel, Vue, Database normalisation, MySQL, REST API, Integration tests

---

# Obsah

Úvod	1
<b>1 Analýza</b>	<b>3</b>
1.1 Analýza súčasnej aplikácie . . . . .	3
1.1.1 Architektúra aplikácie . . . . .	3
1.1.2 Databázový model aplikácie . . . . .	5
1.2 Analýza požiadaviek . . . . .	11
1.2.1 Funkčné požiadavky . . . . .	11
1.2.2 Nefunkčné požiadavky . . . . .	15
1.3 Používateľské role . . . . .	16
1.3.1 Bežný používateľ . . . . .	16
1.3.2 Registrátor zvierat . . . . .	16
1.3.3 Schvaľovateľ vrhov . . . . .	17
1.3.4 Administrátor . . . . .	17
1.4 Prípady použitia . . . . .	18
1.4.1 Prípady použitia zvierat . . . . .	18
1.4.2 Prípady použitia vrhov . . . . .	21
<b>2 Návrh aplikácie</b>	<b>23</b>
2.1 Návrh architektúry aplikácie . . . . .	23
2.1.1 Architektúra MPA aplikácie . . . . .	23
2.1.2 Architektúra SPA aplikácie . . . . .	25
2.1.3 Voľba architektúry . . . . .	27
2.2 Voľba technológií . . . . .	28
2.2.1 PHP . . . . .	28
2.2.2 HTML 5 . . . . .	28
2.2.3 CSS 3 . . . . .	29
2.2.4 JavaScript . . . . .	29
2.3 Návrh databázy . . . . .	30

2.3.1	Normalizácia databázy . . . . .	30
2.3.2	Návrh databázového modelu . . . . .	32
<b>3</b>	<b>Implementácia</b>	<b>37</b>
3.1	Serverová časť . . . . .	37
3.1.1	Štruktúra projektu . . . . .	37
3.1.2	MVC architektúra . . . . .	38
3.2	Klientská časť . . . . .	42
3.3	Migrácia dát . . . . .	46
<b>4</b>	<b>Testovanie</b>	<b>49</b>
4.1	Jednotkové testy . . . . .	49
4.2	Integračné testy . . . . .	49
4.3	Overenie správnosti importu . . . . .	51
	<b>Záver</b>	<b>53</b>
	<b>Literatúra</b>	<b>55</b>
<b>A</b>	<b>Zoznam použitých skratiek</b>	<b>57</b>
<b>B</b>	<b>Používateľské rozhranie</b>	<b>59</b>
<b>C</b>	<b>Obsah priloženého CD</b>	<b>87</b>



---

## Zoznam obrázkov

1.1	Ukážka súboru vrhy_detail_zvirete.php . . . . .	4
1.2	Logický model súčasnej databázy . . . . .	10
2.1	Diagram znázorňujúci architektúru MPA aplikácií . . . . .	24
2.2	Diagram znázorňujúci architektúru SPA aplikácií . . . . .	25
2.3	Logický model navrhutej databázy . . . . .	36
3.1	Ukážka modelovej triedy Animal.php . . . . .	38
3.2	Ukážka šablóny animal_overview.blade.php . . . . .	40
3.3	Ukážka controlleru AnimalController.php . . . . .	41
3.4	Ukážka routeru web.php . . . . .	42
3.5	Ukážka šablóny komponentu AnimalPage.vue . . . . .	43
3.6	Ukážka biznis logiky AnimalPage.vue komponentu . . . . .	44
3.7	Ukážka volania import metód v routri web.php . . . . .	46
4.1	Ukážka integračného testu zmazania používateľa . . . . .	50
B.1	Screenshot prihlasovacej obrazovky . . . . .	59
B.2	Screenshot aplikácie po prihlásení s otvoreným menu . . . . .	60
B.3	Screenshot aplikácie po prihlásení s otvoreným dropdown menu . . . . .	60
B.4	Screenshot načítavania zvoleného zvieraťa . . . . .	61
B.5	Screenshot načítaného zvoleného zvieraťa . . . . .	61
B.6	Screenshot zobrazenia rodokmeňu zvoleného zvieraťa . . . . .	62
B.7	Screenshot zobrazenia poznámok a histórie zvoleného zvieraťa . . . . .	62
B.8	Screenshot zobrazenia sekcie editácie zvieraťa . . . . .	63
B.9	Screenshot zobrazenia ponuky voľby vrhu pri editácii zvieraťa . . . . .	63
B.10	Screenshot aplikácie po zvolení vrhu pri editácii zvieraťa . . . . .	64
B.11	Screenshot aplikácie zobrazujúci možnosť voľby chovateľa . . . . .	64
B.12	Screenshot modálneho okna s ponukou vytvorenia novej osoby . . . . .	65
B.13	Screenshot zobrazenia nasledujúcej sekcie pri editácii zvieraťa . . . . .	65
B.14	Screenshot zobrazenia poslednej sekcie pri editácii zvieraťa . . . . .	66

B.15 Screenshot zobrazenia modálneho okna pri zvolení zmazania zvieráťa	66
B.16 Screenshot zobrazenia poznámky po kliknutí na tlačidlo pridania novej poznámky . . . . .	67
B.17 Screenshot zobrazenia viditeľnosti poznámky . . . . .	67
B.18 Screenshot zobrazenia typov poznámky . . . . .	68
B.19 Screenshot aplikácie po vytvorení poznámky . . . . .	68
B.20 Screenshot aplikácie pri pokuse vymazať poznámku . . . . .	69
B.21 Screenshot aplikácie zobrazujúci nasledujúcu sekciu zvieráťa . . . .	69
B.22 Screenshot aplikácie zobrazujúci proces pridania chybnjej registrá- cie zvieráťa . . . . .	70
B.23 Screenshot aplikácie zobrazujúci proces pridania správnej regsitrá- cie zvieráťa . . . . .	70
B.24 Screenshot aplikácie zobrazujúci registráciu zvieráťa po jej pridaní	71
B.25 Screenshot modálneho okna aplikácie pri ostránení registrácie . . .	71
B.26 Screenshot zobrazujúci prvú stranu vygenerovaného PDF súbor zvieráťa . . . . .	72
B.27 Screenshot zobrazujúci druhú stranu vygenerovaného PDF súbor zvieráťa . . . . .	72
B.28 Screenshot zobrazujúci tabuľku všetkých vrhov . . . . .	73
B.29 Screenshot aplikácie zobrazujúci načítavanie zvoleného vrhu . . . .	73
B.30 Screenshot aplikácie zobrazujúci načítaný vrh . . . . .	74
B.31 Screenshot aplikácie zobrazujúci dodatočné informácie o vrhu . . .	74
B.32 Screenshot aplikácie zobrazujúci rodokmeň vrhu . . . . .	75
B.33 Screenshot aplikácie zobrazujúci ďalšiu sekciu informácií o vrhu . .	75
B.34 Screenshot aplikácie zobrazujúci sekciu editácie vrhu . . . . .	76
B.35 Screenshot aplikácie zobrazujúci ďalšiu sekciu informácií o vrhu . .	76
B.36 Screenshot aplikácie zobrazujúci poslednú sekciu editácie vrhu . . .	77
B.37 Screenshot aplikácie zobrazujúci modálne okno poslania žiadosti o schválenie vrhu . . . . .	77
B.38 Screenshot aplikácie zobrazujúci poslanú žiadosť o schválenie vrhu	78
B.39 Screenshot aplikácie zobrazujúci modálne okno odpovedi na po- slanú žiadosť . . . . .	78
B.40 Screenshot aplikácie zobrazujúci modálne okno vyplnenej odpovedi na poslanú žiadosť . . . . .	79
B.41 Screenshot aplikácie zobrazujúci zamietnutú žiadosť . . . . .	79
B.42 Screenshot aplikácie zobrazujúci všetkých používateľov aplikácie .	80
B.43 Screenshot aplikácie zobrazujúci nastavenie rolí používateľovi ap- likácie . . . . .	80
B.44 Screenshot zobrazujúci menu aplikácie v mobilnom prehliadači . .	81
B.45 Screenshot zobrazujúci zoznam zvierat v mobilnom prehliadači . .	82
B.46 Screenshot zobrazujúci načítavanie zvieráťa v mobilnom prehliadači	83
B.47 Screenshot zobrazujúci načítanie zviera v mobilnom prehliadači . .	84
B.48 Screenshot zobrazujúci detail zvieráťa v mobilnom prehliadači . . .	85
B.49 Screenshot poslednej sekcie zvieráťa v mobilnom prehliadači . . . .	86

---

# Úvod

Používanie webových aplikácií sa stalo bežnou súčasťou našich životov vďaka rozšírenému používaniu mobilných zariadení schopných pripojiť sa na internet. Ich používatelia nie sú len bežní ľudia, ale aj záujmové organizácie, medzi ktoré taktiež patrí organizácia „Český klub potkanů“. Táto organizácia používa webovú aplikáciu pre evidenciu informácií o zvieratách v aktívnom chove a ich vrhoch, pomáha jej členom v plánovaní odchovov a umožňuje pre ne generovať jednoduché preukazy. Avšak problémom existujúcej aplikácie je jej zastaranosť, ťažkopádnosť pri jej používaní a použitie zlých návrhových vzorov pri jej vytváraní.

Táto bakalárska práca sa zaoberá vytvorením novej webovej aplikácie, ktorá nahradí starú aplikáciu a tým uľahčí organizácii evidenciu jednotlivých informácií.

Rešeršná časť práce sa venuje analýzou existujúcej aplikácie z pohľadu architektúry aplikácie a jej databázy. Na túto časť nadviaže analýza požiadaviek s popisom prípadov použitia, ktoré poslúžia ako základ implementácie jednotlivých funkcií. Následne sa práca zaoberá zvolením vhodnej architektúry potrebnej pre beh aplikácie, technológií, ktoré aplikácia bude používať a návrhom databázy, ktorá vyrieši existujúci problém ukladania duplicitných dát.

Implementačná časť práce sa venuje procesu implementácie aplikácie z pohľadu serverovej a klientskej časti. Následne pokračuje opisom procesu migrácie dát z existujúcej databázy do novej databázy navrhutej pre novú aplikáciu.

## ÚVOD

---

Práca sa taktiež venuje testovaniu aplikácie pomocou integračných testov, ktoré overia, či jej implementácia prebehla korektne podľa požiadaviek definovaných v analytickej časti práce. V neposlednom rade sa práca zaoberá kontrolou importovaných existujúcich dát do databázy implementovanej aplikácie.

---

# Analýza

Táto kapitola sa venuje analýze súčasnej webovej aplikácie [1], ktorá zahŕňa analýzu jej architektúry a databázového modelu. Následne sú vymedzené jednotlivé funkčné a nefunkčné požiadavky novej webovej aplikácie, na ktoré nadväzuje sekcia používateľských rolí. Záver kapitoly je určený pre analýzu prípadov použitia, ktoré vyplývajú z funkčných požiadaviek.

## 1.1 Analýza súčasnej aplikácie

Táto sekcia sa zaoberá analýzou súčasnej aplikácie [1], ktorá bola vytvorená pre potreby neziskovej organizácie viesť evidenciu zvierat, vrhov, ich registrácií a ďalších informácií. Hlavným cieľom analýzy je zistenie súčasného riešenia aplikácie z pohľadu architektúry a štruktúry súčasnej databázy.

### 1.1.1 Architektúra aplikácie

Samotný kód aplikácie nie je členený – skladá sa z jednej vrstvy – čo znamená, že kód aplikácie zodpovedný za prístup do databázy, logiku aplikácie a zobrazenie aplikácie nie je od seba oddelený.

## 1. ANALÝZA

---

```
1  <?php
2  get_header();
3  if (!is_user_logged_in()) {
4      echo (<h1>Neautorizováno</h1>\n");
5  } else {
6      ?>
7  <h1>Detail zvířete</h1>
8  <br>
9  <?php
10     if (!empty($_REQUEST['id'])) {
11         $req_id = $_REQUEST['id'];
12         if (is_numeric($req_id)) {
13             $sql_potkan = "SELECT * FROM `". $tabulka_zvirata . "`
14             WHERE `id_prvni_verze` = " . $req_id .
15             " ORDER BY `id` DESC";
16             $potkani = $wpdb->get_results(
17                 $sql_potkan, ARRAY_A, 0
18             );
19             $zvire = $potkani[0];
20             if ($zvire != null) {
21                 ?>
22                 <div id="zvire">
23                     <fieldset style="
24                     border: 1px solid #A0A0A0;
25                     margin: 2px;
26                     padding: 5px;">
27                         <legend style="padding: 5px;">
28                             Základní informace
29                         </legend>
30                         <span id="popis">Pohlaví:</span><span id="hodnota">
31                             <?php
32                                 if ($zvire['pohlavi'] == 'M') {
33                                     echo ("kluk");
34                                 }
35                                 else if ($zvire['pohlavi'] == 'F') {
36                                     echo ("holka");
37                                 }
38                                 else {
39                                     echo ("????");
40                                 }
41                             ?>
42                         </span><br>
43                         ...
44                     </fieldset>
45                 </div>
```

4 Obr. 1.1: Ukážka súboru vrhy\_detail\_zvirete.php

Obrázok 1.1 obsahuje ukážku súboru súčasnej aplikácie, ktorý je zodpovedný za zobrazenie detailov zvieráťa v danom vrhu. Z ukážky je možné postrehnúť vyššie spomenuté miešanie logiky zodpovednej za funkcionálnosť aplikácie (riadok 1–6), kód zodpovedný za prístup do databázy (riadok 10–19) a v neposlednom rade logiku zobrazovania aplikácie (riadok 21–41).

Ako programovací jazyk súčasnej aplikácie bol použitý jazyk PHP, ktorý bol navrhnutý v roku 1994 Rasmusom Lerdorfom. Tento jazyk je univerzálny programovací jazyk na strane servera a je primárne určený na vývoj webových stránok [2] (vlastný preklad).

Pre pridávanie, spracovanie a získavanie dát v aplikácii slúži open-source SQL databázový systém nazývaný MySQL, ktorý je vyvíjaný, distribuovaný a podporovaný spoločnosťou Oracle Corporation [3] (vlastný preklad).

### 1.1.2 Databázový model aplikácie

Neoddeliteľnou súčasťou tejto práce je analýza databázového modelu existujúcej aplikácie. Na základe tejto analýzy bude navrhnutá taká štruktúra databázy, ktorá nebude spôsobovať nekonzistentnosť a duplicitu dát v aplikácii. V tejto podsekcii sú popísané jednotlivé tabuľky, ich štruktúra a najmä význam stĺpcov, do ktorých sa jednotlivé dáta ukladajú. Databáza súčasnej aplikácie sa skladá z tabuliek

`czkp_mimi`, `czkp_vrh`, `czkp_zvire`, `pp_informace`, `pp_miminka` a `pp_zadosti`.

Pre lepšiu predstavu štruktúry databázového modelu aplikácie sa v podsekcii 1.1.2.7 nachádza logický model databázy v grafickej podobe.

Všetky tabuľky obsahujú umelo-vytvorený unikátny primárny kľúč záznamu zvaný `id`, podľa ktorého sa rozlišujú jednotlivé záznamy.

V nasledujúcich dvoch tabuľkách (`czkp_vrh` a `czkp_zvire`) sa vyskytujú tri rovnaké stĺpce, menovite `id_prvni_verze`, `datum_zaznamu` a `uzivatel`. V stĺpci `id_prvni_verze` je uložený identifikátor (`id`) prvej verzie zvieráťa/vrhu. Stĺpec `datum_zaznamu` obsahuje dátum vytvorenia záznamu spolu s jeho časom a v stĺpci `uzivatel` sa nachádza identifikátor používateľa, ktorý daný záznam vytvoril. Tento identifikátor ukazuje na systémovú tabuľku `wp_users`, ktorá obsahuje informácie o používateľoch súčasného systému.

#### 1.1.2.1 Tabuľka `czkp_vrh`

Táto tabuľka má za úlohu správu dát týkajúcich sa jednotlivých vrhov. Stĺpec `id_prvni_verze` odkazuje na prvý záznam daného vrhu.

V stĺpci `wp_id_majitel` je uložený identifikátor majiteľa vrhu ukazujúci na systémovú tabuľku `wp_users`. Stĺpce `otec` a `matka` slúžia k ukladaniu identifikátorov otca, resp. matky. Tento identifikátor pochádza z tabuľky `czkp_zvire`. Následne sú do tejto tabuľky ukladané aj údaje ako označenie vrhu (v stĺpci `oznaceni`), typ vrhu (`typ_priznani`), línia vrhu (`linie`), gény vrhu (`geny_vrhu`), jeho varietnosť (`varietnost`) či dátum narodenia

(`datum_narozeni`). U každého vrhu sa taktiež zaznamenáva chovná stanica (`chov_stanice`), v ktorej sa daný vrh narodil, prípadne kontakt na chovateľa, ktorý je uložený v stĺpci `chov_kontakt`. Dôležité informácie, ako počet narodených a odchovaných mláďat sa ukladajú do stĺpcov `nar_mladat`, prípadne `odchov_mladat`. Počet odchovaných mláďat sa delí na počet odchovaných samcov (`odch_kluci`) a samíc (`odch_holky`). Počet mláďat uvoľnených pre chov je možné nájsť v stĺpci `mlad_chovne`. Rozdiel odchovaných mláďat a mláďat uvoľnených pre chov sa rovná počtu mláďat určených na maznáčika — tento údaj je možné nájsť v stĺpci `mlad_pet`.

Jednotlivé vrhy môžu byť zaregistrované pod klubom ČKP — tieto registrácie sa taktiež nachádzajú v tabuľke `czkp_vrh`. Pre účely registrácie vrhu slúžia stĺpce `datum_registrace`, v ktorom je uložený dátum registrácie, `reg_cislo_vrhu`, ktorý obsahuje registračné číslo vrhu a `rok_reg`, ktorý znamená, v ktorom roku bol vrh zaregistrovaný. Registrátor, ktorý zaregistroval daný vrh, môže pripojiť poznámku počas registrácie vrhu, ktorá je uložená v stĺpci `poznamka_reg`.

Chovateľ daného vrhu môže taktiež vytvoriť poznámku k vrhu — táto poznámka sa následne uloží do stĺpca `poznamka_chov`.

Ako posledný stĺpec nachádzajúci sa v tabuľke `czkp_vrh` je stĺpec `kompletni`, ktorý znamená, či o uloženom vrhu sú dostupné kompletne informácie, avšak tento údaj sa v aplikácii nepoužíva.

### 1.1.2.2 Tabuľka `czkp_zvire`

Tabuľka `czkp_zvire` slúži na ukladanie informácií o jednotlivých zvieratách, ktoré sú sledované organizáciou.

Skladá sa zo stĺpca `id_prvni_verze`, ktorý odkazuje na prvý záznam daného zvierata v rovnakej tabuľke – čo znamená, že pomocou tohto stĺpca sa dajú zistiť všetky úpravy daného vrhu spustením príslušného SQL príkazu. Jej obsahom je taktiež stĺpec `datum_zaznamu`, ktorý je nastavený na dátum a čas vloženia záznamu do tabuľky. Stĺpec `uzivatel` obsahuje identifikátor používateľa, ktorý daný záznam vytvoril. Následne sa v tabuľke nachádzajú stĺpce `uzamceny_upravy`, `uzamceny_kdy` a `uzamceni_kdo`, ktoré sa ale v súčasnej aplikácii nevyužívali. Pri zvierati je potrebné ukladať jeho pohlavie – to je uložené v stĺpci `pohlavi`. Dátum narodenia zvierata sa nachádza v stĺpci `datum_narozeni`.

Informácie o chovateľovi, resp. majiteľovi zvierata sa ukladajú do šiestich stĺpcov (3 a 3). V stĺpci `chovatel_ckp_id` je uložený identifikátor chovateľa (z tabuľky `wp_users`), v `chovatel` meno chovateľa a v `chov_stanice` chovateľská stanica. Informácie o majiteľovi majú podobnú štruktúru, s tým rozdielom že sa dáta ukladajú do stĺpcov `majitel_ckp_id`, `majitel` a `majet_stanice`. V prípade, že chovateľ resp. majiteľ nepochádza z organizácie, jeho identifikátor zostáva prázdny.

Vonkajšie črty zvierata, ako farba očí, typ uší, typ srsti, znaky a farba srsti



sa ukladajú do stĺpcov `barva_oci`, `typ_ucha`, `typ_srsti`, `bila_kresba` a `barva_srsti`. Pre ukladanie otca a matky zvierata slúžia stĺpce `matka` a `otec`, v ktorých sa nachádza jedinečný identifikátor nachádzajúci sa v tejto tabuľke. Pre účely zaznamenania, z akého vrhu dané zviera pochádza, slúži stĺpec `cis_vrhu`, ktoré obsahuje registračné číslo vrhu, z ktorého dané zviera pochádza. Avšak pre jednoznačnú identifikáciu vrhu, odkiaľ zviera pochádza, mal slúžiť stĺpec `id_ckpt_vrhu` — ten sa ale nepoužíva.

Pre potreby organizácie sa do tejto tabuľky ukladajú údaje o registrácii zvierata — a to typ registrácie (stĺpec `reg_ckpt_typ`), číslo registrácie (`reg_ckpt_cislo`) a rok registrácie (stĺpec `reg_ckpt_rok`). Toto platí iba v prípade, že zviera je zaregistrované pod klubom ČKP. Ak je zviera zaregistrované pod iným klubom, tak sa jeho registračné číslo ukladá do stĺpca `reg_c_ostatni`. Navyše sa ukladá aj identifikátor registrátora, ktorý dané zviera zaregistroval pod klubom ČKP do stĺpca `registrator`. V neposlednom rade je nutné ukladať aj dátum registrácie zvierata — pre túto informáciu slúži stĺpec `datum_registrace`.

Informácie o dátume úmrtia a dôvodu úmrtia zvierata sa ukladajú do stĺpcov `datum_umrti`, resp. do stĺpca `duvod_umrti`.

Chovateľ zvierata si taktiež môže pridať poznámku k zvieratu, ktorá je následne uložená v stĺpci `poznamka_chovatel`.

Medzi ďalšie informácie, ktoré sa zbierajú o zvierati, patrí aj informácia ohľadom rizikovosti chovu (stĺpec `rizikovost_chovu`) a prípadná poznámka, prečo je chov rizikový. Táto poznámka sa ukladá samostatne do stĺpca `riziko_pozn`. Predposledný stĺpec `overeno_pk` obsahuje informáciu, či zviera bolo overené podľa plemennej knihy a stĺpec `pozn_edit` obsahuje poznámku, ktorá mohla byť vytvorená pri editácii zvierata.

### 1.1.2.3 Tabuľka `czkp_mimi`

Úlohou tejto tabuľky je ukladať informácie o mláďatách narodených v jednotlivých vrhoch. Rozdiel medzi tabuľkou `czkp_mimi` a `czkp_zvire` spočíva v tom, že na rozdiel od tabuľky `czkp_zvire`, v tejto tabuľke sú uložené iba mláďatá, ktoré pochádzajú zo známeho vrhu.

Tabuľka obsahuje stĺpec `id_vrhu`, v ktorom je uložený identifikátor záznamu z tabuľky `czkp_vrh`. Podľa tohto stĺpca vieme určiť, v akom vrhu sa dané mláďa narodilo. Pre ukladanie mena a pohlavia mláďata slúžia stĺpce `jmeno`, resp. `pohlavi`. Chovnosť mláďata sa ukladá do stĺpca `chov`. Stĺpec `chov_omez` ďalej slovne špecifikuje chovné obmedzenie mláďata. Pre organizáciu je potrebné sledovať, kto dané mláďa chová, a pre tento účel slúži stĺpec `chov_kdo`. Pre uloženie dodatočných informácií ohľadom mláďata, ako napríklad typ uší, typ srsti, farba srsti, farba očí a znaky slúžia stĺpce `typ_ucha`, `typ_srsti`, `barva_srsti`, `barva_oci` a `znaky`.

### 1.1.2.4 Tabuľka `pp_informace`

Do nasledujúcej tabuľky — `pp_informace` — sa ukladajú informácie spojené s jednotlivými vrhmi. Táto tabuľka nemá žiadnu spojitosť s tabuľkou `czkp_vrh`. Dôvod vzniku tejto tabuľky bol vývoj nadstavby nad pôvodnou aplikáciou.

Táto tabuľka obsahuje stĺpec `uzivatel`, ktorý obsahuje identifikátor majiteľa vrhu ukazujúci do tabuľky `wp_users`. Následne sa do tabuľky ukladajú údaje o samotnom vrhu, ako napríklad typ vrhu (`typ_vrhu`), jeho označenie (`vrh_oznaceni`), dátum narodenia (`vrh_narozeni`), varietnosť (`vrh_varietnost`), počet narodených a odchovaných mláďat (`vrh_nar_mladat/vrh_odch_mladat`) a počet odchovaných samcov a samíc (`vrh_odch_kluci/vrh_odch_holky`).

Každý vrh má matku a otca vrhu — tieto údaje sú ukladané do stĺpcov `vrh_id_matka` a `vrh_id_otec`. Ich obsahom sú identifikátory rodičov daného vrhu. Títo rodičia pochádzajú z tabuľky `czkp_zvire`.

Informácie o chovateli vrhu sa ukladajú do dvoch stĺpcov, a to `chovatel` a `chov_kontakt`. Prvý z menovaných stĺpcov obsahuje meno chovateľa a druhý jeho kontakt.

Nakoľko každý vrh môže byť zaregistrovaný, je nutné ukladať údaje o ich registrácii. Pre tieto účely slúžia stĺpce `reg_dat_vyplneni`, ktorý znamená dátum požiadania o registráciu majiteľom vrhu, `reg_dat_schvaleni`, ktorý obsahuje dátum schválenia vrhu a `reg_cislo_vrhu` obsahujúci registračné číslo vrhu.

### 1.1.2.5 Tabuľka `pp_miminka`

Obsahom tabuľky `pp_miminka` sú mláďatá, ktoré boli narodené vo vrhoch uložených v predchádzajúcej tabuľke.

Aby bolo možné zistiť, ku ktorému vrhu jednotlivé mláďa patrí, bolo potrebné, aby tabuľka obsahovala stĺpec `id_pp`, ktorý obsahuje identifikátor vrhu, v ktorom sa mláďa narodilo. Tento identifikátor ukazuje do predchádzajúcej tabuľky, `pp_informace`. Následne sa do tabuľky ukladajú dáta týkajúce sa predovšetkým samotného mláďata, ako napríklad jeho meno (`mimi_jmeno`), pohlavie (`mimi_pohlavi`), typ uší (`mimi_typ_ucha`), typ srsti (`mimi_typ_srsti`), farba srsti (`mimi_barva_srsti`), či farba očí (`mimi_barva_oci`) a jeho znaky (`mimi_znaky`). K danému zvieraťu taktiež prislúcha majiteľ, ktorý sa ukladá do stĺpca `mimi_majitel`. Okrem mena majiteľa sa ukladá nielen jeho kontakt (`mimi_maj_kontakt`), ale aj číslo preukazu majiteľa, v prípade že je zároveň členom organizácie (`mimi_maj_prukaz`). Medzi posledné informácie ukladajúce sa k danému mláďatu patrí taktiež informácia, či je mláďa určené pre chov (`mimi_chov`), prípadne chovné obmedzenie (`mimi_chov_omezeni`).

#### 1.1.2.6 Tabuľka `pp_zadosti`

Posledná tabuľka, ktorá sa nachádza v databáze súčasnej aplikácie, je tabuľka s názvom `pp_zadosti`. Táto tabuľka má za úlohu zhromažďovať dáta o poslaných, resp. (ne)schválených žiadostiach o schválenie vrhu. Na rozdiel od predchádzajúcich tabuliek, v tejto tabuľke sa nachádzajú stĺpce obsahujúce informácie iba k daným žiadostiam.

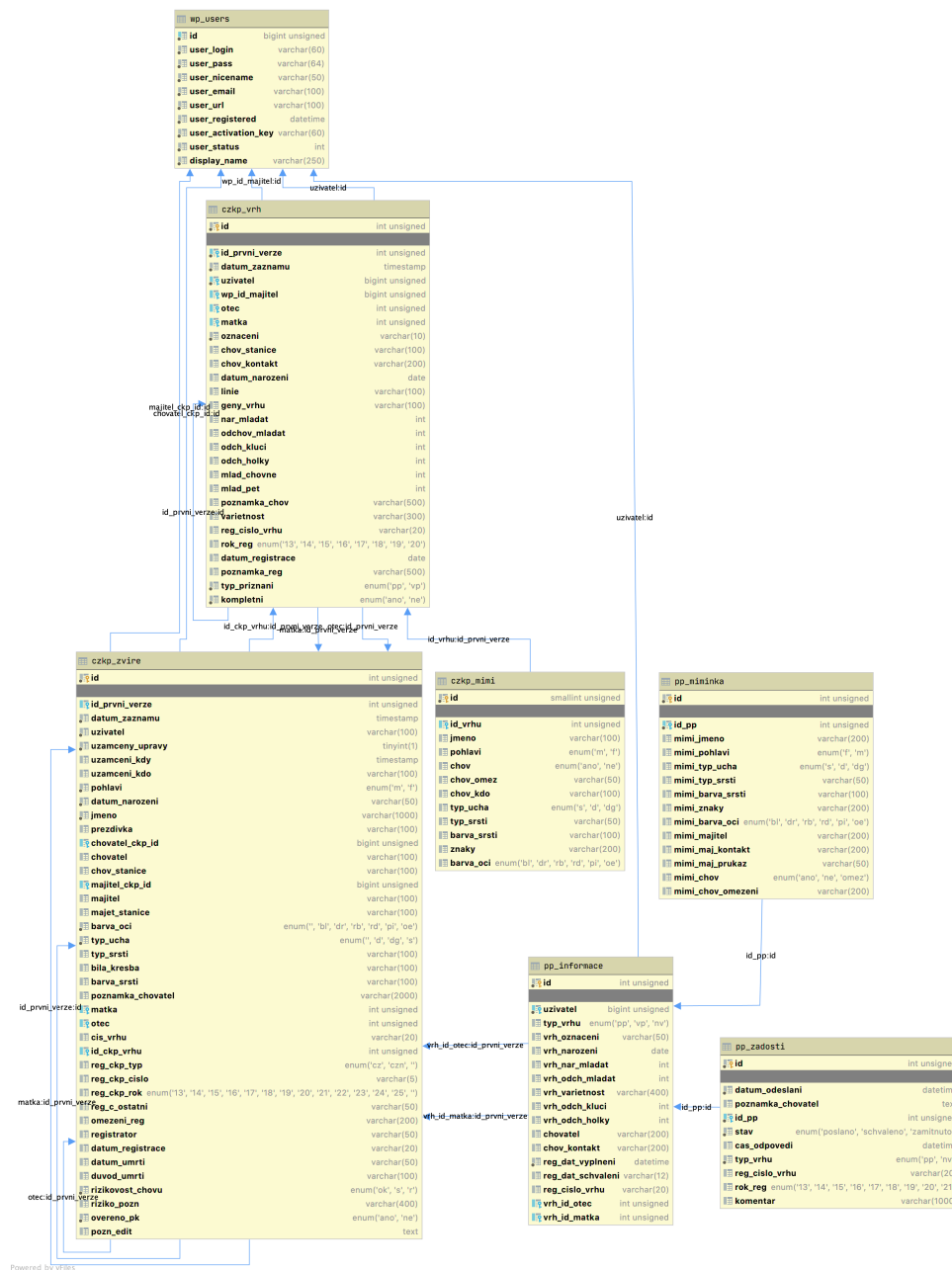
Táto tabuľka obsahuje stĺpec `id_pp`, ktorého obsahom je identifikátor vrhu z tabuľky `pp_informace`, ktorého sa týka daná žiadosť o schválenie vrhu. Tabuľka následne obsahuje stĺpec `datum_odoslani`, ktorý je automaticky nastavený na dátum a čas odoslania žiadosti a `cas_odpovedi`, ktorý indikuje dátum a čas odpovedi registrátora na danú žiadosť.

Okrem týchto informácií sa v tabuľke nachádza informácia o type schvaľovaného vrhu (`typ_vrhu`), poznámka žiadateľa o schválenie vrhu (`poznamka_chovatel`) a komentár registrátora ku žiadosti (`komentar`). V prípade, že je žiadosť schválená, `reg_cislo_vrhu` bude obsahovať registračné číslo vrhu a `rok_reg` rok registrácie daného vrhu.

# 1. ANALÝZA

## 1.1.2.7 Databázová schéma

V tejto podsekcii je možné nájsť logický model databázy v grafickej podobe, ktorý obsahuje schému tabuliek so vzťahmi medzi jednotlivými tabuľkami.



Obr. 1.2: Logický model súčasnej databázy

## 1.2 Analýza požiadaviek

V tejto sekcii sa nachádza popis všetkých požiadaviek kladených na vznikajúcu webovú aplikáciu. Tieto požiadavky sa delia na funkčné a nefunkčné.

### 1.2.1 Funkčné požiadavky

Funkčné požiadavky vo všeobecnosti vymedzujú hranice aplikácie v kontexte jej funkcionality, ktorú používateľ od aplikácie očakáva. Ich úlohou je taktiež spresniť odhad pracnosti na vznikajúcej aplikácii [4] (vlastný preklad).

#### 1.2.1.1 Evidencia zvierat

Aplikácia musí umožniť jednoduchú správu zvierat, od vytvorenia nového zvierata, cez zobrazenie jeho detailov, až po následnú editáciu, prípadne jeho zmazanie.

V evidencii zvierat je pre každé zviera potrebné ukladať nasledujúce údaje:

- Meno a prezývku,
- dátum narodenia,
- majiteľa a chovateľa zvierata,
- vrh, v ktorom sa zviera narodilo,
- matku a otca zvierata,
- pohlavie,
- farbu očí,
- typ uší,
- farbu a typ srsti,
- znaky,
- dátum a dôvod úmrtia.

Pre uľahčenie vytvorenia a editovania zvierata budú textové polia v maximálnej možnej miere interaktívne.

To znamená, že polia majiteľa a chovateľa zvierata vrátia na základe vstupu zoznam majiteľov, resp. chovateľov, ktorí už sú evidovaní v aplikácii. Následne si z tohto zoznamu používateľ zvolí požadovaného človeka. V prípade, že aplikácia požadovaného človeka nevráti, používateľovi bude ponúknutá možnosť ho dodatočne vytvoriť.

Na podobnom princípe bude fungovať aj textové pole pre vrh, s tým rozdielom, že podľa zvoleného vrhu aplikácia predvyplní matku a otca zvierata.

### 1.2.1.2 Evidencia registrácií zvieráťa

Je potrebné, aby aplikácia ponúkala možnosť zaregistrovať evidované zvierá pod záujmový chov. Pri takejto registrácii sa budú zbierať nasledovné údaje:

- Klub, pod ktorým bude zvierá zaregistrované (na výber z možností: ČKP, SOCHP alebo Ostatné),
- typ registrácie (vyžadované iba v prípade registrácie zvieráťa pod klubom ČKP),
- registračné číslo,
- dátum registrácie,
- informácia, či je nasledovný chov povolený,
- informácia o obmedzení chovu.

### 1.2.1.3 Evidencia vrhov

Medzi nevyhnutnú funkčnosť aplikácie sa radí aj evidencie vrhov. Tak ako pri evidencii zvierat, tak aj v tomto prípade používateľské rozhranie umožní vytvoriť nový vrh, zobraziť ho, respektíve ho editovať alebo vymazať.

Pre účel evidencie vrhov sa budú v aplikácii ukladať nasledujúce údaje:

- Typ a označenie vrhu,
- majiteľ vrhu,
- meno a kontakt na chovateľa,
- dátum narodenia,
- matka a otec vrhu,
- lúnia,
- genetické informácie,
- počet narodených a odchovaných mláďat,
- počet odchovaných samcov a samíc,
- počet mláďat určených pre maznanie a pre následný chov.

Rovnako ako pri evidencii zvierat, tak aj tu umožní aplikácia interaktívne zvoliť matku, otca a majiteľa vrhu zobrazením zoznamu uložených zvierat/ludí.

#### 1.2.1.4 Správa žiadostí o schválenie vrhov

Pre organizáciu je žiaduce, aby aplikácia obsahovala správu žiadostí o schválenie vrhu.

V rámci žiadosti o schválenie vrhu sa rozlišujú dva typy osôb – žiadateľ o schválenie vrhu (zväčša majiteľ daného vrhu) a registrátor vrhu.

Aplikácia žiadateľovi umožní poslať žiadosť o schválenie vrhu s možnosťou zanechania poznámky pre registrátora. Po odoslaní a úspešnom spracovaní tejto žiadosti serverom sa odošle e-mail všetkým registrátorom s informáciou o vytvorení novej žiadosti o schválenie vrhu. Na tento e-mail bude môcť zareagovať akýkoľvek registrátor, ktorý sa na základe dostupných informácií o vrhu rozhodne, či danú žiadosť schváli alebo zamietne. O zmene stavu žiadosti bude žiadateľ informovaný e-mailom.

#### 1.2.1.5 Zobrazenie rodokmeňu zvierťa a vrhu

Pre jednoduchšiu vizualizáciu predkov konkrétneho zvierťa aplikácia ponúkne zobrazenie rodokmeňu zvierťa vo forme jednoduchej tabuľky.

V tejto tabuľke budú okrem mien zvierat zobrazené aj dodatočné informácie o zvierati, definované v sekcii 1.2.1.1. Rodokmeň bude zobrazený v rámci jednotlivých zvierat a vrhov.

V prípade vrhu bude rodokmeň zobrazovať predkov matky a otca daného vrhu.

#### 1.2.1.6 Tvorba poznámok pre zviera a vrh

V rámci aplikácie bude potrebné implementovať poznámky, ktoré budú môcť byť priradené jednotlivým zvieratám a vrhom. Takáto poznámka by mala mať nastaviteľnú viditeľnosť (rozdelenie poznámky na verejnú a súkromnú) a typ (všeobecná, upozornenie alebo výstraha). Taktiež musí poskytnúť informáciu, kedy bola vytvorená, respektíve editovaná.

Nakoľko sa jedná o poznámky pre zviera a vrh, budú zobrazené u príslušných zvierat, resp. vrhoch, ku ktorým sa vzťahujú.

### 1.2.1.7 Zobrazenie histórie zmien zvierat a vrhov

Medzi želanú funkcionálnosť novej webovej aplikácie patrí sledovanie a následné zobrazenie histórie zmien pre všetky zvieratá a vrhy.

Aplikácia bude zaznamenávať nasledujúce zmeny v systéme:

- Vytvorenie zvierata/vrhu,
- úprava údajov zvierata/vrhu,
- zmazanie zvierata/vrhu,
- obnova zmazaného zvierata/vrhu.

Pri každej zmene popísanej vyššie je taktiež nutné ukladať, kto a kedy danú zmenu vykonal. V prípade úpravy údajov budú navyše zaznamenané tie údaje, ktoré boli zmenené používateľom. Túto históriu zmien bude možné vidieť vo forme tabuľky u každého zvierata, resp. vrhu.

### 1.2.1.8 Generovanie preukazov

Pre potreby organizácie je nevyhnutné implementovať generovanie preukazov (osvedčení) vo forme PDF súboru. V tomto súbore bude prvá strana vyplnená informáciami o danom zvierati 1.2.1.1, vrátane jeho registrácie zo sekcie 1.2.1.2. V niektorých prípadoch bude taktiež zobrazená registrácia vrhu v ktorom sa zviera narodilo, ktorej obsah je definovaný v 1.2.1.4. Druhá strana súboru bude vyplnená rodokmeňom zvierata (1.2.1.5).

Preukaz bude možné vygenerovať tlačidlom na stránke konkrétneho zvierata.

### 1.2.1.9 Filtrovanie a radenie

Pre vylepšenie používateľskej skúsenosti s aplikáciou bude potrebné implementovať filtrovanie a radenie zvierat ako aj vrhov na príslušných stránkach so zoznamom zvierat, resp. vrhov. Implementácia tejto funkcionality umožní jednoduchšiu prácu s aplikáciou a rýchlejšie nájdenie potrebných informácií.

### 1.2.1.10 Správa používateľov a rolí

Pre administrátorov aplikácie je nutné implementovať možnosť správy jednotlivých používateľov aplikácie, priradovať im príslušné role, alebo im ich naopak odobrať. Na základe tejto skutočnosti je žiaduce vytvoriť pohľad so zoznamom používateľov a ich rolami, s následnou možnosťou im danú rolu zmeniť, prípadne daných používateľov odobrať.



#### 1.2.1.11 Lokalizácia aplikácie

Nakoľko budú aplikáciu používať nielen českí ale aj zahraniční používatelia, je nutné, aby aplikácia poskytovala obsah lokalizovaný do anglického jazyka. Jazyk bude možné jednoducho zmeniť v menu na paneli webovej aplikácie.

#### 1.2.2 Nefunkčné požiadavky

Na rozdiel od funkčných požiadaviek, nefunkčné požiadavky umožňujú určit obmedzenia kladené na aplikáciu. V neposlednom rade majú taktiež zásadný dopad na návrh architektúry webovej aplikácie [4](vlastný preklad).

##### 1.2.2.1 Webová aplikácia

Nakoľko je požadovaný systém navrhnutý ako webová aplikácia, bude potrebné, aby bola prístupná z internetu pomocou moderných webových prehliadačov.

##### 1.2.2.2 Používateľské rozhranie

Webová aplikácia bude musieť obsahovať používateľské rozhranie, s ktorým budú môcť používatelia interagovať. Prostredie bude navyše responzívne, čo uľahčí prípadný prístup do systému z mobilného prehliadača.

##### 1.2.2.3 Technológie

Po konzultácii s vedúcim práce boli vymedzené nasledovné technológie, ktoré budú použité na strane servera.

Ako programovací jazyk bude použitý jazyk PHP vo verzii 7.3, hoci momentálne existuje novšia verzia jazyku – 7.4 [5] (vlastný preklad). Dôvod výberu nižšej verzie jazyka je daný PHP podporou webhostingu (server poskytujúci webovú aplikáciu používateľom na internete, v tomto prípade sa jedná o webhosting Endora – [www.endora.cz](http://www.endora.cz)), na ktorom bude aplikácia nasadená.

Pre ukladanie dát potrebných pre funkčnosť aplikácie a ich následné spracovávanie bude použitý rovnaký databázový systém, ako v súčasnej aplikácii – MySQL. Tento databázový systém bude použitý vo verzii 5.6.

Táto verzia taktiež nie je najnovšou verziou (v skutočnosti bola prvýkrát vydaná v roku 2013, avšak je stále oficiálne podporovaná [6] (vlastný preklad)), opäť z dôvodu neexistujúcej podpory novšieho databázového systému webhostingom.

### 1.3 Používateľské role

Novovznikajúca webová aplikácia bude prístupná iba zaregistrovaným a prihláseným používateľom. Navyše, niektoré akcie budú obmedzené iba pre určitý okruh používateľov.

K tomu, aby sme umožnili prístup k niektorým akciám iba vybraným používateľom, bude potrebné do aplikácie implementovať používateľské role a práva. Následne bude aplikácia riadiť prístup používateľa k jednotlivým akciám na základe príslušnosti k vybranej roli.

V nasledujúcich podsekciiach budú priblížené jednotlivé role a k nim príslušné práva, ktoré sa budú vyskytovať v aplikácii.

#### 1.3.1 Bežný používateľ

Túto rolu bude mať každý používateľ automaticky po registrácii do webovej aplikácie. Bežný používateľ bude môcť v aplikácii:

- Vytvoriť a zobraziť svoje zvieratá,
- upraviť svoje zvieratá v prípade, že nie sú zaregistrované pod klubom ČKP,
- vytvoriť a upraviť registrácie svojich vlastných zvierat, ktoré nespádajú pod klub ČKP,
- zobraziť zoznam všetkých cudzích zvierat spolu s ich detailmi,
- vytvoriť a zobraziť vrhy, u ktorých je používateľ ich majiteľom,
- uraviť vrhy, ktorých je majiteľom, pokiaľ neboli tieto vrhy schválené žiadosťou,
- zobraziť všetky vrhy typu VP alebo schválené vrhy typu PP a NV spolu s ich detailmi,
- pridať poznámku k zvieratám a vrhom, ktoré vlastní.

#### 1.3.2 Registrátor zvierat

Registrátor zvierat je v hierarchii rolí postavený nad bežným používateľom. Tým pádom má všetky práva bežného používateľa a navyše nasledujúce práva:

- Možnosť pridať poznámku k akýmkoľvek zvieratám,
- možnosť pridať, editovať a vymazať akúkoľvek registráciu u každého zvierata.

### 1.3.3 Schvaľovateľ vrhov

Schvaľovateľ vrhov je taktiež postavený v hierarchii rolí nad bežným používateľom podobne ako registrátor zvierat, s tým rozdielom, že schvaľovateľ vrhu môže v aplikácii:

- Vidieť a odpovedať na žiadosti o schválenie vrhov,
- editovať akékoľvek zviera a vrh,
- pridať poznámky k akémukoľvek vrhu,
- vygenerovať preukaz zvierata.

### 1.3.4 Administrátor

Ako obvykle, pre administrátora neplatia žiadne reštrikcie, čo znamená, že bude mať prístup ku všetkej funkcionalite definovanej vo funkčných požiadavkách v sekcii 1.2.1.

### 1.4 Prípady použitia

Prípady použitia sú špecifikácie rôznych činností, ktoré môžu používatelia s aplikáciou vykonávať [4] (vlastný preklad). Tieto prípady použitia budú zachytené vo forme scenáru a budú vychádzať nielen zo známych funkčných požiadaviek popísaných v sekcii 1.2.1, ale aj z jednotlivých používateľských rolí, ktoré boli definované v sekcii 1.3.

V tejto práci som sa rozhodol venovať iba takým prípadom použitia, ktoré sú z môjho pohľadu pre čitateľa prínosnejšie. Tie som následne rozdelil podľa príslušnosti k jednotlivým prvkom aplikácie. Všetky nasledujúce prípady použitia predpokladajú, že používateľ bude v systéme prihlásený.

Poznámka: Prvé štyri scenáre použitia sú aplikovateľné aj na vrhy, avšak z dôvodu neuvádzania duplicitných informácií nebudú ďalej rozvinuté v príslušnej podsekcii. Zároveň pri následnej aplikácii prvých štyroch scenárov na vrhy, sa výskyt slova „zvieratá“ automaticky nahradza slovom „vrh“ v príslušnej podobe.

#### 1.4.1 Prípady použitia zvierat

V tejto podsekcii budú popísané jednotlivé prípady použitia, ktoré sa týkajú zvierat.

##### 1.4.1.1 UC1 – Vyhľadanie zvieratá

Vyhľadanie zvieratá je jeden zo základných prípadov použitia, kedy používateľ chce vyhľadať dané zvieratá.

##### Scenár:

1. Používateľ zvolí možnosť „Zvieratá“ z navigačného menu aplikácie.
2. Aplikácia následne zobrazí tabuľku so zvieratami a filtrom, na základe ktorého si používateľ môže nastaviť dodatočné kritéria filtrovania zvierat.
3. Používateľ nastaví filter podľa kritérií hľadaného zvieratá a klikne na tlačidlo „Filtrovať“.
4. Aplikácia zobrazí všetky zvieratá vyhovujúce zadaným kritériám.

##### Alternatívny scenár:

3. Aplikácia zobrazí hľadané zvieratá už po načítaní tabuľky so zvieratami. V tomto prípade scenár vyhľadávania zvieratá končí.

Tento prípad použitia realizuje viaceré funkčné požiadavky na aplikáciu, konkrétne evidenciu a filtrovanie a radenie zvierat.

#### 1.4.1.2 UC2 – Zobrazenie detailu zvierťa

Tento prípad použitia popisuje zobrazenie detailu zvierťa a zahŕňa prípad UC1 – Vyhľadanie zvierťa.

##### Scenár:

1. UC1 – Vyhľadanie zvierťa
2. Používateľ klikne na meno zvierťa, ktorého detail si želá vidieť.
3. Aplikácia zobrazí detail zvierťa.

Popísaný prípad taktiež realizuje funkčnú požiadavku na aplikáciu, a to evidenciu zvierat.

#### 1.4.1.3 UC3 – Vytvorenie zvierťa

Medzi ďalší základný prípad použitia patrí pridanie zvierťa používateľom do systému. Pri vytváraní zvierťa sa aplikuje dodatočné obmedzenie pre bežného používateľa, ktoré spočíva v nemožnosti zvolenia majiteľa iného ako je on sám (1.3.1).

##### Scenár:

1. Používateľ zvolí možnosť „Zvieratá“ z navigačného menu aplikácie, ktorá následne načíta stránku so zoznamom uložených zvierat.
2. Následne používateľ klikne na tlačidlo „Pridať zvierá“.
3. Aplikácia načíta novú stránku s formulárom pre vytvorenie nového zvierťa.
4. Používateľ vyplní formulár.
5. Po vyplnení všetkých potrebných údajov pre vytvorenie zvierťa aplikácia umožní odoslať formulár kliknutím na tlačidlo „Uložiť“.
6. Používateľ klikne na tlačidlo „Uložiť“.
7. Po úspešnom vytvorení zvierťa na základe vložených údajov systém presmeruje používateľa na stránku so zoznamom uložených zvierat.

Tento prípad použitia realizuje funkčnú požiadavku evidencie zvierat.

### 1.4.1.4 UC4 – Úprava zvieráťa

V tomto prípade sa používateľ aplikácie snaží upraviť existujúce zviera v systéme. Tento prípad použitia taktiež zahŕňa prípady UC1 – Vyhľadanie zvieráťa alebo UC2 – Zobrazenie detailu zvieráťa, v závislosti od spôsobu úpravy zvieráťa, ktorý si používateľ zvolí.

#### Scenár:

1. UC1 – Vyhľadanie zvieráťa
2. Používateľ klikne na rozbaľovacie menu u zvieráťa, ktoré chce editovať.
3. Po tomto kroku aplikácia ponúkne možnosť editácie alebo zmazania zvieráťa.
4. Používateľ zvolí možnosť editácie.
5. Aplikácia následne presmeruje používateľa na stránku s editáciou zvieráťa, ktorá bude obsahovať formulár s predvyplnenými údajmi zvieráťa.
6. Používateľ upraví údaje zvieráťa.
7. Kliknutím na tlačidlo „Odoslať“ sa odošle vyplnený formulár na server.
8. Po úspešnom spracovaní formuláru je používateľ presmerovaný na stránku s detailmi upravovaného zvieráťa.

#### Alternatívny scenár:

1. UC2 – Zobrazenie detailu zvieráťa
2. Používateľ klikne na tlačidlo „Upraviť“ na pravom postrannom paneli.
3. Následne scenár pokračuje bodom 5 v pôvodnom scenári.

Popísaný prípad použitia realizuje funkčnú požiadavku evidencie zvieráťa.

### 1.4.1.5 UC5 – Vytvorenie registrácie zvieráťa

Tento prípad použitia popisuje vytvorenie registrácie zvieráťa. Pri samotnom vytváraní novej registrácie sa aplikujú dodatočné obmedzenia pre bežného používateľa, tak ako je popísane v 1.3.1.

Tento prípad použitia zahŕňa prípad zobrazenia detailov zvieráťa.

**Scenár:**

1. UC2 – Zobrazenie detailu zvierata
2. Kliknutím na tlačidlo „Pridať novú registráciu“ aplikácia zobrazí registračný formulár v modálnom okne (v menšom okne aplikácie prekrývajúce jej hlavný obsah).
3. Používateľ vyplní potrebné polia.
4. Po stlačení tlačidla „Uložiť“ sa vyplnený formulár odošle na server.
5. Po úspešnom spracovaní formuláru serverom sa modálne okno zavrie.

Tento prípad použitia realizuje funkčnú požiadavku kladenú na aplikáciu, konkrétne evidenciu registrácií zvierata.

### 1.4.2 Prípady použitia vrhov

Táto podsekcia popisuje jednotlivé prípady použitia, ktoré sa týkajú vrhov.

#### 1.4.2.1 UC6 – Vytvorenie žiadosti o schválenie vrhu

Tento prípad použitia reálne zahŕňa prípad použitia „zobrazenia detailov vrhu“, avšak pre neuvádzanie duplicitných informácií môže čitateľ vychádzať z prípadu použitia zobrazenia detailov zvierata. Scenáre sa v oboch prípadoch obsahovo nelíšia, tak ako bolo uvedené poznámke v sekcii 1.4.

Na tento prípad sa taktiež vzťahujú dodatočné obmedzenia pre bežného používateľa vyplývajúce z 1.3.1.

**Scenár:**

1. UC2 – Zobrazenie detailu zvierata
2. Používateľ klikne na tlačidlo „Vytvoriť žiadosť“.
3. Aplikácia následne zobrazí modálne okno s formulárom obsahujúcim textové pole slúžiace pre poznámku registrátorovi.
4. Po stlačení tlačidla „Uložiť“ sa formulár odošle na server.
5. Po úspešnom spracovaní formuláru serverom sa vytvorí nová žiadosť o schválenie vrhu.
6. Modálne okno sa automaticky zavrie.

Popísaný prípad použitia realizuje funkčnú požiadavku správy žiadostí o schválenie vrhu.

### 1.4.2.2 UC7 – Odpoveď na žiadosť o schválenie vrhu

Tak ako v predchádzajúcom prípade, tento prípad použitia bude zahŕňať „zobrazenie detailov vrhu“.

Pre tento prípad použitia sa vzťahujú dodatočné obmedzenia – na žiadosť o schválenie vrhu bude môcť odpovedať používateľ majúci rolu schvaľovateľa žiadostí vrhov alebo administrátora.

#### Scenár:

1. UC2 – Zobrazenie detailu zvierata
2. Používateľ stlačí tlačidlo „Odpovedať“ pri príslušnej žiadosti o schválenie vrhu.
3. Aplikácia zobrazí modálne okno s formulárom pre schválenie vrhu.
4. Používateľ vyplní formulár a kliknutím na tlačidlo „Uložiť“ sa odošle na server.
5. Server spracuje odoslaný formulár a ihneď odošle e-mail žiadateľovi o stave jeho žiadosti.

#### Alternatívny scenár:

1. Používateľ klikne na link v e-maile, ktorý mu prišiel po odoslaní novej žiadosti o schválenie vrhu žiadateľom.
2. Ako reakcia na kliknutie na link, operačný systém otvorí predvolený webový prehliadač s adresou vedúcou na detail vrhu.
3. Následne scenár pokračuje bodom 2 v pôvodnom scenári.

Tak ako v predchádzajúcom prípade použitia, aj v tomto prípade použitia je realizovaná funkčná požiadavka správy žiadostí o schválenie vrhu.



---

# Návrh aplikácie

Táto kapitola sa venuje návrhu architektúry budúcej aplikácie, s priblížením dvoch hlavných architektúr používaných pri tvorbe webových aplikácií. Následne sú obe porovnané a podľa výsledkov porovnania je vybraná taká, ktorá bude použitá pri vývoji aplikácie. V ďalšej sekcii sú vymedzené použité technológie budúcou aplikáciou. V neposlednom rade sa kapitola zaoberá návrhom databázy s popisom normalizácie tabuliek a ich zobrazením v schéme v grafickej podobe.

## 2.1 Návrh architektúry aplikácie

Dôležitý aspekt pri návrhu aplikácie spočíva vo výbere architektúry, na ktorej bude aplikácia postavená. Na základe jej výberu sa odvíjajú technológie, ktorými bude daná aplikácia disponovať.

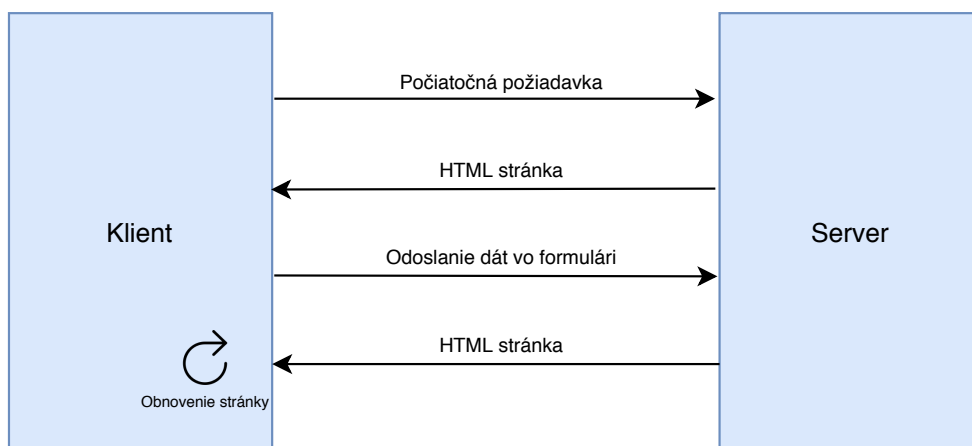
V nasledujúcich sekciiach sú zdokumentované dva typy architektúr, z ktorých bude na základe porovnania a vlastných skúseností s tvorbou webových aplikácií vybraná jedna, ktorá bude definovať podobu budúcej webovej aplikácie.

### 2.1.1 Architektúra MPA aplikácie

MPA je skratka pre viac-stránkovú webovú aplikáciu. Takáto aplikácia vždy načíta celú stránku a zobrazí novú v prípade interakcie s aplikáciou – zvyčajne po odoslaní formuláru používateľom alebo presmerovaním na inú stránku aplikácie [7] (vlastný preklad).

#### Priebeh komunikácie medzi serverom a klientom

Celkový priebeh komunikácie medzi klientom a serverom zachytáva nasledujúci diagram s jeho popisom.



Obr. 2.1: Diagram znázorňujúci architektúru MPA aplikácií

1. Klient požiadá server o stránku.
2. Server vráti klientovi požadovanú stránku.
3. Klient vrátenú stránku serverom vykreslí.
4. Klient vyplní formulár, ktorý sa následne odošle na server.
5. Server prijme údaje od klienta, ktoré následne spracuje.
6. Po spracovaní údajov klientovi vráti späť stránku s aktualizovanými údajmi.
7. Klient túto stránku načíta, čím príde k obnoveniu stránky.

Technológia AJAX čiastočne rieši problém znovunačítavania celej stránky dynamickým aktualizovaním tých častí aplikácie, ktoré boli používateľom zmenené. Avšak zakomponovanie tejto technológie do MPA sťažuje a komplikuje celý vývojový proces aplikácie [8] (vlastný preklad).

### Výhody použitia MPA architektúry

- Jednoduchá optimalizácia stránok pre webové vyhľadávače [7] (vlastný preklad),
- umožňuje jednoduchú správu informácií na jednotlivých stránkach [9] (vlastný preklad),
- je potrebný menší rozsah nástrojov a znalostí na rozdiel od vývoja SPA aplikácie [10] (vlastný preklad),
- veľa dostupných riešení pre vývoj MPA aplikácií [10] (vlastný preklad).

### Nevýhody použitia MPA architektúry

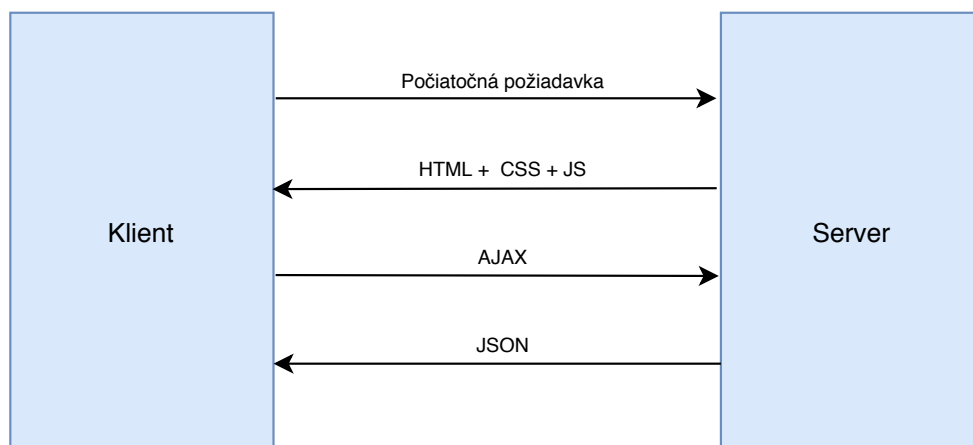
- Zvýšený čas načítavania stránok kvôli neustálemu obnovovaniu stránok (neplatí pri použití technológie AJAX) [7] (vlastný preklad),
- znížený výkon aplikácie, kvôli neustálemu načítavaniu väčšieho množstva informácií naraz pred následným poslaním stránky používateľovi [7] (vlastný preklad),
- úzka previazanosť vývoja aplikácie na strane servera a klienta, čo zneumožňuje prípadné neskoršie nasadenie rôznych technológií [10] (vlastný preklad).

### 2.1.2 Architektúra SPA aplikácie

SPA je skratka pre single-page aplikáciu. Single-page aplikácia je aplikácia, ktorá nepotrebuje znovunačítanie stránky počas jej používania. Na tomto type architektúry sú postavené aplikácie ako Gmail, Google Mapy, Facebook či GitHub. V tomto prípade sa o aktualizáciu obsahu na stránke nestará server, ale klient, zväčša pomocou frameworku (sady podporných programov uľahčujúca vývoj aplikácie) bežiacom v prostredí webového prehliadača [9] (vlastný preklad).

#### Priebeh komunikácie medzi serverom a klientom

Celkový priebeh komunikácie medzi klientom a serverom zachytáva nasledujúci diagram s jeho popisom.



Obr. 2.2: Diagram znázorňujúci architektúru SPA aplikácií

## 2. NÁVRH APLIKÁCIE

---

1. Klient požiada server o stránku.
2. Server vráti klientovi požadovanú HTML stránku so všetkým obsahom aplikácie.
3. Klient vrátenú stránku serverom vykreslí.
4. Klient vyplní formulár, ktorý sa následne odošle na server pomocou AJAXu.
5. Server prijme údaje od klienta, ktoré následne spracuje.
6. Po spracovaní údajov server vráti späť klientovi aktualizované údaje (v JSON formáte),
7. Klient zahodí staré údaje na stránke a nahradí ich novými, čím príde k obnoveniu údajov ale nie stránky.

### **Výhody použitia SPA architektúry**

- Rýchlosť a responzivnosť aplikácie založenej na aktualizovaní iba tej časti aplikácie, ktorá sa zmenila [7] (vlastný preklad),
- zjednodušený proces vývoja, nakoľko sa o vykreslenie obsahu nestará server ale klientský framework [9] (vlastný preklad),
- minimálna previazanosť kódu aplikácie na strane servera a klienta, čo umožňuje jednoduchú správu používaných knižníc bez ovplyvnenia ďalších častí aplikácie [10] (vlastný preklad).

### **Nevýhody použitia SPA architektúry**

- Ťažká optimalizácia aplikácie pre webové vyhľadávače, nakoľko sa o vykreslenie obsahu stará Javascript (programovací jazyk bežiaci vo webovom prehliadači), ktorý webové vyhľadávače neinterpretujú [7] (vlastný preklad),
- prvé načítanie aplikácie trvá dlhší čas v porovnaní s MPA architektúrou, pretože sa musí všetok obsah aplikácie stiahnuť do zariadenia [7] (vlastný preklad),
- nemožnosť zobrazenia aplikácie pri vypnutom Javascripte vo webovom prehliadači [10] (vlastný preklad).

### 2.1.3 Voľba architektúry

Pre budúcu aplikáciu som sa rozhodol vybrať SPA architektúru, nielen z dôvodu rýchlejšieho a jednoduchšieho procesu vývoja, ale aj vďaka vlastným skúsenostiam z oblasti vývoja SPA aplikácií. Nakoľko aplikácia bude dostupná iba vopred vybraným používateľom, nebude potrebné ju optimalizovať pre webové vyhľadávače.

### 2.2 Voľba technológií

V tejto sekcii sa presunieme od voľby architektúry, na ktorej bude aplikácia postavená, ku voľbe technológií, ktoré budú použité vo výslednej aplikácii.

#### 2.2.1 PHP

Medzi zvolené technológie bude jednoznačne patriť programovací jazyk PHP, nakoľko jeho voľba je jedným z nefunkčných požiadaviek kladených na budúcu aplikáciu.

#### Framework

Vývoj softvéru je komplexný proces, ktorý zahŕňa nielen výslednú implementáciu konkrétneho softvéru, ale aj analýzu, návrh, a v neposlednom rade aj testovanie softvéru. Softvérové frameworky uľahčujú prácu vývojárom tým, že im umožňujú prevziať kontrolu nad celým procesom vývoja softvéru alebo jeho väčšiny [11] (vlastný preklad).

Medzi niektoré výhody použitia frameworku patrí:

- Pomoc pri zavádzaní lepších programovacích postupov a vhodnom využívaní návrhových vzorov,
- väčšia bezpečnosť výsledného kódu,
- možnosť vyhnúť sa duplicite kódu,
- skrátenie času vývoju výslednej aplikácie.

Pre vývoj webových aplikácií v PHP existujú viaceré frameworky, ako napríklad CakePHP, Nette, Symfony, Laravel a iné. Spomedzi menovaných bol vybraný na základe doterajších skúseností posledný z nich – Laravel.

#### 2.2.2 HTML 5

HTML – HyperText Markup Language – je značkovací jazyk používaný pre definovanie významu a štruktúry dokumentov. Ako už z názvu vyplýva, HTML používa značky pre definovanie elementov v dokumente s rozdielnymi vlastnosťami, ako napríklad nadpis, odstavec, tabuľky a iné. Tieto značky sú interpretované webovým prehliadačom, ktorý ich následne vykreslí na stránku [12] (vlastný preklad).

Najnovšou verziou tohto jazyka je verzia HTML 5.

### 2.2.3 CSS 3

Cascading Style Sheets (CSS) je jazyk, ktorý sa používa na ilustráciu vzhľadu, štýlu a formátu dokumentu a jeho elementov napísaného v akomkoľvek značkovacom jazyku. Využíva sa najmä v spojitosti s webovými stránkami. CSS je tvorený pravidlami aplikovanými na jednotlivé elementy dokumentu. Tieto pravidlá sú definované selektorom elementu, ktorý určuje polohu elementu v dokumente, a CSS pravidlami rôzneho typu, ktoré definujú konkrétny štýl daného elementu [13] (vlastný preklad).

CSS3 je momentálne najnovšia verzia jazyka CSS.

### 2.2.4 JavaScript

JavaScript je jednoduchý programovací jazyk, pomocou ktorého môžu vývojári webových stránok vytvárať interaktívne a dynamické webové stránky. Bol vyvinutý spoločnosťou Netscape, ale v súčasnosti ho používa väčšina webových prehliadačov. Je to programovací jazyk s otvoreným zdrojovým kódom, ktorý môže ktokoľvek používať [14] (vlastný preklad).

#### Framework

Nakoľko novovznikajúca aplikácia bude postavená na SPA architektúre, bude potrebné, aby sa stránky vykreslovali na strane klienta. To sa dá samozrejme dosiahnuť bez použitia frameworku, ktorý by sa postaral o aktualizáciu komponentov na stránke, avšak z dôvodu rýchlejšieho procesu vývoja som sa rozhodol použiť framework Vue.js.

Vue.js je progresívny framework pre vytváranie používateľských rozhraní. Tento framework bol navrhnutý od základov tak, aby bol postupne prispôsobiteľný. Nakoľko je zameraný iba na vrstvu zobrazenia, je ľahké ho integrovať s inými knižnicami a existujúcimi projektmi [15] (vlastný preklad).

Pre vytváranie používateľského rozhrania existujú aj iné frameworky, ako napríklad React a Angular, avšak z dôvodu doterajších skúseností bol zvolený Vue.js framework.

### 2.3 Návrh databázy

V predchádzajúcej kapitole v podsekcii 1.1.2 bol zanalyzovaný databázový model súčasnej aplikácie.

Táto sekcia popíše normalizáciu súčasnej databázy a na základe poznatkov vyplývajúcich z analýzy o súčasnom databázovom modeli bude navrhnutý databázový model pre vznikajúcu aplikáciu, ktorý bude brať do úvahy funkčné požiadavky kladené na aplikáciu.

#### 2.3.1 Normalizácia databázy

Normalizácia databázy je proces efektívneho usporiadania údajov v databáze. Tento proces má dva ciele: odstránenie nadbytočných údajov (napríklad neukladanie tých istých údajov vo viac ako jednej tabuľke) a zabezpečenie ukladania iba súvisiacich údajov do tabuľky. Normalizovaná databáza znižuje množstvo miesta potrebného na ukladanie dát a navyše zabezpečí, že dáta sú logicky usporiadané [16] (vlastný preklad).

Pri návrhu databázy je žiaduce dosiahnuť tretiu normálovú formu (3NF) pre každú tabuľku v databáze, pretože až takáto databáza sa považuje za normalizovanú [17] (vlastný preklad).

Proces normalizácie tabuľky zahŕňa jej konverziu do prvej normálovej formy (ak v nej nie je), následne do druhej normálovej formy (ak ju nespĺňa) a na koniec do tretej normálovej formy.

Tabuľka je v prvej normálovej forme (1NF), ak spĺňa nasledujúce vlastnosti:

- Obsahuje iba atomické hodnoty v každom stĺpci,
- hodnoty uložené v stĺpci sa viažu iba na daný stĺpec,
- všetky stĺpce v tabuľke majú unikátny názov,
- nezáleží na poradí ukladania dát do stĺpcov [18] (vlastný preklad).

Tabuľka je v druhej normálovej forme (2NF), ak má nasledujúce vlastnosti:

- Tabuľka spĺňa 1NF,
- nemôže obsahovať čiastočnú závislosť [18] (vlastný preklad).

Čiastočná závislosť v tabuľke existuje, pokiaľ atribút v tabuľke závisí od časti primárneho kľúča a nie od celého kľúča [19] (vlastný preklad).



Na to, aby tabuľka bola v tretej normálovej forme (3NF), musí navyše mať nasledujúce vlastnosti:

- Tabuľka spĺňa 2NF,
- nemôže obsahovať tranzitívnu závislosť [18] (vlastný preklad).

Tranzitívna závislosť predstavuje tri alebo viac atribútov, ktoré majú funkčnú závislosť medzi sebou, čo znamená, že stĺpec A závisí na stĺpci B cez stĺpec C [20] (vlastný preklad).

### 2.3.1.1 Tabuľka `czkp_vrh`

Na základe analýzy a splnenia požiadavky 3NF bude potrebné, aby údaje o chovateľovi sa nenachádzali priamo v tabuľke vrhov, ale cez cudzí kľúč smerujúci do tabuľky určenej pre chovateľov/majiteľov. Taktiež bude potrebné presunúť údaje o registrácii vrhu a poznámok do tabuliek pre ne určených.

### 2.3.1.2 Tabuľka `czkp_zvire`

Z vykonanej analýzy tejto tabuľky vyplýva, že tabuľka okrem údajov o samotnom zvierati obsahuje navyše informácie o entitách ako majiteľ/chovateľ, registrácia zvierata, prípadne poznámky viažuce sa k danému zvieratu. Pretože výsledná tabuľka má byť normalizovaná, nemala by obsahovať tieto údaje priamo, ale cez cudzí kľúč.

Pre každú horeuvedenú entitu by mala vzniknúť jedna tabuľka — zvlášť pre chovateľov a majiteľov, registrácie zvierat a poznámky. Z pohľadu databázy netreba rozoznávať ľudí, či sú daní ľudia chovateľmi alebo majiteľmi, preto môžu byť uložené v jednej tabuľke.

### 2.3.1.3 Tabuľka `czkp_mimi`

Táto tabuľka nespĺňa 3NF, pretože obsahuje text namiesto cudzieho kľúča v stĺpci chovateľa. Pri porovnaní tejto tabuľky a tabuľky `czkp_zvire` je možné vidieť, že stĺpce definované v tejto tabuľke sa štruktúrou rovnajú stĺpcom v tabuľke `czkp_zvire`, z čoho vyplýva, že táto tabuľka môže byť spojená s tabuľkou `czkp_zvire`.

### 2.3.1.4 Tabuľka `pp_informace`

Táto tabuľka je obdobou tabuľky `czkp_vrh`, čo znamená, že tieto dve tabuľky môžu byť zlúčené do jednej, po oddelení údajov o registrácii vrhu a presunutí údajov o chovateľovi vrhu do samostatnej tabuľky.

### 2.3.1.5 Tabuľka pp\_miminka

Tak ako tabuľka czkp\_mimi, tak aj táto obsahuje mláďatá narodené vo vrhu (ktorý je uložený v pp\_informace). Pozitívum oproti tabuľke czkp\_mimi je jej kompaktnosť (obsahuje menej informácií, ktoré sa viažu nepriamo k tejto tabuľke), avšak stále nespĺňa 3NF vďaka ukladaniu informácií o majiteľovi priamo v tejto tabuľke. Po presunutí dát o majiteľoch do samostatnej tabuľky, môže byť táto tabuľka zlúčená s tabuľkou czkp\_mimi (ktorá bude zlúčená s tabuľkou czkp\_zvire).

### 2.3.1.6 Tabuľka pp\_zadosti

Štruktúra tejto tabuľky porušuje 3NF tým, že si ukladá typ vrhu, ku ktorému sa daná žiadosť viaže, pretože typ vrhu je uložený už pri samotnom vrhu a nie je potrebné ho ukladať aj pri žiadosti o schválenie vrhu.

## 2.3.2 Návrh databázového modelu

Na základe informácií o štruktúre databázy a funkčných požiadaviek sa táto podsekcia zaoberá výsledným návrhom databázového modelu, ktorý zahŕňa popis navrhnutých tabuliek a zjednodušenú schému logického modelu v grafickej podobe, ktorého základom tvoria navrhnuté tabuľky.

Všetky navrhnuté tabuľky používajú stĺpec id ako jednoznačný identifikátor daného záznamu. Tento identifikátor je unikátny a pri každom vložení nového záznamu do databázy sa tento identifikátor inkrementuje o 1. Prvý záznam v danej tabuľke má id nastavený na hodnotu 1.

### 2.3.2.1 Tabuľka animals

Táto tabuľka bude zoskupovať všetky zvieratá, ktoré organizácia sleduje bez ohľadu na to, či sa jedná o mláďa z vrhu alebo o dospelé zviera.

Jej štruktúra sa skladá zo stĺpcov creator\_id, čo je identifikátor používateľa, ktorý dané zviera vytvoril. Následne štruktúra tabuľky obsahuje identifikátory breeder\_id a owner\_id. V prvom prípade sa jedná o identifikátor človeka, ktorý je chovateľom zvierata, kde v druhom sa jedná o identifikátor majiteľa zvierata. Pre identifikáciu matky a otca zvierata sa používajú stĺpce mother\_id a father\_id. Obsahom stĺpca litter\_id je identifikátor vrhu, z ktorého zviera pochádza.

Meno zvierata, jeho prezývka, pohlavie, dátum narodenia, farba očí, typ uší, farba srsti, typ srsti a jeho znaky budú uložené do príslušných stĺpcov, a to name, nickname, sex, birthdate, eyes\_color, ear\_type, fur\_color, fur\_type a markings.

Údaje o úmrtí zvierata ako dátum úmrtia a dôvod úmrtia, budú uložené v stĺpcoch death\_date, respektíve death\_reason.

Informácia, či je povolený chov, alebo je nejakým spôsobom limitovaný, bude možné nájsť v stĺpcoch `breeding_available`, respektíve `breeding_limitation`. Aplikácia bude obsahovať podporu pre generovanie PDF preukazov. V danom preukaze sa budú nachádzať údaje z tejto tabuľky a z tabuľky `litters`. Jedným z týchto údajov je aj meno majiteľa zvierata, jeho kontakt a číslo členského preukazu, ktorý vydáva organizácia. Tieto údaje by normálne mohli byť uložené v tabuľke určenej pre chovateľov a majiteľov. Avšak v prípade, že by sa tieto údaje zmenili priamo v tabuľke, v ktorej sa nachádza daný majiteľ zvierata, nastala by situácia, že by sa pri následnom generovaní rovnakého preukazu líšili údaje o majiteľovi zvierata medzi týmto preukazom a preukazom vygenerovaným pred zmenou údajov. Preto sa tieto údaje budú taktiež ukladať v tabuľke `zvierat` – konkrétne v stĺpcoch `owner_name`, `owner_contact` a `owner_member_card_number`.

### 2.3.2.2 Tabuľka `animal_registrations`

Nakolko zvieratá môžu byť zaregistrované pod rôznymi klubmi, je potrebné, aby sa tieto údaje ukladali do tabuľky na to určenej.

Štruktúra tejto tabuľky sa skladá z `animal_id`, čo je identifikátor zvierata, ktorého sa daná registrácia zvierata týka. Následne sa ukladá identifikátor registrátora, ktorý dané zviera zaregistroval (`registrator_id`). Medzi ďalšie údaje, ktoré sa ukladajú do tejto tabuľky, patrí klub, v ktorom je zviera zaregistrované (`club`), typ registrácie (`type`), jeho registračné číslo (`registration_number`) a rok registrácie (`year`). Každý klub má možnosť obmedziť následný chov zvierata a špecifikovať dodatočné informácie, prečo je následný chov zvierata obmedzený.

Pre tieto účely budú slúžiť stĺpce `breeding_available/breeding_limitation`.

### 2.3.2.3 Tabuľka `litters`

Táto tabuľka bude zhromažďovať všetky vrhy sledované organizáciou.

Medzi údaje, ktoré je potrebné ukladať o každom vrhu, patria identifikátory oboch rodičov vrhu – v prípade matky sa jedná o stĺpec `mother_id` a v prípade otca o stĺpec `father_id`. Okrem iného sa bude ukladať aj identifikátor majiteľa vrhu do stĺpca `owner_id` a identifikátor používateľa, ktorý daný záznam vytvoril (`creator_id`).

Následne sa do tabuľky budú ukladať údaje ako typ vrhu (`type`), jeho označenie (`label`), dátum narodenia vrhu (`birthdate`), línia vrhu (`line`) a genetické informácie (`genetic_information`).

Vo vrhu je nutné sledovať počet narodených a odchovaných mláďat (v stĺpcoch `babies_born` a `babies_reared`), odchovaných samcov a samíc (`reared_boys` a `reared_girls`) a počet mláďat určených pre chov a pre maznanie — tieto údaje budú uložené v stĺpcoch `for_breeding` a `for_petting`.

## 2. NÁVRH APLIKÁCIE

---

Nakoľko aplikácia bude podporovať generovanie preukazov vo formáte PDF, budú v tejto tabuľke uložené údaje o mene chovateľa a jeho kontaktných údajov z rovnakých dôvodov, ktoré boli uvedené v 2.3.2.1.

Meno chovateľa a jeho kontaktné údaje budú uložené v stĺpcoch `breeder_name`, resp. `breeder_contact`.

### 2.3.2.4 Tabuľka `litter_approval_requests`

Účelom tejto tabuľky je spracovávať žiadosti o schválenie vrhov. Úlohou schvaľovateľa je posúdiť jednotlivé žiadosti a na základe poskytnutých údajov o vrhu rozhodnúť, či daný vrh schváli alebo nie.

Pre tie účely bude potrebné, aby tabuľka obsahovala identifikátor vrhu, ku ktorému sa daná žiadosť viaže.

Tento identifikátor bude uložený v stĺpci `litter_id`. Okrem tohto identifikátora sa bude taktiež ukladať identifikátor žiadateľa, ktorý danú žiadosť vytvoril — tento údaj sa automaticky uloží pri vytvorení žiadosti do stĺpca `creator_id`. Po posúdení žiadosti a rozhodnutí, či bude daný vrh schválený alebo nie, sa bude ukladať identifikátor schvaľovateľa, ktorý danú žiadosť schválil alebo zamietol, a to do stĺpca `registrator_id`.

Žiadosti môžu mať tri stavy, a to stav „poslaná“, „schválená“ a „zamietnutá“. Stav jednotlivých žiadostí sa bude ukladať do stĺpca `state`.

Žiadateľ o schválenie vrhu bude mať možnosť pridať poznámku schvaľovateľovi – táto poznámka sa bude ukladať do stĺpca `creator_note`.

Pre schválenú žiadosť je potrebné ukladať jej registračné číslo, dátum registrácie, prípadne poznámku schvaľovateľa pre žiadateľa (tá môže byť uložená i v prípade odmietnutia schválenia vrhu). Tieto údaje sa budú ukladať do stĺpcov `registration_number`, `registration_date`, resp. `registrator_note`.

### 2.3.2.5 Tabuľka `stations`

Chovné stanice, ktorých súčasťou môže byť majiteľ resp. chovateľ zvierat/vrhu, budú uložené v tejto tabuľke. Štruktúra tabuľky je jednoduchá — obsahuje stĺpec `name`, v ktorom je uložené meno chovnej stanice, a stĺpec `creator_id`, ktorý obsahuje identifikátor osoby, ktorý danú stanicu vytvoril.

### 2.3.2.6 Tabuľka `people`

Táto tabuľka je určená pre zhromažďovanie údajov o chovateľoch a majiteľoch zvierat a vrhov.

O týchto ľuďoch je potrebné evidovať osobné údaje ako ich meno, email, telefónne číslo a číslo členského preukazu.

Tieto informácie sa budú ukladať do stĺpcov `name`, `email`, prípadne do stĺpcov `telephone_number` a `member_card_number`.

Okrem týchto základných dát sa budú ukladať dodatočné informácie, ako identifikátor chovateľskej stanice, ktorej je daný človek súčasťou (`station_id`),

identifikátor používateľa, ktorý daný záznam vytvoril (`creator_id`) a identifikátor používateľa (`user_id`), ak sa používateľ nachádza v systémovej tabuľke `users`. Ak je identifikátor `user_id` vyplnený, znamená to, že daný človek je systémovým používateľom aplikácie, ale aj zároveň chovateľom, resp. majiteľom nejakého zvieratá/vrhu.

### 2.3.2.7 Tabuľka `users`

Systémová tabuľka `users` bude obsahovať dáta o používateľoch systému. O týchto používateľoch je potrebné ukladať ich meno (`name`) a email (`email`), podľa ktorého sa budú daní používatelia identifikovať pri prihlásení do aplikácie. Taktiež bude potrebné ukladať ich heslo v zahešovanej podobe do stĺpca `password`.

Pre účely budúceho rozšírenia aplikácie — konkrétne prihlasovania sa pomocou systému WordPress do novej aplikácie — sa bude ukladať do stĺpca `wordpress_id` identifikátor používateľa vo WordPress systéme.

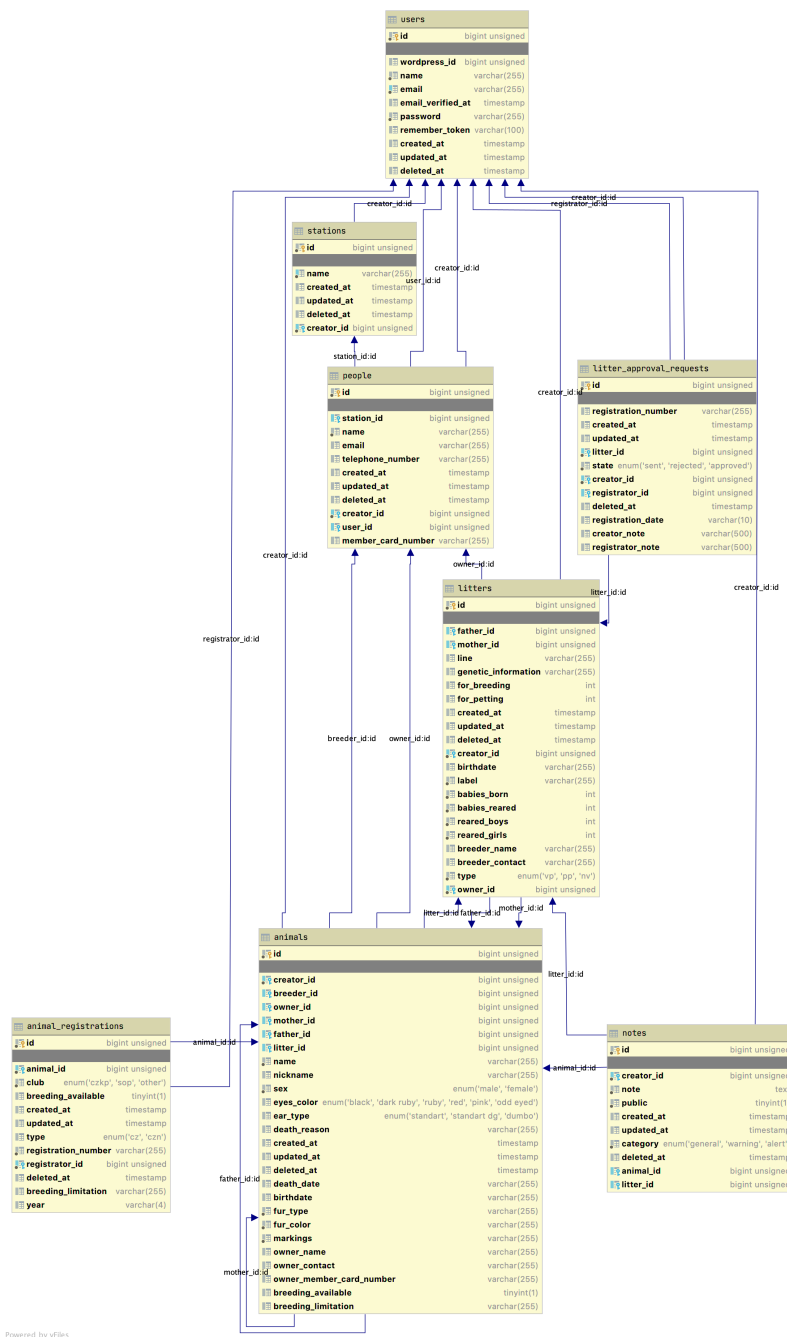
### 2.3.2.8 Tabuľka `notes`

Nakoľko je žiaduce ukladať poznámky k jednotlivým zvieratám a vrhom, bude vytvorená tabuľka `notes`, do ktorej sa jednotlivé poznámky budú ukladať. Táto tabuľka bude obsahovať stĺpec `creator_id`, do ktorého sa bude ukladať identifikát používateľa, ktorý daný záznam vytvoril. Ak sa bude jednať o poznámku ku zvierati, jeho identifikátor sa uloží do stĺpca `animal_id`. V opačnom prípade – pri ukladaní poznámky viažucej sa k vrhu – sa identifikátor daného vrhu uloží do stĺpca `litter_id`. Ďalej pre účel poznámok bude potrebné ukladať ich kategóriu, viditeľnosť a obsah. Tieto informácie sa budú ukladať do stĺpcov `category`, `public` a `note`.

## 2. NÁVRH APLIKÁCIE

### 2.3.2.9 Databázová schéma

Pre lepšiu vizualizáciu navrhnutých tabuliek a ich vzťahov medzi nimi sa v tejto podsekcii nachádza zjednodušený logický model databázy v grafickej podobe.



Obr. 2.3: Logický model navrhutej databázy

---

# Implementácia

Cieľom tejto kapitoly je na základe predošlej analýzy súčasnej aplikácie a návrhu databázy priblížiť implementačnú časť novej aplikácie. Najprv sa kapitola venuje implementácii aplikácie na serverovej časti s priblížením architektonického vzoru MVC, ktorý sa využíva pri tvorbe webových aplikácií. Následujúca sekcia je vymedzená implementáciou klientskej časti aplikácie. Záver je venovaný migrácii dát z pôvodnej do novej aplikácie.

## 3.1 Serverová časť

Implementácia serverovej časti webovej aplikácie sprevádzala využitie zvoleného PHP frameworku, ktorým bol Laravel.

### 3.1.1 Štruktúra projektu

Pre inicializáciu nového Laravel projektu som použil príkaz `laravel new pedigree-book`. Tento príkaz vytvoril nasledujúcu adresárovú štruktúru.

```
| app ..... Adresár určený pre implementáciu aplikácie  
| bootstrap ..... Adresár určený pre aplikačnú cache  
| config ..... Adresár obsahujúci konfiguračné súbory aplikácie  
| database ..... Adresár určený pre databázové migrácie, továrne a seeds  
| public ..... Adresár určený pre všetky verejne dostupné súbory  
| resources ..... Adresár určený pre klientské skripty a Vue.js elementy  
| routes ..... Adresár obsahujúci router  
| storage ..... Adresár určený pre súbory generované frameworkom  
| tests ..... Adresár určený pre aplikačné testy  
| vendor ..... Adresár obsahujúci nainštalované serverové balíčky
```

Po spustení príkazu `php artisan serve` sa spustil server na adrese `localhost:8000`. Po navštívení tejto adresy vo webovom prehliadači bolo možné vidieť predvolenú úvodnú stránku Laravelu.

#### 3.1.2 MVC architektúra

Laravel je implementovaný na základe architektonického vzoru MVC, ktorý je akronymom pre Model-View-Controller. Použitie tohto vzoru rozdeľuje aplikáciu do troch hlavných skupín — na modely, views a controllery. Nasledujúce podsekcie vysvetlia význam jednotlivých skupín.

##### 3.1.2.1 Model

Model (modelová trieda) obsahuje logiku aplikácie a všetko, čo do tejto logiky spadá. Môžu to byť rôzne výpočty, databázové dotazy, validácia prichádzajúcich dát a podobne. Funkcia modelu spočíva v prijatí parametrov, ktoré spracuje s možným využitím dát z databázy. Následne takto spracované dáta vráti späť metóde, ktorá danú funkciu modelu volala [21] (vlastný preklad). Platí, že pre každú tabuľku v databáze by mala byť vytvorená príslušná modelová trieda. Takáto modelová trieda je následne úzko spätá s danou tabuľkou, nakoľko poskytuje rôzne metódy pre prácu s dátami uloženými v tejto tabuľke [22] (vlastný preklad).

Laravel pristupuje k týmto dátam pomocou frameworku Eloquent, ktorý je určený pre objektovo-relačné mapovanie. Objektovo-relačné mapovanie (ORM) je technika prístupu k dátam uloženým v databáze. Táto technika spočíva v mapovaní riadkov príslušnej tabuľky v relačnej databáze do objektov, ktoré sú instance jednotlivých modelov. Táto technika funguje aj opačným smerom.

```
1  /**
2   * Create new animal
3   * @param $data
4   * @return Animal
5   * @throws Throwable
6   */
7  public static function createAnimal($data) {
8      $animal = new Animal($data);
9      $animal->creator_id = Auth::id();
10     $animal->saveOrFail();
11     return $animal->refresh();
12 }
```

Obr. 3.1: Ukážka modelovej triedy `Animal.php`

Na obrázku 3.1 je možné vidieť metódu `createAnimal`, ktorá je zodpovedná za vytvorenie nového zvierata. Táto metóda sa nachádza v modelovej triede



`Animal`, ktorá je modelovou triedou tabuľky `animals`. V prvom riadku tela metódy prebieha vytvorenie novej instance modelu, ktorá je naplnená dátami nachádzajúcimi sa v parametri `$data`. Následne je parameter `creator_id` objektu `Animal` nastavený na identifikátora používateľa, od ktorého požiadavka na vytvorenie nového zvieräta prišla. Nie je náhoda, že názov tohto parametru je možné nájsť ako stĺpec v tabuľke `animals`. Zavolaním Eloquent metódy `saveOrFail` sa daný objekt pretransformuje do SQL príkazu určeného pre pridanie dát do tabuľky. Následne sa tento príkaz vykoná na pozadí. V prípade, že uloženie objektu do databázy z rozličných dôvodov zlyhá, táto metóda vráti výnimku. Nakoniec sa zavolaním metódy `refresh` aktualizuje daná instance modelu aktuálnymi dátami z databázy (o identifikátor zvieräta a systémové stĺpce).

#### 3.1.2.2 View

View je časť aplikácie, ktorá zodpovedá za vykreslenie obsahu používateľom. Jej obsahom je šablóna, obsahujúca HTML stránku a tagy nejakého značkovacieho jazyka, ktorý umožňuje do šablóny vkladať premenné, prípadne vykonávať iterácie a podmienky. View nie je len šablóna, ale zobrazovač výstupu obsahujúca minimálne množstvo logiky potrebnej pre výpis dát. Podobne ako model, view nevie, odkiaľ mu dáta prišli, iba sa stará o ich finálne zobrazenie používateľovi [21] (vlastný preklad).

V prípade Laravelu sa používa v šablónach značkovací jazyk Blade. Blade umožňuje nie len vytvárať jednotlivé šablóny, ale aj komponenty, ktoré môžu byť použité viackrát v rôznych šablónach. Tieto šablóny sa nachádzajú v adresári `resources/views`.

```
1 <section id="breeding-info" class="bold">
2     <span class="breeding_available">
3         @isset($isAnimalAvailableForBreeding)
4             {{ $isAnimalAvailableForBreeding ?
5                 'Chov povolen' : 'Chov není povolen'
6             }}
7         @endisset
8     </span>
9     <p class="breeding_limitation">
10        @isset($animalBreedingLimitation)
11            {{ !empty($animalBreedingLimitation()) ?
12                ('Poznámka: ' . $animalBreedingLimitation) : ''
13            }}
14        @endisset
15    </p>
16 </section>
```

Obr. 3.2: Ukážka šablóny `animal_overview.blade.php`

Na obrázku 3.2 sa nachádza ukážka časti šablóny, ktorá je zodpovedná za vygenerovanie preukazu obsahujúci detaily zvierata. Ako si je možné všimnúť, obsah tejto šablóny sa skladá z použitia HTML jazyka pre definovanie štruktúry preukazu, zo špeciálnych Blade príkazov ako `isset` a `endisset`, a z funkcie s premennými v syntaxe jazyka PHP.

### 3.1.2.3 Controller

Controller je poslednou časťou MVC, ktorá ozrejní funkčnosť tohto vzoru. Jedná sa o prostredníka, s ktorým komunikuje používateľ, model i view. Tieto komponenty drží pohromade a zabezpečuje komunikáciu medzi nimi [21] (vlastný preklad). Samotné controllery aplikácie sa nachádzajú v adresári `app/Http/Controllers`.

```
1  /**
2   * Show view with animal's template for .pdf export
3   *
4   * @param int $id
5   * @return Factory|View
6   * @throws AuthorizationException
7   */
8  public function export(int $id) {
9      $animal = Animal::getAnimalForPdf($id);
10
11     // Check, whether user is authorized to view this page
12     $this->authorize('canExportAnimal', $animal);
13
14     $litter = $genealogy = null;
15
16     // Is animal part of some litter?
17     if (isset($animal->litter_id)) {
18         $litter = Litter::getLitter($animal->litter_id);
19         $genealogy = Animal::getGenealogy($litter, 0);
20     }
21
22     (...)
23
24     return view('generated_docs.pdf', [
25         'animal' => $animal,
26         'litter' => $litter,
27         'genealogy' => $genealogy,
28         'orientation' => $orientation
29     ]);
30 }
```

Obr. 3.3: Ukážka controlleru `AnimalController.php`

Na obrázku 3.3 sa nachádza ukážka metódy `export`. Z obrázku je možné vidieť volanie metód modelov `Animal` a `Litter`. Po tom čo controller dostane dáta z jednotlivých metód modelov, sú tieto dáta predané view, ktorá ich následne doplní do šablóny `pdf`. Nakoniec controller šablónu s dátami vráti,

### 3. IMPLEMENTÁCIA

---

a framework sa postará o jej odoslanie do klientského prehliadača, ktorý danú šablónu zobrazí používateľovi.

Avšak, na akom základe sa spustí táto metóda? Odpoveďou na túto otázku je časť aplikácie zvaná „Router“. Router je zodpovedný za spustenie metód controllerov na základe prijatého HTTP požiadavku (buď iniciovaného používateľom navštívením danej adresy alebo skriptom, ktorý pošle požiadavku na danú adresu). Na základe URL adresy požiadavku router rozhodne, ktorú metódu daného controlleru spustí.

Router aplikácie sa nachádza v priečinku `routes`.

```
1 Route::middleware('auth:sanctum')->group(function() {
2     Route::get(
3         '/animals/{id}/export',
4         'AnimalsController@export'
5     )->name('animals.id.export.get');
6
7     (...);
8 });
```

Obr. 3.4: Ukážka routeru web.php

Na obrázku 3.4 je zobrazená ukážka routeru. V prípade, ak HTTP požiadavka prichádzajúca na server bude mať adresu v tvare „`https://domena.sk/animals/<id>/export`“ (<id> sa nahradí ľubovoľným číslom), router spustí metódu `export` v controlleri `AnimalsController`.

## 3.2 Klientská časť

Klientská časť aplikácie sa skladá prevažne z implementácie jednotlivých stránok, ktoré sa skladajú z viacerých nezávislých komponent, ktoré spolu komunikujú. Tieto stránky sú zobrazované klientskym frameworkom `Vue.js`, ktorý sa stará o správne zobrazenie jednotlivých stránok na základe adresy, na ktorej sa používateľ nachádza.

Spustením príkazu `npm install vue` sa nainštaloval framework `Vue.js` spolu s jej závislosťami do adresára `node_modules`, ktoré sú potrebné pre beh samotného frameworku.

Komponent, ktorý zobrazí stránku zvieráťa, vyzerá nasledovne:

```
1 <template>
2   <div id="animal-page" class="columns section">
3     <left-panel
4       :animal="animal"
5       :user="user">
6   </left-panel>
7   <animal-information
8     :animal-id="animalId"
9     :animal="animal"
10    :is-loading="isLoading"
11    :user="user">
12 </animal-information>
13 <right-panel
14   :animal-id="animalId"
15   :is-loading="isLoading"
16   :animal="animal"
17   :user="user"
18   :key="animalId">
19 </right-panel>
20 </div>
21 </template>
```

Obr. 3.5: Ukážka šablóny komponentu AnimalPage.vue

Komponenty vo Vue.js sa definujú pomocou štandardných HTML značiek. Okrem iného jednotlivým komponentom môžu byť predané premenné rôzneho typu. Vue.js taktiež podporuje vkladanie existujúcich komponentov do výsledného komponentu. Týmto spôsobom sa dá dosiahnuť znovupoužiteľnosť jednotlivých komponentov naprieč veľkým množstvom stránok.

V horeuvedenom prípade komponent `AnimalPage` obsahuje tri komponenty:

- `LeftPanel`
- `AnimalInformation`
- `RightPanel`

Samotný komponent obsahuje nie len jej deklaráciu v rámci HTML jazyka, ale aj biznis logiku v jazyku JavaScript. Väčšinou sa jedná o metódy, ktoré buď načítavajú dáta zo servera, alebo reagujú na používateľský vstup.

### 3. IMPLEMENTÁCIA

---

V prípade `AnimalPage` komponentu sa jedná o nasledujúcu logiku:

```
1 <script>
2   export default {
3     name: "AnimalPage",
4     components: {RightPanel, LeftPanel, AnimalInformation},
5     data() {
6       return {
7         animal: null,
8         isLoading: false,
9       }
10    },
11    computed: {
12      animalId() {
13        return Number(this.$route.params.animal);
14      }
15    },
16    methods: {
17      async loadData() {
18        this.isLoading = true;
19        const url = `/api/animals/${this.animalId}`;
20        try {
21          const request = await axios.get(url);
22          this.animal = request.data;
23        } catch (e) {
24          this.$buefy.toast.open({
25            message:
26              this.$t('animal.index.page_load_fail'),
27            type: 'is-danger'
28          });
29          throw e;
30        } finally {
31          this.isLoading = false;
32        }
33      }
34    },
35    async created() {
36      await this.loadData();
37    }
38  }
39 </script>
```

Obr. 3.6: Ukážka biznis logiky `AnimalPage.vue` komponentu

Hodnota atribútu `name` určuje meno daného komponentu naprieč celou klientskou aplikáciou. Obsah objektu `components` deklaruje použité komponenty v rámci `AnimalPage`. Metóda `data` vracia objekt s reaktívnymi atribútami — pri zmene jednotlivých atribútov Vue.js notifikuje komponent o zmene atribútu. V prípade, že sa daný atribút používa v komponente, Vue.js zaistí jeho znovuykreslenie. Dané atribúty sa taktiež používajú na ukladanie stavu jednotlivých komponentov. Objekt `methods` obsahuje biznis metódy, ktoré môžu byť volané inými metódami. V tomto prípade sa používateľská metóda `loadData` spustí po tom, čo sa spustí exekúcia metódy `created`. Metóda `created` je ale systémová – konkrétne sa exekuuje po vytvorení daného komponentu na stránke.

Zjednodušený životný cyklus tohto komponentu bude nasledovný:

1. Vue.js vytvorí a zobrazí komponent `AnimalPage`.
2. Po vytvorení tohto komponentu bude automaticky spustená metóda `created`, ktorá vzápätí zavolá metódu `loadData`.
3. Daná metóda sa postará o načítanie informácií o zvierati s identifikátorom `animalId`.
4. V prípade zlyhania načítavania dát sa zobrazí chybová hláška, v opačnom prípade sa informácie o zvierati uložia do premennej `animal`.
5. Obsah premennej `animal` sa spropaguje do komponentov, ktoré prijímajú objekt `animal`.
6. Ak daný komponent závisí pri jeho zobrazení od obsahu premennej `animal`, bude znovu vykreslený.

Po zmene jednotlivých komponentov je potrebné spustiť príkaz `npm run prod`, ktorý jednotlivé komponenty skompiluje do formy zrozumiteľnej prehliadaču. Pri vývoji aplikácie je však komfortnejšie použiť príkaz `npm run watch`, ktorý na pozadí zaistí automatickú kompiláciu komponentov po každej ich zmene. Po následnom znovunačítaní stránky prehliadača (`localhost:8000`) sa zobrazia najnovšie zmeny aplikácie.

### 3.3 Migrácia dát

Po implementácii novej aplikácie bolo potrebné zmigrovať existujúce dáta do novej aplikácie. Proces migrácie spočíval vo vytvorení starých tabuliek s dátami v databáze novej aplikácie. Následne boli pre tieto tabuľky vytvorené modelové triedy, ktoré obsahujú logiku pre transformáciu dát zo starých do nových tabuliek. Tieto modelové triedy je možné nájsť v priečinku `app/ImportModels`.

Každá modelová trieda obsahuje metódu `import`. Tieto metódy sú volané v jednotlivých metódach `controllera ImportController`, ktoré sú zaregistrované v `routri web.php` 3.7.

```
1 Route::get(
2     '/import/contacts',
3     'ImportController@importContacts'
4 )->name('import.contacts.get');
5 Route::get(
6     '/import/czkp_animals',
7     'ImportController@importCZKPAanimals'
8 )->name('import.czkp_animals.get');
9 Route::get(
10    '/import/czkp_litters',
11    'ImportController@importCZKPLitters'
12 )->name('import.czkp_litters.get');
13 Route::get(
14    '/import/czkp_animals/litters',
15    'ImportController@connectAnimalWithLitter'
16 )->name('import.czkp_animals.litters.get');
17 Route::get(
18    '/import/czkp_babies',
19    'ImportController@importCZKPBabies'
20 )->name('import.czkp_babies.get');
21 Route::get(
22    '/import/pp_information',
23    'ImportController@importPPInformation'
24 )->name('import.pp_information.get');
```

Obr. 3.7: Ukážka volania `import` metód v `routri web.php`

Pred importom dát muselo byť veľké množstvo záznamov manuálne očistených, z dôvodu nekonzistentnosti mien majiteľov/chovateľov, ich kontaktných údajov a príslušnosti k jednotlivým chovným staniciam medzi súčasnými tabuľkami. Táto nekonzistentnosť dát bola spôsobená ich ukladaním do nenor-



malizovaných tabuliek. Čistenie dát zahŕňalo manuálnu kontrolu a dodatočnú úpravu vyše 3500 záznamov v súčasnej databáze.

Ak by sa záznamy neočistili, nebolo by možné nie len spárovať jednotlivé zvieratá a vrhy k ich majiteľom a chovateľom, ale ani importovať týchto ľudí do tabuľky `people` a spárovať ich s jednotlivými chovnými stanicami z tabuľky `stations`.



---

# Testovanie

Testovanie aplikácie je nevyhnutnou súčasťou pre potvrdenie, že implementovaná funkcionálna funguje podľa špecifikácie. Táto kapitola popisuje jednotkové a integračné testy, ktoré boli použité na otestovanie správnej funkcionality aplikácie. Následne sa kapitola venuje procesu kontroly a overenia správnosti importu existujúcich dát do novej databázy.

## 4.1 Jednotkové testy

Jednotkové testy sú testy, ktoré testujú jednotlivé komponenty softvéru. Účelom týchto testov je overiť, či každý komponent softvéru funguje tak, ako je navrhnutý. Jednotka je najmenšia testovateľná súčasť akéhokoľvek softvéru. Zvyčajne má jeden alebo niekoľko vstupov a zvyčajne jeden výstup [23] (vlastný preklad).

Nakoľko biznis logika tejto aplikácie nie je rozsiahla, boli namiesto jednotkových testov vytvorené integračné testy.

## 4.2 Integračné testy

Integračné testovanie je úroveň testovania aplikácie, pri ktorej sa jednotlivé komponenty kombinujú a testujú ako skupina. Účelom tejto úrovne testovania je odhaliť poruchy v interakcii medzi integrovanými komponentami [24] (vlastný preklad).

Samotné integračné testy sa zameriavajú na testovanie komunikácie medzi klientom a serverom, pri ktorých sa overuje aktuálna a očakávaná odpoveď servera. Počas behu jednotlivých testov dochádza k overeniu autentifikácie používateľa, prístupu do databázy, respektíve overeniu práv používateľa k danej akcii.

#### 4. TESTOVANIE

---

Príkazom `php artisan test` sa spustia všetky implementované testy. Tento príkaz je potrebné spustiť z hlavného adresára webovej aplikácie.

```
1 public function testUserDelete() {
2     // Create new user
3     $user = factory(User::class)->create();
4
5     // Deleting unauthenticated should result in 401
6     $this->deleteJson(
7         route('users.destroy',
8             ['user' => $user->id])
9     )->assertStatus(401);
10
11    // Login as user
12    $this->loginAsUser();
13
14    // Deleting as user should result in 403
15    $this->deleteJson(
16        route('users.destroy',
17            ['user' => $user->id])
18    )->assertStatus(403);
19
20    // Login as admin
21    $this->loginAsAdmin();
22
23    // Now the deletion should be alright
24    $this->deleteJson(
25        route('users.destroy',
26            ['user' => $user->id])
27    )->assertStatus(204);
28 }
```

Obr. 4.1: Ukážka integračného testu zmazania používateľa

Na ukážke testu 4.1 je zobrazený priebeh testu zmazania používateľa. V tomto teste je využitá kontrola prihlásenia a následná kontrola práv. Ak je používateľ odhlásený, respektíve nemá dostatočné práva pre zmazanie používateľa, server by mal vrátiť odpoveď s HTTP status kódom 401, resp. 403. Po prihlásení sa ako administrátor by malo byť možné zmazať vytvoreného používateľa poslaním DELETE požiadavky na server.

Na podobnom princípe sú postavené ostatné integračné testy. Integračné testy boli vytvorené pre všetky entity, ktoré sú dostupné cez API servera. Tieto testy sa nachádzajú v adresári `tests/Feature`.

Pri jednotlivých testoch sa nevyužíva MySQL databáza ako v prípade samotnej webovej aplikácie, ale databáza SQLite, ktorá je nakonfigurovaná tak, aby fungovala v pamäti počítača. Jej podrobnú konfiguráciu spolu s konfiguráciou testov je možné nájsť v súbore `phpunit.xml` nachádzajúci sa v hlavnom adresári webovej aplikácie.

### 4.3 Overenie správnosti importu

Nakoľko dáta museli byť pred samotným importom očistené z dôvodu ich nekonzistencie a duplicity, nebolo možné overiť správnosť importu oproti originálnym dátam. Preto som pristúpil ku kontrole importovaných dát voči očisteným dátam. Táto kontrola bola vykonaná porovnaním počtu záznamov importovaných a očistených dát

Zo začiatku boli niektoré informácie zle importované — táto skutočnosť sa týkala najmä previazanosti záznamov medzi jednotlivými tabuľkami. Následné opravy dát v databáze a v logike importu tieto problémy odstránili. Keďže dáta prešli takýmto náročnejším procesom importu, nie je možné 100% garantovať kompletnú zhodu s originálnymi dátami. Pri prípadnom výskyte tejto nekonzistencie musia byť dotknuté dáta manuálne upravené.



---

## Záver

Analytická časť práce sa zamerala na analýzu súčasnej aplikácie, prevažne z pohľadu jej architektúry a štruktúry používanej databázy. Následne boli zozbierané funkčné a nefunkčné požiadavky na základe funkcionality doterajšej aplikácie. Nakoľko aplikáciu budú používať ľudia s rôznymi právami k rôznym častiam aplikácie, bolo potrebné implementovať správu rolí a práv. Nadväzujúce prípady použitia definovali presnejšiu podobu implementovanej aplikácie. V časti návrhu aplikácie boli porovnané MPA a SPA architektúry, ktoré sa používajú pri implementácii webových aplikácií. Na základe ich porovnania a doterajších skúseností bola vybraná SPA architektúra, ktorá spolu s následnou voľkou technológií bola implementačným základom pre vznikajúcu aplikáciu. Okrem výberu architektúry a voľby použitých technológií bolo nevyhnutné normalizovať tabuľky v existujúcej databáze. Tieto normalizované tabuľky poslúžili ako základ navrhnutých tabuliek v novom databázovom modeli aplikácie, ktorý bol navrhnutý s cieľom predísť duplikácii dát.

Implementačná časť obsahuje detaily ohľadom implementácie serverovej a klientskej časti, s bližším popisom MVC architektúry. Okrem samotnej implementácie bolo nutné premigrovať existujúce dáta do novej databázy, aby aplikáciu mohli ihneď začať používať existujúci používatelia starej aplikácie. Nakoľko boli dáta z veľkého množstva v existujúcej databáze duplicitné, a ich očistenie nebolo možné zautomatizovať, musela byť veľká časť dát očistená manuálne.

Po implementácii samotnej aplikácie nasledovalo jej otestovanie, aby sa overilo, či aplikácia neobsahuje implementačné chyby a či zodpovedá zadaným požiadavkám. V neposlednom rade nasledovala kontrola správnosti importu existujúcich dát do databázy novej aplikácie. Keďže originálne dáta museli byť pred importom očistené, nebolo možné so 100% istotou overiť správnosť importu týchto dát voči originálnym dátam.

Výsledná webová aplikácia pokrýva všetky funkčné a nefunkčné požiadavky, ktoré boli na ňu kladené v analytickej časti práce.





---

## Literatúra

- [1] Maleček, L.: *Plemenná kniha*. [software] [přístup 2020-05-04]. Dostupné z: <http://www.czkp.cz/plemenna-kniha/>
- [2] *Pros and Cons of PHP Programming Language*. [online], [cit. 2020-05-04]. Dostupné z: <https://www.pros-cons.net/pros-and-cons-of-php-programming-language/>
- [3] *What is MySQL?* [online], [cit. 2020-05-04]. Dostupné z: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>
- [4] Arlow, J.; Neustadt, I.: *UML 2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press, 2007, ISBN 978-80-251-1503-9, 95 s.
- [5] *PHP: Releases*. [online], [cit. 2020-05-04]. Dostupné z: <https://www.php.net/releases/index.php>
- [6] *MySQL 5.6 Release Notes*. [online], [cit. 2020-05-04]. Dostupné z: <https://dev.mysql.com/doc/relnotes/mysql/5.6/en/>
- [7] Adomavicius, S.: *Which and when to use: SPA vs MPA - JAXenter*. [online], 2020, [cit. 2020-05-05]. Dostupné z: <https://jaxenter.com/spa-vs-mpa-160963.html>
- [8] Quanit, M.: *Single Page Application vs. Multi-page Application*. 2020, [cit. 2020-05-05]. Dostupné z: <https://dev.to/mquanit/single-page-application-vs-multi-page-application-p2m>
- [9] Kukhnavets, P.: *SPA vs MPA: Key Difference Between Single-Page and Multi-Page Apps*. [online], 2020, [cit. 2020-05-05]. Dostupné z: <https://cuspy.io/blog/spa-vs-mpa-what-is-the-difference/>
- [10] Melnik, I.: *Single Page Application (SPA) vs Multi Page Application (MPA): Pros and Cons*. [online], 2020, [cit. 2020-05-05].

- Dostupné z: <https://merehead.com/blog/single-page-application-vs-multi-page-application/>
- [11] Singh, V.: *What is Frameworks? [Definition] Types of Frameworks*. [online], 2020, [cit. 2020-05-06]. Dostupné z: <https://hackr.io/blog/what-is-frameworks>
- [12] *HTML: Hypertext Markup Language*. [online], 2020, [cit. 2020-05-06]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTML>
- [13] *What is CSS3?* [online], 2020, [cit. 2020-05-07]. Dostupné z: <https://www.educba.com/what-is-css3/>
- [14] Belyh, A.: *JavaScript*. [online], 2020, [cit. 2020-05-07]. Dostupné z: <https://www.cleverism.com/lexicon/javascript/>
- [15] *Guide - Introduction*. [online], 2020, [cit. 2020-05-08]. Dostupné z: <https://vuejs.org/v2/guide/>
- [16] Chapple, M.: *The Basics of Normalizing a Database*. [online], 2020, [cit. 2020-05-08]. Dostupné z: <https://www.lifewire.com/database-normalization-basics-1019735>
- [17] Foster, E.; Godbole, S.: *Database Systems: A Pragmatic Approach*. Berkeley, CA: Apress, druhé vydání, 2016, ISBN 978-1-4842-1191-5, 98 s.
- [18] *Normalization of Database*. [online], 2020, [cit. 2020-05-18]. Dostupné z: <https://www.studytonight.com/dbms/database-normalization.php>
- [19] *What is Second Normal Form?* [online], 2020, [cit. 2020-05-19]. Dostupné z: <https://www.studytonight.com/dbms/second-normal-form.php>
- [20] *Third Normal Form (3NF)*. [online], 2020, [cit. 2020-05-19]. Dostupné z: <https://www.studytonight.com/dbms/third-normal-form.php>
- [21] Čápka, D.: *MVC architektura*. [online], 2020, [cit. 2020-05-20]. Dostupné z: <https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor>
- [22] *Eloquent: Getting Started*. [online], 2020, [cit. 2020-05-20]. Dostupné z: <https://laravel.com/docs/7.x/eloquent>
- [23] *Unit Testing*. [online], 2020, [cit. 2020-05-21]. Dostupné z: <http://softwaretestingfundamentals.com/unit-testing/>
- [24] *Integration Testing*. [online], 2020, [cit. 2020-05-22]. Dostupné z: <http://softwaretestingfundamentals.com/integration-testing/>

## Zoznam použitých skratiek

- 1NF** 1. normálová forma
- 2NF** 2. normálová forma
- 3NF** 3. normálová forma
- AJAX** Asynchronous JavaScript and XML
- API** Application programming interface
- CSS** Cascading Style Sheets
- ČKP** Český klub potkanů z. s.
- HTML** HyperText Markup Language
- HTTP** Hypertext Transfer Protocol
- JSON** JavaScript Object Notation
- MPA** Multi-page aplikácia
- MVC** Model-View-Controller
- NV** Nulový vrh
- ORM** Objektovo-relačné mapovanie
- PDF** Portable Document Format
- PHP** Personal Home Page Tools
- PP** Preukaz pôvodu
- SOCHP** Specializovaná organizace chovatelů potkanů

## A. ZOZNAM POUŽITÝCH SKRATIEK

---

**SPA** Single-page aplikácia

**SQL** Structured Query Language

**VP** Výpis predkov

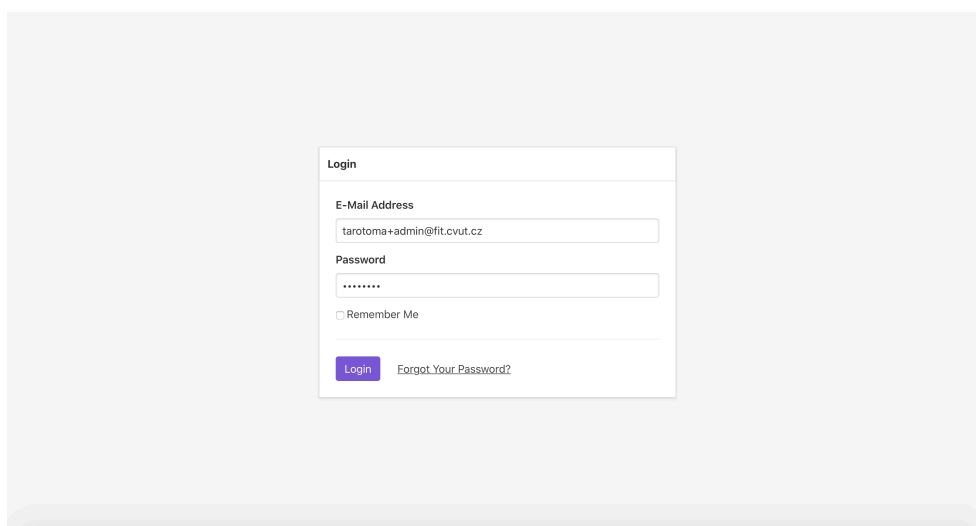
**UC** Use Case

**URL** Uniform Resource Locator

---

## Používateľské rozhranie

V tejto prílohe sa nachádzajú snímky obrazoviek finálnej webovej aplikácie s ich popisom. Dáta, ktoré je možné vidieť na jednotlivých snímkoch, boli vytvorené PHP knižnicou **Faker**, ktorá umožňuje generovanie falošných dát so špecifikáciou ich formy. Nakoľko je webová aplikácia responzívna, na konci prílohy sa nachádzajú snímky obrazovky aplikácie pri jej mobilnom zobrazení.



Obr. B.1: Screenshot prihlasovacej obrazovky

## B. POUŽÍVATELSKÉ ROZHRAŇIE

The screenshot shows the application interface for 'Zvířata'. At the top, there is a purple navigation bar with the text 'Databáze zvířat', 'Zvířata', 'Vrhy', and 'Uživatelé'. Below the navigation bar, the title 'Zvířata' is centered. To the right, a user profile dropdown menu is open, showing 'Přihlášen jako: Admin' and a 'Odhlásit se' button. Below the title, there are four search filters: 'Zadejte jméno zvířete', 'Zadejte příjmení zvířete', 'Zadejte jméno chovatele', and 'Zadejte jméno majitele'. There are also two buttons: 'Zobraz moje zvířata' and 'Filtrovat'. Below the filters, there is a table with the following columns: 'Jméno', 'Chovatel', 'Majitel', 'Datum narození', 'Pohlaví', and 'Registrován'. The table contains 10 rows of data. The 'Registrován' column has buttons that are either green ('Ano') or red ('Ne').

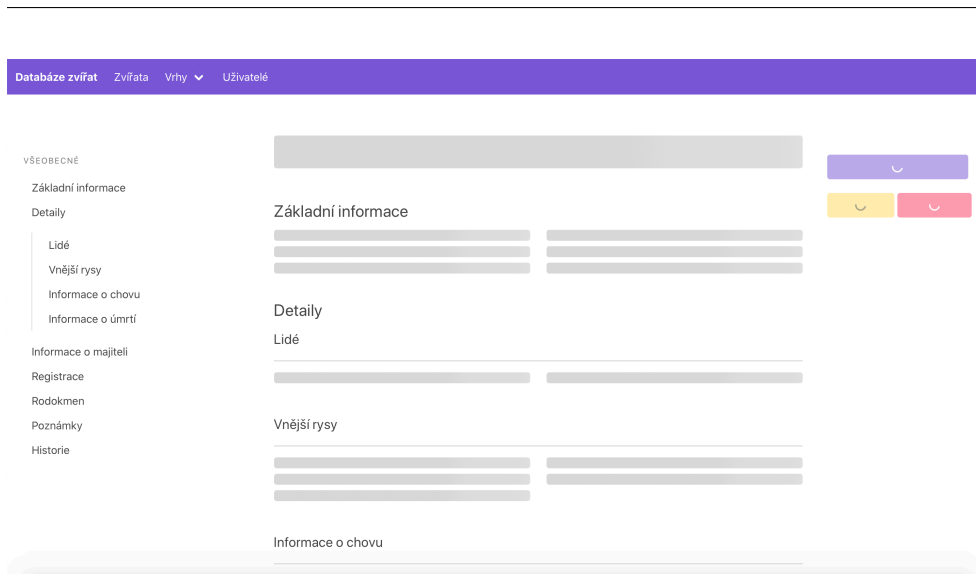
Jméno	Chovatel	Majitel	Datum narození	Pohlaví	Registrován
Prof. Dimitri Kovalis	Barney Sipes Jr.	Kiarra Conn	07.09.1982	♂ Samec	Ano
Noemi Miller	Bruce Marvin	Filomena Greenholt	28.03.1998	♂ Samec	Ano
Darrell Lubowitz	Prof. Hailee Corkery	Mrs. Bette Walter	-	♂ Samec	Ne
Amir Lakin	Larue Fadel	Kiarra Conn	30.12.1993	♂ Samec	Ano
Westley Goyette	Mr. Peyton McClure IV	Sonia King	01.10.2015	♂ Samec	Ano
Michaela Jacobi MD	Registrator	Betty Hoeger DVM	-	♂ Samec	Ano
Shanelle Schultz	Kiarra Conn	Hellen Bernier	-	♂ Samec	Ano
Juwan Oberbrunner	Dr. Mortimer Treutel	Nigel Bernhard	-	♂ Samec	Ne
Fern Barrows	Alena Fritsch DVM	Lucienne DuBuque V	-	♂ Samec	Ne

Obr. B.2: Screenshot aplikácie po prihlásení s otvoreným menu

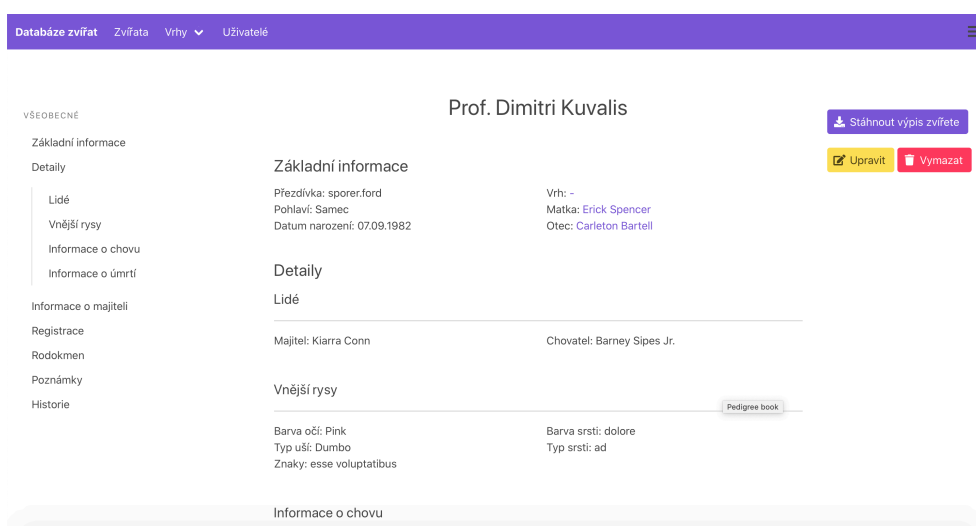
The screenshot shows the application interface for 'Zvířata'. At the top, there is a purple navigation bar with the text 'Databáze zvířat', 'Zvířata', 'Vrhy', and 'Uživatelé'. Below the navigation bar, the title 'Zvířata' is centered. Below the title, there are four search filters: 'Zadejte jméno zvířete', 'Zadejte příjmení zvířete', 'Zadejte jméno chovatele', and 'Zadejte jméno majitele'. There are also two buttons: 'Zobraz moje zvířata' and 'Filtrovat'. Below the filters, there is a table with the following columns: 'Jméno', 'Chovatel', 'Majitel', 'Datum narození', 'Pohlaví', and 'Registrován'. The table contains 10 rows of data. The 'Registrován' column has buttons that are either green ('Ano') or red ('Ne'). A dropdown menu is open over the 'Registrován' column, showing 'Upravit' and 'Vymazat' options.

Jméno	Chovatel	Majitel	Datum narození	Pohlaví	Registrován
Prof. Dimitri Kovalis	Barney Sipes Jr.	Kiarra Conn	07.09.1982	♂ Samec	Ne
Noemi Miller	Bruce Marvin	Filomena Greenholt	28.03.1998	♂ Samec	Ano
Darrell Lubowitz	Prof. Hailee Corkery	Mrs. Bette Walter	-	♂ Samec	Ano
Amir Lakin	Larue Fadel	Kiarra Conn	30.12.1993	♂ Samec	Ano
Westley Goyette	Mr. Peyton McClure IV	Sonia King	01.10.2015	♂ Samec	Ano
Michaela Jacobi MD	Registrator	Betty Hoeger DVM	-	♂ Samec	Ano
Shanelle Schultz	Kiarra Conn	Hellen Bernier	-	♂ Samec	Ano
Juwan Oberbrunner	Dr. Mortimer Treutel	Nigel Bernhard	-	♂ Samec	Ne
Fern Barrows	Alena Fritsch DVM	Lucienne DuBuque V	-	♂ Samec	Ne

Obr. B.3: Screenshot aplikácie po prihlásení s otvoreným dropdown menu

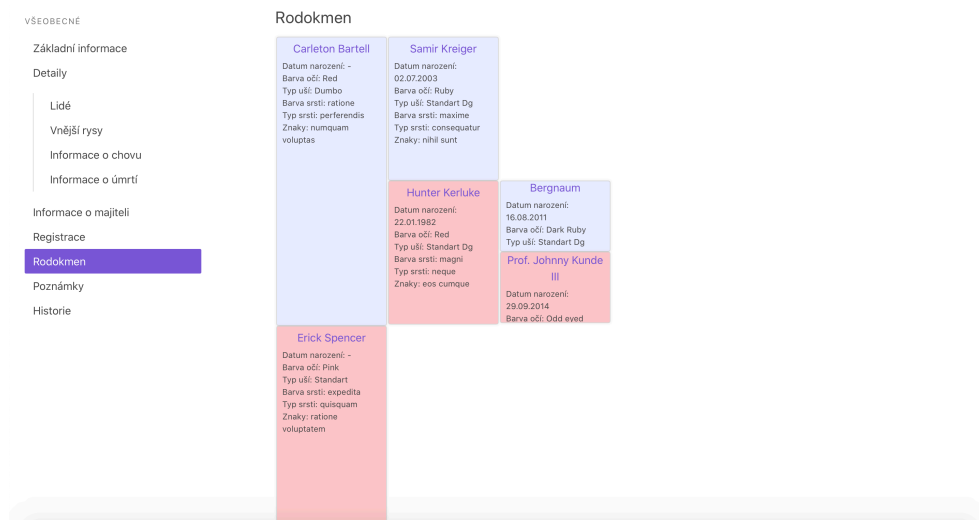


Obr. B.4: Screenshot načítavania zvoleného zvierťa

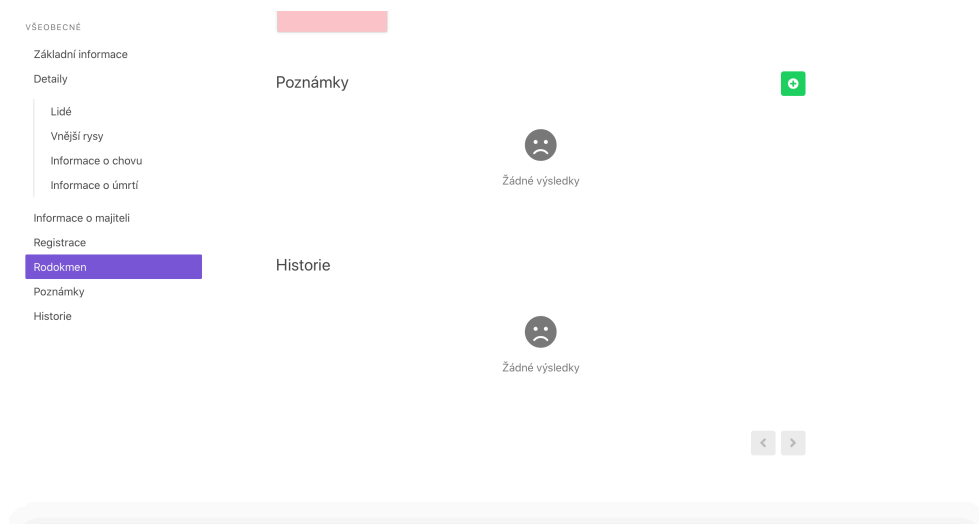


Obr. B.5: Screenshot načítaného zvoleného zvierťa

## B. POUŽÍVATELSKÉ ROZHRAŇIE

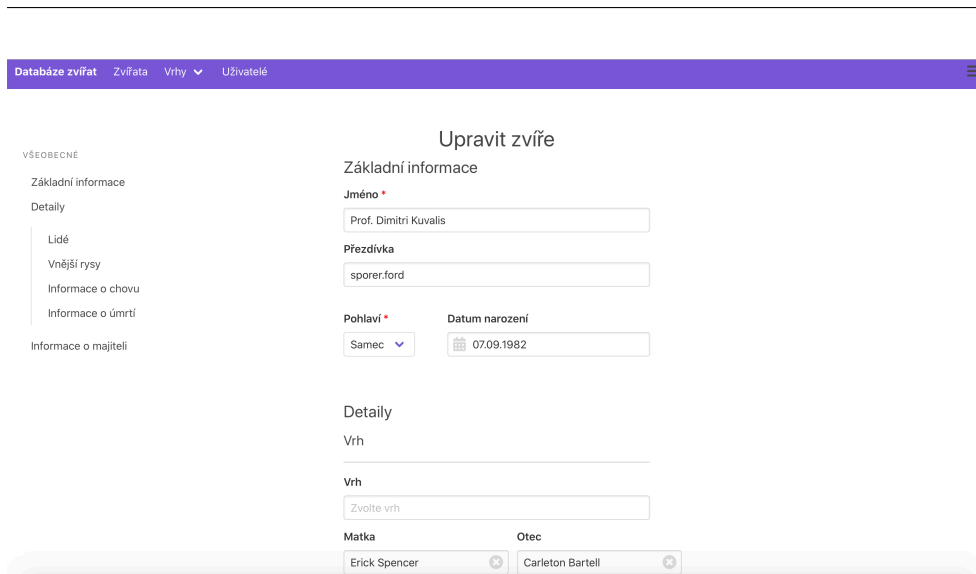


Obr. B.6: Screenshot zobrazenia rodokmeňu zvoleného zvierata

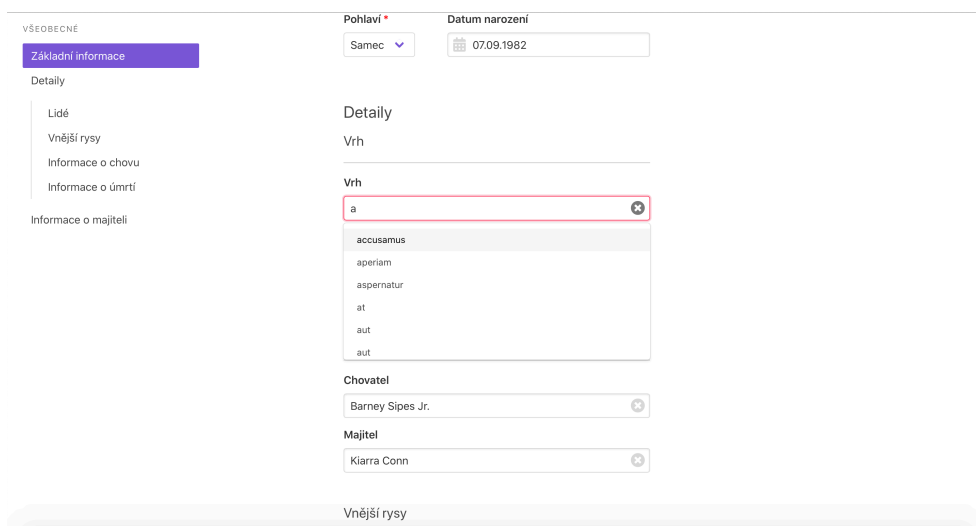


Obr. B.7: Screenshot zobrazenia poznámok a histórie zvoleného zvierata





Obr. B.8: Screenshot zobrazenia sekcie editácie zvierťa



Obr. B.9: Screenshot zobrazenia ponuky voľby vrhu pri editácii zvierťa

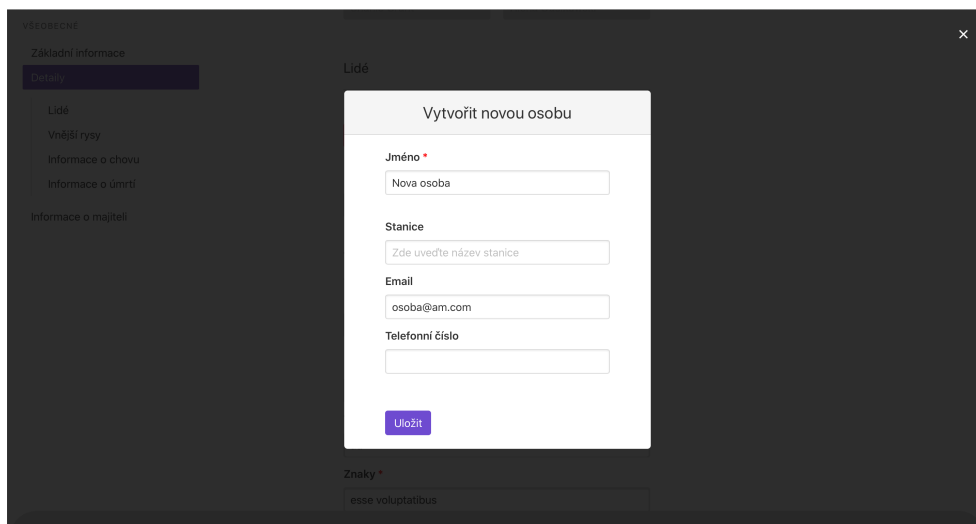
## B. POUŽÍVATELSKÉ ROZHRAŇIE

The screenshot shows a web application interface for editing an animal's details. On the left, there is a sidebar with a menu containing: 'VŠEOBECNÉ', 'Základní informace', 'Detaily', 'Lidé', 'Vnější rysy', 'Informace o chovu', 'Informace o úmrtí', and 'Informace o majiteli'. The 'Detaily' section is currently active. The main content area is titled 'Detaily' and contains several fields: 'Pohlaví' (Sex) set to 'Samec', 'Datum narození' (Date of birth) set to '07.09.1982', 'Vrh' (Breed) set to 'aspermatur', 'Matka' (Mother) set to 'Shanie Grant', and 'Otec' (Father) set to 'Alena Gusikowski'. Below these are fields for 'Lidé' (People), 'Chovatel' (Breeder) set to 'Barney Sipes Jr.', and 'Majitel' (Owner) set to 'Kiarra Conn'. At the bottom, there is a section for 'Vnější rysy' (External features).

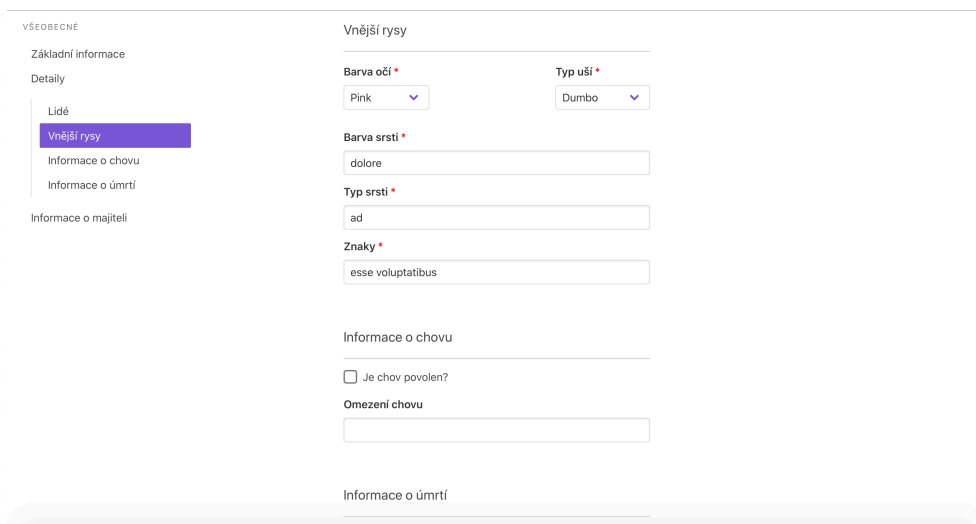
Obr. B.10: Screenshot aplikácie po zvolení vrhu pri editácii zvierata

The screenshot shows the same application interface, but now the 'Chovatel' (Breeder) field is selected. The 'Detaily' section is still active in the sidebar. The 'Chovatel' field is highlighted with a red border and contains the text 'Tomas'. Below this field, there is a message 'Žiadne výsledky' (No results) and a link 'Přidat nového chovatele' (Add new breeder). The 'Vnější rysy' (External features) section is expanded, showing fields for 'Barva očí' (Eye color) set to 'Pink', 'Typ uší' (Ear type) set to 'Dumbo', 'Barva srsti' (Coat color) set to 'dolore', 'Typ srsti' (Coat type) set to 'ad', and 'Znaky' (Signs) set to 'esse voluptatibus'.

Obr. B.11: Screenshot aplikácie zobrazujúci možnosť voľby chovateľa



Obr. B.12: Screenshot modálneho okna s ponukou vytvorenia novej osoby



Obr. B.13: Screenshot zobrazenia nasledujúcej sekcie pri editácii zvierata

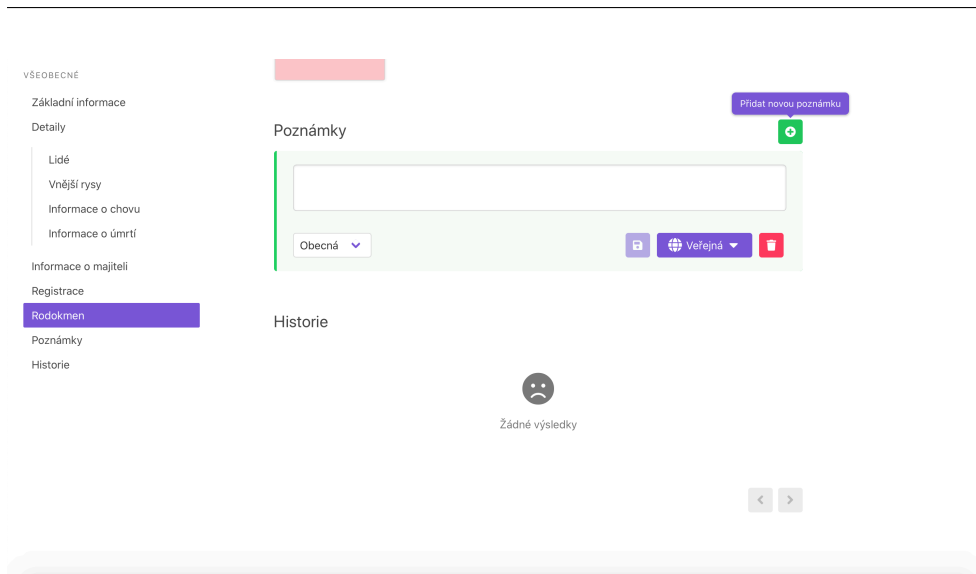
## B. POUŽÍVATELSKÉ ROZHRAŇIE

The screenshot shows a web application interface for editing an animal's record. On the left is a sidebar with a menu under the heading 'VŠEOBECNÉ'. The menu items are: 'Základní informace', 'Detaily', 'Lidé', 'Vnější rysy', 'Informace o chovu', 'Informace o úmrtí' (highlighted in purple), and 'Informace o majiteli'. The main content area is titled 'Informace o úmrtí' and contains several form fields: 'Datum úmrtí' with a date picker set to '03.11.2000', 'Důvod úmrtí' with a text input containing 'Aut id voluptates dolorum impedit aperiam reiciendis. Facer', 'Informace o majiteli' with a text input and a blue tooltip that says 'Tyto informace budou zobrazeny na vygenerovaných dokumentech', 'Jméno', 'Kontakt', and 'Číslo členského průkazu'. At the bottom center of the form area is a purple button labeled 'Uložit'.

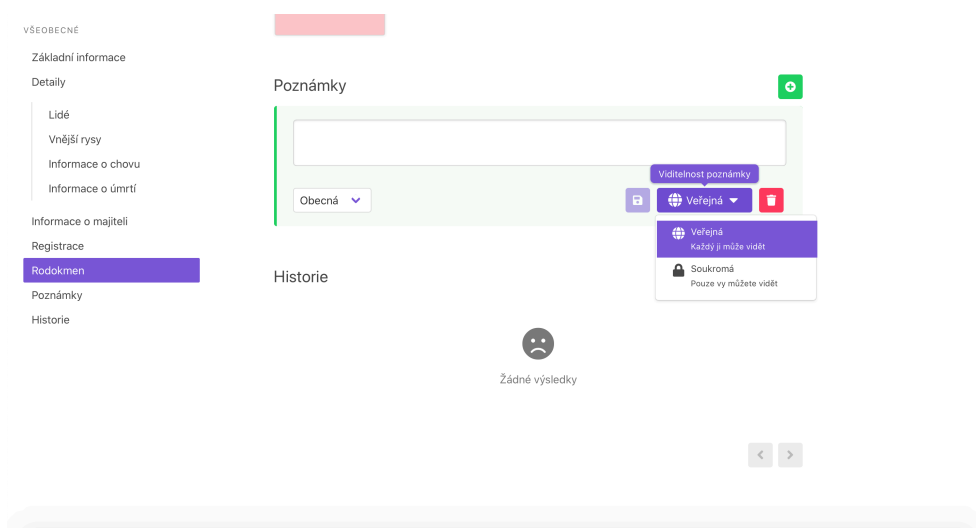
Obr. B.14: Screenshot zobrazenia poslednej sekcie pri editácii zvierata

The screenshot shows a modal dialog box titled 'Odstránení zvířete' (Deletion of animal) overlaid on a dark background. The dialog contains a red exclamation mark icon and the text: 'Opravdu chcete toto zvíře odstranit? Tuto akci nelze vrátit zpět.' (Are you really sure you want to delete this animal? This action cannot be undone). At the bottom of the dialog are two buttons: 'Cancel' and 'Odstranit zvíře' (Delete animal).

Obr. B.15: Screenshot zobrazenia modálneho okna pri zvolení zmazania zvierata

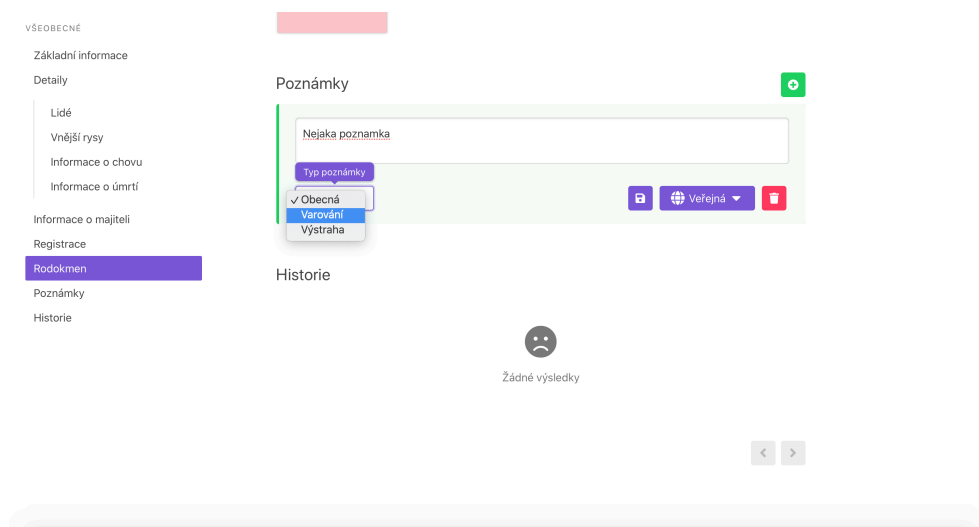


Obr. B.16: Screenshot zobrazenia poznámky po kliknutí na tlačidlo pridania novej poznámky

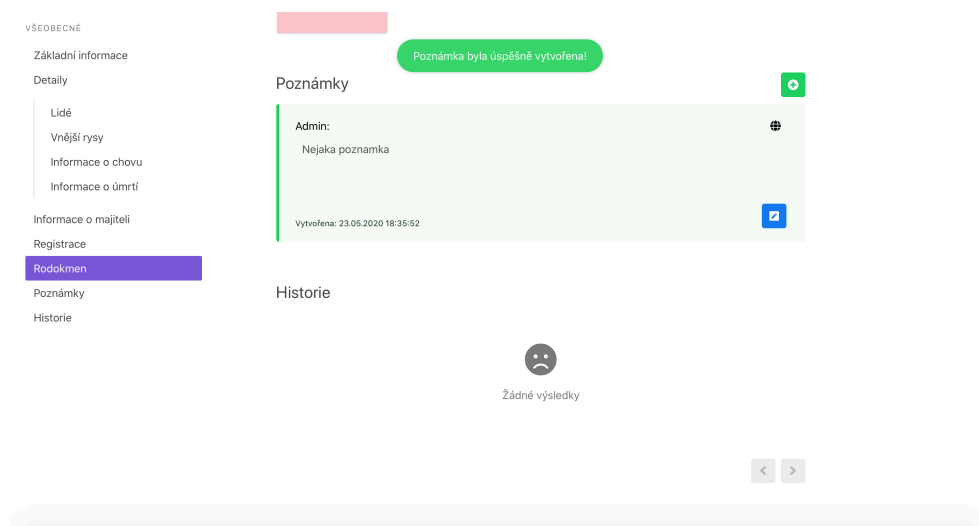


Obr. B.17: Screenshot zobrazenia viditelnosti poznámky

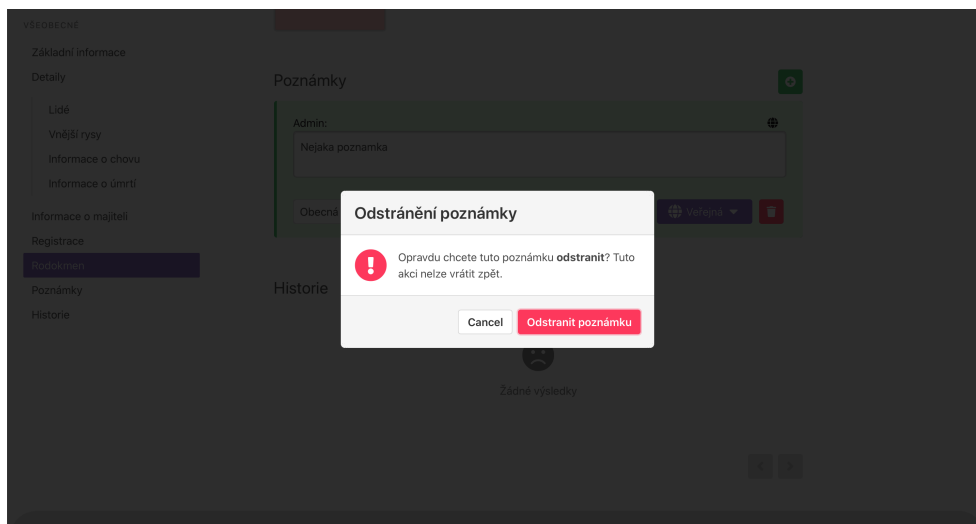
## B. POUŽÍVATELSKÉ ROZHRAŇIE



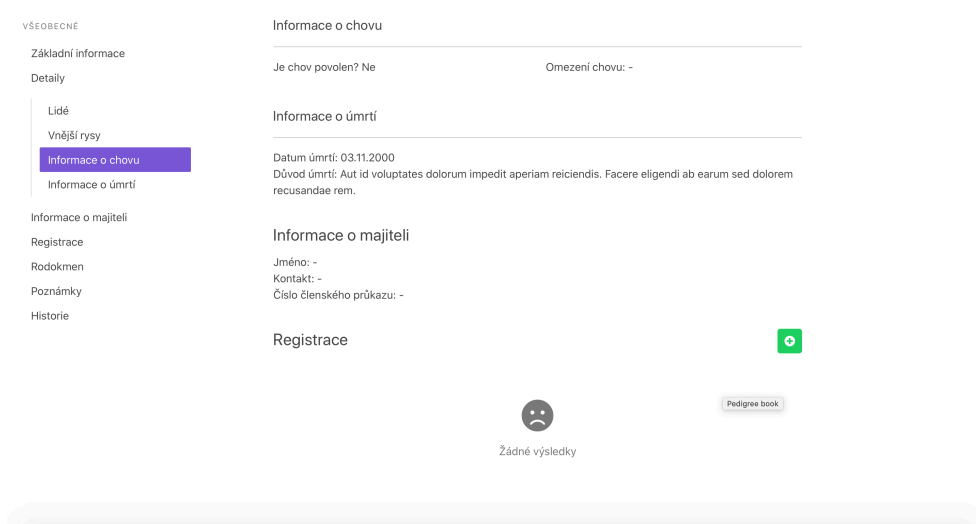
Obr. B.18: Screenshot zobrazenia typov poznámky



Obr. B.19: Screenshot aplikácie po vytvorení poznámky

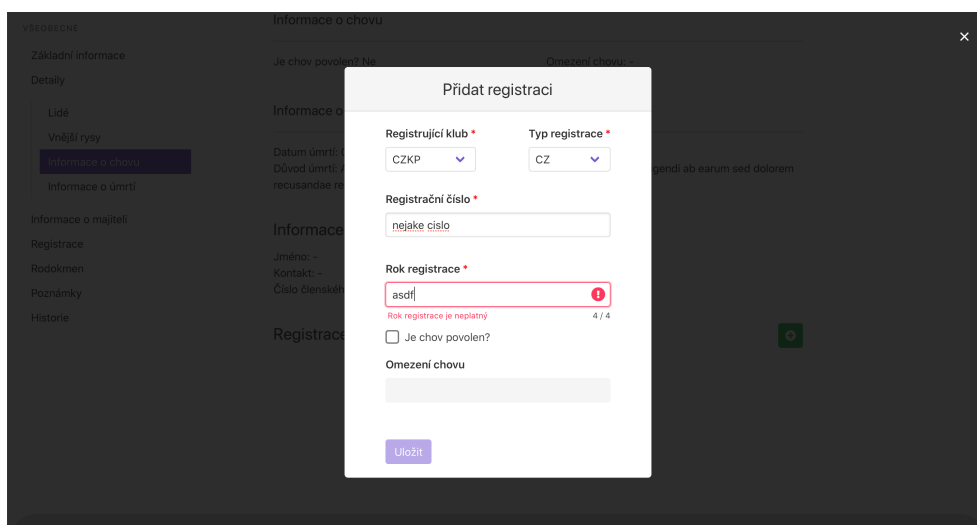


Obr. B.20: Screenshot aplikace při pokuse vymazat poznámku

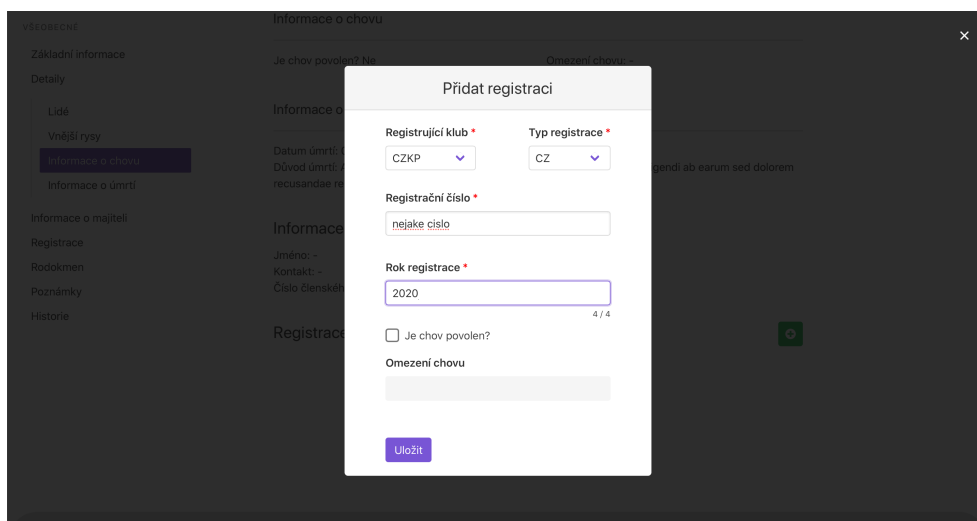


Obr. B.21: Screenshot aplikace zobrazující nasledující sekciu zvierata

## B. POUŽÍVATELSKÉ ROZHRAŇIE

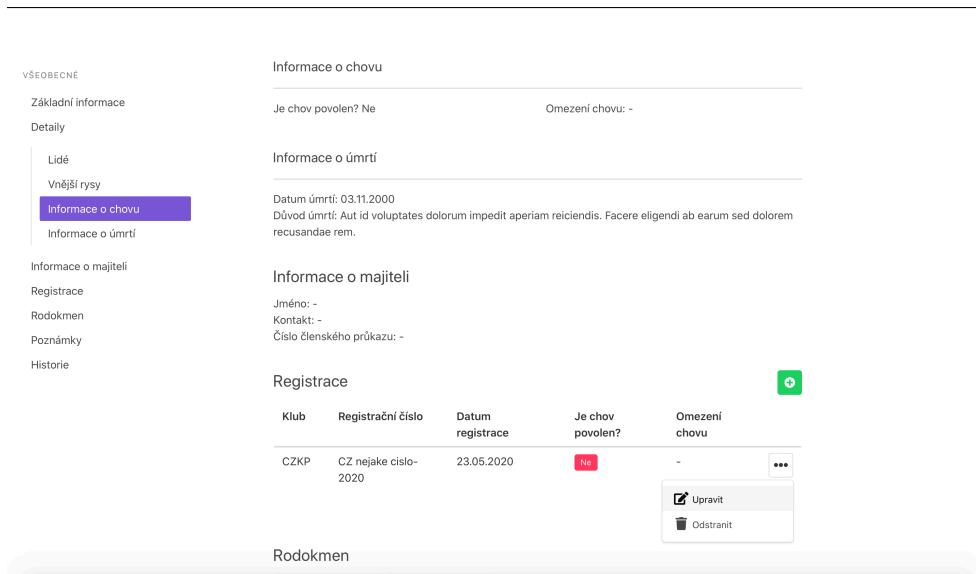


Obr. B.22: Screenshot aplikácie zobrazujúci proces pridania chybnjej registrácie zvieratá

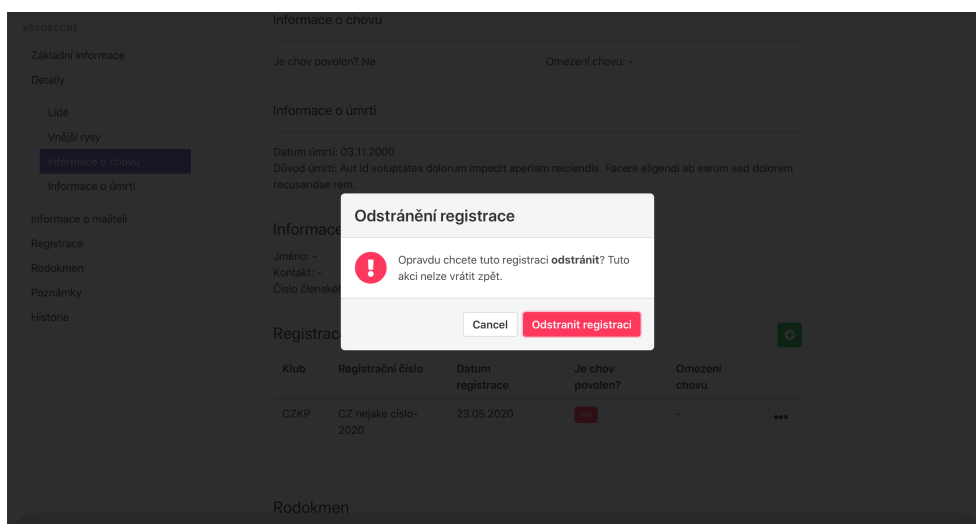


Obr. B.23: Screenshot aplikácie zobrazujúci proces pridania správnej registrácie zvieratá





Obr. B.24: Screenshot aplikácie zobrazujúci registráciu zvierata po jej pridaní




Obr. B.25: Screenshot modálneho okna aplikácie pri ostránení registrácie

## B. POUŽÍVATELSKÉ ROZHRAŇIE

Chovateľská stanice Grady-Durgan vydáva pod záštitou Českého klubu potkanů

### PRŮKAZ PŮVODU potkana laboratorního



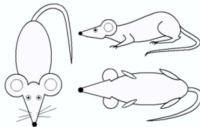
Jméno potkana **Helmer Ryan**  
 Pohlaví **Female**  
 Datum narození **25.06.1987**  
 Uší **Standart**  
 Srst **eum**  
 Barva **perferendis**  
 Znaky **laboriosam quia**  
 Barva očí **Odd eyed**  
 Odchovaná mláďata **7 / 28**  
 Poměr pohlaví **9 M / 8 F**

Chovatel **Dr. Joseph Greenfelder**  
 Kontakt **bayer.stella@gmail.com**

Majitel  
 Kontakt  
 Číslo členského průkazu

**CHOV POVOLEN**

**ZÁKRES ROZLOŽENÍ ZNAKŮ**



**REGISTRACE VRHU**

datum vyplnění  
 datum schválení  
 registrační číslo

bezpečnostní kolek

© ČKP z.s.

**REGISTRACE POTKANA**

datum registrace  
 registrační číslo  
 poznámka

registroval

Obr. B.26: Screenshot zobrazující první stranu vygenerovaného PDF souboru zvierata

???			
<p><b>Brandy Kessler II</b></p> <p>RC CZN 948406-2019 * 20.05.1971</p> <p>Dark Ruby / Standart Dg / ut / reprehenderit / quam dolor</p> <p>Majitel: Barney Sipes Jr. Chovatel: User</p>	<p><b>Anselay Conn</b></p> <p>RC ??? * 09.01.1976</p> <p>Odd eyed / Dumbo / est / voluptas / fuga sint</p> <p>Majitel: Miss Nannie Hyatt PhD</p>	<p><b>Sabrina Flatley</b></p> <p>RC ??? * 12.08.1986</p> <p>Black / Standart / et / sapiente / non nam</p> <p><b>Eveline Hermiston</b></p> <p>RC ??? * 01.09.1999</p> <p>Ruby / Standart / quisquam / culpa / beatae consequatur</p>	<p><b>Noemi Miller</b></p> <p>Black / Standart / eum / ut / voluptas qui</p> <p><b>Lilian Little</b></p> <p>Dark Ruby / Standart Dg / sit / occaecat / quae doeribus</p> <p><b>Prof. Melody Grant</b></p> <p>Ruby / Standart / odio / consequuntur / quosque dolorem</p> <p><b>Anabel Doolley</b></p> <p>Pink / Standart / qui / opta / in dolore</p>
	<p><b>Anselay Pollich</b></p> <p>RC CZ 440425166-1998 * 09.08.1987</p> <p>Odd eyed / Dumbo / quo / laborum / omnis in</p> <p>Majitel: Anibal Kutch</p>	<p><b>Skytia Abshire</b></p> <p>RC ??? * ??, ??, ????</p> <p>Ruby / Standart / repellendus / eum / iste natus</p> <p><b>Susana Rogahn</b></p> <p>RC CZN 247557036-1989 * 22.08.1980</p> <p>Pink / Dumbo / voluptatem / consequatur / enim asperiores</p>	<p><b>Alena Gustikowski</b></p> <p>Red / Standart Dg / non / dolorum / magni ex</p> <p><b>Cynus Bergstrom</b></p> <p>Red / Standart Dg / reprehenderit / possimus / ut et</p>
<p><b>Maxine Crooks</b></p> <p>RC ??? * 06.04.2003</p> <p>Pink / Dumbo / magnam / nam / sunt sed</p> <p>Majitel: Larue Fadel Chovatel: Prof. George Lemke IV</p>	<p>Předek neznámý</p>	<p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p>	<p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p> <p>Předek neznámý</p>
	<p>Předek neznámý</p>	<p>Předek neznámý</p>	<p>Předek neznámý</p>

Varietnost vrhu: Odd eyed / Standart / perferendis / eum / laboriosam quia

Obr. B.27: Screenshot zobrazující druhou stranu vygenerovaného PDF souboru zvierata

Databáze zvířat Zvířata Vrhů Uživatelé

### Vrhů

Typ: 
 Označení: 
 Majitel: 
 Stav schválení: 
[Zobraz moje vrhy](#) [Filtrovat](#) [+](#)

Typ	Registrační číslo	Označení	Majitel	Datum narození	Stav schválení	
NV	25522371	quod	Barney Sipes Jr.	10.07.1971	<a href="#">Čeká na schválení</a>	⋮
NV	-	inventore	Miss Nannie Hyatt PhD	01.02.2014	Není dostupné	⋮
NV	38720484	eaque	Dr. Mortimer Treutel	22.12.2009	<a href="#">Čeká na schválení</a>	⋮
PP	58366079	ipsa	Admin	18.04.2017	<a href="#">Čeká na schválení</a>	⋮
VP	-	aut	Betty Hoeger DVM	25.02.1972	Není dostupné	⋮
NV	-	quam	Maryse Jacobs	02.11.1986	Není dostupné	⋮
NV	45004519	autem	Joanny Bogan	03.04.1989	<a href="#">Čeká na schválení</a>	⋮
NV	42263350	beatae	Nigel Bernhard	01.12.1972	<a href="#">Zamhruzo</a>	⋮
PP	99618535	quia	Miss Nannie Hyatt PhD	03.07.1991	<a href="#">Schváleno</a>	⋮

Obr. B.28: Screenshot zobrazující tabulku všech vrhů

Databáze zvířat Zvířata Vrhů Uživatelé

VŠEOBECNĚ

Základní informace

Rodiče

Detaily

Mimina

Chovatel

Zvířata

Rodokmen

Poznámky

Historie

[↶](#) [↷](#)

### Základní informace

Rodiče

### Detaily

Mimina

Chovatel

Obr. B.29: Screenshot aplikace zobrazující načítavanie zvoleného vrhu

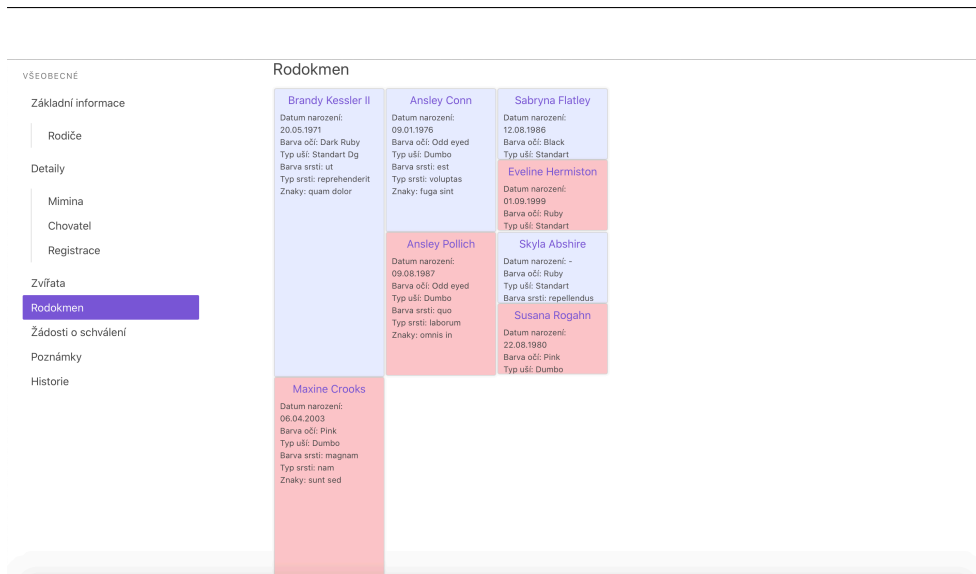
## B. POUŽÍVATELSKÉ ROZHRAŇIE

The screenshot shows a web application interface for a dog database. At the top, there is a navigation bar with 'Databáze zvířat', 'Zvířata', 'Vrhy', and 'Uživatelé'. The main content area is titled 'dolorum - Koby Medhurst' and includes 'Upravit' and 'Odstranit' buttons. A sidebar on the left lists various sections: 'VŠEOBECNÉ', 'Základní informace', 'Rodiče', 'Detaily', 'Mimina', 'Chovatel', 'Registrace', 'Zvířata', 'Rodokmen', 'Žádosti o schválení', 'Poznámky', and 'Historie'. The main content is divided into three sections: 'Základní informace', 'Rodiče', and 'Detaily'. The 'Základní informace' section displays: Typ: PP, Označení: dolorum, Datum narození: 02.10.1988, Stanice: Grady-Durgan, and Majitel: Koby Medhurst. The 'Rodiče' section shows Matka: Maxine Crooks and Otec: Brandy Kessler II. The 'Detaily' section includes Linie: cupiditate, Genetické informace: sunt distinctio impedit voluptas ut, and Varietnost: Odd eyed / Standart / perferendis / eum / laboriosam quia. At the bottom, there are statistics for Mimina: Narozených mimin: 28, Odchovaných mimin: 7, Narozených kluků: 9, Narozených holek: 8, Mimina pro chov: -, and Mimina pro mazlení: 0.

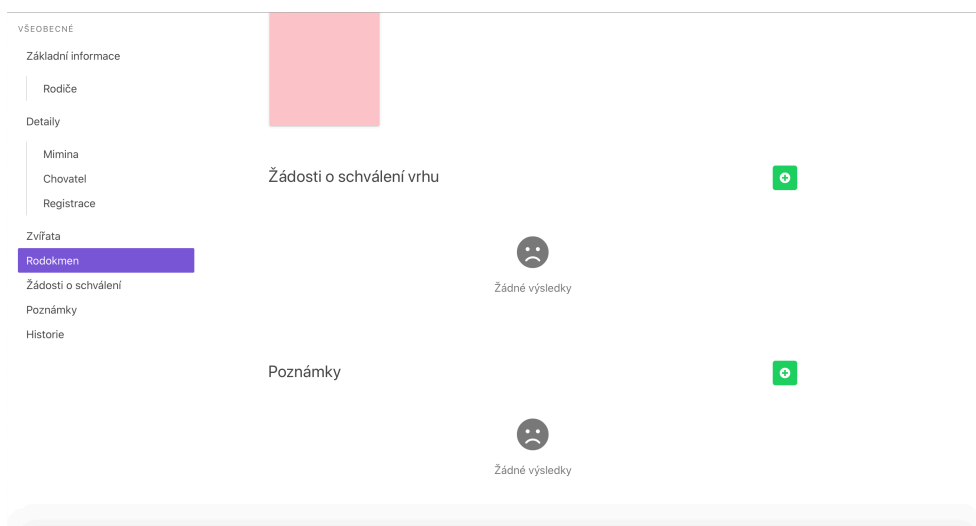
Obr. B.30: Screenshot aplikácie zobrazujúci načítaný vrh

The screenshot shows a detailed view of a dog's profile. The top navigation bar is the same as in the previous screenshot. The main content area is titled 'Odchovaných mimin: 7', 'Narozených holek: 8', and 'Mimina pro mazlení: 0'. The sidebar on the left is the same, but 'Mimina' is highlighted. The main content is divided into three sections: 'Chovatel', 'Registrace', and 'Zvířata'. The 'Chovatel' section displays: Jméno chovatele: Dr. Joseph Greenfelder, Kontakt na chovatele: bayer.stella@gmail.com. The 'Registrace' section displays: Jméno registrátora: -, Datum registrace: -, Číslo registrace: -. The 'Zvířata' section displays a table with columns: Jméno, Přezdivka, Datum narození, and Pohlaví. The table has one row: Helmer Ryan, -, 25.05.1987, ♀ Samice. Below the table, there is a 'Detaily' section with: Typ uší: Standart, Barva srsti: perferendis, Typ srsti: eum, Barva očí: Odd eyed, and Znaky: laboriosam quia. At the bottom, there is a 'Majitel' section with: Jméno: -, Kontakt: -, Číslo členského průkazu: -.

Obr. B.31: Screenshot aplikácie zobrazujúci dodatočné informácie o vrhu



Obr. B.32: Screenshot aplikácie zobrazujúci rodokmeň vrhu



Obr. B.33: Screenshot aplikácie zobrazujúci ďalšiu sekciu informácií o vrhu

## B. POUŽÍVATELSKÉ ROZHRAŇIE

Databáze zvířat Zvířata Vřhy Uživatelé

### Upravit vrh

VŠEOBECNÉ

Základní informace

Rodiče

Detaily

Mimina

Chovatel

**Základní informace**

**Typ \***  
PP

**Označení \***  
dolorum

**Majitel \***  
Koby Medhurst

**Datum narození**  
02.10.1988

**Rodiče**

**Matka**  
Maxine Crooks

**Otec**  
Frankly Keebler II

Obr. B.34: Screenshot aplikácie zobrazujúci sekciu editácie vrhu

VŠEOBECNÉ

Základní informace

Rodiče

**Detaily**

Mimina

Chovatel

**Detaily**

**Linie**  
cupiditate

**Genetické informace**  
sunt distinctio impedit voluptas ut

**Miminka**

**Narozených mimin \***  
- 28 +

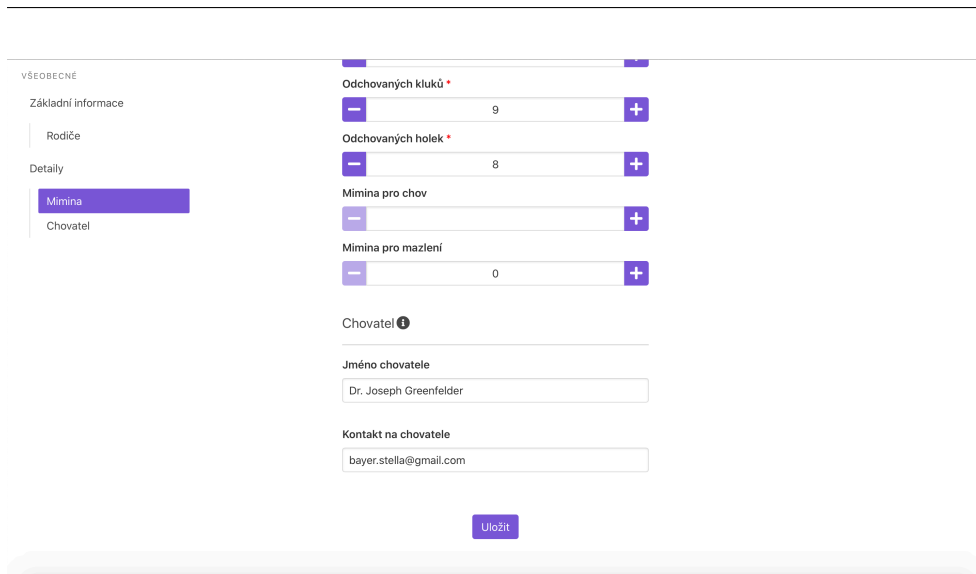
**Odchovaných mimin \***  
- 7 +

**Odchovaných kluků \***  
- 9 +

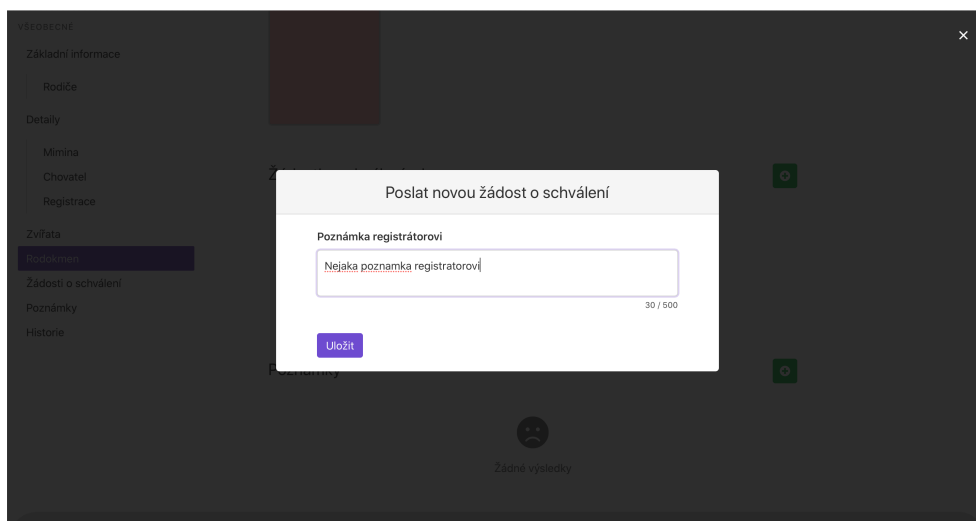
**Odchovaných holek \***  
- 8 +

**Mimina pro chov**  
- +

Obr. B.35: Screenshot aplikácie zobrazujúci ďalšiu sekciu informácií o vrhu

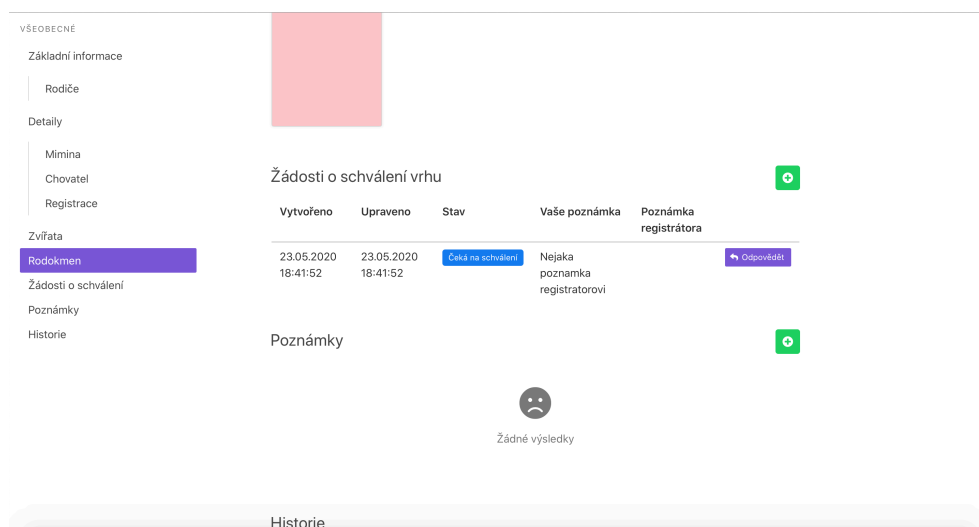


Obr. B.36: Screenshot aplikácie zobrazujúci poslednú sekciu editácie vrhu

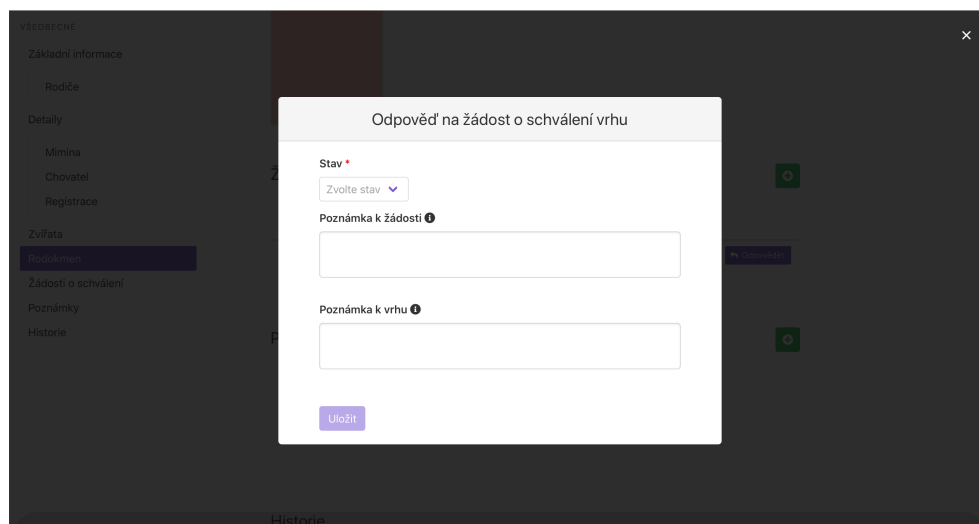


Obr. B.37: Screenshot aplikácie zobrazujúci modálne okno poslania žiadosti o schválenie vrhu

## B. POUŽÍVATELSKÉ ROZHRAVIE

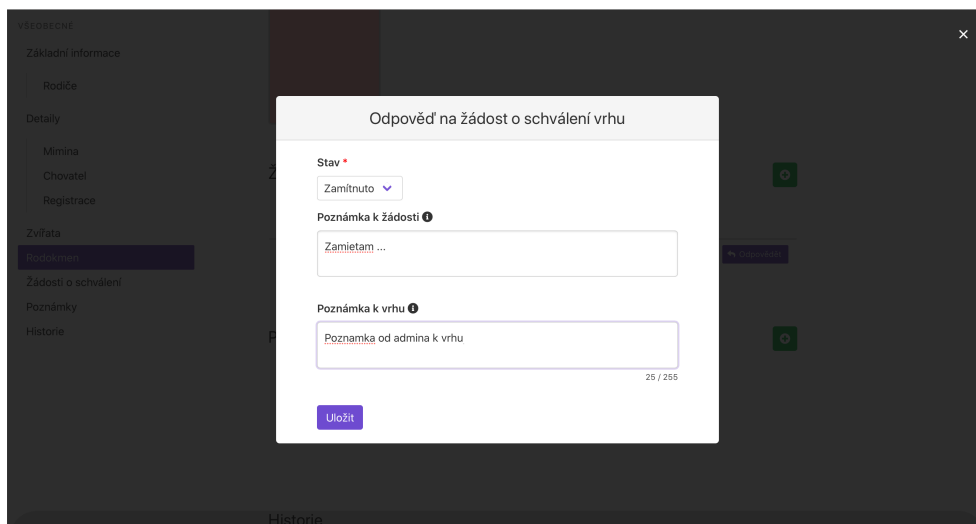


Obr. B.38: Screenshot aplikácie zobrazujúci poslanú žiadosť o schválenie vrhu

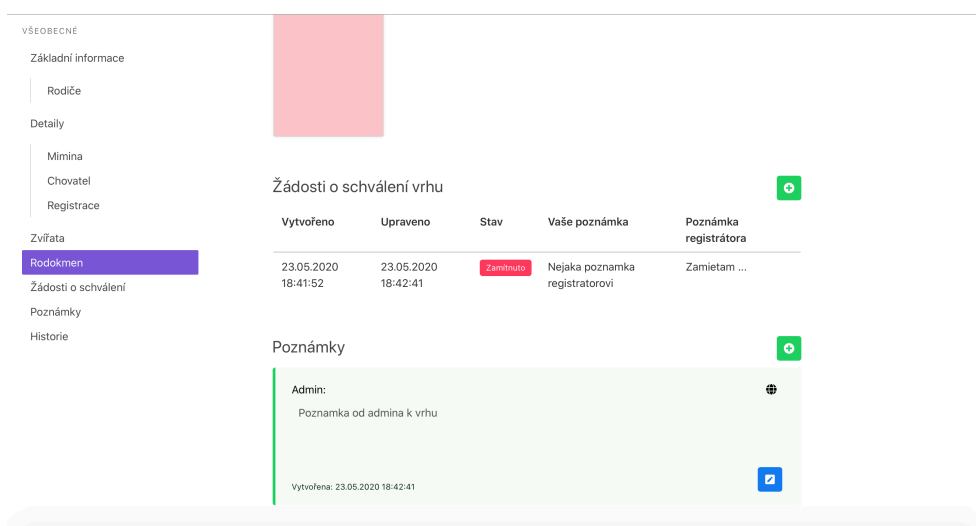


Obr. B.39: Screenshot aplikácie zobrazujúci modálne okno odpovedi na poslanú žiadosť



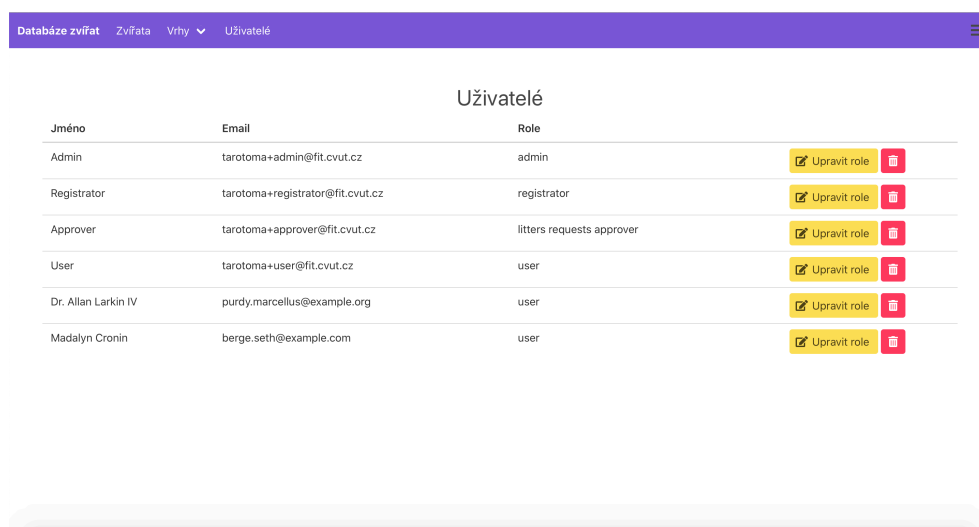


Obr. B.40: Screenshot aplikace zobrazující modálne okno vyplnenej odpovedi na poslanú žiadosť

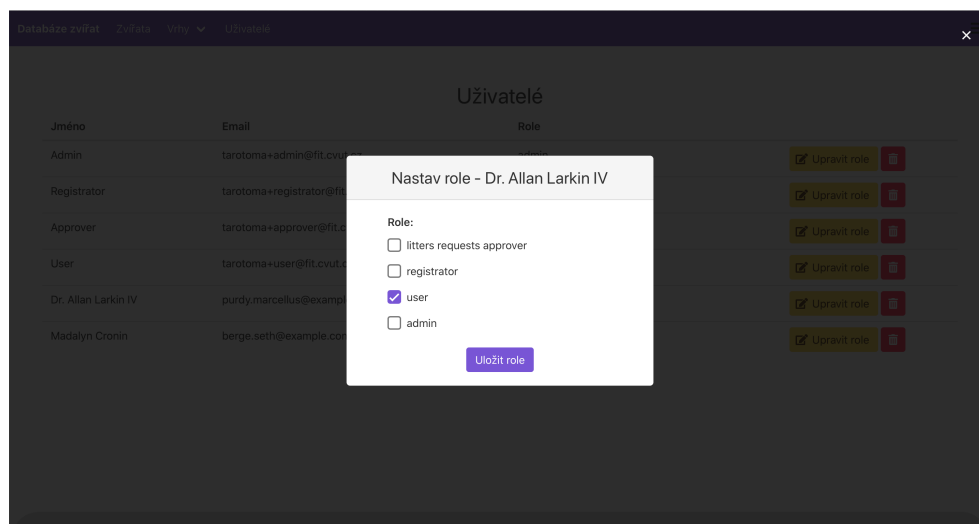


Obr. B.41: Screenshot aplikace zobrazující zamietnutú žiadosť

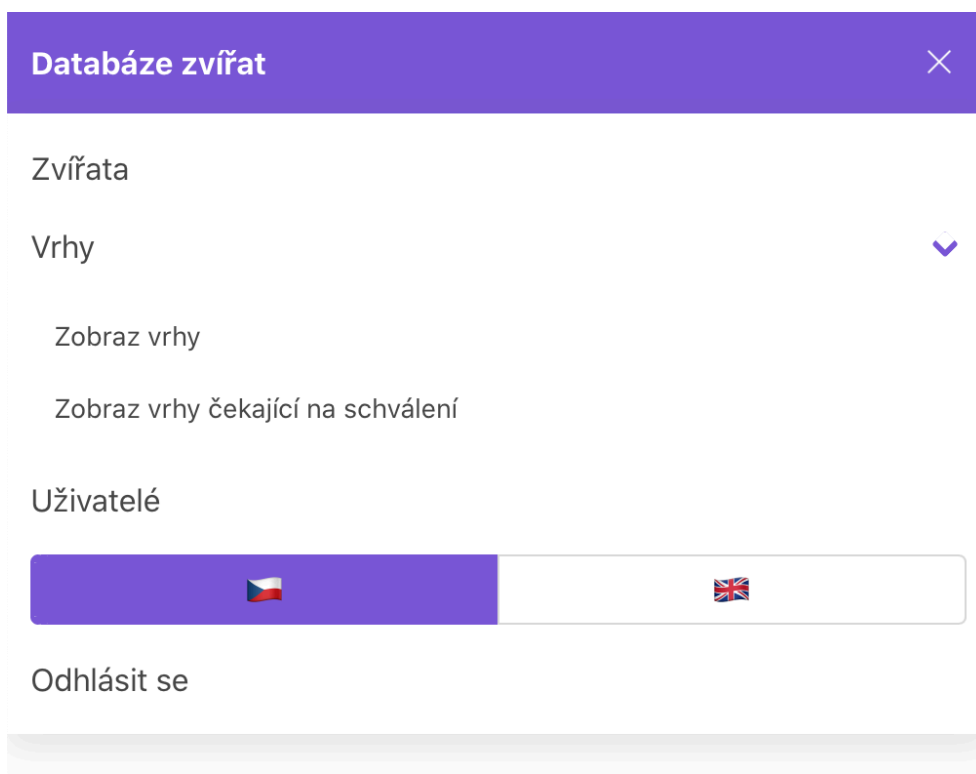
## B. POUŽÍVATELSKÉ ROZHRAŇIE



Obr. B.42: Screenshot aplikácie zobrazujúci všetkých používateľov aplikácie



Obr. B.43: Screenshot aplikácie zobrazujúci nastavenie rolí používateľovi aplikácie



## Zvířata

### Jméno

### Přezdívka

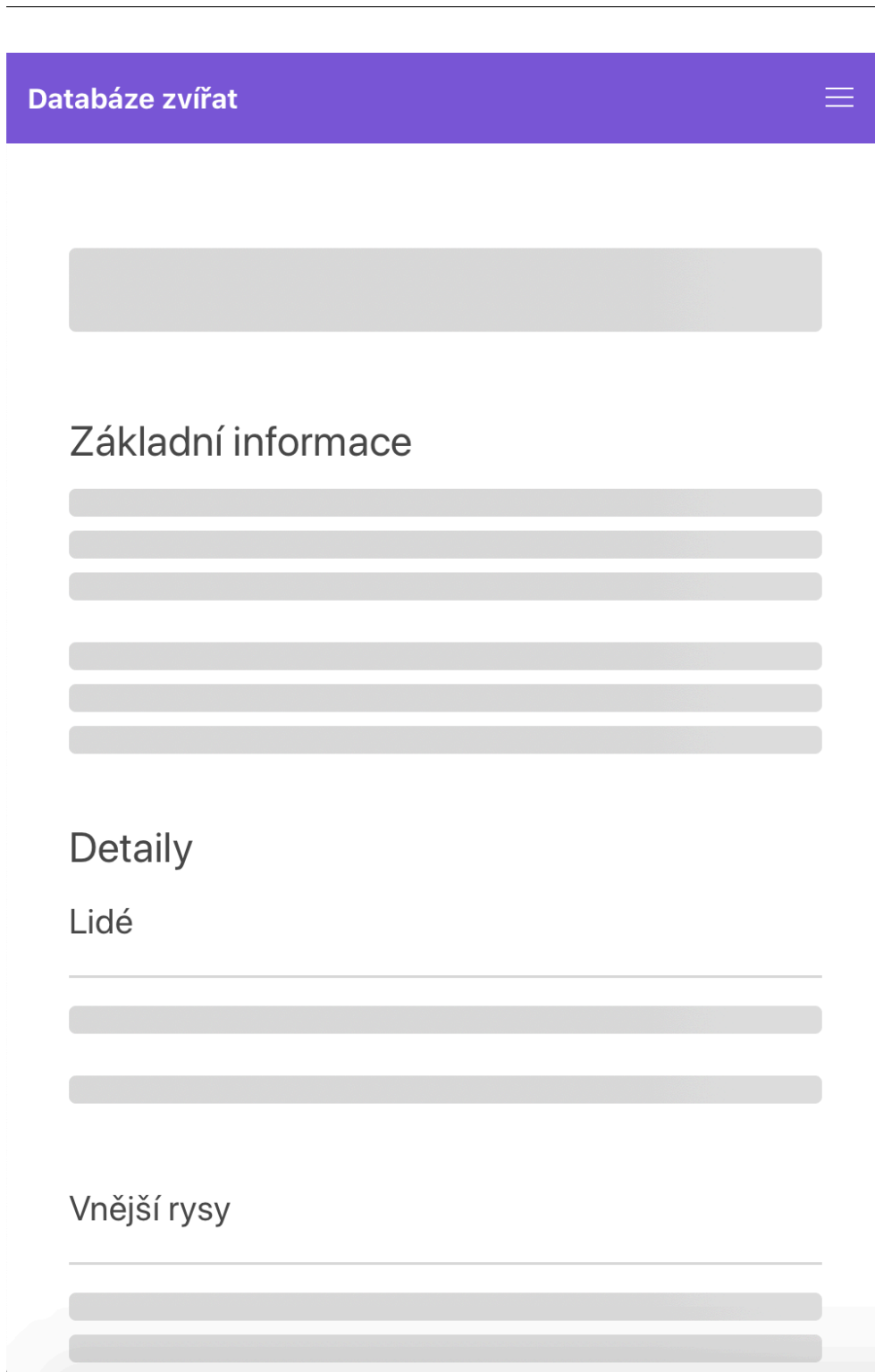
### Chovatel

### Majitel

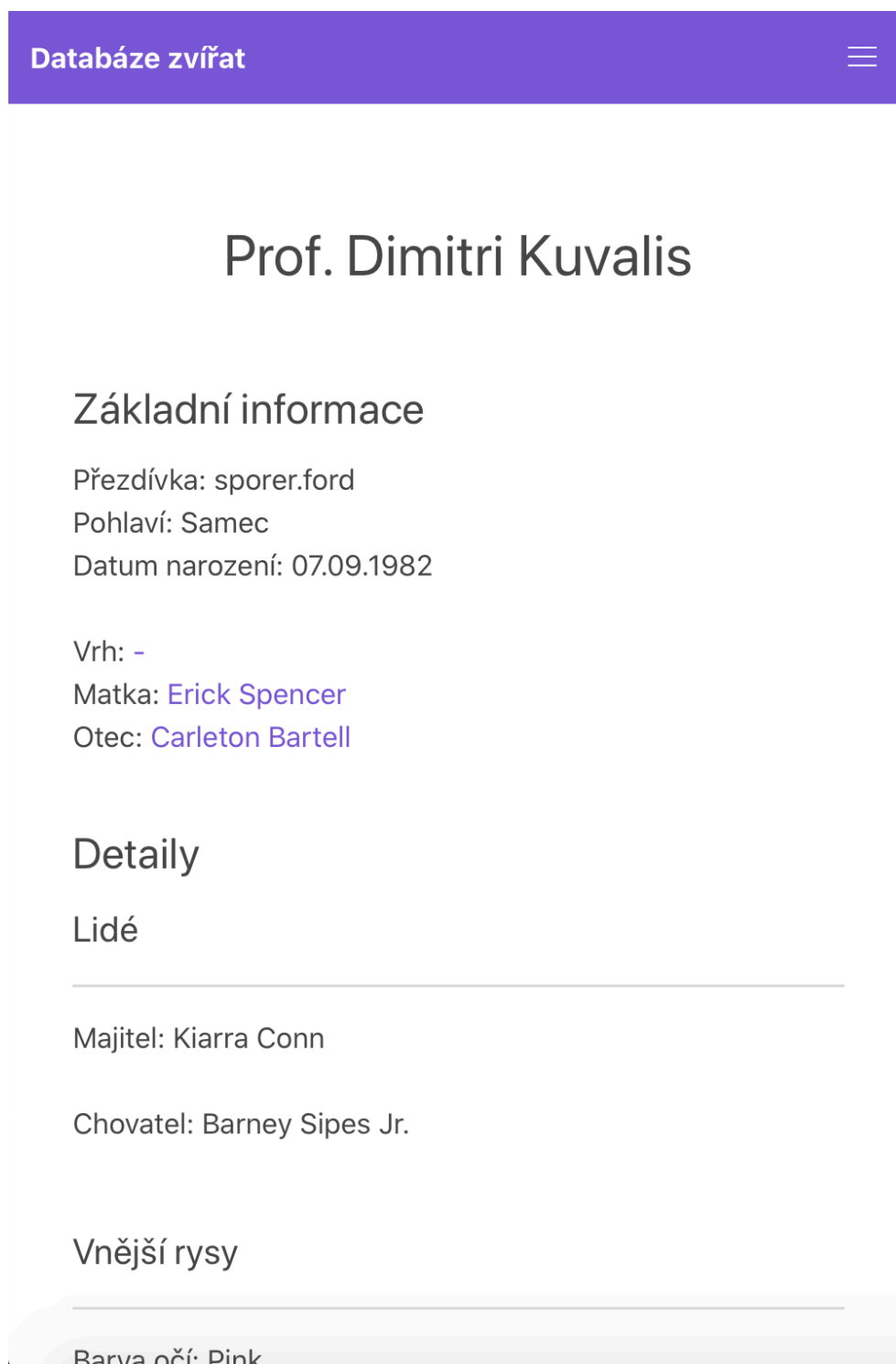
Obr. B.44: Screenshot zobrazující menu aplikace v mobilnom prehliadači

Filtrovat		+
		↓ ↑
Jméno	Prof. Dimitri Kuvalis	
Chovatel	Barney Sipes Jr.	
Majitel	Kiarra Conn	
Datum narození	07.09.1982	
Pohlaví	♂ Samec	
Registrován	Ano	
		⋮
Jméno	Noemi Miller	
Chovatel	Bruce Marvin	
Majitel	Filomena Greenholt	
Datum narození	28.03.1998	
Pohlaví	♂ Samec	
Registrován	Ano	
		⋮
Jméno	Darrell Lubowitz	

Obr. B.45: Screenshot zobrazující zoznam zvierat v mobilnom prehliadači



Obr. B.46: Screenshot zobrazující načítavanie zvierata v mobilnom prehliadači



Obr. B.47: Screenshot zobrazující načítanie zviera v mobilnom prehliadači

---

## Vnější rysy

---

Barva očí: Pink

Typ uší: Dumbo

Znaky: esse voluptatibus

Barva srsti: dolore

Typ srsti: ad

## Informace o chovu

---

Je chov povolen? Ne

Omezení chovu: -

## Informace o úmrtí

---

Datum úmrtí: 03.11.2000

Důvod úmrtí: Aut id voluptates dolorum impedit aperiam reiciendis. Facere eligendi ab earum sed dolorem recusandae rem.

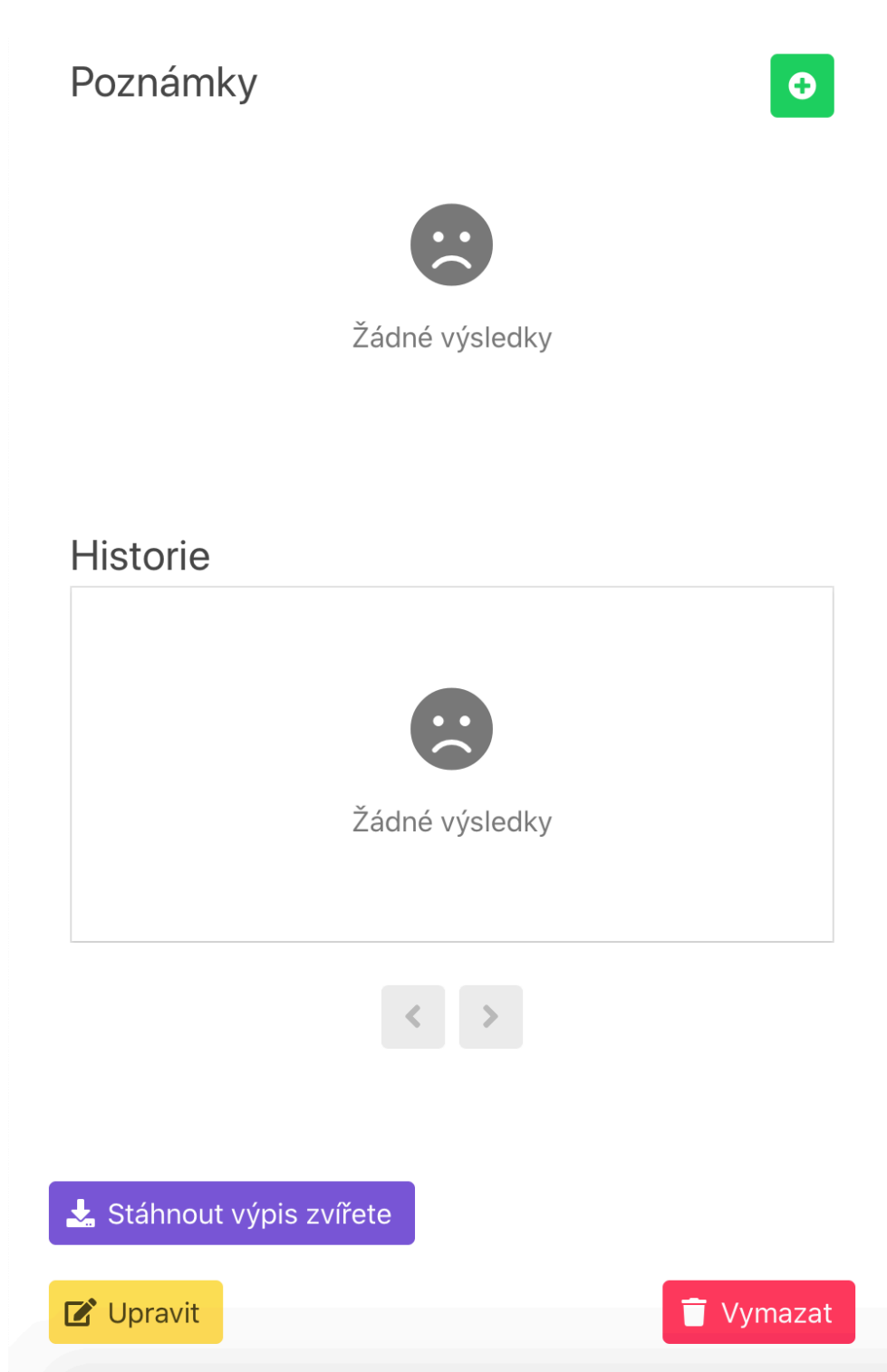
## Informace o majiteli

Jméno: -

Kontakt: -

Číslo členského průkazu: -

Obr. B.48: Screenshot zobrazující detail zvířata v mobilnom prehliadači



Obr. B.49: Screenshot poslednej sekcie zvierata v mobilnom prehliadači



## Obsah priloženého CD

src.....	adresár so zdrojovými kódmi
├ app.....	adresár so zdrojovými kódmi aplikácie
├ thesis.....	adresár so zdrojovým kódom práce vo formáte $\text{\LaTeX}$
└ text.....	text práce
├ BP_Tomáš_Taro_2020.pdf .....	text práce vo formáte PDF