FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Interoperabilní informační systém pro zvířecí útulky
Student:	Anton Ovchinnikov
Vedoucí:	Ing. Marek Suchánek
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	

Pokyny pro vypracování

Zvířecí útulky a záchranné stanice musí evidovat umístěná zvířata a další informace o nich. Současně řada těchto zařízení je modernizována, což otevírá nové možnosti automatizace a sledování pomocí různých typů senzorů. Cílem této práce je na základě analýzy a rešerše navrhnout informační systém pro zařízení, která poskytuje dočasnou péči toulavým, zraněným a opuštěným zvířatům. Tento systém musí být snadno použitelný, rozšiřitelný a především nabízet možnosti interoperability s dalšími systémy a technickými prostředky.

- Analyzujte problematiku útulků a záchranných stanic, popište klíčové procesy a požadavky na evidenci informací.
- Proveďte stručnou rešerši existujících či obdobných řešení.
- Sestavte požadavky na nový informační systém a vytvořte případy užití systému.
- Navrhněte systém splňující stanovené požadavky a zohledněte použitelnost, rozšiřitelnost a interoperabilitu. Při návrhu datových struktur se v rámci zaměření na interoperabilitu inspirujte v definicích z schema.org.
- Implementujte prototyp systému dle návrhu s využitím frameworku Spring Boot, řádně jej otestujte a zdokumentujte.
- Zhodnoťte přínosy systému a navrhněte další možný rozvoj.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Interoperabilní informační systém pro zvířecí útulky

Ovchinnikov Anton

Katedra softwarového inženýrství
Vedoucí práce: Ing. Marek Suchánek, Ph.D. et Ph.D.

14. května 2024

Poděkování

Chtěl bych nejprve poděkovat svému vedoucímu bakalářské práce Ing. Marku Suchánkovi, Ph.D. et Ph.D. za jeho pomoc a konzultace. Dále bych chtěl poděkovat své rodině a blízkým za morální podporu.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 14. května 2024

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Ovchinnikov, Anton. *Interoperabilní informační systém pro zvířecí útulky.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Abstrakt

Cílem této bakalářské práce bylo na základě provedené analýzy vytvořit prototyp informačního systému pro útulky pro zvířata. V průběhu práce jsme zvážili potřebné zákony a technologie, vytvořili plány a schémata. Výsledkem byl plně funkční prototyp vytvořený v jazyce Java s využitím frameworku Spring Boot. Kromě hotového prototypu jsme navrhli další kroky pro vývoj systému, které udávají směr další práce.

Klíčová slova Java, Spring Boot, webová aplikace, útulek pro zvířata, záchranné stanice.

Abstract

The aim of this bachelor's thesis was to create a prototype of an information system for animal shelters based on our analysis. In the course of our work we considered the necessary laws and technologies, made plans and diagrams. As a result, we got a fully working prototype created in Java language using Spring Boot framework. In addition to the finished prototype, we proposed further steps for the development of the system, which give direction for future work.

Keywords Java, Spring Boot, web application, animal shelter, animal rescue stations.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Dnes je modernizace a automatizace informačních systémů různých služeb aktuální potřebou. Těžko si lze představit organizaci, která by se obešla bez počítačů. Nicméně, i přes všeobecnou digitalizaci, ne všude jsou tyto technologie využívány dostatečně efektivně.

Moderní společnost se potýká s řadou výzev a problémů, které vyžadují pozornost a péči. Jeden důležitý aspekt péče o slabé a zranitelné členy společnosti spočívá v organizaci efektivní péče a podpory pro opuštěné zvířata. Tento problém je aktuální v mnoha zemích a vyžaduje systémový přístup k jeho řešení.

Právě za tímto účelem bylo rozhodnuto pracovat na této práci. Vzhledem k rostoucímu počtu opuštěných zvířat vzrůstá potřeba vytvoření informačních systémů, které pomohou zlepšit organizaci a řízení útulků. V teoretické části bude provedena analýza útulků a podobných institucí. V praktické části bude veškerá práce věnována vytvoření systému pro útulek. Tento systém bude přispívat k optimalizaci procesů evidence, péče a údržby zvířat, což zajistí efektivnější a pohodlnější správu útulků.

Hlavním motivem pro práci na tomto projektu byla hluboká láska k zvířatům. Tato bakalářská práce pomůže zlepšit život zvířat v útulcích.

Kapitola 1

Cíle práce

Hlavním cílem bakalářské práce je vytvořit informační systém pro útulky. Tento cíl lze rozdělit na několik menších.

Cílem teoretické části této práce je analýza problémů útulků a záchranných stanic. Je nutné najít a popsat klíčové procesy a požadavky nezbytné pro provoz útulků. Také je důležité analyzovat existující řešení nebo podobná řešení, aby se předešlo častým chybám a bylo možné integrovat systém do již existujících. Důležitou součástí bakalářské práce je stanovení požadavků systému a vytváření případů užití.

Cílem praktické části práce je návrh systému, který bude splňovat požadavky, použitelnost, rozšiřitelnost a interoperabilitu. Při práci na datových strukturách musí být použity definice ze speciálních webových stránek. Dalším krokem bude implementace, dokumentace a testování prototypu systému s využitím frameworku Spring Boot.

Cílem finální částí práce bude zhodnocení výhod systému a navrhnutí dalšího vývoje na základě získaných zkušeností.

Kapitola 2

Analýza zvířecích útulků

2.1 Zákony

Pro vytvoření informačního systému pro útulky je nutné porozumět domén, na kterém budeme pracovat. Vzhledem k tomu, že vytvoření takového systému nemá jasný regulační rámec, bude se tato kapitola týkat pouze zákonů a návodů, které se týkají tohoto odvětví. Projdeme si hlavní zákony, které regulují provoz útulků a záchranných stanic a současně mají vztah k našemu vytvářenému systému. Projednáme 3 zákony, konkrétně: zákon č. 246/1992 Sb., zákon č. 166/1999 Sb. a zákon č. 89/2012 Sb. Ale než to uděláme, definujeme útulek a záchrannou stanici.

- Podle zákona č. 114/1992 Sb., §5: Záchranná stanice – zařízení, kde se mohou zvířata léčit a pokud možno vrátit většinu z nich do přirozeného prostředí [12].
- Podle zákona č. 246/1992 Sb., §3: Útulek pro zvířata – zařízení, které poskytuje dočasnou péči toulavým a opuštěným zvířatům [1].

2.1.1 Zákon č. 246/1992 Sb., na ochranu zvířat proti týrání

První věcí, kterou zákon [1] dělá, je poskytnutí definic používaných termínů. Zde jsou hlavní tři, které budou používány po celou dobu analýzy:

- Toulavé zvíře (§ 3 písm. h) – *„Toulavým zvířetem se rozumí zvíře v lidské péči, které není pod trvalou kontrolou nebo dohledem fyzické osoby nebo chovatele a které se pohybuje volně mimo své ustájení, výběhové prostory nebo mimo domácnost svého chovatele.“* [2].
- Opuštěné zvíře (§3 písm. i) – *„Opuštěným zvířetem se rozumí zvíře původně v lidské péči, které není pod přímou kontrolou nebo dohledem fyzické osoby nebo chovatele a ze zjištěných skutečností vyplývá, že ho jeho chovatel opustil s úmyslem se jej zbavit nebo ho vyhnal.“* [3].
- Zařízení (§3 písm. v) – *„Zařízením se rozumí stavba, budova, komplexy budov nebo jiné prostory, v nichž je provozována činnost se zvířaty; může se jednat o zařízení, která nejsou úplně oplocena nebo zastřešena, jakož i o pohyblivá zařízení.“* [4].

Dalším krokem bude zvážení účelu zákona. Podle zákona č. 89/2012 Sb. (viz 2.1.2) je zvíře živým tvorem, a proto s ním nelze nakládat jako s obyčejnou věcí, což také reguluje zákon č. 246/1992 Sb., jehož účel je popsán v § 1, bodě 1:

„Účel zákona – chránit zvířata, která jsou živými tvory schopnými cítit bolest a utrpení, před krutým zacházením, způsobováním újmě na jejich zdraví a jejich zabitím bez příčiny, jestliže je způsobeno i nedbalostí člověka.“[5].

Z definice můžeme zjistit, na co bychom měli dbát při návrhu našeho informačního systému, a to konkrétně na uspání zvířete a povinnosti člověka vůči nim. Podívejme se na § 5, kde je upraveno právo útulku uspat zvíře. Důvody, pro které může útulek nebo záchranná stanice zvíře uspat:

- Slabost, vážné zranění, nemoci, stáří a další důvody, které způsobují zvířeti utrpení.
- Nebezpečí, které dané zvíře představuje pro člověka.
- Regulace populace.
- Ochrana před epidemiemi.

Nyní se podívejme na povinnosti provozovatelů (majitelů) útulků a záchranných stanic. Odpověď najdeme v § 25, bodech 3 a 4:

- zajistění ochrany a péče o zvíře,
- seznam přijatých zvířat a jejich údaje,
- seznam zvířat, propuštěných na svobodu nebo předaných novým majitelům, včetně veškerých údajů o nich,
- seznam uprchlých zvířat,
- prokázání odborné způsobilosti,
- zveřejnit seznam nalezených zvířat,
- zdarma předat novým majitelům informace o péči o zvíře.

2.1.2 Zákon č. 89/2012 Sb., občanský zákoník

S odvoláním na § 494 zákona [6], podle kterého je zvíře živým tvorem:

„Živé zvíře má zvláštní význam a hodnotu již jako smysly nadaný živý tvor. Živé zvíře není věcí a ustanovení o věcech se na živé zvíře použijí obdobně jen v rozsahu, ve kterém to neodporuje jeho povaze.“[7].

To vše je zapotřebí, abychom si uvědomili, že i když je vlastnictví zvířat obecně regulováno stejnými pravidly jako vlastnictví věcí, není to totéž, a proto existují samostatná regulující pravidla. Tři z nich nyní projednáme.

2.1.2.1 Pravidla přechodu vlastnického práva k domestikovaným zvířatům, uvedené v §1047 [8]

1. Domestikované zvíře, které vlastník nechytil a dokonce ho nevrátil vlastníkovi v rozumné lhůtě, i když v tom není nikým překáženo, se stává zvířetem bez majitele a může být vlastníkem přisvojeno na základě soukromého vlastnictví nebo někým vlastníkem státního majetku. Skutečně rozumnou lhůtou pro vrácení zvířete vlastníkovi je období šesti týdnů.

2. Bod 1 se nevztahuje, jestliže je zvíře označeno takovým způsobem, že lze identifikovat jeho vlastníka.

2.1.2.2 Odpovědnost nálezce, uvedená v §1058 [9]

1. Jestliže je zvíře nalezeno a je zřejmé, že mělo vlastníka, musí nálezce okamžitě informovat městský úřad o nálezu, jestliže je kvůli okolnostem nemožné určit, komu má být vráceno.

2. Osoba, která se stará o nalezené zvíře, o něj pečuje jako o řádný majitel, dokud se o něj nepostará vlastník.

2.1.2.3 Podmínky nabývání a převodu, uvedené v §1059 [10]

1. Jestliže je do dvou měsíců od oznámení o nálezu nalezeno zvíře zjevně určené pro amatérské chování a nikdo na něj nepodal žádost, nálezce nabývá právo vlastnictví na něj.

2. Jestliže nálezce zvíře oznámí městskému úřadu, že si nepřeje získat zvíře, a jestliže městský úřad zvíře trvale předá osobě, která provozuje útulek pro zvířata, tato osoba může s tímto zvířetem volně nakládat, jestliže se do čtyř měsíců od dne, kdy jí bylo zvíře svěřeno, nikdo nehlásí. Pokud bylo o nálezu oznámeno až po předání zvířete, lhůta začíná běžet od okamžiku oznámení o nálezu.

Shrnující výše uvedené, lze říci, že podle tohoto zákona je nálezce povinen informovat městský úřad o nálezu. V případě, že nálezce nechce zvíře u sebe držet, je povinen o něj pečovat městský úřad, kde bylo zvíře nalezeno. Útulek, kterému bylo zvíře svěřeno, je také povinen se o něj starat, ale nemá právo s ním nakládat po dobu čtyř měsíců od oznámení o nálezu. Nálezce získává právo vlastnictví na zvíře po dvou měsících od podání žádosti o nálezu, pokud není možné identifikovat vlastníka zvířete; v opačném případě je zvíře povinně vráceno vlastníkovi.

2.1.3 Zákon č.166/1999 Sb., veterinární zákon

Podle § 61 zákonu [11] veterináři musí informovat příslušné orgány o zahájení a ukončení své činnosti, o podezření na nebezpečné nemoci a kruté zacházení se zvířaty, a vést záznam o přijatých preventivních opatřeních, použitých léčivých přípravcích a utracených zvířatech. Veterináři jsou také povinni zajistit dodržování požadavků na používání léčivých přípravků a prevenci jejich negativního vlivu na produkty živočišného původu. Kromě toho musí uchovávat příslušnou dokumentaci po dobu tří let a na žádost ji poskytovat oficiálním zástupcům.

2.2 Metodické návody

Péče o zvířata v útulku nebo podobném zařízení vyžaduje individuální přístup ke každému druhu zvířat. Nemůžeme v našem zařízení vytvořit místnosti stejné velikosti a připravit stejnou stravu pro všechna zvířata. Je zřejmé, že různá zvířata mají různou velikost, různou potravní základnu a jsou přizpůsobena k životu v různých podmínkách. Bylo by nerozumné a dokonce nezodpovědné umístit mořskou želvu do chladného výběhu určeného pro psa. Proto je nezbytné alespoň minimálně zjistit požadavky na péči o každé zvíře, abychom zajistili pohodlné podmínky pro umístěná zvířata.

Dalším krokem bude prozkoumání metodických pokynů státní veterinární správy [18].

2.2.1 Inventář a prostory

Zde budeme zvažovat pokyny uvedené v bodech 6 a 7 týkající se skladování a přípravy krmiva pro zvířata, stejně jako nezbytného vybavení a prostor zařízení.

1. Pro suché krmivo používat uzavřené, suché a chladné prostory bez přístupu slunečního světla.
2. Pro přírodní krmivo se používají samostatné místnosti s chladící nebo mrazicí technikou, stejně jako povinný technický nábytek (stoly, nádobí atd.).
3. Tyto místnosti musí být vybaveny ochranou proti hmyzu (sítě na oknech atd.).
4. Místa pro skladování životních potřeb zvířat (boxy pro skladování).
5. Místa pro karanténní zónu, jejíž kapacita činí nejméně 10% celkové kapacity.
6. Místa pro volný pohyb nebo pro agresivní zvířata (2 dveře pro zabránění útěku při otevření jedné z dveří).
7. Prostory pro kočky musí být vybaveny vybavením pro lezení a obrušování drápků. Také se doporučuje umístit tam hračky.
8. Vybavení pro bezpečný odchyt.
9. Vedlejší prostory pro zvířata (mytí, veterinární prohlídky atd.).

2.2.2 Hygienická péče

Probereme základní aspekty péče o zvířata a také pravidla pro dodržování hygieny v zařízení, které jsou popsány v bodě 8.

1. Denní prohlížení stavu zvířat a jejich zabezpečení.
2. Zajištění denního krmení a dlouhodobého přístupu k vodě.
3. Denní čištění výběhů.

4. Dezinfekce nově příchozích zvířat a míst, kde bylo zjištěno nemocné zvíře.

5. Nemocné zvíře umístit do karantény.

6. Jednou za 3 měsíce celkové úklid celého zařízení.

2.2.3 Principy chovu

Jelikož zvíře může přijít do útulku těhotné nebo je stanovena cílová populace zvýšení, jsme povinni projednat pokyny pro správné chování, které jsou popsány v bodě 9.

1. V případě kojení psů, koček a samiček jiných druhů domácích zvířat, která jsou v útulku, mohou být kojená mláďata podle posouzení veterinárního lékaře [13] kojena pouze takovým množstvím, které odpovídá stavu zdraví kojící samice.

2. Minimální věk pro odstavení štěňat od feny je 50 dnů, minimální věk pro odstavení koťat od kočky je 84 dny.

2.2.4 Velikost prostorů

Jak bylo zmíněno výše, musíme zajišťovat zvířatům pohodlné podmínky v zařízení. Projednáme minimální požadavky na velikost klecí pro kočky a psů, protože jsou to nejběžnější domácí zvířata. Tato požadavky jsou uvedeny v zákoně č. 419/2012 Sb. [14]

Minimální prostor pro držení chovné samice a vrhu je prostor určený pro jednu kočku, který je postupně zvětšován tak, že jsou mláďata do čtyř měsíců po narození přemístěna do prostor odpovídajících požadavkům na prostor dospělých zvířat.

Plocha pro krmení a plocha pro nádobu na výkaly musí být vzdáleny nejméně 0,5 m a tyto plochy se nesmějí zaměňovat.

	Podlaha (m^2)	Římsy (m^2)	Výška (m)
Minimum pro jedno dospělé zvíře	1,5	0,5	2
Dodatečný prostor pro každé další zvíře	0,75	0,25	–

Tabulka 2.1: Kočky [15].

Prostor pro každého ze psů umístěných v páru nebo ve skupině může být v průběhu uplatňovaných pokusů omezen na polovinu celkového vyhrazeného prostoru (2 m^2 u psa do 20 kg, 4 m^2 u psa nad 20 kg), jak stanoví zákon nebo tato vyhláška, je-li toto oddělení zvířete důležité pro vědecké účely. Doba, během níž je prostor psa takto omezen, nesmí přesahovat 4 hodiny bez přestávky.

Kojící feně a mláďatům musí být přidělen stejný prostor jako samotné feně stejné hmotnosti. Porodní kotec musí být řešen tak, aby se mohla fena od štěňat vzdálit do jiné části prostoru nebo na vyvýšenou plošinu.

Váha (kg)	Minimální plocha uzavřeného prostoru (m^2)	Minimální podlahová plocha pro jedno nebo dvě zvířata (m^2)	Pro každé další zvíře rozšíření nejméně o (m^2)	Minimální výška (m)
až do 20	4	4	2	2
nad 20	8	8	4	2

Tabulka 2.2: Psi [16].

Hmotnost psa (kg)	Minimální plocha uzavřeného prostoru (m^2)	Minimální podlahová plocha na jedno zvíře (m^2)	Minimální výška (m)
až do 5	4	0,5	2
od 5 do 10	4	1,0	2
od 10 do 15	4	1,5	2
od 15 do 20	4	2	2
nad 20	8	4	2

Tabulka 2.3: Psi – odstavená štěňata [17].

Podle údajů v tabulce vidíme, že rozměry jsou pevně regulovány a nemohou být menší než uvedené parametry. Z tohoto důvodu je nutné omezit přiřazování zvířatům nevhodných klecí.

Kapitola 3

Rešerše existujících řešení

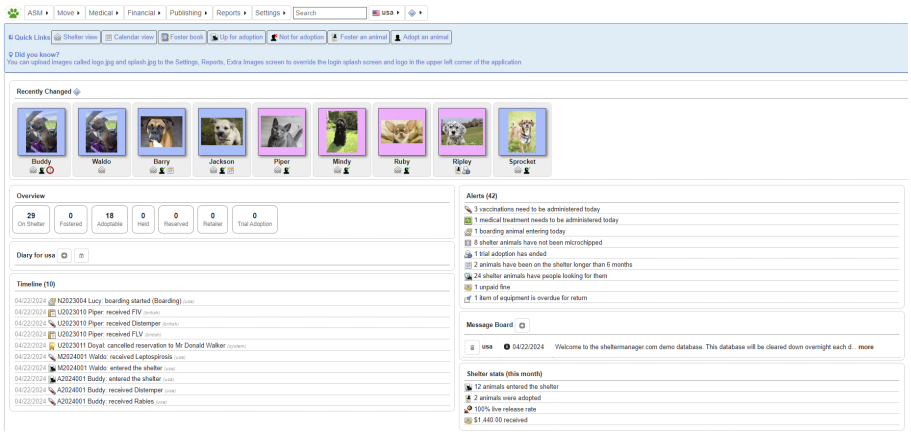
V této části našeho výzkumu představíme analýzu existujících informačních systémů, které byly vyvinuty pro použití v útulcích pro zvířata a dalších podobných zařízeních. Tato analýza je nezbytná k tomu, abychom identifikovali již existující řešení a na jejich základě vylepšili naši vlastní vyvíjenou informační systém.

V současné době existuje na trhu značné množství informačních systémů určených pro útulky. Ve svém výzkumu se omezíme na zkoumání nejrozšířenějších (podle vyhledávaní) a bezplatných řešení. Tento přístup bude více vhodný, s ohledem na skutečnost, že náš vlastní systém bude také k dispozici k použití zdarma. Poté provedeme stručný přehled každého z vybraných informačních systémů a představíme obecné výsledky jejich analýzy. Budeme zkoumat takové parametry jako: dostupnost, informativnost, zajímavá řešení, snadnost použití a dostupnost dokumentace. Některé parametry budou spíše subjektivně hodnoceny, ale jsou dostatečně důležité, abychom je nezmínili. Tyto parametry budou označeny speciálním symbolem *.

3.1 Animal Shelter Manager

Animal Shelter Manager(ASM) [20] je bezplatná webová aplikace, která se objevila na začátku roku 2000 pro správu zvířat v útulcích a záchranných stanicích. ASM je stále udržován a přístup k němu je zajištěn prostřednictvím webového rozhraní, které umožňuje přístup z libovolného zařízení s připojením k internetu. Data lze uchovávat lokálně, na vlastním SQL [19] serveru nebo za 305 eur ročně získat přístup na jejich webu.

V budoucnu se budeme opírat o ASM, protože se jedná o plně funkční webovou stránku s možností prohlížení demo verze (obrázek 3.1).



Obrázek 3.1: Demo verze ASM (Snímek obrazovky z [20]).

3.1.1 Výhody

- Dostupnost z prohlížeče.
- Bohaté množství informací o zvířatech.
- Lékařské a finanční záznamy.
- Možnost exportu a importu souborů.
- Hotové šablony pro vyplňování.
- 15 dostupných jazyků.
- Import dat z jiných řešení.
- Interakce s Google Maps.

3.1.2 Nevýhody

- Zastaralé uživatelské rozhraní*.
- Chybí funkcionalita pro správu útulku.
- Vlastní placená databáze.
- Platba za aplikaci a podporu.

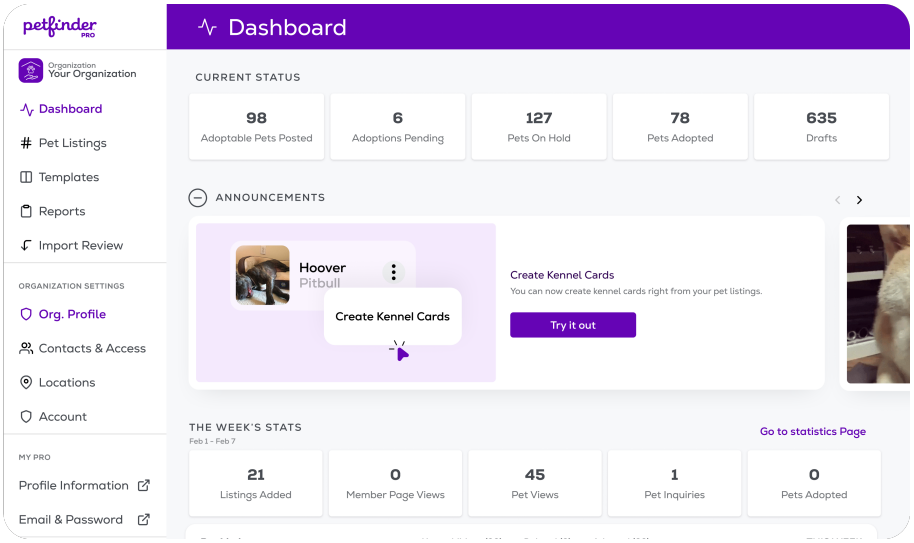
3.2 Petstablished

Petstablished [21] je bezplatný webový software, který se objevil na trhu v roce 2011 a je registrován v USA. Díky přítomnosti webové verze je také k dispozici na libovolném zařízení. Bezplatný přístup je zajištěn podmíněným každoročním

registrováním jejich čipů. Pro malé útulky je to registrace 5 čipů a pro velké (s více než 100 zvířaty) 10 čipů. V opačném případě se účtuje měsíční poplatek ve výši 24 a 49 USD pro malé a velké útulky, respektive.

3.2.1 Výhody

- Dostupný z prohlížeče.
- Bohaté množství informací o zvířatech.
- Medikální a finanční záznamy.
- Hotové šablony pro vyplňování.
- Přítomnost tutoriálů.
- Moderní uživatelské rozhraní*.



Obrázek 3.2: Petstablished. Dashboard (Snímek obrazovky z [21]).

Obrázek 3.2 slouží k ukázání moderního uživatelského rozhraní.

3.2.2 Nevýhody

- Poplatek při nízké aktivitě zařízení.
- Přístup k demo na vyžádání.
- Chybí správa případů.
- Chybí kontrola zvířat.
- Chybí funkcionalita pro správu útulku.

3.3 PetPoint

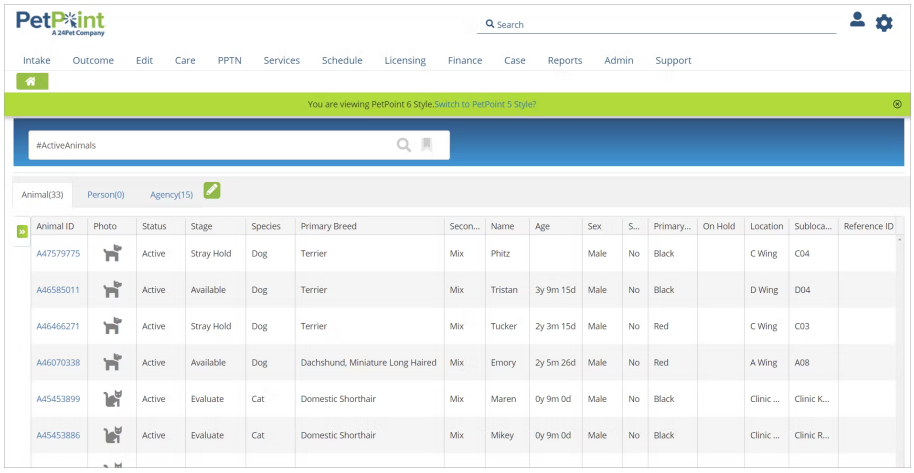
PetPoint [22] je webová aplikace od společnosti 24PetCare, mezinárodní společnosti specializující se na poskytování produktů a služeb pro domácí zvířata. PetPoint je cloudový systém pro správu dat, určený k pomoci organizacím pro ochranu zvířat při efektivním řízení útulků, včetně efektivního sledování, jak zvíře přichází do útulku, všechny podrobnosti o něm, péče o něj během pobytu v útulku, jeho konečný výsledek a vše, co se děje mezi nimi.

3.3.1 Výhody

- Dostupný z prohlížeče.
- Bohaté množství informací o zvířatech.
- Medikální a finanční záznamy.
- Hotové šablony pro vyplňování.

3.3.2 Nevýhody

- Neintuitivní uživatelské rozhraní*.
- Přístup k demo na vyžádání.



Obrázek 3.3: PetPoint. Seznam zvířat (Snímek obrazovky z [22]).

3.4 Závěr

V této kapitole jsme zkoumali několik existujících řešení v oblasti správy útulků pro zvířata s cílem využít jejich zkušeností a funkcí pro vývoj našeho vlastního systému. Zaměřili jsme se na nejoblíbenější a bezplatná řešení, což odpovídá našemu plánu poskytnout bezplatný systém. Celkově lze říci, že přezkoumávaná řešení nabízejí cenné nástroje pro řízení útulků pro zvířata, ale každé z nich má své silné a slabé stránky.

Kapitola 4

Katalog požadavků a případy užití

4.1 Funkční a nefunkční požadavky

Nedílnou součástí při vytváření našeho systému bude sestavení požadavků. Tyto požadavky rozdělíme do dvou skupin: funkční (functional requirements) a nefunkční požadavky (non-functional requirements) [23]. Před tím, než začneme s tvorbou a rozdělením požadavků do skupin, poskytneme jejich stručné definice a charakteristiku.

4.1.1 Funkční požadavky

Funkční požadavky jsou hlavní funkcionální schopnosti nebo vlastnosti, které by měl softwarový systém mít k dosažení svého zamýšleného cíle. Jednoduše řečeno, odpovídají na otázku: „Co by měl program dělat?“. Musí být konkrétní a přesně popisovat funkčnost systému.

Funkčním požadavkům přidělíme také prioritu. K tomu použijeme metodu MoSCoW [24]. Podle této metody rozdělíme funkční požadavky do 4 skupin: Must, Should, Could a Would.

4.1.1.1 Must

Must jsou povinné a musíme je implementovat.

FR01: Systém bude mít účet pro každého zaměstnance.

FR02: Systém umožňuje registrovat nová zvířata a sleduje jejich stav.

FR03: Systém umožní nastavit stav zvířete (adoptováno, podléhající předání atd.).

FR04: Systém bude udržovat historii zvířat.

FR05: Systém umožní hledání zvířat v útulku.

FR06: Systém umožní zaměstnancům přihlašovat do vlastního účtu.

4.1.1.2 Should

Should se vytvářejí po povinných a v případě problémů můžeme odložit.

FR07: Systém umožní nastavit charakteristiku prostor.

FR08: Systém zaznamená majetek útulku.

FR09: Systém bude sledovat umístění zvířat.

FR10: Systém poskytne zaměstnancům možnost vést dokumentaci (Kdo, koho, jak, kdy, ...).

FR11: Systém obsahuje údaje o zaměstnancích a jejich kvalifikaci.

4.1.1.3 Could

K realizaci Could požadavků přistoupíme, pokud budou k dispozici zdroje, včetně času, na jejich vytvoření.

FR12: Systém poskytuje údaje o provozu útulku.

FR13: Systém vytváří zprávy.

FR14: Systém bude sledovat stravu zvířat.

FR15: Systém umožní zaměstnancům vést záznamy o výdajích.

FR16: Systém bude poskytovat doporučení (například při převozu nových zvířat, doporučí vytvoření dalších karanténních míst).

FR17: Systém bude obsahovat materiály o péči o zvířata, určené těm, kteří si z útulku vezmou zvíře.

FR18: Systém bude uchovávat fotografie zvířat.

4.1.1.4 Would

Do Would požadavků zahrneme požadavky, které můžeme odložit a zrušit bez poškození našeho systému.

FR19: Systém umožní export dokumentů do různých formátů.

FR20: Systém bude obsahovat různé formuláře k vyplnění.

FR21: Systém bude shromažďovat statistiky.

FR22: Systém umožní nastavení jednorázových úkol.

FR23: Systém umožní přepínání mezi zařízeními.

FR24: Systém bude upozorňovat na povinné akce (krmení, úklid, ...).

FR25: Systém bude sledovat možnost splnění úkolu.

4.1.2 Nefunkční požadavky

Nefunkční požadavky doplňují funkční požadavky a určují, jak by měl softwarový systém provádět určité funkce. Definují vlastnosti, charakteristiky a omezení systému, nikoli jeho konkrétní vlastnosti.

NFR01: Systém bude dostupný z webového prohlížeče. Tento požadavek byl vytvořen za účelem zajištění přístupu k systému z co nejvíce zařízení.

NFR02: Systém bude uchovávat hesla uživatelů v zašifrované podobě. Tento požadavek zajistí bezpečnost dat, protože i zaměstnanci s přístupem k databázi nebudou moci ukrást uživatelské heslo.

NFR03: Systém bude napsán v jazyce Java [25] s využitím frameworku SpringBoot [26]. Podrobnosti o tomto požadavku jsou uvedeny v bodech 5.1 a 5.2.

NFR04: Systém nevyžaduje instalaci dodatečného softwaru kromě prohlížeče. Tento požadavek byl vytvořen, protože všechny akce by měly probíhat na serveru. Server potřebuje od uživatele pouze vhodný požadavek.

NFR05: Autentizace bude provedena pomocí JSON Web Token (JWT) [27]. Tento požadavek umožní umístit aplikaci na několik serverů. Také urychluje práci serveru tím, že mu odpadá nutnost nadměrného dotazování se databáze k ověření uživatele.

4.2 Případy užití

Dalším krokem bude psaní případů užití (use cases). Tento krok je velmi důležitý, protože v případě práce na našem systému více vývojářům je třeba, aby věděli, jak program funguje a co má vykonávat. Tyto případy užití jim pomohou pochopit, jak systém funguje. Případy užití také názorně ukážou očekávaný výsledek použití našeho systému. Posledním důvodem je to, že UC usnadní testování našeho systému, protože nám bude stačit projít jimi a porovnat očekávaný výsledek s reálným.

UC01: Registrace zaměstnanců (FR01). Tento případ užití umožní administrátorovi registrovat nové zaměstnance. Administrátor bude muset vyplnit všechny potřebné údaje o zaměstnanci, které bude později moci změnit.

UC02: Autorizace a odhlášení (FR01, FR06). Tento případ užití umožní zaměstnancům přihlašovat se a odhlašovat se ze systému.

UC03: Zamítnout přiřazení (FR25). Tento případ užití umožní systému odmítnout přiřazení zaměstnance k plnění úkolu.

UC04: Registrace zvířete (FR02). Tento případ užití umožní zaměstnancům registrovat nová zvířata v útulku. Zaměstnanec bude povinen vyplnit všechna povinná pole a může také vyplnit další informace.

UC05: Vyhledávání zvířat v systému (FR05, FR09). Tento případ užití umožní zaměstnancům najít zvíře v útulku. Zaměstnanec obdrží seznam zvířat s jejich údaji. Zaměstnanec si ze seznamu vybere požadované zvíře.

UC06: Přiřadit zaměstnance (FR22). Tento případ užití umožní přiřadit zaměstnance k plnění úkolu.

UC07: Správa prostoru (FR07, FR08, FR12). Tento případ užití umožní administrátorům měnit parametry prostoru a přidávat inventář (velikost, inventář atd.). Administrátor změní potřebný parametr, přidá nový nebo ho odstraní.

UC08: Správa účtu zaměstnanců (FR11, FR12). Tento případ užití umožní administrátorům měnit účetní údaje zaměstnanců (upravovat osobní údaje, měnit oprávnění, pozice atd.)

UC09: Vyřadit zvíře z útulku (FR03). Tento případ užití umožní zaměstnancům vyřadit zvířata z útulku.

UC10: Správa zvířat (FR03, FR05, FR09, FR12, FR14). Tento případ užití umožní zaměstnancům spravovat zvířata v útulku. Zaměstnanec bude moci upravovat údaje o zvířeti, přidělovat mu stravu, měnit umístění, přidělovat procedury a mnoho dalšího.

UC11: Správa dokumentace (FR04, FR10, FR15, FR17 – FR20). Tento případ užití umožní zaměstnancům uchovávat a spravovat dokumentaci o útulku.

UC12: Zobrazení statistik a reportů (FR01 – FR04, FR12, FR14, FR21). Tento případ užití umožní administrátorům zobrazit statistiky útulku, ve kterých bude uvedena informace o počtů přijatých zvířat, kolik bylo vráceno majitelům atd.

UC13: Tisk a export dokumentace (FR19). Tento případ užití umožní zaměstnancům tisknout a exportovat různé smlouvy, dokumenty a protokoly vytvořené při vydání/přijetí zvířete do útulku atd.

UC14: Rozesílání upozornění (FR13, FR16, FR24). Tento případ užití umožní systému posílat upozornění na nadcházející události (krmení, úklid, prohlídka atd.)

UC15: Plnění úkolu (FR22). Tento případ užití umožní zaměstnancům vytvořit vlastní úkol a pracovat na něm.

UC16: Označit úkol jako splněný (FR25). Tento případ užití umožní označit úkol jako splněný.

UC17: Registrace a autorizace do útulku (FR23). Tento případ užití umožní registrovat a autorizovat se v útulku. Případ užití umožní přepínat mezi různými útulky.

4.3 Aktéři

Při psaní případů užití jsme popisovali, kdo vykonává různé akce. Podrobněji popišme roli každého aktéra.

- Zaměstnanec (Staff).

  Aktér, který vykonává většinu práce. Stará se o zvířata, vedou dokumentaci, pracuje na útulku a plánuje úkoly.

- Administrátor (Admin).

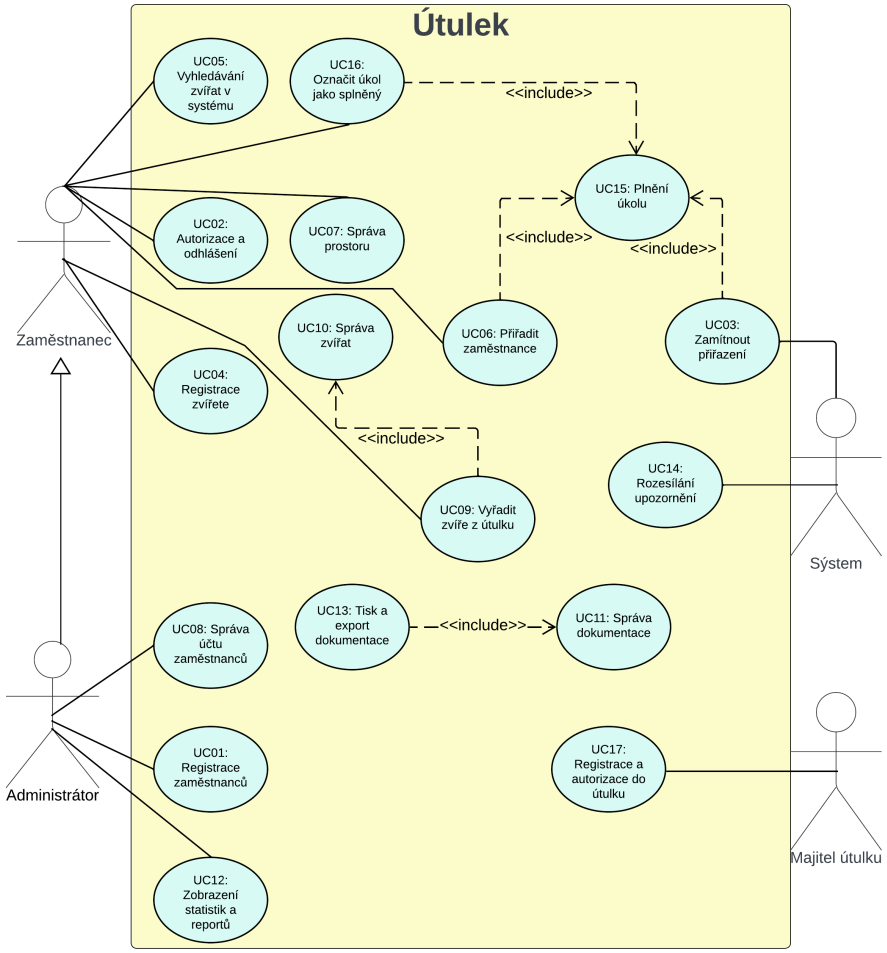
  Aktér, který je také zaměstnancem, ale má větší oprávnění, jako je změna údajů zaměstnanců, prohlížení statistik a reportů atd.

- Systém (System).

  Aktér, jehož cílem je zasílat upozornění, řídit útulek a kontrolovat dodržování podmínek při činnostech uživatele.

- Majitel útulku (Shelter owner).

  Aktér, který může vytvořit nový útulek.



Obrázek 4.1: Celkový pohled na systém na use case diagramu (UC diagram podle [23]).

4.4 Scénáře

Posledním krokem bude vytvoření různých scénářů. Cílem scénářů je modelování a návrh interakce uživatele se systémem v rámci jednoho scénáře. Podrobné postupné interakce uživatele se systémem.

Jelikož existuje mnoho možných vývojových variant událostí, uvedeme příklad dvou z nich. Obě scénáře (hlavní scénáře) budou zahrnovat odbočky (alternativní scénáře).

4.4.1 Scénář č. 1 „Plnění úkolu“ (UC15).

1. Uživatel může vybrat možnost „Autorizace“, aby se přihlásil do systému pod svým účtem. Pokud uživatel neexistuje, spustí se alternativní scénář č. 1. Zaměstnanec může vybrat možnost „Odhlášení“, čímž dokončí hlavní scénář.

2. Systém zobrazuje zaměstnanci úkoly, které je třeba provést.

3. Zaměstnanec je přiřazen k jejich provedení. V případě odmítnutí systémem se spustí alternativní scénář č. 2.

4. Zaměstnanec označuje úkol jako splněný po jeho provedení.

5. Zaměstnanec stiskne tlačítko „Odhlášení“ a odhlásí se ze systému.

4.4.2 Scénář č. 2 „Registrace zvířete“ (UC04).

1. Uživatel může vybrat možnost „Autorizace“, aby se přihlásil do systému pod svým účtem. Pokud uživatel neexistuje, spustí se alternativní scénář č. 1. Zaměstnanec může vybrat možnost „Odhlášení“, čímž dokončí hlavní scénář.

2. Zaměstnanec vybere možnost „Registrace zvířete“a vyplní všechny příslušné údaje o zvířeti. Zvíře je uloženo do systému. Pokud je zvíře již registrováno, systém odesílá o tomto oznámení a scénář končí.

3. Zaměstnanec vyhledává zvíře ve systému pomocí možnosti „Vyhledávání zvířat“a zadává odpovídající kritéria vyhledávání.

4. Zaměstnanec zvíře přiřadí k veterinárnímu vyšetření, po čemž systém vytvoří úkol pro diskusi o novém zvířeti.

5. Spustí se scénář „Plnění úkolu“od bodu č. 2 až po bod č. 4 včetně.

6. Pokud je zvíře nemocné, zaměstnanec ho umístí do karantény, jinak ho zaměstnanec umístí do vhodného ubytovacího zařízení. Zvíře je úspěšně umístěno. Pokud systém neumožní umístit zvíře, spustí se alternativní scénář č. 3.

7. Zaměstnanec stiskne tlačítko „Odhlášení“a odhlásí se ze systému.

4.4.3 Alternativní scénář č. 1 „Registrace“ (UC01).

1. Administrátor se přihlásí do systému.
2. Administrátor vybere možnost „Registrace nového zaměstnance“pro přidání nového zaměstnance do systému.
3. Administrátor vyplní potřebné údaje o novém zaměstnanci, jako je jméno, pozice, kontaktní informace atd.
4. Po vyplnění všech povinných polí administrátor uloží informace a nový zaměstnanec je úspěšně přidán do systému.
5. V případě potřeby administrátor upraví údaje o zaměstnanci.

4.4.4 Alternativní scénář č. 2 „Odmítnutí úkolu“ (UC15).

Předpoklad: V útulku je zaměstnanec s potřebnými oprávněními k provedení úkolu. Zaměstnanec je přihlášen do systému.

1. Systém zamítne zaměstnance k provedení úkolu.
2. Další zaměstnanec je přiřazen k provedení úkolu.

4.4.5 Alternativní scénář č. 3 „Odmítnutí umístění“ (UC10).

Předpoklad: V útulku jsou k dispozici vhodné prostory pro umístění. Zaměstnanec je přihlášen do systému.

1. Systém odmítne umístit zvíře zaměstnanci.
2. Zaměstnanec zkontroluje důvod odmítnutí.
3. Zaměstnanec vyhledá vhodné prostory na základě důvodu odmítnutí.
4. Zaměstnanec umístí zvíře do prostor.

Kapitola 5

Návrh vlastního řešení

V této kapitole budeme zkoumat možnosti řešení, které budou použity v rámci této bakalářské práce. Vzhledem k nutnosti vytvořit systém tak, aby bylo možné provádět změny, rozšiřovat funkcionalitu a zároveň zapojovat nové lidi do projektu, musí systém dodržovat veškeré přijaté standardy a postupy pro maximální efektivitu a pohodlí. V této kapitole budou zkoumány základní aspekty navrženého řešení, jeho cíle a úkoly, stejně jako metody a nástroje použité při jeho vývoji.

5.1 Programovací jazyk

Jedním z klíčových aspektů při vytváření aplikace je volba programovacího jazyka a dostupných nástrojů. V našem případě jsme zvolili programovací jazyk Java.

Java [25] je přísně typovaný objektově orientovaný programovací jazyk obecného použití, který patří společnosti Oracle.

Aplikace v Javě jsou obvykle přeloženy do speciálního bajtového kódu, což jim umožňuje běžet na libovolné počítačové architektuře, pro kterou existuje implementace virtuální Java stroje (JVM). Java rovněž zaujímá vysoká místa v žebříčcích oblíbenosti programovacích jazyků. Obrovská popularita, což znamená velké množství informací, a existence JVM, která umožňuje nebrat ohled na výběr platformy, hrály klíčovou roli při výběru programovacího jazyka.

5.2 Framework

Nicméně nelze Java zkoumat bez zmínky o frameworku, protože při pohledu na zdrojový kód našeho systému je zřejmé jeho široké použití a výrazné zjednodušení práce. Zvolili jsme nejznámější framework pro Javu – Spring Boot.

Spring Framework (dále jenom Spring) [26] je univerzální framework s otevřeným zdrojovým kódem pro Java platformu. I když Spring nenabízí konkrétní programovací model, stal se široce rozšířeným v Javové komunitě, především jako alternativa a náhrada zastaralého modelu Enterprise JavaBeans. Spring poskytuje vývojářům v Javě větší volnost při návrhu a nabízí dobře zdokumentované a snadno použitelné nástroje pro řešení problémů, které vznikají při vytváření aplikací.

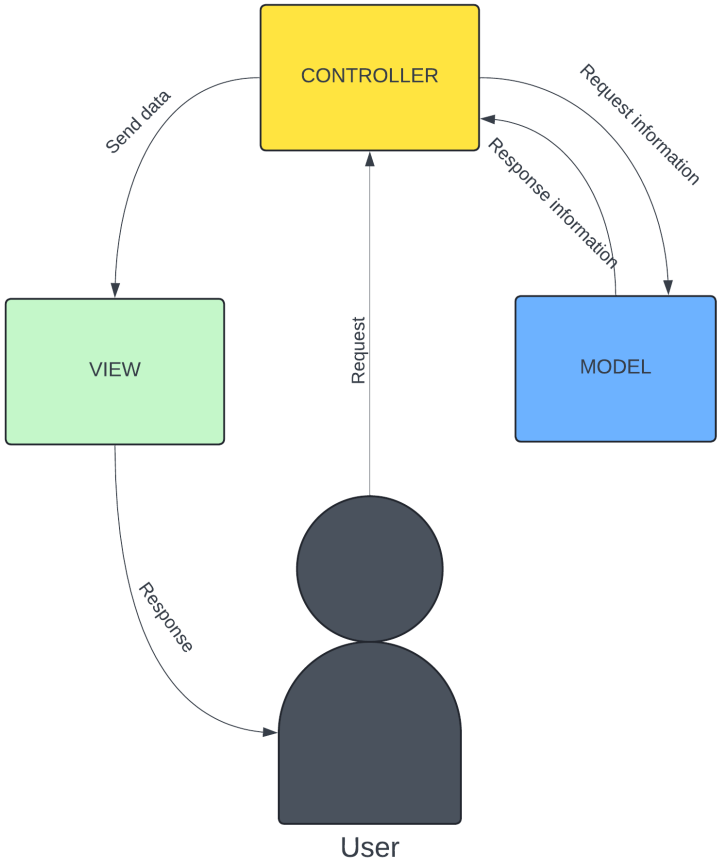
Důvody použití Springu:

- Inversion of Control kontejner – delegovaná konfigurace komponent aplikací a řízení životního cyklu objektů v Javě.
- Framework pro autentizaci a autorizaci – nástroje pro autentizaci a autorizaci podporující mnoho populárních protokolů a standardů pomocí Spring Security.
- MVC [28] framework – framework založený na HTTP [29] a servletech, poskytující mnoho možností pro rozšíření a konfiguraci.
- Framework vzdáleného přístupu – konfigurovatelné předávání Java objektů přes síť pomocí HTTP protokolů.
- Testování – podpora tříd pro psaní modulárních a integračních testů.

5.3 Architektonický vzor

Pro náš systém byl vybrán nejpopulárnější architektonický vzor Model-View-Controller (MVC) [28]. MVC je softwarový návrhový vzor, který je obvykle používán při vývoji uživatelských rozhraní a rozděluje provázanou softwarovou logiku do tří vzájemně propojených prvků: Model (Model), Pohled (View) a Řadič (Controller). Model je zodpovedný za business logiku aplikace data, pohled zobrazuje data uživateli, zatímco řadič interaguje s uživatelem, přijímá jeho vstupy a na základě těchto interakcí aktualizuje pohled a model.

I když je tento vzor tradičně používán pro grafická uživatelská rozhraní na stolních počítačích, stal se populárním i v webovém vývoji. Mnoho běžně používaných programovacích jazyků, včetně Javy, poskytuje struktury, které usnadňují implementaci vzoru MVC.



Obrázek 5.1: Komunikace v MVC architektuře (MVC diagram podle [28]).

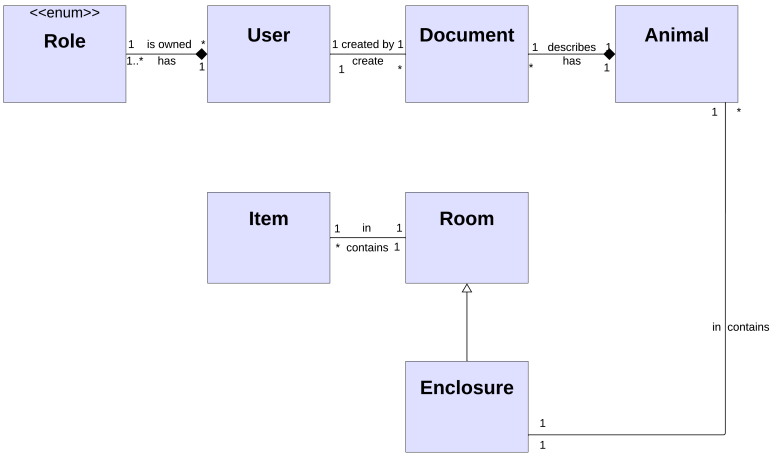
Na obrázku 5.1 je zobrazena podrobná schéma komunikace mezi uživatelem a aplikací, stejně jako vnitřní komunikace mezi různými komponentami aplikace. Uživatel odesílá požadavek, který je přijat řadičem. Řadič poté tento požadavek předává modelu, kde je zpracován a následně vrácen zpět. Poté, co je požadavek vyřízen, výsledek je zobrazen uživateli pomocí pohledu.

5.4 Návrh řešení

Dalším důležitým krokem bude navržení implementace. Z kapitoly 4 vybereme komponenty, na které je třeba zaměřit pozornost. Vzhledem k tomu, že vytváříme systém pro útulek, musí v něm být zahrnuty zvířata, místo jejich pobytu a zaměstnanci, kteří se budou o ně starat. Zaměstnanec bude mít různé odpovědnosti, což je také třeba zohlednit. V části bylo často zmíněno, že je třeba sledovat každý krok při práci se zvířaty, takže zaměstnanci musí vést dokumentaci. V útulku určitě bude inventář, jehož správa je také třeba zvážit.

5.4.1 Diagram tříd

Výše popsané komponenty jsou také třídami, které budou implementovány. Pro zobrazení interakce mezi nimi použijeme UML [23] diagram tříd.

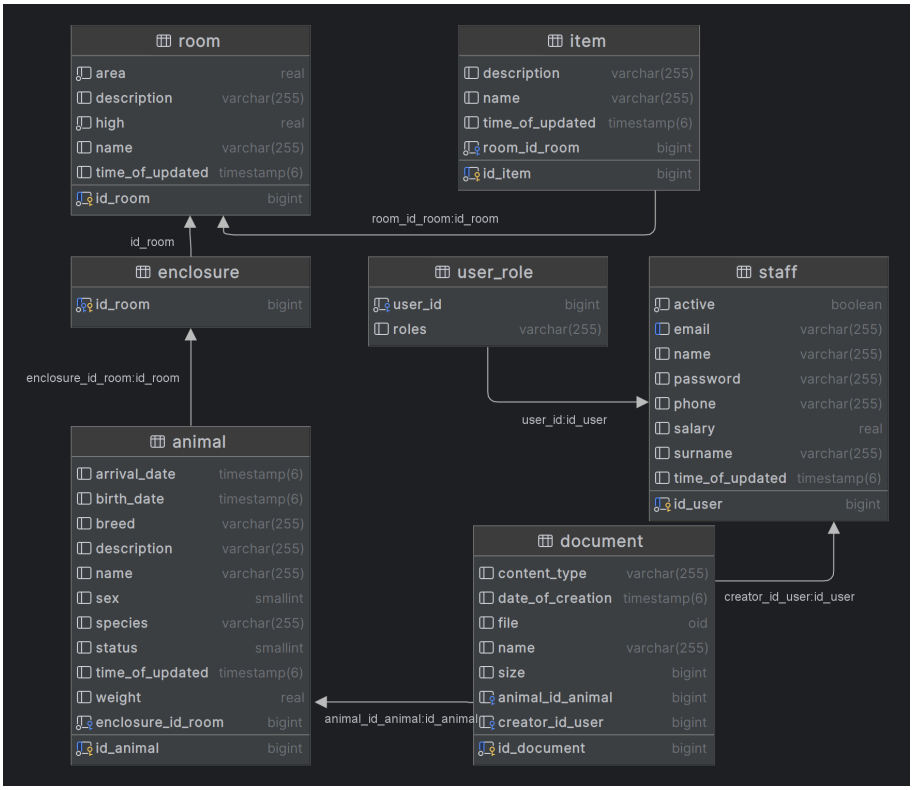


Obrázek 5.2: Diagram tříd (Diagram tříd podle [23]).

Odpovědnosti zaměstnanců (User) byly rozhodnuto realizovat pomocí běžného výčtového typu, kde budou uvedeny všechny existující odpovědnosti (Role). Současně může uživatel mít více rolí, například zaměstnanec může být současně lékařem a administrátorem. Toto řešení usnadní budoucí rozšíření funkcionality, protože pro vytvoření nové role stačí přidat její záznam do třídy Role. Dokumentace (Document) musí být vedena určitým zaměstnancem a věnována konkrétnímu zvířeti. Jeden zaměstnanec může vést libovolné množství dokumentace a jednomu zvířeti (Animal) může být přiřazeno libovolné množství dokumentace. Zvířata musí být umístěna ve speciálních místnostech – v kleci (Enclosure). Protože klec je pouze typem místnosti, bylo rozhodnuto udělat ji potomkem třídy místnost (Room), která zachová všechny její atributy. Jedno zvíře může být umístěno pouze v jedné kleci, zatímco v jedné kleci může být umístěno několik zvířat. Věci (Item) budou umístěny v místnostech. V jedné místnosti může být libovolné množství věcí, a jedna věc může být umístěna pouze v jedné místnosti.

5.4.2 Relační schéma

Další otázkou, která vyžaduje řešení, je otázka ukládání dat a jejich atributů. Data budou uložena v databázi PostgreSQL, díky její rozsáhlé funkčnosti a kompatibilitě s Javou. Nyní musíme porozumět, jaké informace budou uchovávány našimi třídami. Pro lepší představu se podíváme na diagram 5.3 a okomentujeme nezřejmé atributy nebo jejich účel.



Obrázek 5.3: Relační schéma (Relační schéma podle [23]).

- Staff – tabulka zaměstnance útulku. Pole email slouží nejen jako způsob kontaktu, ale také jako uživatelské jméno účtu při autentizaci. Pole active ukazuje dostupnost účtu a má výchozí hodnotu true.
- User_role – tabulka, která zobrazuje seznam uživatelských rolí.
- Document – tabulka obsahující samotný soubor jako pole bytů a veškeré potřebné informace o souboru, včetně tvůrce a zvířete, ke kterému je dokument připojen.
- Animal – tabulka zvířete v útulku. Všechny pole slouží k popisu zvířete. Obsahuje klíč klece, ve které se nachází.
- Enclosure – pomocná tabulka, která spojuje Animal a Room.

- Room – tabulka popisující místnost.
- Item – tabulka předmětu. Obsahuje klíč místnosti, ke které je připojen.

5.5 Representational State Transfer

Representational State Transfer (REST) je architektonický styl pro interakci komponentů distribuované aplikace v síti [33].

Jinými slovy, REST je soubor pravidel, jak programátorovi organizovat psaní kódu serverové aplikace tak, aby všechny systémy snadno vyměňovaly data a aplikaci bylo možné škálovat. REST představuje sjednocenou sadu omezení, které se berou v úvahu při návrhu distribuované hypermédiální systému.

V případě, že je aplikace postavena v souladu s paradigmatem REST (neodporuje mu), takovou aplikaci nazýváme RESTful. Dalšími charakteristickými principy architektury REST jsou: bezestavovost (viz 5.5.1), vlastnosti zdrojů (viz 5.5.2), vlastnosti metod (viz 5.5.3), stavové kódy (viz 5.5.4) a verzování (viz 5.5.5).

5.5.1 Bezestavovost

Koncept REST nepředpokládá, že objekt na serveru má stav (komunikace vždy probíhá jako poprvé) a všechny objekty mají slovesa:

- GET – získání prostředku
- POST – vytvoření prostředku
- PUT – aktualizace prostředku
- DELETE – odstranění prostředku

V případě REST by požadavek na získání, aktualizaci nebo odstranění objektu s id 123 a vytvoření nového objektu vždy vypadal následovně:

```
GET http://myshelter.com/items/123
POST http://myshelter.com/items
PUT http://myshelter.com/items/123
DELETE http://myshelter.com/items/123
```

V těle POST a PUT požadavků jsou předávány informace objektů. Obvykle ve formátu JSON nebo XML.

5.5.2 Vlastnosti zdrojů

Tato vlastnosti předpokládá, že pro každý zdroj se používá vlastní URL [34]. To znamená, že pokud máme API, pak pro každou akci s každým zdrojem bude použita vlastní URL, například pro nahrání souboru zaměstnancem a načtení všech zaměstnanců budou použity následující URL adresy:

```
http://myshelter.com/staffs/upload-file
http://myshelter.com/staffs/read-all
```

5.5.3 Vlastnosti metod

Tato vlastnosti předpokládá vytvoření operací CRUD v zdrojích, proč bychom neměli najít způsob, jak tyto operace použít pro všechny zdroje.

Tuto myšlenku lze realizovat použitím HTTP metod. Pokud chceme získat seznam zvířat, můžeme provést požadavek Get na zdroj, ale pokud chceme vytvořit nové zvíře, můžeme použít POST metody namísto GET na stejný zdroj, například:

```
GET http://myshelter.com/animals
POST http://myshelter.com/animals
```

5.5.4 Stavové kódy

Při návrhu REST API je velmi důležité používat odpovídající HTTP stavové kódy, zejména při vývoji a testování RESTful API. Nejčastěji používané stavové kódy [29] jsou:

- 200 – OK: Vše funguje.
- 201 – CREATED: Byl vytvořen nový zdroj.
- 204 – NO CONTENT: Zdroj byl úspěšně odstraněn, není žádný obsah odpovědi.
- 304 – NOT MODIFIED: Vrácená data jsou dočasně uložená (data nebyla změněna).
- 400 – BAD REQUEST: Požadavek není správný nebo nemůže být zpracován. Konkrétní chyba by měla být vysvětlena v těle odpovědi.
- 401 – UNAUTHORIZED: Požadavek vyžaduje ověření uživatele.
- 403 – FORBIDDEN: Server přijal požadavek, ale odmítl ho, nebo přístup k zdroji je zakázán.
- 404 – NOT FOUND: Zdroj na dané URL není nalezen.
- 500 – INTERNAL SERVER ERROR: Při návrhu API je vhodné se vyhnout této chybě. Všechny vnitřní chyby by měly být zpracovány serverem a vráceny v čitelné formě.

5.5.5 Verzování

Verzování umožňuje poskytovat různé verze zdroje a vždy udržovat zpětnou kompatibilitu zdroje. To se dělá proto, aby uživatelé starší verze API měli možnost pokračovat v používání aplikace po provedení nekompatibilních změn. Toto je snadno dosaženo pomocí hlavičky.

Namísto

```
GET http://myshelter.com/animals
POST http://myshelter.com/animals
```

Použít

```
GET http://myshelter.com/v1/animals
POST http://myshelter.com/v1/animals
```

5.6 Frontend

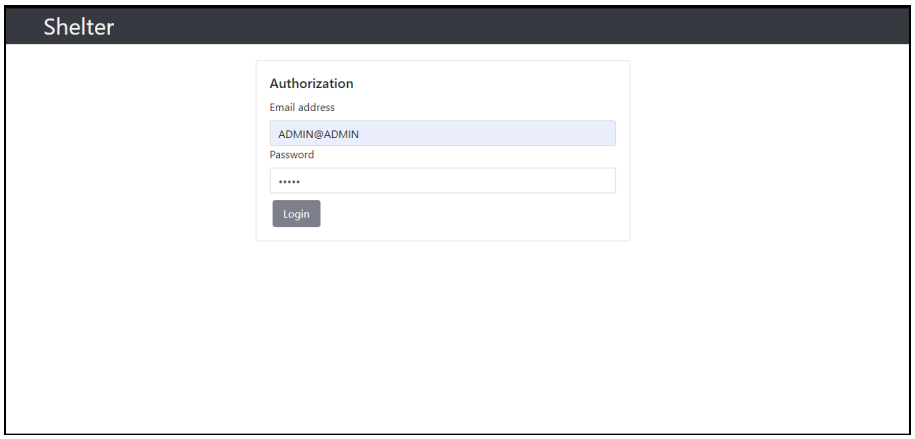
Pro ovládaní s aplikací potřebujeme uživatelské rozhraní (dále jen UI). K tomuto účelu používáme webové stránky. Připravili jsme hi-fi prototypy. Abychom si udělali lepší představu o implementaci, podíváme se na ně i na další použité nástroje.

5.6.1 Hi-fi prototypy

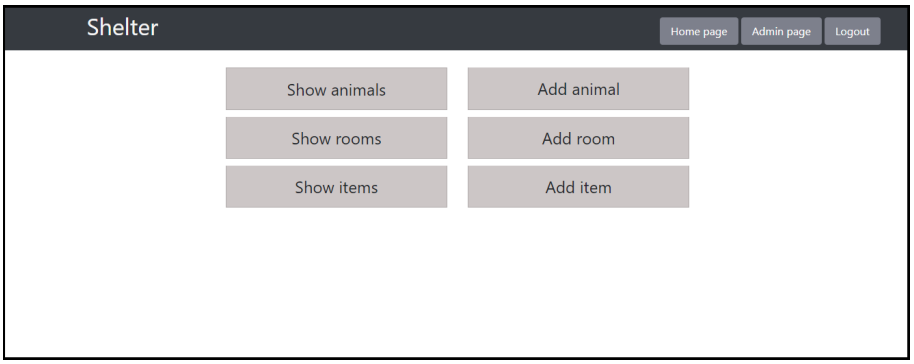
Hi-fi prototyp – vysoce detailní prototyp, tj. co nejblíže skutečnému produktu, kde všechny prvky vypadají jako na skutečném webu.

Pro rychlý přístup k hlavním funkcím bude na všech webových stránkách k dispozici navigační panel. Vlevo se nachází název stránky a vpravo tlačítko pro přihlášení/odchod, tlačítko domovské stránky a tlačítko panelu správce.

Zobrazení tlačítek na navigačním panelu bude různé. Pokud tedy uživatel není autorizován, budou mu skryta všechna tlačítka.

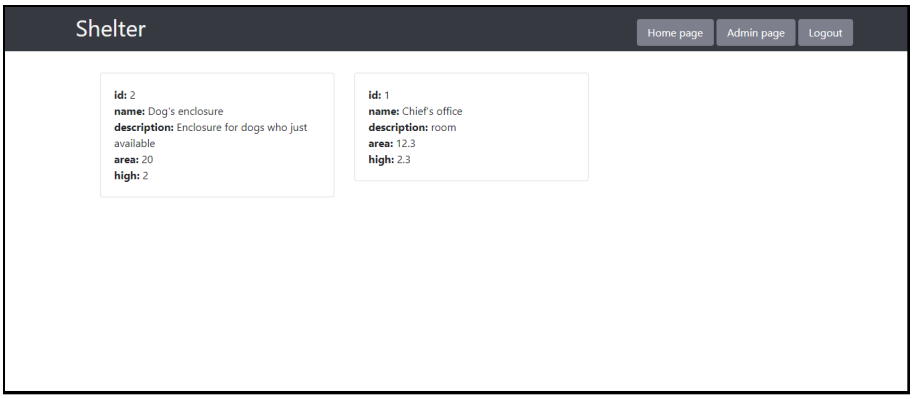


Obrázek 5.4: Stránka pro autorizaci uživatele (Vlastní Hi-fi prototyp).



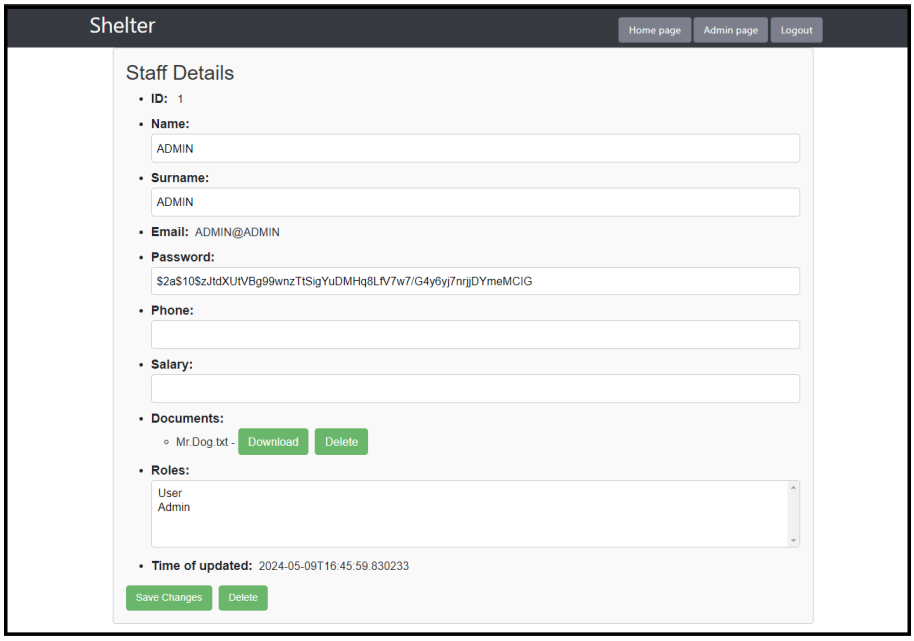
Obrázek 5.5: Domovská stránka (Vlastní Hi-fi prototyp).

Na obrázku 5.5 je domovská stránka. Uživatel bude po autorizaci automaticky přesměrován na domovskou stránku. Tato stránka slouží k přístupu na další stránky, na kterých můžete prohlížet, upravovat, odstraňovat nebo přidávat záznamy.
Po kliknutí na tlačítko Show ... se uživatel dostane na stránku se seznamem všech prvků. Příklad je uveden na obrázku 5.6.
Po kliknutí na tlačítko Add ... se otevře formulář se všemi informacemi, které je třeba vyplnit pro přidání nového prvku. Příklad je uveden na obrázku 5.8.



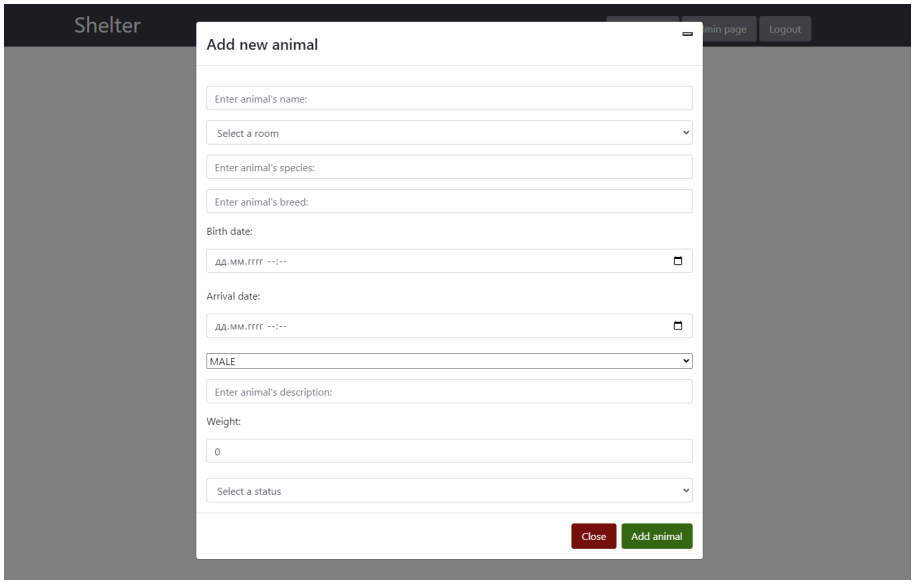
Obrázek 5.6: Seznam místností (Vlastní Hi-fi prototyp).

Na této stránce se uživateli zobrazí stručné informace a po kliknutí na požadovaný prvek bude uživatel přesměrován na stránku s úplnými informacemi o daném prvku (Příklad je uveden na obrázku 5.7).

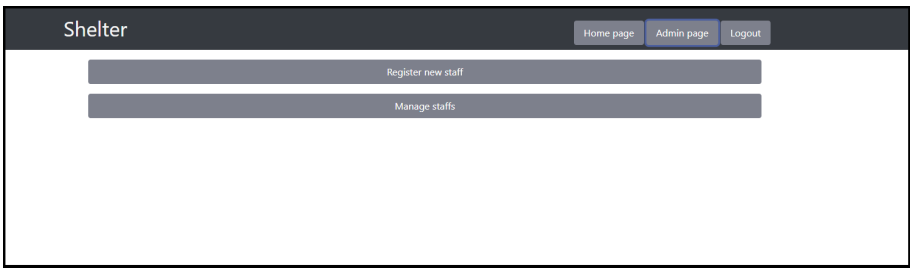


Obrázek 5.7: Stránka zaměstnance (Vlastní Hi-fi prototyp).

Na stránce zaměstnance vidíme všechny informace o zaměstnanci. Uživatel zde může změnit některé údaje a poté změny uložit nebo je zcela vymazat.



Obrázek 5.8: Přidání zvířete (Vlastní Hi-fi prototyp).



Obrázek 5.9: Stránka správce (Vlastní Hi-fi prototyp).

Pokud je uživatel autorizovaný a má roli správce, je mu k dispozici tlačítko administrátorské stránky. Po kliknutí na něj bude uživatel přesměrován na stránku s dalšími funkcemi. Příklad je uveden na obrázku 5.9.

5.6.2 React

K napsání webové stránky byl zvolen React [32]. React je knihovna jazyku JavaScript [31] od společnosti Facebook pro snadný vývoj UI webových stránek a aplikací. Hlavním rysem této knihovny jsou komponenty a stavy.

Komponenta je kus kódu, který je zodpovědný za vzhled jednoho z prvků webu nebo aplikace. K vytvoření komponenty se používá speciální jazyk JSX, který vypadá jako směs JavaScriptu a HTML [30]. Stav jsou všechny informace o prvku, včetně jeho zobrazení.

Jinými slovy, použití knihovny React je způsob, jak jednoduše prezentovat kód JavaScript a HTML.

5.6.3 Schema.org

„Schema.org je společná komunitní aktivita, jejímž cílem je vytvářet, udržovat a propagovat schémata pro strukturovaná data na internetu, na webových stránkách, v emailových zprávách i mimo ně.“ [35]

Rozdíl mezi vyhledávacím algoritmem a člověkem spočívá v tom, že algoritmus není schopen pochopit význam obsahu webové stránky bez speciálních pokynů. Zatímco člověk dokáže analyzovat obsah, zvýraznit vzory, klíčová slova a další technické charakteristiky, vyhledávací algoritmus není schopen interpretovat význam obsahu. Aby tento problém vyřešili, byly vytvořeny pro vyhledávače sémantické mikroznaky. Jedná se o speciální značení webové stránky pomocí značek a atributů, které pomáhá robotům vyhledávačů, robotům sociálních sítí a dalším automatickým systémům pochopit obsah každé stránky nebo samostatného bloku obsahu a také téma celého webu jako celku. Sadu těchto mikroznaků poskytuje Schema.org.

Příklad použití bez mikroznaků

```
<div>
    <span>San Francisco 49ers</span>
    <div>
        <span>Joe Montana</span>
        <span>1979</span>
        <span>1992</span>
        <span>Quarterback</span>
    </div>
</div>
```

Příklad použití s mikroznaky

```
<div itemscope itemtype="https://schema.org/SportsTeam">
    <span itemprop="name">San Francisco 49ers</span>
    <div itemprop="member" itemscope
    ↪  itemtype="https://schema.org/OrganizationRole">
        <div itemprop="member" itemscope
        ↪  itemtype="https://schema.org/Person">
            <span itemprop="name">Joe Montana</span>
        </div>
        <span itemprop="startDate">1979</span>
        <span itemprop="endDate">1992</span>
        <span itemprop="roleName">Quarterback</span>
    </div>
</div>
```

Pomocí atributů itemscope itemtype určujeme název šablony a pomocí atributu itemprop určujeme vlastnosti objektu.

Po přezkoumání webových stránek našeho útulku byly identifikovány následující typy mikroznaček, které lze použít: **Person, Thing, Room, CreativeWork, DataFeedItem, ListItem.**

- **Person** se používá pro stránku zaměstnance (staff) a zvířete (animal). Má následující vlastnosti: name, birthDate, gender, description, weight, email, telephone atd.
- **Thing** se používá pro stránku věci (item). Má následující vlastnosti: name a description.
- **Room** se používá pro stránku místnosti (room). Má následující vlastnosti: name, description, floorSize, petsAllowed
- **CreativeWork** se používá pro dokumentace (document). Má následující vlastnosti: name a description.
- **DataFeedItem** se používá pro atribut čas aktualizace (timeOfUpdated). Má následující vlastnosti: dateModified.
- **ListItem** se používá pro seznam věcí (items) na stránce Room. Má následující vlastnosti: item.

Kapitola 6

Implementace, dokumentace a testování

Během vytváření informačního systému se také objevily různé problémy. Jaký je nejlepší způsob realizace nějakého UC? Jak bude na dalším vývoji pracovat člověk zcela neznalý systému? Anebo jak se vyhnout novým problémům při snaze rozšířit stávající funkcionalitu? Odpovědím na tyto otázky je věnována tato kapitola. Probereme zde zajímavá či problematická řešení, způsob tvorby projektové dokumentace a samozřejmě testování našeho programu.

6.1 Implementace

Během realizaci nápadů popsaných v předchozích kapitolách jsme se museli potýkat s některými obtížemi: od hledání různých triků, které by usnadnily práci, až po několikadenní hledání nejasné chyby. To vše bude popsáno v odstavcích 6.1.1 až 6.1.3.

6.1.1 Použití dědičnosti a šablon

Trik, který výrazně snížil počet řádků kódu – použití dědičnosti a šablon. Protože mnoho služeb (services) a řadičů (controllers) má operace CRUD a ne vždy vyžadují jedinečnou implementaci, bylo vytvoření třídy předka s připravenou implementací těchto operací prvním úkolem, který bylo třeba provést. Ke CRUD operacím byla přidána také operaci pro ctění seznamu všech prvků.

CrudService class

```
1 public abstract class CrudService <T extends DataID<ID>, ID, R
  ↪  extends CrudRepository<T, ID>>{
2     protected final R repository;
3     public T create(T data){...}
4     public Iterable<T> readAll(){...}
5     public Optional<T> readById(ID id){...}
6     public void update(ID id, T data){...}
7     public void deleteById(ID id){...}
8 }
```

CrudController class

```
public abstract class CrudController<
        T extends DataID<ID>, ID,
        S extends CrudService<T, ID, R>,
        R extends CrudRepository<T, ID>,
        M extends SuperMapper<T, D>,
        D extends DataID<ID>
        > {
    protected final S service;
    protected final M mapper;
    @PostMapping
    public ResponseEntity<?> create(@RequestBody D data){...}

    @GetMapping
    public ResponseEntity<Iterable<D>> readAll(){...}

    @GetMapping("/{id}")
    public ResponseEntity<?> readById(@PathVariable ID id){...}

    @PutMapping("/{id}")
    public ResponseEntity<?> update(@PathVariable ID id,
    ↪  @RequestBody D data){...}

    @DeleteMapping("/{id}")
    public ResponseEntity<?> deleteById(@PathVariable ID
    ↪  id){...}
}
```

K předání požadovaných parametrů byly použity šablony. Implementace **CrudService (S)** a **CrudController** byla uvedena výše. **CrudRepository (R)** je již implementován ve Spring Boot. **SuperMapper (M)** je rozhraní, ze kterého se dědí mapovač (**mapper**). **DataID** je rozhraní, ze kterého se dědí entity **(T)** a DTO **(D)**.

Toto řešení eliminuje nutnost opakovat stejný kód pro různé třídy. Stačí předat šabloně potřebné parametry a nezapomenout, aby třída byla dědicem **CrudService** nebo **CrudController**.

6.1.2 Mapovač

Další problém, který se během běhu vyskytl, bylo přetečení zásobníku. Pro řešení tohoto problému byly vytvořeny DTO. Kromě samotných DTO však byly zapotřebí také převodníky. Naštěstí Spring Boot již disponuje potřebnou funkcí pro rychlé vytvoření konvertoru – mapovač. K jeho implementaci stačí vytvořit rozhraní a specifikovat, co chceme převádět. Všechny třídy použité v mapovači musí mít implementovány gettery a settery. Zde nám opět pomohla dědičnost, díky které jsme se vyhnuli nutnosti psát řádky kódu.

SuperMapper class

```
public interface SuperMapper <E, D>{
    D toDTO(E entity);
    E toEntity(D dto);
    Iterable<D> toDTO(Iterable<E> entities);
    Iterable<E> toEntity(Iterable<D> dtos);
}
```

6.1.3 Anotace

Spring Boot má obrovský výběr hotových funkcí, které dělají práci programátora. Stačí jen zadat, kde a co je třeba udělat. Jedná se o anotace. Například pomocí anotace byla vytvořena kompletní dokumentace (viz 6.2). Dalším příkladem využití anotací je generování getterů, setterů, kontruktorů, REST řadičů a mapovačů (viz 6.1.2). Níže je uvedeno několik příkladů použití těchto poznámek.

- **AnimalController class. REST řadič.**

```
@RestController
@RequestMapping("api/v1/animals")
public class AnimalController{...}
```

  Příklad použití anotací v metodách je v 6.1.1.

- **Room class. Gettery, settery a konstruktor.**

```
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Room{...}
```

- **UserMapper class. Mapovač.**

```
@Mapper(componentModel = "spring", uses =
↪  {DocumentMapper.class})
public interface UserMapper{...}
```

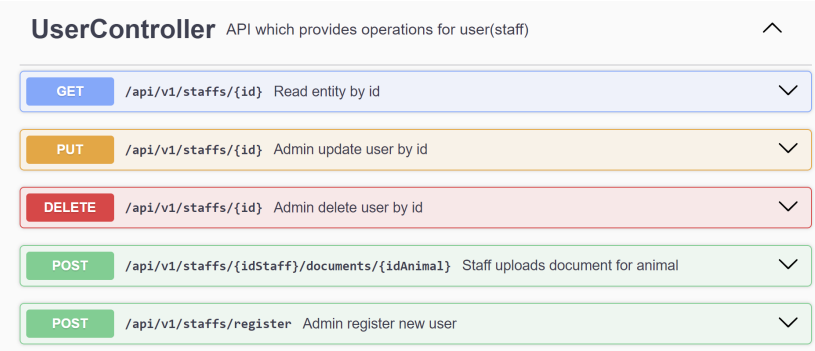
  V **uses** jsou specifikovány mapovače vnořených objektů.

- **UserController class. Dokumentace.**

```
@Operation(summary = "Admin delete user by id")
@ApiResponse(responseCode = "204",
             description = "User was deleted successfully")
@ApiResponse(responseCode = "404",
             description = "User not found")
public ResponseEntity<?> deleteById(@PathVariable Long id)
```

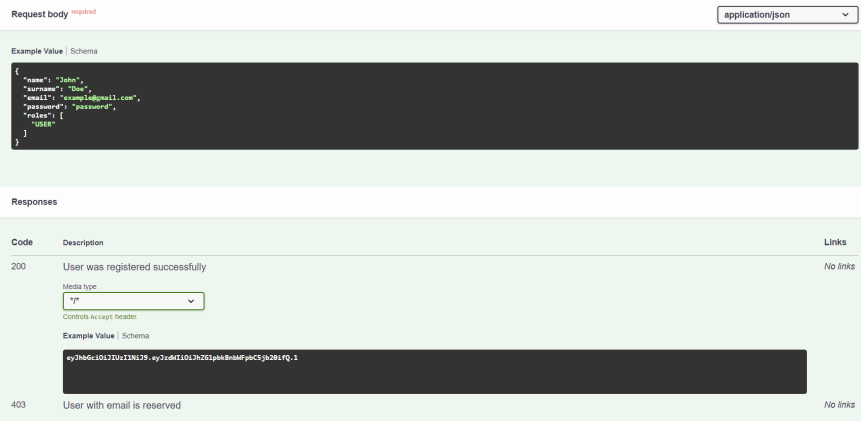
6.2 Dokumentace

Aby jsme vyřešili problém seznámení nových uživatelů s rozhraním API, rozhodli jsme se vytvořit k němu dokumentaci. Pro tento účel jsme zvolili Swagger [36]. Protože Spring Boot má možnost generovat dokumentaci automaticky, zvolili jsme tento způsob. Pro generování stačilo použít anotace (viz 6.1.3). Podívejme se na atributy, které jsou uvedeny v naší dokumentaci. Rozebereme si je na konkrétních příkladech.



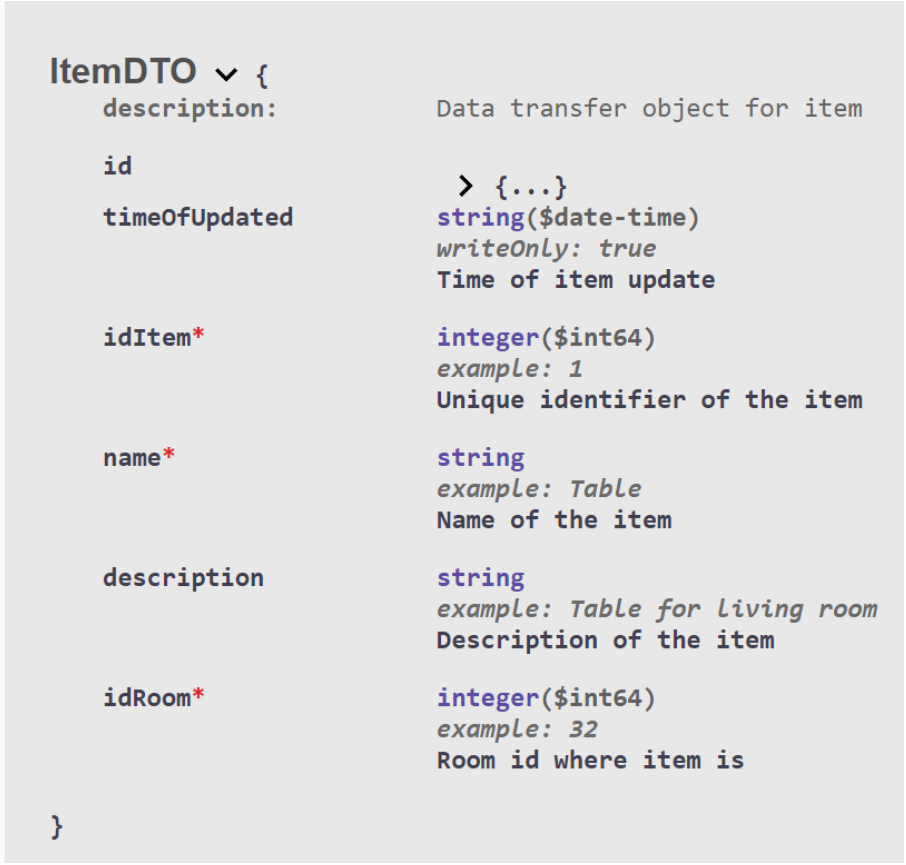
Obrázek 6.1: Dokumentace k UserController ve swaggeru (Snímek obrazovky z generované dokumentace. Koncové body UserController).

Na obrázku 6.1 jsou všechny koncové body, které UserController poskytuje. U každého koncového bodu je také uveden jeho popis a operace, kterou provádí (viz 5.5.1).



Obrázek 6.2: Další informace o koncovém bodě pro registrace (Snímek obrazovky z generované dokumentace. Další informace o koncovém bodě /register).

Na obrázku 6.2 je další informace o koncovém bodě pro registrace nového uživatele. Uvádí příklady request body a stavů odpovědí (status code) pro každý možný případ.



Obrázek 6.3: Schéma ItemDTO (Snímek obrazovky z generované dokumentace. Schéma ItemDTO).

Na obrázku 6.3 jsou informace o schématu ItemDTO. Vidíme všechna pole, jejich datový typ a příklad.

6.3 Testování

Testování hraje při psaní programů důležitou roli. I když jsme si jisti, že vše funguje tak, jak bylo navrženo, neznamená to, že pokud se v budoucnu pokusíme něco změnit, neporušíme funkčnost. Rozdělme si tuto část na dvě. První část bude věnována technické realizaci testů a ve druhé části se zbývá předmětem testováni.

6.3.1 Mockito, MockMvc a JUnit5

- Mockito je framework pro izolované testování [37]. Díky Mockito můžeme určit určité chování pro část programu, kterou právě netestujeme, ale od které očekáváme určité chování.

Moduly, chování kterých chceme konfigurovat, budou označeny anotací @Mock. Modul, který chceme testovat, bude označen anotací @InjectMocks. Nyní nastavíme chování modulu.

```
//Očekáváme, že objekt s id 1 neexistuje.
Mockito.when(service.readById(1)).thenReturn(Optional.empty());
//Očekáváme, že metoda readById na objektu s id 1 zavolána
↪  2krát.
Mockito.verify(service, Mockito.times(2)).readById(1);
```

Nyní si můžeme být jisti, že při testování našeho modelu se nemusíme bát, že by chyba v jiném modulu jakkoli ovlivnila výsledek.

- MockMvc je framework pro testování na straně serveru, který umožňuje ověřit funkčnost aplikace Spring MVC pomocí snadných a cílených testů [38].

```
//Voláme metodu get s parametrem 1.
mockMvc.perform(get(URL + "/{id}", 1))
    // Očekáváme vrácení kódu 200.
    .andExpect(status().isOk())
    .andExpect(result -> {
        String json =
        ↪  result.getResponse().getContentAsString();
        String expected =
        ↪  objectMapper.writeValueAsString(dto);
        //Očekáváme, že vrácený výsledek bude totožný s
        ↪  očekávaným.
        Assertions.assertEquals(expected, json);
    });
```

- JUnit5 je framework určený pro automatické testování jednotek (tzv. unit testing) programů [39].

```
@BeforeEach // Pomocí této anotace provedeme nastavení před
↪  spuštěním testů.
void setUp() {
    mockMvc =
    ↪  MockMvcBuilders.standaloneSetup(controller).build();
    objectMapper = new ObjectMapper();
}

@Test // Pomocí této anotace označujeme metodu jako
↪  testovací.
void TestReadAll(){...}
```

6.3.2 Testování funkcionality

Pro kontrolu provozuschopnosti našeho informačního systému byly vytvořeny unit testy. Zahrnují všechny služby a řadiče. Vytvořili jsme také samostatný

test pro jeden repozitář, protože SQL dotaz tam byl napsán ručně. Celkový počet jednotkových testů je 73 ve 13 modulech.

Při tvorbě testů nebyly zjištěny žádné chyby, ale po změně malé funkcionality začalo několik testů zobrazovat chyby. Díky tomu byly chyby rychle odstraněny a testy zdůvodnily svou existenci.

Kromě testování jednotek bylo provedeno také testování podle případů užití. Pouze UC 3, 6 a 12 až 17 (viz 4.2) nebyly testovány, protože FR, které je pokrývají, mají nízkou prioritu (viz 4.1.1) a nebyly implementovány. Také toto testování bylo většinou pokryto jednotkovým testováním. Zbývající funkčnost (včetně frontendu) byla pokryta uživatelským testováním.

Během uživatelského testování byly zjištěny a odstraněny chyby s nesprávným zobrazením údajů, předčasné vyvolání modulů pro přidávání věcí a zvířat a také duplicita tlačítek na autorizační obrazovce.

Kapitola 7

Zhodnocení a další rozvoj

Přes veškerou vykonanou práci je náš informační systém stále ve stavu prototypu. Základní funkce byly splněny, ale pokud porovnáme náš systém s již existujícími, je zřejmé, že je třeba udělat ještě mnoho práce na jeho vylepšení a rozšíření. V této kapitole vyhodnotíme náš prototyp a porovnáme ho s řešeními popsanými v kapitole 3. Na základě tohoto porovnání a na základě nerealizovaných FR s nízkou prioritou (viz 4.1.1) navrhneme další kroky vývoje. Všechna řešení jsou dostupná z prohlížeče, proto se o nich v tomto porovnání nebudeme zmiňovat.

7.1 Porovnání s ASM

Jak již bylo řečeno dříve (viz 3.1), porovnání s ASM bude popsáno ve větším rozsahu než s jinými řešeními, a to z důvodu jednoduchého přístupu k demoverzi.

Při prvním zkoumání ASM (viz 3.1) je okamžitě patrná rozsáhlá funkčnost programu. Části funkcí, které jsou v ASM k dispozici, ale v našem systému chybí, se budeme věnovat v této části (viz 7.3).

Nejdříve se zamysleme nad výhodami systému ASM (viz 3.1.1). Jednou z výhod je rozsáhlejší systém pro práci s různými dokumenty. Zatímco v našem systému je dokument jednoduše přiřazen pracovníkovi a zvířeti, v systému ASM je rozdělen do čtyř samostatných kategorií: léčba, finance, publikace a zprávy.

Další výhodou je dostupnost velkého počtu jazyků, což zvyšuje dostupnost v místech, kde neumí anglicky. Je také interakce s Google Maps, což umožňuje rychle zobrazit místo nálezu zvířete a podniknout další kroky.

Pokud se zamyslíme nad výhodami našeho systému, zjistíme, že již v prototypu existuje malá funkcionalita pro správu samotného zařízení (správa inventáře). Také přístup je zcela zdarma.

7.2 Porovnání s PetPoint a Petstablished

PetPoint (viz 3.3) nijak nevybočuje z řady Petstablished a ASM, všechny jeho výhody a nevýhody již byly popsány v předchozím bodě (viz 7.1) nebo budou popsány v bodě 7.3. Je třeba také poznamenat, že vzhledem k velmi nízké

funkčnosti našeho prototypu nemá komplexní rozhraní a omezuje se na jednoduchou sadu tlačítek (viz 5.6.1). Což se zase nedá říci o PetPointu.

Petstablished (viz 3.2) má zase výukové programy, ale v případě nízké aktivity útulku vyžaduje platbu, která v našem systému zcela chybí. Její nevýhodou je také chybějící kontrola zvířat a funkcionalita pro správu útulku.

7.3 Další rozvoj

Provedeme shrnutí bodů 7.1 a 7.2. Funkcionalita, kterou vhodně přidat:

- **Interakce s Google Maps,** U každého zvířete doplňte informace o místě jeho nálezu. Odtud by mělo být možné na první pohled zobrazit polohu na mapách.
- **Rozšíření možností dokumentace,** Pro finance a další dokumenty, které se netýkají zvířat, vytvoříme samostatné stránky. Rovněž je vhodné vylepšit stávající funkce a přidat možnost upravovat dokumenty v rámci systému.
- **Přidání nových jazyků,** Přidání samostatné možnosti pro výběr jazyka. Ještě předtím proveďte analýzu měst/zemí, kde se má náš informační systém používat. Poté jej lokalizujeme do požadovaných jazyků.
- **Vytvoření výukových programů.** Vytvoříme samostatný koncový bod pro výukové programy. Umístěme materiály ve formátu textu nebo videa a přidat k nim přístup všem zaměstnancům.

Dalším krokem bude prověření nerealizovaných FR. Vrátíme se k bodu 4.1.1 a pro každý nerealizovaný FR navrhneme způsob, jak jeho začlenit do vyvíjeného systému. Všechny realizované FR mají prioritu must a should, zbývá zvážit FR s prioritou could a would.

FR12: Realizujeme řídicí panel neboli dashboard (příklad na obrázku 3.2), kde si správci mohou zobrazit základní ukazatele o provozu zařízení, péče o zvířata, jako je počet přijatých a odcházejících zvířat, náklady na péči, počet zaměstnanců a dobrovolníků a další statistické údaje.

FR13: Realizujeme modul pro vytváření a odesílání zpráv, např. pro upozornění na události v útulku. Takové zprávy mohou být vázány na čas, akci nebo nějaký signál. Vypadalo by to jako vyskakovací okno na webové stránce. Správci by měli mít možnost tyto zprávy přizpůsobit.

FR14: Vytvoříme sekci pro evidenci a sledování výživy zvířat, kam budou moci zaměstnanci zadávat informace o stravě, množství a době krmení, ale i o speciálních dietách či požadavcích.

FR15: Vytvoříme funkcionalitu pro evidenci výdajů kachny, včetně nákupů krmiv, léčivých přípravků, přípravků na ošetřování zvířat a dalších materiálových nákladů.

FR16: Zavedeme systém doporučení na základě údajů o činnosti útulku, např. o potřebě vytvořit další karanténní prostory při příchodu nových zvířat nebo o každodenním připomínání krmení zvířat.

FR17: Přidáme sekci s informacemi o tom, jak pečovat o různé druhy zvířat, tipy na výchovu a výcvik, informace o nemocech a další užitečné materiály pro budoucí majitele.

FR18: Do databáze přidáme další tabulku a spojíme ji s tabulkou zvířat. V nové tabulce bude seznam fotografií a zvířete, ke kterému patří.

FR19: Přidáme funkci pro export dokumentů do různých formátů, jako je PDF, CSV nebo Excel, aby zaměstnanci mohli snadno sdílet informace nebo vytvářet přehledy.

FR20: Vytvoříme soubor formulářů pro různé operace, jako je přijímání zvířat, vyřizování adopcí, registrace dobrovolníků atd. Všechny formuláře budou uloženy na jednom místě a budou mít odkaz ke stažení nebo tisk.

FR21: Pro každou předepsanou akci uložíme log do samostatného souboru. Na základě logů budeme realizovat obecné statistiky. Například o počtu uprchlých nebo nemocných zvířat.

FR22: Realizujeme funkci pro zadávání a sledování dočasných úkolů a upomínek pro zaměstnance, např. pro lékařské prohlídky, ošetření zvířat proti parazitům a další denní úkoly.

FR23: Zavedeme systém účtů pro útulky, podobný tomu, který jsme zavedli pro zaměstnance. Majitel útulku bude muset svůj útulek zaregistrovat a poté bude mít personál přístup k autorizaci.

FR24: Na základě zpráv z **FR13** bude moci správce vytvořit seznam zpráv a frekvenci jejich zasílání. Takové zprávy se budou automaticky zobrazovat, pokud nebyla provedena akce. Například pokud zaměstnanec nezadal informace o úklidu.

FR25: Implementujeme funkce pro sledování plnění úkolů a cílů stanovených pro zaměstnance, na základě těchto informací také závisí oznámení z **FR24**.

Závěr

První kapitola teoretické části práce byla věnována rozboru problematiky útulků a záchranných stanic pro zvířata. Za tímto účelem jsme analyzovali zákony České republiky upravující pravidla péče o zvířata a jejich chov a také pokyny Státní veterinární správy. V poslední řadě byly zkoumány požadavky na ustájení zvířat na příkladu koček a psů.

Cílem další kapitoly bylo prozkoumat existující řešení. Za tímto účelem byly vzaty tři systémy: Animal Shelter Manager, Petstablished a PetPoint. U každého z nich byly analyzovány výhody a nevýhody.

Dále byl vytvořen katalog požadavků a případů užití. Funkční požadavky byly rozděleny do 4 částí, aby bylo možné stanovit jejich priority podle metody MoSCoW. Nefunkční požadavky se týkaly zabezpečení, dostupnosti a výběru implementačních nástrojů, jako jsou Java, Spring Boot a JWT. Funkční požadavky, které pokrývají, byly přiřazeny ke katalogu případů užití, byl vytvořen diagram s identifikovanými aktéry a pro některé případy užití byly vytvořeny scénáře.

Od páté kapitoly se práce věnovala praktické části. V samotné kapitole byl popsán vlastní návrh na vytvoření systému. Byly popsány důvody, proč byla jako programovací jazyk zvolena Java a framework Spring Boot. Rovněž bylo navrženo použití architektonického stylu MVC. Pomocí schémat a diagramů byla navržena struktura projektu. Posledním bodem byl popis metodiky REST, podle které byla aplikace vytvořena.

Šestá kapitola popisuje různé detaily z implementace a upozorňuje na nejzajímavější řešení nebo problémy, které se vyskytly. Velká pozornost byla věnována dokumentaci a testování.

V závěrečné části a zároveň v poslední kapitole byl náš informační systém vyhodnocen. Byly také zváženy možnosti dalšího vývoje, spočívající v nízkoprioritních funkčních požadavcích popsaných v kapitole 4.

Shrneme-li práci na tvorbě informačního systému, můžeme konstatovat, že všech stanovených cílů bylo dosaženo. Vytvořený systém splňuje požadavky definované v zadání a plně realizuje funkcionalitu požadovanou prototypem.

Literatura

[1] Česká republika. Zákon č. 246/1992 Sb. Zákon České národní rady na ochranu zvířat proti týrání. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: `https://www.zakonyprolidi.cz/cs/1992-246`.

[2] Česká republika. Zákon č. 246/1992 Sb, § 3 písm. h. Zákon České národní rady na ochranu zvířat proti týrání. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: `https://www.zakonyprolidi.cz/cs/1992-246`.

[3] Česká republika. Zákon č. 246/1992 Sb, § 3 písm. i. Zákon České národní rady na ochranu zvířat proti týrání. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: `https://www.zakonyprolidi.cz/cs/1992-246`.

[4] Česká republika. Zákon č. 246/1992 Sb, § 3 písm. v. Zákon České národní rady na ochranu zvířat proti týrání. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: `https://www.zakonyprolidi.cz/cs/1992-246`.

[5] Česká republika. Zákon č. 246/1992 Sb, § 1. Zákon České národní rady na ochranu zvířat proti týrání. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: `https://www.zakonyprolidi.cz/cs/1992-246`.

[6] Česká republika. Zákon č. 89/2012 Sb. Zákon občanský zákoník. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2012-89`.

[7] Česká republika. Zákon č. 89/2012 Sb, § 494. Zákon občanský zákoník. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2012-89`.

[8] Česká republika. Zákon č. 89/2012 Sb, § 1047. Zákon občanský zákoník. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2012-89`.

[9] Česká republika. Zákon č. 89/2012 Sb, § 1058. Zákon občanský zákoník. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: https://www.zakonyprolidi.cz/cs/2012-89.

[10] Česká republika. Zákon č. 89/2012 Sb, § 1059. Zákon občanský zákoník. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: https://www.zakonyprolidi.cz/cs/2012-89.

[11] Česká republika. Zákon č. 166/1999 Sb. Zákon o veterinární péči a o změně některých souvisejících zákonů (veterinární zákon). In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: https://www.zakonyprolidi.cz/cs/1999-166.

[12] Česká republika. Zákon č. 114/1992 Sb. Zákon České národní rady o ochraně přírody a krajiny. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 04.03.2024]. Dostupné z: https://www.zakonyprolidi.cz/cs/1992-114.

[13] Česká republika. Zákon č. 21/2013 Sb. Vyhláška o stanovení podmínek při chovu psů a koček. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 12.08.2021]. Dostupné z: https://www.zakonyprolidi.cz/cs/2013-21.

[14] Česká republika. Zákon č. 419/2012 Sb. Vyhláška o ochraně pokusných zvířat. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 12.05.2024]. Dostupné z: https://www.zakonyprolidi.cz/cs/2012-419.

[15] Česká republika. Zákon č. 419/2012 Sb, Příloha č. 7, Tabulka 3 Kočky. Vyhláška o ochraně pokusných zvířat. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 12.05.2024]. Dostupné z: https://www.zakonyprolidi.cz/cs/2012-419.

[16] Česká republika. Zákon č. 419/2012 Sb, Příloha č. 7, Tabulka 4.1 Psi. Vyhláška o ochraně pokusných zvířat. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 12.05.2024]. Dostupné z: https://www.zakonyprolidi.cz/cs/2012-419.

[17] Česká republika. Zákon č. 419/2012 Sb, Příloha č. 7, Tabulka 4.2 Psi – odstavená štěňata. Vyhláška o ochraně pokusných zvířat. In: *Zákony pro lidi* [Online]. AION CS, 2010–2024 [cit. 12.05.2024]. Dostupné z: https://www.zakonyprolidi.cz/cs/2012-419.

[18] Česká republika. Veterinární podmínky pro zřizování, provoz a kontrolu útulků a obdobných podnikatelských zařízení. In: *SVS ČR č.2000/05/EPIZ ze dne 28.2.2000 ve znění novely ze dne 1.8.2010.*

[19] ISO/IEC JTC 1 (Joint Technical Committee 1) / SC 32 (Subcommittee 32) /. *Structured Query Language* [software]. Stabilní verze: 2023. [přístup 17. 04. 2024]. Dostupné z: https://www.iso.org/standard/76583.html.

[20] *Sheltermanager.com* [Online]. Sheltermanager Ltd. (Last update: 12.03.2024). [cit. 24.03.2024]. Dostupné z: https://sheltermanager.com.

[21] *Petstablished* [Online]. PerfectPetMatch.com LLC. (Last update: 12.01.2023). [cit. 24.03.2024]. Dostupné z: https://petstablished.com.

[22] *Petpoint* [Online]. Pethealth Inc. [cit. 24.03.2024]. Dostupné z: https://www.24petcare.com/solutions/petpoint.

[23] Kulak, D., & Guiney, E. *Use Cases: Requirements in Context*. 2. vydání. Addison-Wesley Professional, 2003. 272 stránky. ISBN-13: 978-0321154989.

[24] Miranda, E.: Time boxing planning: buffered moscow rules. *ACM SIGSOFT Software Engineering Notes*, ročník 36, 11 2011: s. 1–5, doi: 10.1145/2047414.2047428.39.

[25] Oracle Corporation. *Java* [software]. Stabilní verze: 19.03.2024. [přístup 22. 04. 2024]. Dostupné z: https://www.java.com.

[26] VMware. *Spring Boot* [software]. Stabilní verze: 21.03.2024. [přístup 22. 04. 2024]. Dostupné z: https://spring.io/.

[27] Internet Engineering Task Force. *JSON Web Token* [software]. Stabilní verze: 18.05.2018. [přístup 22. 04. 2024]. Dostupné z: https://jwt.io/.

[28] Reenskaug, T. *Model–view–controller* [software]. 1970. [přístup 22. 04. 2024].

[29] CERN, IETF, W3C. *Hypertext Transfer Protocol* [software]. Aktualizace 2022. [přístup 22. 04. 2024]. Dostupné z: https://httpwg.org/specs.

[30] W3C. *HyperText Markup Language* [software]. Aktualizace 2017. [přístup 22. 04. 2024]. Dostupné z: https://html.spec.whatwg.org/.

[31] Eich, B. *JavaScript* [software]. Stabilní verze: Červen 2023. [přístup 22. 04. 2024]. Dostupné z: https://www.javascript.com/.

[32] Walke, J. *React* [software]. Stabilní verze: 14.06.2022. [přístup 22. 04. 2024]. Dostupné z: https://react.dev/.

[33] Pang, A. *Rest: Why You Get More Done When You Work Less*. Basic Books, 2016. 320 stránky. ISBN-13: 978-0465074877.

[34] Berners-Lee, T. *Uniform resource locator* [software]. Aktualizace 2023. [přístup 22. 04. 2024]. Dostupné z: https://url.spec.whatwg.org/.

[35] Google, Inc., Yahoo, Inc., Microsoft Corporation and Yandex. *Schema.org* [software]. Stabilní verze: 12.02.2024. [přístup 22. 04. 2024]. Dostupné z: https://schema.org/.

[36] SmartBear Software. *Swagger* [software]. Stabilní verze: 09.05.2024. [přístup 10. 05. 2024]. Dostupné z: `https://swagger.io/`.

[37] Faber, S., Dutheil, B. *Mockito* [software]. Stabilní verze: 18.06.2023. [přístup 10. 05. 2024]. Dostupné z: `https://site.mockito.org/`.

[38] VMware. *MockMvc* [software]. Aktualizace 2022 . [přístup 10. 05. 2024]. Dostupné z: `https://docs.spring.io/spring-framework/reference/testing/spring-mvc-test-framework.html`.

[39] Beck, K., Gamma, E. *JUnit5* [software]. Stabilní verze: 23.03.2023 . [přístup 09. 04. 2024]. Dostupné z: `https://junit.org/junit5/`.

Příloha **A**

Seznam použitých zkratek

API – Application programming interface

ASM – Animal shelter manager

CRUD – Create read update delete

DTO – Data transfer object

FR – Functional requirement

Hi-fi High fidelity

HTTP – Hypertext transfer protocol

HTML – Hypertext markup language

JVM – Java virtual machine

JWT – Json web token

MoSCoW – Must should could would

MVC – Model view controller

NFR – Non-functional requirement

REST – Representational state transfer

SQL – Structured query language

UC – Use case

UI – User interface

UML – Unified modeling language

URL – Uniform resource locator

XML – Extensible markup language

Příloha **B**

Obsah příloh

- `readme.md` ..... Informace o archivu
- `shelter` ..... Serverový projekt v IntelliJ IDEA
  - `readme.md` ..... Informace o serveru
  - `src`
    - `main` ..... Zdrojové kódy implementace
    - `test` ..... Testy
  - `docker-compose.yml` ..... Soubor s nastavením nástroje Docker
  - `build.gradle` ..... Soubor s nastavením projektu
  - `[...]` ..... Soubory s nastaveními projektu potřebnými pro provoz
- `client` ..... Projekt klienta
  - `readme.md` ..... Informace o klientu
  - `public` ..... Loga a soubory pro React
  - `src` ..... Zdrojové kódy implementace
  - `package.json` ..... Soubor s nastavením knihoven a závislostí
  - `package-lock.json` ..... Soubor s nastavením knihoven a závislostí
- `API Documentation.json` ..... Offline verze dokumentace
- `BP-ovchiant.zip` ..... Zdrojová forma práce ve formátu LaTeX