FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Návrh aplikace pro evidenci psů
Student:	Adam Štursa
Vedoucí:	Ing. Michal Rost
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem této práce je analýza, návrh a implementace server-klient aplikace pro evidenci psů, jejich plemen a rodokmenů. Serverová část aplikace bude přístupná v podobě RESTového API a její prototyp impementován
pomocí Java frameworku Spring. Protoyp webového klienta bude toto API využívat.

Jednotlivé kroky postupu:
1. Analýza funkčních a nefunkčních požadavků
2. Analýza stavajících řešení
3. Návrh řešení na základě předchozí analýzy
4. Implementace prototypu aplikace
5. Dokumentace a vhodné otestovaní iplementace

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 28. ledna 2024 v Praze.

Bakalářská práce

NÁVRH WEBOVÉ APLIKACE PRO EVIDENCI PSŮ

Adam Štursa

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Michal Rost
16. května 2024

Odkaz na tuto práci: Štursa Adam. *Návrh webové aplikace pro evidenci psů.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především svému vedoucímu Ing. Michalu Rostovi za odborné vedení a rady, které mi v průběhu zpracování této práce věnoval. Dále bych rád poděkoval své rodině a blízkému okolí ze jejich podporu a trpělivost vůči mé osobě, bez které by tato práce nevznikla.

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. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na jejímž základě se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 16. května 2024

Abstrakt

Tato bakalářská práce je zaměřena na vytvoření aplikace typu server-klient, která se věnuje evidenci psů, jejich rodokmenů, nemocí a dalších klíčových částí spojenými se psy. Serverová část aplikace je přístupná prostřednictvím rozhraní REST API implementovaného pomocí frameworku Java Spring, přičemž prototyp webového klienta toto rozhraní využívá. Mezi klíčové součásti práce patří analýza funkčních a nefunkčních požadavků, průzkum existujících řešení a návrh vycházející z předchozích analýz. Práce se dále věnuje implementaci prototypu aplikace a jejímu testování. Cílem práce je poskytnout majitelům psů komplexní a uživatelsky přívětivou platformu pro správu a prezentaci jejich domácích mazlíčků a vyřešit tak nedostatky v současných existujících řešeních.

Klíčová slova webová aplikace, evidence psů, REST API, Java, Spring Boot, Symfony, PostgreSQL

Abstract

This bachelor thesis is focused on the creation of a server-client application dedicated to the registration of dogs, their pedigrees, diseases and other key parts related to dogs. The server side of the aplication is accessed through a REST API implemented using the Java Spring framework, and the web client prototype uses this interface. The key components of the work include analysis of functional and non-functional requirements, exploration of existing solutions and software design based on these analyses. The thesis also discusses the implementation of the prototype application and its testing. The aim of the work is to provide dog owners with a comprehensive and user-friendly platform for managing and presenting their pets, addressing the shortcomings of currently existing solutions.

Keywords web application, dog registry, REST API, Java, Spring Boot, Symfony, PostgreSQL

Seznam zkratek

API	Application Programming Interface
CSS	Cascading Style Sheet
DTO	Data Transfer Object
FCI	Fédération Cynologique Internationale
HTML	Hypertext Markaup Language
HTTP(S)	Hypertext Transfer Protocol (Secure)
JS	JavaScript
NoSQL	Not Only SQL
ORM	Object Relational Mapping
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
URI	Uniform Resource Identifier

Kapitola 1
Úvod

Pro mnoho páníčků není pes pouze jejich dobrý kamarád a věrný společník, ale zároveň i chlouba. A proč by přece být nemohl? Za takové štěně s rodokmenem se dnes platí zlatem a následně majitel pejska tráví dlouhé měsíce až roky jeho výchovou. Za tuhle těžkou dřinu je následně páníčkovi odměnou právě ta chvíle, kdy se jeho mazlíček stane úspěšným v nějakém závodě, či na výstavě. Za tento úspěch následně dostane majitel papírový certifikát, který si může akorát tak zarámovat, případně vyfotit na vlastní sociální síť. Zkrátka v současné době neexistuje žádný snadný způsob, kterým by se mohl páníček rodokmenem a úspěchy svého pejska chlubit. Zároveň nemá jednoduchý způsob, jak zjistit úspěchy psích kamarádů svého mazlíčka, nebo mu najít kamarády nové.

Snad největším strachem každého majitele psa je, že se mu jeho milovaný mazlíček ztratí. Ať už majiteli pes přeskočí plot, nebo se během procházky vyvlíkne z vodítka a uteče přímo před očima jeho páníčka, k jeho následnému navrácení slouží, dnes již v České republice povinný, čip a jemu přiřazené unikátní číslo. Po získání tohoto čísla naskenováním čipu, lze pejska nahlásit jako nalezeného na policii. Zároveň pokud majitel zaregistroval svého mazlíčka do registru psů, je možné nález nahlásit i tomuto registru. Ten pak kontaktuje přímo majitele s informacemi o nálezu jeho mazlíčka.

Z vlastní zkušenosti ovšem vím, že to s takovou registrací psa není úplně jednoduché. Když jsem se snažil Samuela – našeho rodinného psa přihlásit do registru psů, byl jsem, po zadání klíčových slov „registr psů“ do internetového vyhledávače, zavalen nepřeberným množstvím webů poskytujících službu registrace psa. Zatímco některé poskytovaly omezené funkce, jiné měly nesmyslné poplatky, nebo byly „sešité horkou jehlou“. S žádným z těchto webů jsem tedy nebyl příliš spokojený a neměl jsem tedy tušení, kde Samuela zaregistrovat.

Ovšem i poklidný večer, kdy se pes nikde netoulá a majitel ho má pod dozorem, se může stát velmi rychle hektickým. Stačí totiž, aby si páníček během drbání svého mazlíčka všiml neznámé vyrážky a v tu chvíli nastává panika. Jak to tak bývá, po krátkém hledání na internetu páníček usoudí, že musí se svým pejskem okamžitě k veterináři. Ovšem ten, ke kterému běžně chodí, už má po ordinačních hodinách, a tak nastává druhé hledání na internetu.

Všechny tyto zmíněné problémy mají společné dvě věci: lásku psích majitelů k jejich mazlíčkům a absenci jednotné a přehledné aplikace, která by jim tyto jejich trápení pomohla vyřešit. Pro některé z těchto problémů sice již samostatná řešení existují, ale který páníček by neocenil, kdyby měl vše přehledně na jednom místě? Proto, když za mnou přišli z firmy Triton IT s nápadem na aplikaci, která by tyto problémy řešila, neváhal jsem a chtěl jsem být součástí jejího vývoje.

Kapitola 2
Cíle

Cílem teoretické části práce je sběr funkčních a nefunkčních požadavků od zadavatele a následná analýza existující řešení a jejich nabízené funkcionality. Na základě této analýzy následně vybrat a popsat vhodné technologie pro vypracování praktické části práce.

Cílem praktické části práce je navrhnout, implementovat a vhodně otestovat prototyp serverové a klientské části webové aplikace s požadovanými funkcionalitami.

Kapitola 3
Analýza současných řešení

Cílem této kapitoly je představení podobných, již existujících, webových aplikací, které se svými funkcemi protínají s tématem této bakalářské práce. U jednotlivých aplikací popíši jejich nabízené funkce a své subjektivní dojmy z používání těchto aplikací.

3.1 Pedigree Online

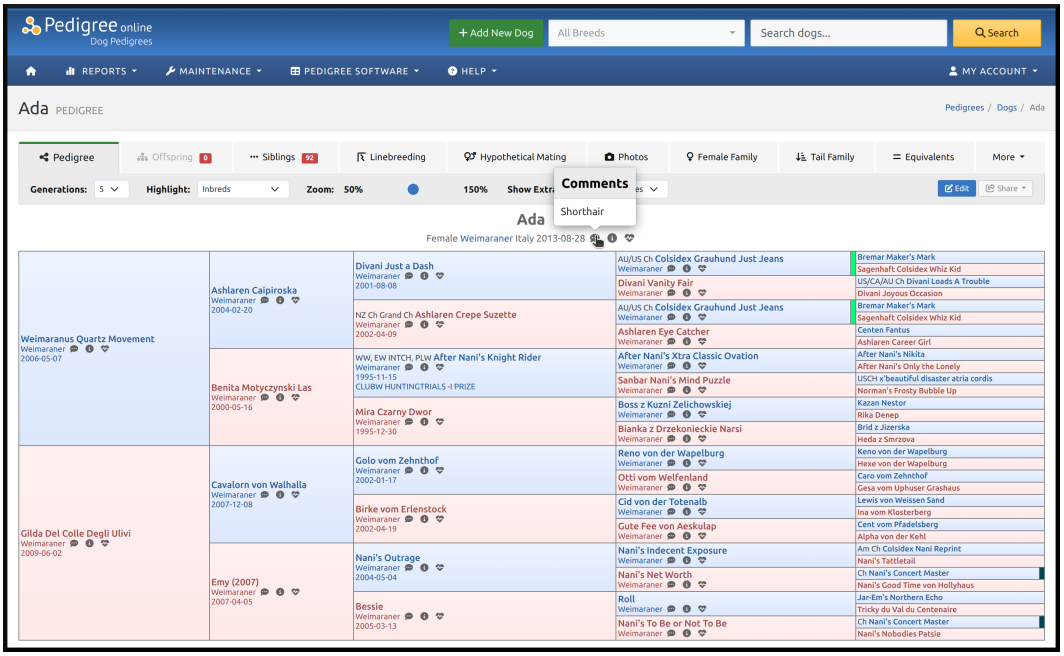
Pedigree Online je největší celosvětovou databází psích rodokmenů, která poskytuje vytváření rodokmenu zdarma a je kompletně otevřená veřejnosti. [1] Tato aplikace nabízí svému uživateli především tyto tři funkcionality:

Vytváření a editace psa: Umožňuje uživateli přidat jakéhokoli psa do této veřejné databáze a následně provádět jeho úpravy. K těmto oběma úkonům slouží jeden a ten samý formulář obsahující informace o psovi jako je jméno, rasa, datum narození, informace o rodičích, či informace o chovné stanici, ze které pes pochází. Dále má uživatel možnost nahrát k psovi fotografie a vyplnit různé zdravotní informace. Zarážející je potom ten fakt, že jakýkoli přihlášený uživatel má právo změnit informace u jakéhokoli psa a to bez žádné další kontroly. Jediný způsob kontroly dat u psa je tak volně dostupná historie změn.

Zobrazení informací o psovi: Nabízí uživateli hlavně náhled na vyplněné informace o psovi a jeho rodokmen ve formě tabulky, skrze kterou se může dostat k informacím o předcích daného psa. Dále má uživatel možnost si zobrazit fotografie, seznam potomků, seznam sourozenců a vyfiltrované přehledy o rodině psa (např. zobrazení pouze mužských/dámských předků).

Vyhledávání psa: Dává uživateli možnost vyhledat psa pouze podle jeho jména, rasy a pohlaví. Aplikace nenabízí žádné pokročilejší vyhledávání (například podle registračního kódu). Výsledky vyhledávání jsou zobrazeny v tabulce s těmito údaji: jméno, datum narození, pohlaví, rasa, registrační číslo, otec a matka psa. Tuto tabulku lze podle těchto údajů seřadit.

Celkově aplikace působí velice nepřívětivě – není pro uživatele intuitivní, místy není responzivní a design působí zastarale. Tím, že aplikace není intuitivní, mám namysli například to, že veškeré informace o psovi na jeho detailu jsou schované za nenápadnými ikonami a zobrazí se pouze tehdy, když uživatel najede kurzorem myši na tuto ikonu. Dále za neintuitivní považuji třeba již zmíněnou omezenou funkce vyhledávání a celkově složitou navigaci skrz aplikaci. To, že data mohou být upravovány i nepřihlášenými uživateli, pak způsobuje ztrátu na důvěryhodnosti informací obsažených na této stránce. Důvodem pro představení této aplikace je fakt, že vytváření rodokmenu psa bude důležitou součástí aplikace, která je předmětem této bakalářské práce.



Obrázek 3.1 Ukázka detailu psa v aplikaci Pedigree Online

3.2 Paper-Dog

Hlavním cílem portálu Paper-Dog je usnadnit majitelům chov psů s průkazem původu. Tohoto cíle chce portál dosáhnout pomocí usnadnění přístupu k informacím týkajících se všech oblastí chovu psů, čímž zvýší vzdělanost majitelů a chovatelů psů, a zároveň jim pomohou vyřešit potřeby nutné pro správný chov psů. [2] Tato aplikace nabízí svým uživatelům tyto funkcionality:

Přehled psích služeb: Dává uživateli možnost zobrazení seznamu služeb podle jejich kategorií (veterinární stanice, chovatelské kluby, salóny pro psy, výcviková centra, atp.) v podobě tabulky. Seznam je vždy filtrovatelný podle kraje ve kterém se daná instituce nachází. Dále, pokud může mít daný typ služby zaměření pouze na určitá plemena, je možné tyto služby filtrovat právě podle plemene (například chovatelské kluby, výcviková centra). Kromě seznamu těchto služeb by se v aplikaci měla nacházet mapa, ve které jsou dané služby zanesené. Ke dni 15.04.2024 ovšem tato mapa není funkční a místo ní je tak uživateli zobrazená pouze chybová hláška.

Atlas psích plemen: Je jednoduchý filtrovatelný seznam existujících psích plemen. Uživatel v tomto seznamu může filtrovat podle názvu, výšky a hmotnosti plemene. V detailu plemene uživatel najde kromě základních textových informací o plemeni také vazby na konkrétní instituce, inzerce a profily psů, které jsou s tímto plemenem spojené.

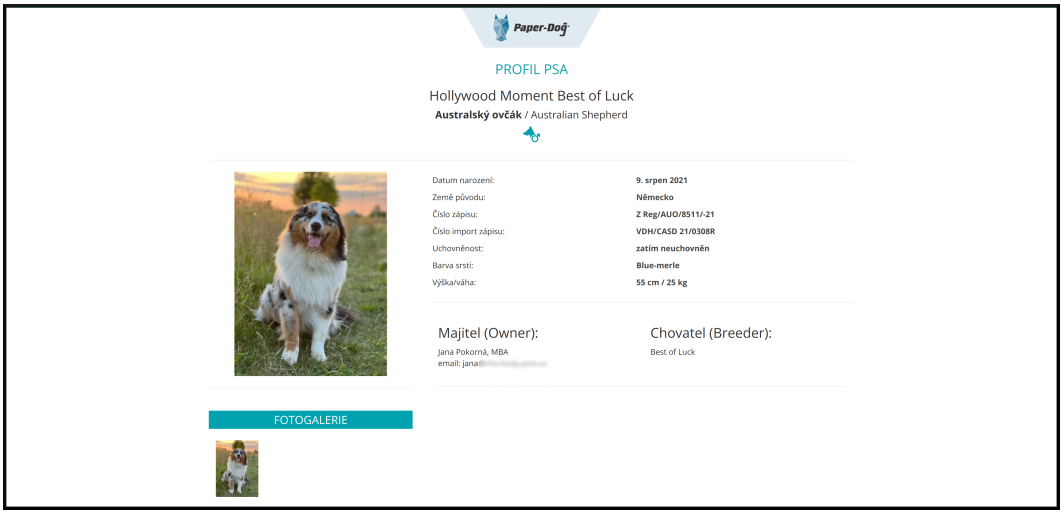
Online poradna: Nabízí přihlášeným uživatelům možnost položit dotaz, či na dotaz jiného uživatele odpovědět. Seznam dotazů lze filtrovat dle plemene a kategorie dotazu, mezi které patří například „Výživa“, „Výchova a výcvik“, nebo „Veterinární poradna“. Seznam dotazů je zobrazený jako tabulka, ve které je název, plemeno, datum vytvoření a počet odpovědí. Dle očekávání na detailu dotazu uživatel najde jeho text, seznam odpovědí a tlačítko pro přidání odpovědi.

Inzerce: Umožňuje uživateli vytvářet a prohlížet inzeráty. Vytvoření inzerátu je prováděno vyplněním formuláře, do kterého uživatel zadá údaje o kategorii, nadpisu, kraji a popisu. Nepovinně pak uživatel může vyplnit i cenu, fotografii či plemeno psa. Seznam inzerátů je filtrovatelný podle jeho lokality, stáří, kategorie a plemena.

Profil psa: Je částí aplikace ve které může uživatel v aplikaci evidovat své psy a prohlížet profily psů jiných uživatelů. Aplikace je zaměřená na psy s průkazem původu, a proto mezi povinné údaje patří i registrační číslo psa, chovatelská stanice ze které pochází a dvě předchozí generace. Tato omezení nejen zabraňují evidování psů bez průkazu původu, ale celkově znepříjemňují proces vytváření psa (co když si uživatel například tyto údaje jednoduše nepřeje vyplňovat). Veškeré vyplněné údaje o psovi jsou volně dostupné přihlášeným i nepřihlášeným uživatelům v profilu psa. Profily všech psů jsou v aplikaci uživatelům dostupné formou seznamu, který lze filtrovat podle lokality, plemene, barvy, pohlaví, atd.

Profil uživatele: Dává uživateli, kromě možnosti editace údajů jeho profilu, přehled jím vytvořených psů, inzerátů a příspěvků v online poradně. Uživatel tak získává snadný přístup k zobrazení a úpravě veškerého jím přidaného obsahu.

První dojmy z portálu pro mne, jako uživatele, byly příjemné, aplikace má vcelku moderní design a na první pohled se zdá být přehledná. Avšak čím déle jsem aplikaci používal, tím méně jsem byl s aplikací spokojený. Prvním zaražením pro mne nastalo hned při registraci do aplikace – namísto očekávaného vytvoření hesla pomocí formuláře, mi bylo heslo vygenerováno a zasláno emailem. Tato praktika je považována za bezpečnostní riziko. [3] Kromě tohoto je aplikace místy neresponzivní a uživatelsky nepřívětivá (schované nepovinné pole u formuláře, nepřehledné menu aplikace, neočekávané chování tlačítek, atp.). V neposlední řadě není uživatelský obsah v aplikaci nikým kontrolován, a tak se mezi inzeráty, příspěvky v online poradně, ale i profily psů nachází spousta spamového[1] obsahu. V důsledků těchto nedostatků je tak výsledný dojem z aplikace při delším používání spíše nepříjemný.



Obrázek 3.2 Ukázka detailu psa v aplikaci Paper-Dog

[1] Spam – nevyžádaný obsah v digitální komunikaci, typicky se jedná o reklamní sdělení [4]

3.3 PesWeb

PesWeb je portál zaměřený především na pomoc psům v útulku skrze přinášení aktuálních informací, novinek a zajímavostí z tohoto tématu. Dále se snaží o centralizaci nabídek útulkových psů, ale i o to být portálem pro všechny majitele a milovníky psů. [5] Webová aplikace svým uživatelům zprostředkovává tyto funkcionality:

Seznam psů a koček k adopci: Portál nabízí svým uživatelům možnost zobrazit si detailní výpisy psů a koček, které čekají na adopci. Tyto výpisy obvykle zahrnují fotografie, popis a specifické informace o každém zvířeti, včetně věku, plemene a velikosti. Tyto seznamy jsou filtrovatelné jak podle údajů zvířete, tak i podle lokality útulku.

Seznam ztracených a nalezených psů: V aplikaci se nachází sekce, ve které uživatelé mohou nahlásit ztrátu nebo nález psa. Zároveň má uživatel možnost si zobrazit seznam těchto zvířat a ten filtrovat dle jejich údajů a lokality. Na detailu těchto inzerátů najde uživatel kromě detailních informací o ztrátě/nálezu také možnost kontaktovat uživatele, který inzerát vytvořil.

Atlas psů: Tato část aplikace nabízí identickou funkcionalitu jakou je „Atlas psích plemen“ v portálu Paper-Dog (3.2).

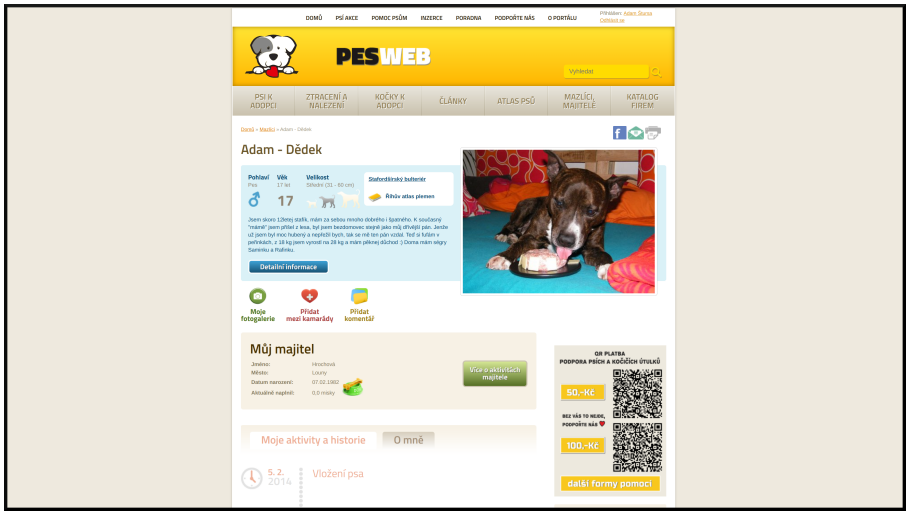
Inzerce: Sekce inzercí umožňuje uživatelům vytvářet a procházet inzeráty týkající se psů a psích potřeb, ale slouží i nabízení psích služeb. Seznam inzerátů je filtrovatelný podle kategorie, lokality, plemene a dalších specifikací. Uživatelé mohou inzeráty vytvářet pomocí formuláře, ve kterém zadají kategorii, nadpis, lokalitu a detailní popis a nepovinně i fotografie.

Online poradna Tato sekce umožňuje uživatelům pokládat otázky, ale nenabízí jim možnost na ně odpovídat. Pro odpovědi na uživatelské dotazy mají na webu dedikované odborníky, kteří by na ně měli odpovídat. Ti se ale vzhledem k velkému množství, i několik měsíců starých, nezodpovězených dotazů, této sekci aktivně nevěnují. Díky tomu tato sekce postrádá veškerý smysl.

Kalendář akcí Je část aplikace, která poskytuje uživatelům přehled nadcházejících událostí, soutěží, výstav a dalších aktivit spojených se psy. Uživatelé mají dny s událostmi vyznačené v malém kalendáři a mohou akci filtrovat podle typu události nebo lokality.

Uživatelský profil Nabízí především možnost spravovat své osobní údaje a informace o vlastněných psech. Dále se v této sekci nachází přehled historie aktivit uživatele a tlačítka pro přidávání uživatelského obsahu. Celá sekce působí mírně nepřehledným dojmem.

Můj celkový dojem z aplikace je opět spíše negativní. Až na výjimky, kterými jsou například formuláře, nepůsobí aplikace svým vzhledem zastarale a chová se responzivně. I přes to, že je aplikace responzivní, jsou některé prvky na mobilních zařízeních příliš malé – nečitelné, či náročné na interakci. Na každé stránce se nachází postranní panel obsahující různé obrázky a odkazy, který působí na uživatele rušivým efektem a odvádí uživatelovu pozornost od hlavního obsahu stránky. Tento fakt spolu s přítomností dvou různých hlavních menu a dalšími drobnostmi dohromady způsobují mírnou uživatelskou nepřívětivost aplikace. Za pozitivní považuji velké množství funkcionalit aplikace, jako je například kalendář akcí a jejich technické zpracování.

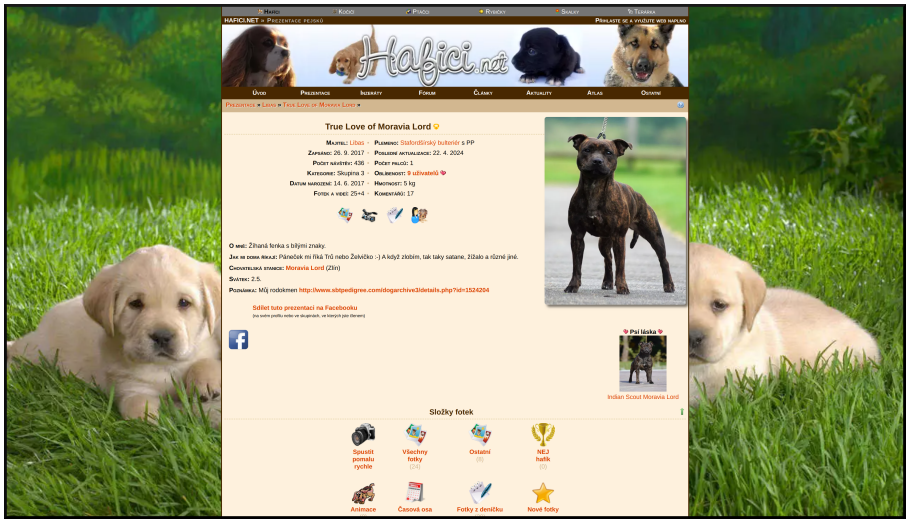


Obrázek 3.3 Ukázka detailu psa v aplikaci PesWeb

3.4 Hafici.net

Posledním českým webem, který se mi podařilo najít, s tématikou podobnou této bakalářské práci, je aplikace Hafici.net. Jedinou zajímavou funkcionalitou navíc, o proti předchozím aplikacím, je možnost uspořádávat přidané fotky psů do alb. Jinak aplikace nabízí v podstatě identické funkcionality, jako předchozí zmiňované aplikace, a to tedy: inzerce, vytváření psích profilů, online fórum (s možností uživatelských odpovědí), atlas psů a přehled institucí. Vzhledem k podobnosti těchto funkcionalit s předchozími aplikacemi, je již dále podrobně popisovat nebudu.

Tato webová aplikace svým vzhledem za těmi předchozími velmi zaostává, a nepůsobí tak na mne, jakožto uživatele, dobrým dojmem již od prvního vstupu do aplikace. Díky spamovému obsahu v sekci inzerátů a zastaralému redakčnímu obsahu, aplikace působí dojmem, že již delší dobu není udržovaná. Naopak z hlediska uživatelské přívětivosti a responzivity tento web nad těmi předchozími vyhrává. Zde bych vytkl pouze články, které nelze uspořádat podle data vydání a absenci odkazu na instituce v hlavním menu aplikace.



Obrázek 3.4 Ukázka detailu psa v aplikaci Hafici.net

3.5 Shrnutí

I přesto, že jsou aplikace PesWeb a Paper-Dog svými funkcionalitami velmi podobné, tyto dvě současná řešení rozdělují majitele psů na dva tábory – ty se psy s průkazem původu a ty s mazlíčky z útulků. Zatímco v jedné aplikaci nelze rodokmen psa vytvářet, v druhé profil svému psovi bez existence rodokmenu vůbec profil nevytvoříte a i přesto jsou možnosti správy rodokmenu v této aplikaci omezené. Třetí aplikaci Hafici.net, také nepovažuji za jejich vhodnou alternativu. Skutečnost, že jsem s žádnou z těchto aplikací nebyl spokojený, mě utvrzuje v mém přesvědčení, že pořádný portál pro majitele psů chybí.

Kapitola 4
Analýza požadavků

Sběr a analýza požadavků jsou, v softwarovém inženýrství, považovány za klíčové kroky pro úspěšný vývoj aplikace. Slouží především k ujasnění, vymezení a správnému porozumění funkcí, které zadavatel požaduje, a které má výsledný software obsahovat. Bez řádné specifikace požadavků může při vývoji docházet k prodloužení vývoje softwaru a tím i zvýšení jeho výsledné ceny, což často vede ke sporům s klientem a jeho následné nespokojenosti. [6] Právě proto je cílem této kapitoly specifikovat uživatele (a jejich role) a současně funkční a nefunkční požadavky aplikace. Požadavky byly sbírány od zadavatele této bakalářské práce, kterým je firma Triton IT s.r.o.

4.1 Seznam funkčních požadavků

Funkční požadavky jasně popisují jednotlivé funkcionality, které má aplikace pokrývat. Seznam těchto požadavků je většinou základním kamenem pro návrh a testování aplikace, protože nám poskytuje komplexní náhled na zadavatelem vyžadované funkcionality, které má výsledná aplikace obsahovat. [6]

F1: Seznam psích ras
Aplikace umožňuje uživateli zobrazit přehled a detail psích ras. V detailu rasy uživatel najde kromě textových informací o dané rase i číselné hodnocení jejích charakteristik (např. *línání: 1/5, slintavost: 3/5*, atd.) a seznam podobných ras.

F2: Seznam psích nemocí
Uživatel má možnost si zobrazit seznam a detail psích nemocí. Zároveň má možnost tento seznam filtrovat na základě projevovaných symptomů.

F3: Seznam psích institucí
V aplikaci budou evidovány různé psí instituce, ve kterých bude uživatel moci vyhledávat podle jejich typu, názvu a lokality. Po vyhledání se uživateli zobrazí seznam odpovídajících institucí.

F4: Vyhledávání psa
Psy bude v aplikaci možné vyhledávat pomocí jména, čísla čipu, čísla tetování, nebo registračního kódu uvedeného v jeho rodokmenu. Toto vyhledávání půjde dále filtrovat podle chovné stanice, ze které pes pochází. Výsledkem tohoto vyhledávání bude seznam psů, kteří tomuto hledání odpovídají.

F5: Zobrazení detailu psa
V detailu psa se uživateli zobrazí všechny jemu dostupné informace o daném psovi. Více o tom, k jakým informacím má který uživatel přístup, je napsáno v sekci 4.3

F6: Registrace a přihlašování
Pokud bude uživatel chtít využívat všechny části aplikace, bude pro něj nezbytná registrace a vytvoření uživatelského profilu. Při následném přihlašování bude identifikován kombinací uživatelského jména nebo emailu a hesla.

F7: Vytvoření psa
Přihlášený uživatel má možnost vyplněním příslušného formuláře vytvořit nový záznam o psovi. Pokud uživatel vytváří psa v rámci rodokmenu a tento pes již v systému existuje, je uživateli nabídnuta možnost propojení na tento existující záznam namísto vyváření záznamu nového. Jinak systém duplikaci dat neřeší.

F8: Správa psa a jeho záznamů
Pakliže má uživatel u psa příslušné oprávnění (více rozvedeno v 4.3.2), může upravovat údaje, záznamy a historii nemocí tohoto psa. Záznamy a nemoci do jeho historie může takový uživatel také přidávat. Důležitou součástí údajů psa, nad kterou bude mít uživatel kontrolu, je jeho psí rodokmen.

F9: Sledování psa
Aplikace nabízí uživatelům možnost sledovat profily psů dle jejich výběru. Veřejné záznamy všech sledovaných psů jsou následně pro uživatele dostupné na jednom místě, seřazené od nejnovějších dle data publikování záznamu.

F10: Přidání inzerátu
Další důležitou částí aplikace jsou inzeráty a jejich vytváření. Každý přihlášený uživatel může vytvořit inzerát, který může zařadit do některé z kategorií (např. ztracený/nalezený pes, prodej psa, prodej psích doplňků). Uživatelé budou také, za jistých podmínek (více o těchto podmínkách v sekci 4.3.2), moci k inzerátu připojit odkaz na konkrétního psa, či vydat inzerát za instituci.

F11: Přehled a detail inzerátů
Pro snadné hledání bude aplikace obsahovat seznam inzerátů, který bude filtrovatelný podle kategorie a půjde řadit dle data, nebo ceny. Na detailu inzerátu bude uživatel mít možnost na inzerát odpovědět vyplněním formuláře. Pokud uživatel inzerát vytvořil, pak na detailu inzerátu najde přehled všech odpovědí spolu s kontaktními informacemi odpovídajícího uživatele. Další komunikace mezi uživateli není aplikací zajištěna.

F12: Uživatelský profil
Na detailu uživatelského profilu je v aplikaci zobrazen přehled psů, ke kterým má uživatel vazbu jako Majitel nebo Editor, dále přehled uživatelem vytvořených inzerátů a neposlední řadě přehled institucí, u kterých má uživatel roli Editor. Pokud je uživatel na svém vlastním profilu, pak navíc vidí své kontaktní údaje s možností jejich úpravy. Kromě uživatelského jména nejsou jiné informace ostatním uživatelům zobrazeny.

F13: Podpora více jazyků

Uživatel si bude moci zvolit jazyk, ve kterém mu bude aplikace zobrazovat její obsah. Pokud obsah nebude v daném jazyce dostupný, aplikace se pokusí zobrazit obsah v aplikací specifikovaném, prioritním pořadí jazyků. U obsahu, který je přidávaný uživatelem (informace o psovi, záznamy o psovi, inzeráty), se jazykové varianty neřeší a to s výjimkou názvů a popisů institucí.

4.2 Seznam nefunkčních požadavků

Požadavky, které sice přímo nespecifikují funkcionality aplikace, ale přesto mají velký vliv na koncové uživatele, údržbu a budoucí vývoj, nazýváme **nefunkčními požadavky**. Mezi nefunkční požadavky patří například požadavky na škálovatelnost, efektivnost, dostupnost, výkonnost, ale i například specifické technologie použité pro návrh a vývoj systému. [7]

N1: Uživatelská přívětivost

Aplikace by měla být srozumitelná a jednoduchá na používání pro všechny generace a typy uživatelů.

N2: Rozšířitelnost

Aplikace bude snadno rozšířitelná o nové funkcionality.

N3: REST API

Serverová část aplikace bude řešená formou REST API[1] implementovaného ve Spring Boot frameworku v jazyce Java. Existence tohoto API nám do budoucna zajišťuje snadné vytvoření aplikací pro jiné platformy je ta webová.

4.3 Aktéři

Roli subjektu (uživatele nebo jiného systému), který není součástí aplikace, ale interaguje sní – vyžívá a řídí průběh jejích funkcionalit, nazýváme *aktérem*. [9] V aplikaci můžeme role uživatelů rozdělit do tří kategorií podle toho zdali se jedná o roli vůči:

- aplikaci jako takové,
- jednotlivému psovi,
- nebo instituci.

4.3.1 Uživatelské role vůči aplikaci

Role v této kategorii ovlivňují, ke kterým částem aplikace má daný uživatel přístup. Konkrétně se jedná o tyto tři role:

Nepřihlášený uživatel má omezený přístup k aplikaci a vidí pouze její veřejné části. Může si zobrazit informace o rasách a nemocích psů, vyhledávat v seznamu institucí a prohlížet veřejné inzeráty. Psy může vyhledávat, ale uvidí pouze základní informace, jako je například jméno, rasa, číslo čipu, barva a fotografie. Uživatel s touto rolí nemá žádné práva na změnu jakéhokoli obsahu v aplikaci.

[1] Z anglického *Representational state transfer application programming interface, jedná se o soubor pravidel pro komunikaci mezi klientskou a serverovou částí aplikace. [8]*

Běžný přihlášený uživatel může navíc oproti nepřihlášenému uživateli vytvářet psy a instituce, které může za určitých podmínek také upravovat (tyto podmínky jsou rozvedené v sekcích 4.3.2 a 4.3.3). Kromě psů a institucí smí vytvářet a následně upravovat inzeráty. Dále má přístup ke všem veřejným informacím a záznamům o psovi. Pro tuto roli dále v textu používám zkrácený název *Běžný uživatel*

Administrátor je přihlášený uživatel, který má neomezený přístup do celé aplikace. Tato role slouží pouze pro případné řešení problémů a počet jejích uživatelů je velmi omezený.

4.3.2 Uživatelské role vůči psovi

Protože je v aplikaci žádoucí kontrola nad tím, kdo může měnit údaje psa, mohou mít vůči jednotlivým psům *běžní uživatelé* tyto různé role:

Majitel psa je role, která jejímu uživateli umožňuje měnit veškeré údaje psa a jeho rodokmenu, a to včetně případného převodu na nového majitele. Zároveň má právo přidávat, měnit a mazat veškeré záznamy o daném psovi, nebo ho přiřadit k, tímto uživatelem vytvořenému, inzerátu. V neposlední řadě může tento uživatel k tomuto psovi přidávat a odebírat uživatele v roli *editor psa*. Tuto roli získává automaticky uživatel, který psa vytvoří a při jeho vytváření se přihlásí jako jeho majitel.

Editor psa má stejná práva jako *majitel psa* až na možnost převést psa na nového majitele a přidávání a odebírání uživatelů v roli *editor psa*. Tato role tak umožňuje spravovat profil psa například rodinnému příslušníkovi. Kromě manuálního přiřazení uživatele do této role *majitelem psa*, může tuto roli uživatel získat také získat automaticky, a to v případě kdy uživatel psa vytvoří, ale při jeho vytváření se nepřihlásí jako jeho majitel (například při přidání předka do rodokmenu psa). Při každé změně majitele psa tuto roli u daného psa všichni uživatelé automaticky ztratí.

Sledující psa je role, která uživateli nedává žádné práva vůči psovi. Ke sledování psa se může přihlásit jakýkoli *běžný uživatel*, který tak získá přehled všech nově přidaných, veřejných záznamů u všech psů, které sleduje.

4.3.3 Uživatelské role vůči instituci

U jednotlivých institucí chceme dát oprávněným osobám kontrolu nad údaji této instituce, a právě k tomto účelu slouží tato role:

Editor instituce má práva pro změnu údajů dané instituce a vytváření inzerátů jejím jménem.

4.4 Případy užití

Případy užití zachycují scénáře použití konkrétních funkčností aplikace z pohledu aktéra, čímž pomáhají identifikovat očekávané chování systému. [10] Případ užití by měl mít název a obsahovat aktéra (kdo scénář vykonává) a tok událostí (uživatelský scénář).

UC1: Zobrazení informací o rase

Aktér: jakýkoli uživatel

Tok událostí: V hlavním menu uživatel klikne na tlačítko „Atlas psů“. Aplikace uživateli zobrazí seznam ras, kde každá položka obsahuje ilustrační obrázek rasy a její název. Dále aplikace zobrazí vyhledávací pole, které uživateli

dovolí textové vyhledávání podle názvu rasy. Po kliknutí na některou z položek seznamu bude uživatel přesměrován na stránku detailu dané rasy. Na detailu rasy uživatel najde kromě informací o dané rase i seznam nejnovějších inzerátů a seznam oblíbených psích profilů spojených s touto rasou.

UC2: Zobrazení psích nemocí

Aktér: jakýkoli uživatel

Tok událostí: Po kliknutí na tlačítko „Atlas nemocí“ v hlavním menu, aplikace uživateli zobrazí seznam psích nemocí. Položka seznamu bude vždy obsahovat název nemoci, její krátký popis a částečný seznam jejích symptomů. Uživatel může seznam filtrovat pomocí zvolení kombinace symptomů, nebo jejího názvu. Po kliknutí na položku seznamu je uživatel přesměrován na detail dané nemoci, kde mu aplikace zobrazí podrobnější popis, popis léčby, kompletní seznam symptomů a další informace o nemoci.

UC3: Zobrazení psích institucí

Aktér: jakýkoli uživatel

Tok událostí: Uživatel v hlavním menu a klikne na tlačítko „Instituce“. Aplikace zobrazí seznam evidovaných psích institucí s názvem a základními informacemi. Uživatel může využít vyhledávací pole, aby filtroval instituce podle názvu, typu nebo lokality. Po výběru konkrétní instituce je uživatel přesměrován na stránku s detailnějšími informacemi o této instituci.

UC4: Zobrazení inzerátů

Aktér: jakýkoli uživatel

Tok událostí: Uživatel klikne v hlavním menu na tlačítko „Inzeráty“. Aplikace zobrazí seznam aktuálních inzerátů, každý s náhledem obsahu, kategorie a datem publikace. Uživatel může inzeráty filtrovat podle kategorie nebo vyhledávat pomocí jejich štítků. Po kliknutí na inzerát uživatel zobrazí jeho plný obsah a detaily, včetně formuláře pro odpovězení na inzerát. Tvůrce inzerátu na jeho detailu uvidí seznam těchto odpovědí s kontaktními informacemi tvůrce odpovědi.

UC5: Registrace a přihlášení

Aktér: nepřihlášený uživatel

Tok událostí: V hlavním menu uživatel klikne na ikonu osoby a bude přesměrován na formulář pro přihlášení. Pro přihlášení zadá uživatelské jméno a heslo, které aplikace zkontroluje. V případě zadání chybných údajů je uživateli zobrazena chybová hláška. Pro registraci uživatel klikne na stránce pro přihlášení na tlačítko „Vytvořit účet“, které ho přesměruje na formulář určený pro registraci. Po vyplnění požadovaných údaje jako uživatelské jméno, e-mail a heslo a odešle formulář a aplikace provede kontrolu na duplicitu e-mailu a uživatelského jména a dostatečné bezpečnosti hesla. V případě neúspěchu je opět uživateli zobrazena chybová hláška. Po úspěšné registraci nebo přihlášení je uživatel přesměrován na domovskou stránku.

UC6: Vytvoření psa

Aktér: přihlášený uživatel

Tok událostí: Přihlášený uživatel přejde na stránku pro správu svého profilu a vybere „Přidat psa“. Vyplní formulář s informacemi o psovi, jako je jméno, číslo čipu a plemeno, a poté odesílá údaje. Systém zkontroluje vyplněnost povinných údajů a následně vytvoří záznam o novém psovi. V případě chyby je uživateli zobrazena chybová hláška.

UC7: Přidání záznamu psa

Aktér: majitel/editor psa

Tok událostí: Majitel nebo editor psa přejde na detail psa, ke kterému chce přidat záznam. Na detailu psa klikne na tlačítko „Přidat záznam“, které mu zobrazí formulář pro přidání záznamu. Uživatel formulář vyplní a klikne na tlačítko „Přidat“. Data formuláře jsou aplikací zkontrolována a v případě chyby je uživateli zobrazena chybová hláška.

UC8: Přidání záznamu do historie nemocí psa

Aktér: majitel/editor psa

Tok událostí: Tok událostí je identický s tokem událostí UC7. Jediným rozdílem je kliknutí na tlačítko „Přidat záznam nemoci“ na detailu psa a vyplnění jiného, příslušného, formuláře.

UC9: Sledování psa

Aktér: přihlášený uživatel

Tok událostí: Uživatel přejde na detail psa a na jeho profilové stránce klikne na tlačítko „Sledovat“. Pes bude přidán do seznamu sledovaných psů uživatele. Pokud uživatel již daného psa sleduje je mu namísto tlačítka „Sledovat“ zobrazeno tlačítko „Zrušit sledování“. Po kliknutí na toto tlačítko je pes odebrán ze seznamu uživatelem sledovaných psů.

UC10: Zobrazení nejnovějších záznamů sledovaných psů

Aktér: přihlášený uživatel

Předpoklad: uživatel sleduje alespoň jednoho psa

Tok událostí: Seznam nejnovějších záznamů psa je přihlášenému uživateli zobrazen na hlavní stránce. Pokud uživatel žádného psa nesleduje, je mu místo tohoto seznamu zobrazena hláška „Zatím nesledujete žádného psa“.

UC11: Vytvoření inzerátu

Aktér: přihlášený uživatel

Tok událostí: Uživatel na stránce seznamu inzerátů vybere možnost „Přidat inzerát“. Systém přesměruje uživatele na formulář pro vytvoření inzerátu.

Uživatel vyplní potřebné údaje v inzerátu, jako jsou kategorie, titulek, popis a případné obrázky. V případě, že je uživatel editorem instituce, nebo majitelem psa, má právo k inzerátu přidat vazbu na danou instituci, nebo daného psa. Po odeslání formuláře systém provede kontrolu údajů, aby zjistil, zda jsou všechny povinné údaje vyplněny. Po úspěšné kontrole je inzerát vytvořen a uživatel přesměrován na jeho detail.

UC12: Přepnutí jazyka aplikace

Aktér: jakýkoli uživatel

Tok událostí: Uživatel v hlavním menu vybere preferovaný jazyk z nabídky dostupných jazyků. Aplikace změní jazyk rozhraní podle uživatelova výběru.

Kapitola 5
Návrh

5.1 Architektura aplikace

Aplikace bude postavená na třívrstvé klient-server architektuře. Mezi tři vrstvy této architektury patří:

Prezentační vrstva obsahuje pouze logiku zobrazování informací a sběr dat uživatele.

Aplikační vrstva slouží pro zpracování a manipulaci s daty od uživatele. Tato vrstva je prostředníkem mezi prezentační a datovou vrstvou, které nesmějí komunikovat přímo mezi sebou. Komunikace s prezentační vrstvou je zajištěna pomocí API.

Datová vrstva následně tyto zpracované data ukládá a spravuje.

Díky tomuto rozdělení aplikace do vrstev, může být každá vrstva nasazena do vlastního dedikovaného prostředí, které tak může být dané vrstvě přizpůsobené a optimalizované na míru. Mezi další výhody této architektury patří například snazší škálovatelnost a jednodušší vývoj a údržba aplikace, protože je možné se při těchto akcích zaměřit pouze na jednu z vrstev. [11]

5.1.1 Klient

Klientskou částí tohoto systému bude webová aplikace. Z pohledu třívrstvé architektury se jedná o vrstvu prezentační.

5.1.2 Server

Většina dnešních aplikací pro komunikaci mezi klientem a serverem využívá API (z angl. application protocol interface). API je množina pravidel a instrukcí, která definuje způsob komunikace mezi klientem a serverem. Mezi, v dnešní době, nejpoužívanější API patří REST, GraphQl a již starší SOAP.

Jak je již definováno v nefunkčních požadavcích (4.2) aplikace bude pro komunikaci využívat REST API. REST API pro komunikaci vyžívá HTTP(S) protokol a je založeno na takzvaných „zdrojích“. Každý zdroj je identifikován unikátní URI (uniform resource identifier) a uživatel k měnu může přistupovat pomocí GET, POST, PUT a DELETE metod z protokolu HTTP(S). Dalším důležitým aspektem REST API je jeho bezstavovost – každý požadavek od klienta obsahuje všechny potřebné informace pro jeho zpracování a je zpracován nezávisle na předchozích požadavcích. [12]

5.1.3 Databáze

Při výběru vhodné databáze máme na výběr mezi SQL (Structured query language) databázemi a NoSQL (Not only SQL) databázemi. Největším rozdíl mezi těmito databázemi je struktura ukládaných dat. Zatímco SQL databáze ukládají strukturovaná data v předem definovaných tabulkách, NoSQL databáze nemusí mít strukturu dat definovanou a jsou tak více flexibilní. Za tento fakt ovšem NoSQL databáze platí integritou, konzistencí a bezpečností dat. [13] A právě z tohoto důvodu jsem se pro tuto aplikaci ve výsledku rozhodl využít klasickou SQL databázi.

5.2 Použité technologie

V této sekci blíže popíšu konkrétní technologie, které budou použity pro vývoj aplikace a odůvodním jejich výběr. V předchozím i následujícím textu se opakovaně vyskytuje slovo *framework*, pojďme si tedy jeho význam vysvětlit. **Framework** je sada komponent a návrhových vzorů, které jsou opakovaně a často využívány při vývoji aplikace. Tyto komponenty by měly být snadno upravitelné a rozšířitelné a měly by být modulární, tak aby se vývojáři mohli zaměřit na vývoj částí specifických pro konkrétní aplikaci namísto „znovuobjevování Ameriky“. [14]

5.2.1 Klient

V jazyce PHP je vytvořeno přes 77 % webových aplikací a je tak dlouhodobě nejpopulárnějším jazykem pro vytváření webových aplikací. [15] Symfony je jedním z nejpoužívanějších PHP frameworků, který je známý pro svou flexibilitu, rychlost a velikou komunitu.

Na frontend aplikace použiji kombinaci HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets) a JavaScriptu. Jedná se o standardní vykreslovací technologie používané ve webových prohlížečích pro zobrazování obsahu webových aplikací.

Bootstrap je CSS a JavaScriptový framework, který dodává vývojářům základní stylovací a interaktivní komponenty a umožňuje jim tak snadno a rychle vytvářet uživatelsky přívětivé a responzivní webové aplikace.

Pro vytvoření interaktivních map, které jsou vhodné pro všechny typy zařízení, použiji JavaScriptovou knihovnu **Leaflet**. Tato knihovna nabízí jednoduché, efektivní a dobře zdokumentované řešení interaktivních map s možností použití velkého množství existujících rozšíření

5.2.2 Server

Jak už bylo uvedeno v nefunkčních požadavcích, serverová část aplikace bude postavena na Java frameworku Spring Boot. Spring Boot je nástavbou na Java framework Spring a jeho hlavním cílem je usnadnění vývoje aplikace v jazyce Java. Tohoto chce docílit například pomocí přesunu konfigurace aplikace z XML souborů do Java kódu, čehož dosahuje například díky skenování a následnému automatickému předávaní závislostí vytvořených komponent za pomoci anotací. Další z jeho výhod je, v základním balíčku, obsažený webový server Tomcat, díky kterému je proces nasazení aplikace výrazně ulehčen. Díky své přívětivosti k vývojářům a velkému množství existujících rozšiřujících balíčků Spring Boot patří mezi nejpopulárnější technologie pro vytváření RESTových API. Jeho stinnou stránkou je, že obsahuje velké množství závislostí, které nemusí být vždy všechny využity. [16]

5.2.3 Databáze

I přesto, že už jsem se rozhodl pro použití relační SQL databáze, stále zbývá vybrat, kterou konkrétní použít. Mezi nejpopulárnější z nich patří konkrétně MySQL a jeho odnož MariaDB, MSSQL, Oracle, PostgreSQL a SQLite. Protože databáze Oracle a MSSQL jsou proprietární

a placené softwary určené především pro veliké podniky, nejsou pro potřeby této bakalářské práce vhodné a zbývají nám tedy čtyři možnosti.

SQLite je bez-serverová souborová databáze nabízející především snadnou přenosnost a jednoduché použití bez konfigurací. To jsou ale zároveň i její nevýhody, protože její maximální počet současných operací je díky zamykání souborů velmi omezený a bezpečnost dat je ohrožována neexistencí možnosti vytvoření uživatelských přístupů. Toto databázové řešení je vhodné spíše pro malé aplikace s omezeným množstvím uživatelů, nebo pro potřeby testování. [17]

MySQL byla původně opensource databáze, ale dnes je již vlastněná firmou Oracle, díky čemuž je v dnešní době opensource již pouze z části. To způsobilo že mnohé z funkcí, které jsou dnes považovány za standard, jsou dostupné pouze v její placené části a vývoj její veřejné části je velmi pomalý. Tento problém se snaží řešit její odnož **MariaDB**, která je vyvíjena původním autorem projektu MySQL. To se databázi MariaDB vcelku úspěšně daří a ta tak nabízí větší množství funkcí a lepší výkon databáze. [18]

PostgreSQL je objektově-relační databáze, díky čemuž nabízí kromě standardních relací, také funkce jako je například dědění tabulek. PostgreSQL si zakládá na vysokém množství podporovaných funkcí, jednoduché rozšiřitelnosti a otevřeném zdrojovém kódu s velkou komunitou. Za tyto zásady ovšem platí daň v podobě pomalejší rychlostí při čtení záznamů a vysokou spotřebou paměti. [19]

Po bližším prozkoumání možností jsem se rozhodoval mezi databázemi MariaDB a PostgreSQL. Nakonec jsem se rozhodl pro tento projekt použít databázi PostgreSQL, protože věřím, že se jedná o do budoucna robustnější řešení.

5.3 Databázový model

Pro bližší pochopení entit (a vztahů mezi nimi) vyskytujících se v aplikaci jsem vytvořil databázový model. Návrh databázového modelu vznikal na základě analýzy požadavků provedené v kapitole 4. V následujícím textu vybrané důležité entity popíši a definuji jejich vztahy. Diagram databázového jádra aplikace je k vidění na obrázku 5.1. Kompletní databázové schéma je k nalezení v přiloženém médiu v souboru *docs/database-diagram.png*.

appuser: Tato entita spravuje informace týkající se uživatele. Název „appuser“ byl zvolen, protože slovo „user“ je v mnoho databázových systémech rezervováno jako klíčové slovo.

- **username:** (povinný atribut) Unikátní přezdívka uživatele.
- **password:** (povinný atribut) Šifrované heslo pro ověření uživatel.
- **e-mail:** (povinný atribut) E-mailová adresa uživatele.
- **enum_role_id:** (povinný atribut) Cizí klíč odkazující na roli, kterou má uživatel vůči celé aplikaci.

dog: Je hlavní entita pro ukládání všech klíčových informací o vytvořených psech, jako jsou údaje o jejich plemeni, jménu, nebo číslu jejich čipu.

- **name:** (povinný atribut) Jméno psa.
- **birth_date:** Datum narození psa.
- **death_date:** Datum úmrtí psa.
- **enum_gender_id:** (povinný atribut) Cizí klíč s vazbou na pohlaví psa.
- **enum_breed_id:** (povinný atribut) Cizí klíč s vazbou na rasu psa.
- **dog_pedigree_id:** Cizí klíč s vazbou na entitu znázorňující dokument (*„papíry“*) psa.

dog_media_id: Cizí klíč s vazbou na hlavní médium zobrazující psa.

sire_id: Cizí klíč s vazbou na otce psa (vazba na tuto entitu).

dam_id: Cizí klíč s vazbou na matku psa (vazba na tuto entitu).

tattoo_number: Číslo tetování.

chip_number: Číslo čipu.

color: Textový atribut barvy.

biography: Textový atribut umožňující majitelům přidat popisek k psovi.

dog_record Je entitou uchovávající vytvořené psí záznamy.

dog_id: (povinný atribut) Cizí klíč

s:

appuser_id: (povinný atribut) Cizí klíč s vazbou na uživatele, který záznam vytvořil.

created: (povinný atribut) Datum vytvoření záznamu, měl by být automaticky generovaný při vložení záznamu.

title: (povinný atribut) Nadpis záznamu.

value: Textová hodnota záznamu.

public: Hodnota znázorňující zdali je záznam viditelný pro veřejnost (true/false).

institution: Je obecná entita ukládající informace, které jsou společné pro všechny typy institucí. Na tuto entitu poté mají vazbu konkrétní entity institucí.

name: (povinný atribut) Název instituce.

info: Textový atribut umožnující uživatelům přidávat k spravovaným institucím textový popis, či jiné informace.

address_id: (povinný atribut) Cizí klíč s vazbou na adresu sídla instituce.

ad: Slouží k ukládání informací o vytvořených inzerátech.

enum_ad_type_id: (povinný atribut) Cizí klíč s vazbou na typ inzerátu (ztráta psa/nalezení psa/nabídka/...)

dog_id: Cizí klíč s vazbou na psa, dává uživatelům možnost provázat psa s inzerátem.

appuser_id: (povinný atribut) Cizí klíč s vazbou na uživatele. Ukládá informaci, kdo inzerát vytvořil.

enum_city_id: (povinný atribut) Cizí klíč s vazbou na město, ve kterém je inzerát nabízen.

title: (povinný atribut) Název inzerátu.

ad_text: (povinný atribut) Text inzerátu.

published: (povinný atribut) Datum vytvoření inzerátu, měl by být automaticky generovaný při vložení záznamu.

price: Nabízená/požadovaná cena.

media: Uchovává informace o mediích nahraných uživatelem.

enum_media_type_id: (povinný atribut) Cizí klíč s vazbou na typ nahraného média.

appuser_id: (povinný atribut) Cizí klíč s vazbou na uživatele, který medium nahrál.

path: (povinný atribut) Obsahující cestu k médiu.

title: Název média.

alt: Alternaticní popis média.

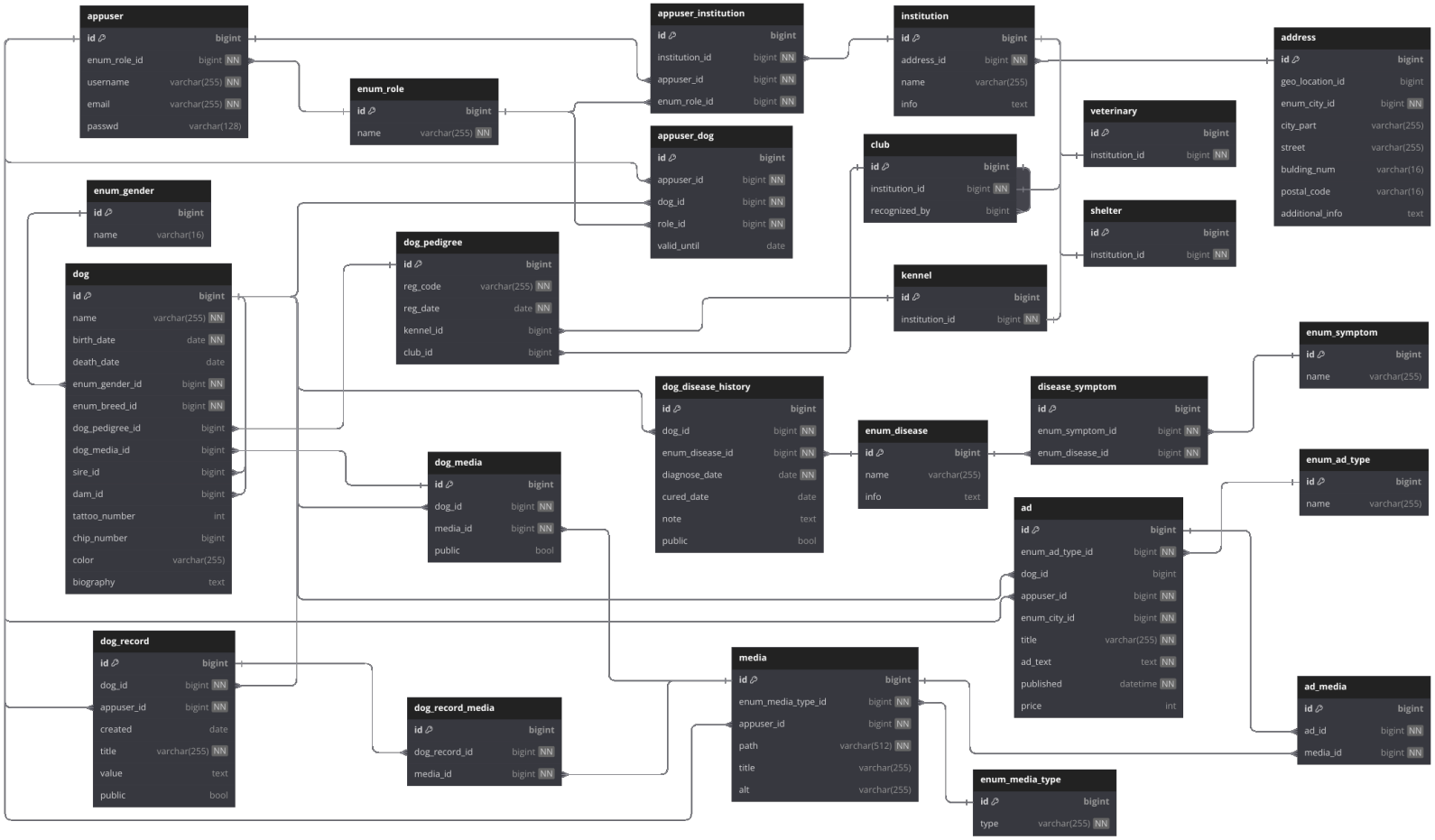
Entity ras: Informace o rasách jsou rozděleny do tří různých entit, které spolu tvoří jeden celek informací o rase. Těmito entitami jsou:

enum_breed: Entita uchovávající obecné informace o rase jako je její (povinný atribut) název, zkratka a textové pole uchovávající alternativní názvy. Dále má tato entita nepovinnou vazbu na FCI (Fédération Cynologique Internationale) sekci, ve které je rasa kategorizována.

breed_characteristics: Slouží k uchovávání číselných charakteristik rasy, každá rasa může mít n různých charakteristik. Její atributy jsou povinný název a číselná hodnota charakteristiky.

breed_text_field: Tato entita slouží k zachycování textových popisů jednotlivých ras. Důvodem pro vlastní entitu, namísto ukládání těchto informací v textovém poli hlavní entitě, je větší kontrola nad těmito informacemi a umožnění rozšíření do budoucna (například o prioritu zobrazování informací). Entita ukládá povinný název textového pole a jeho text samotný.

Entity překladů: Veškerý textový obsah aplikace, který není přidávaný běžným uživatelem (+ názvy a informace k institucím), má tyto textové atributy oddělené do separátních entit, které mají vztah k jazyku, ve kterém je obsah napsán. Tyto entity překladů mají vždy stejný název s původní entitou zakončený příponou „_tranls“.



Obrázek 5.1 ER diagram jádra databáze

Kapitola 6
Implementace

Cílem této kapitoly je popsat průběh vývoje aplikace s konkrétními ukázkami implementačních řešení.

6.1 Verzovací systém

Verzovací systém (anglicky Version Control System – VSC) je důležitý nástroj, který při vývoji softwaru umožňuje vývojářům efektivně spravovat a sledovat změny v kódu. Kromě toho také umožňuje jednoduché slučování kódu, díky čemuž usnadňuje týmovou spolupráci. Díky ukládání historie změn v kódu se, při používaní verzovacího systému, vývojáři nemusí bát nechtěného smazání částí kódu, a může být nápomocný při potřebě rychlé opravy nově vzniklých chyb. [20]

V dnešní době je pro vývojáře použití verzovacího sytému již standardem a i já ho při vývoji této práce používal. Konkrétně jsou používal repozitář v systému GitLab, který je ovšem vhledem k licenci této práce soukromý.

6.2 Server

V této sekci naleznete popis struktury serverové části aplikace a ukázky konkrétních implementačních řešení jejích vybraných částí.

6.2.1 Struktura

Hlavní balíček aplikace je rozdělen na několik pod-balíčků sdružujících dohromady logické celky aplikace. Těmito balíčky jsou:

api: balíček obsahující logiku potřebnou k zobrazování odpovědí uživateli, je rozdělen na další tři pod-balíčky:

- **controller:** obsahuje zdrojové kódy kontrolerů aplikace.
- **converter:** tento balíček obsahuje třídy, které převádí objekty entit na DTO objekty.
- **dto:** v tomto balíčku se nachází DTO (Data Transfer Object) třídy. Tento návrhový vzor slouží k definici struktury odpovědí na požadavky, pomáhají omezovat množství volaných metod a zabraňují jejich zacyklení. [21]

config: obsahuje konfigurační třídy frameworku Spring Boot. Jedná převážně o třídy spojené s bezpečnostní konfigurací aplikace.

exception: je balíček s definicemi vlastních výjimek používaných v aplikaci.

model: v sobě má třídy entit mapované na entity databázové pomocí ORM (Object Relational Mapping).

repository: balíček obsahující třídy sloužící ke komunikaci mezi aplikací a datbází.

service: a její pod-balíček translation obsahují servisní třídy, které většinou obsahují logiku pro práci s entitami. Mouhou ale obsahovat i jinou důležitou aplikační logiku. Její podsložka **translations** pak obsahuje logiku specifickou pro řešení překladů entit.

specification obsahuje logiku pro vytváření vlastních specifikací, které slouží k filtrování entit.

V této základní adresářové struktuře se mohou nacházet další podadresáře, jejichž význam je rozdělení tříd podle jejich souvislostí z hlediska modelu. Znázornění souborové struktury serverové části aplikace:

```
cz.tritonit.aport.backend
├── api
│   ├── controller
│   ├── converter
│   └── dto
├── config
├── exception
├── model
├── repository
├── service
│   └── translation
└── specification
```

6.2.2 CrudController

Kontrolery jsou třídy zodpovědné za mapování url adres a odpovídaní na uživatelské požadavky. CrudController je abstraktní třída, která umožňuje snadné vytváření ostatních kontrolerů a pomocí přepisování jejích metod umožňuje jednoduše definovat, které ze základních požadavků jsou uživateli povolené.

Jak můžete v kódu 6.1 vidět, abstraktní třída CrudController implementuje obecnou logiku pro mapování a vracení odpovědí pro základní CRUD (Create, Read, Update a Delete) požadavky (tedy POST, GET, PUT a DELETE HTTP metody). Před vrácením odpovědi je ale v každé z těchto funkcí zavolaná jí odpovídající „before“ funkce, která ve výchozím nastavení vrací HTTP odpověď „Method not allowed“, která uživateli značí použití nepovolené HTTP metody na dané url adrese. Díky tomuto přístupu máme v následně oděděných kontrolerech plnou kontrolu nad povolenými operacemi pomocí jednoduchého přetížení „before“ funkcí a nemusíme opakovaně psát stejný kód pro různé entity. Navíc nám toto přetěžování umožňuje například dodatečnou kontrolu práv uživatele pro vykonání daného požadavku.

6.2.3 Překlady entit

Dle specifikace funkčních požadavků v kapitole 4 má aplikace vracet uživateli obsah v jeho preferovaném jazyce, je-li dostupný a není-li dostupný, má mu vrátit obsah dle uložené priority jazyků. Tato logika je v aplikaci řešená v abstraktní servisní třídě **TranslationReadService**. V ukázce kódu 6.2 nalézt konkrétní implementaci této logiky. V podstatě se jedná o načtení seznamu jazyků z databáze, vložení uživatelem preferovaného jazyka začátek tohoto seznamu a následně postupné vyhledávání entity v daném jazyce. V případě že pro entitu nebyl nalezen žádný překlad, je vyhozena výjimka značící chybu ve stavu entity.

```java
/**
* Inspired by CrudController class created in BI-TJV course
*/
public abstract class CrudController<E extends AbstractPersistable<ID>, // entity
                  L extends AbstractDto, // entity list dto
                  D extends AbstractDto, // entity detail dto
                  ID extends Serializable>
{
  protected AbstractCrudService<E, ID> service;
  protected AbstractConverter<E, D> converter; // detial dto converter
  protected AbstractConverter<E, L> listConverter; // list dto converter

  // contructor ...

  @GetMapping
  @ResponseBody
  public ResponseEntity<?> readAll(
            @RequestParam(value = "search", defaultValue = "") String search,
            Pageable pageable,
            Locale locale
  ) {
    // ...

    Pair<Specification<E>, Pageable> altered = beforeReadAll(spec, pageable, locale);
    spec = altered.a;
    pageable = altered.b;

    // ... logic for retriving and returning dto of expected entities
  }

  @GetMapping("/{id}")
  public D readOne(@PathVariable ID id, Locale locale) {
    beforeReadOne(id);

    // ... logic for retriving and returning dto of expected entity
  }

  protected Pair<Specification<E>, Pageable> beforeReadAll(
                          Specification<E> spec,
                          Pageable pageable,
                          Locale locale
  ){
    throw new ResponseStatusException(HttpStatus.METHOD_NOT_ALLOWED);
  }

  protected void beforeReadOne(ID id) {
    throw new ResponseStatusException(HttpStatus.METHOD_NOT_ALLOWED);
  }

  // ... mapping of POST, PUT and DELTE methods
}
```

Výpis kódu 6.1 Ukázka implementace CrudControlleru

```java
/**
 * Abstract service class for reading translations of entities.
 *
 *  @param <D> entity type
 *  @param <T> translation entity type
 *  @param <ID> entity key type
 */
public abstract class TranslationReadService<D extends AbstractPersistable<?>,
                      T extends AbstractTranslation<D, ID>,
                      ID extends Serializable>
{
  // ...

  /**
   * Reads translation of an entity by locale.
   * @param translated entity to be translated
   * @param locale user preferred locale
   * @return translation
   */
  public T readOneByTranslatedAndLocale(D translated, Locale locale) {
    // get ordered list of languages with preferred language at top
    List<EnumLang> allLangs = getOrderedLangs(locale);

    // try to find translation by order of languages
    for (EnumLang l : allLangs) {
      Optional<T> result = readOneByTranslatedAndLang(translated, l);
      if (result.isPresent())
        return result.get();
    }

    // if no translation is found, throw an exception illegal state of entity
    throw new EntityStateException("No translation for this entity exists");
  }

  /**
   * Gets ordered list of languages bby priority with preferred language at top.
   * @param locale user preferred locale
   * @return list of languages
   */
  private List<EnumLang> getOrderedLangs(Locale locale) {
    // get preferred language
    EnumLang lang = this.langService.getEnumLangByShortcut(locale.getLanguage());
    // get all languages ordered by priority
    List<EnumLang> allLangs = langService.getEnumLangsOrederdByPriority();
    // remove preferred language from list and add it at top
    allLangs.remove(lang);
    allLangs.addFirst(lang);
    return allLangs;
  }

  // ... methods for reading multiple translated entities

}
```

Výpis kódu 6.2 Ukázka implementace TranslationReadService

6.3 Klient

Tato sekce nabízí pohled do implementačního řešení klientské aplikace v PHP frameworku Symfony. Až na datovou vrstvu je v klientské aplikaci využita standardní souborová struktura projekty Symfony.

6.3.1 Repozitáře a DTO

Vzhledem k tomu, že klientská aplikace získává data pomocí RESTového API a data jsou zpracovávána v serverové části aplikace, tak byla datová vrstva v klientské aplikaci nahrazena za DTO a k získávání zdrojů z tohoto API slouží repozitáře. Logika získávání zdrojů je zajištěna pomocí dvou tříd: **ResourceRequest** (ukázka 6.4) a **AbstractRepository** (ukázka 6.4).

ResourceRequest třída je zodpovědná za obecné vytváření a zpracovávaní požadavků na API.

AbstractRepository využívá ResourceRequest třídu k vytváření konkrétních požadavků na konkrétní zdroj API.

Za deserializovaní odpovědí od API je také zodpovědná třída ResourceRequest, která k řešení tohoto problému vyžívá Symfony komponenty **Serializer** (viz 6.3). Komponenta Serializer využívá enkodérů, které umí převést strukturované texty do PHP polí a následně využívá normalizátorů k převodu těchto polí na objekty. Pro speicifké potřeby umožňuje i implementace a použití vlastních enkodérů a normalizátorů. [22] Pro potřeby klienta bylo dostačující použít základní enkodér JsonEncoder a tři základní normalizátory: ObjectNormalizer, ArrayDenormalizer a DateTimeNormalizer.

```php
// ... Inside ResourceRequest class
public function responseToObject(ResponseInterface $response,
              string $responseModel, // class name of expected DTO
              bool $expectedArray=false): mixed
{
  // If response is expected to be an array, add [] to the response model
  $responseModel = $expectedArray ? $responseModel."[]" : $responseModel;
  try {
    // Try to deserialize the response
    return $this->serializer->
          deserialize($response->getContent(), $responseModel, 'json');
  } catch (ClientExceptionInterface|ServerExceptionInterface){
    // If the response is an error, try to deserialize error message
    try {
      return $this->serializer->
          deserialize($response->getContent(false), $responseModel, 'json');
    } catch (\Exception | \Throwable){
      return null;
    }
  } catch (TransportExceptionInterface) {
    // If the response is an error on transport layer, throw a 503 error
    throw new HttpException(503);
  }
}
```

Výpis kódu 6.3 Ukázka implementace metody pro deserializaci odpovědní na DTO

```php
class ResourceRequest
{
  private string $api_url; // loaded from .env

  // ...

  // Create a GET request to the API
  public function get(string $path, ?array $query = null): ?ResponseInterface
  {
    return $this->makeRequest("GET", $path, query: $query);
  }

  // ... POST, PUT and DELETE methods implementation

  // Create a request to the API
  private function makeRequest(string $method, string $path,
          ?string $body = null, ?array $query = null): ResponseInterface
  {
    $header = $this->getHeader($query);
    $path = trim($path, "/");
    if($body !== null)
      $header['body'] = $body;

    try {
      return $this->client->request($method, $this->api_url."/".$path, $header);
    } catch (TransportExceptionInterface $e) {
      throw new HttpException(503);
    }
  }

  // Build the header for the request
  private function getHeader(?array $query = null): array{
    $header =  [
      'headers' => [
        'Accept' => 'application/json',
        'Content-Type' => 'application/json',
        // add user preferred language to the header
        'Accept-Language' => $this->requestStack->getSession()->get('_locale'),
      ],
    ];

    // Add query parameters to the header if defined
    if(!empty($query))
      $header['query'] = $query;

    // Add the jwt token to the header if the user is logged in
    $user = $this->security->getUser();
    if($user !== null and $user::class === User::class )
      $header['headers']['Authorization'] = 'Bearer ' . $user->getToken();

    return $header;
  }
}
```

Výpis kódu 6.4 Ukázka implementace třídy ResourceRequest

```php
abstract class AbstractRepository
{
  private ResourceRequest $resourceRequest;
  // uri of specific resource
  private string $resource;
  // class name of list DTO
  private string $listModelClass;
  // class name of detail DTO
  private string $modelClass;
  // class name of reponse to enity creation
  private ?string $createResponseClass;
  // bool whether there is pagination object response or simple list
  private bool $listExpectedArrayResponse;

  // constructor

  // Find an object by its id
  public function findById(int $id): mixed
  {
    // make the request to the API
    $response = $this->resourceRequest->get($this->resource."/".strval($id));

    // parse the response to the expected object
    try {
      // if the response is an error response
      // throw an HttpException with the error messages or the status code
      if ($response->getStatusCode() >= 400 ) {
        $error = $this->resourceRequest->
            responseToObject($response, ErrorResponseDTO::class);
        if($error instanceof ErrorResponseDTO)
          throw new HttpException($response->getStatusCode(),
                      $error->getMessages());

        throw new HttpException($response->getStatusCode());
      }
    } catch (TransportExceptionInterface $e) {
      throw new HttpException(503);
    }

    // return the object if it is of the correct type
    return $this->resourceRequest->responseToObject($response, $this->modelClass);
  }

  // ... other functions such as findAll, create, update, ect.
}
```

Výpis kódu 6.5 Ukázka implementace třídy AbstractRepository

6.4 Autentizace uživatele

Pro to, aby mohl uživatel naplno využívat všech funkcionalit aplikace, musíme znát jeho identitu, abychom mu mohli přístup povolit. Tento proces potvrzování identity uživatele se nazývá autentizace. Pro autentizaci přes REST API mnoho řešení. V tomto projektu využiji pro autentizaci JWT (Json Web Token). JWT funguje na principu předávání informací o uživateli pomocí JSON objektu, který je zakódovaný v Base64. Tento token je následně většinou buď kryptograficky podepsaný, nebo dokonce zašifrovaný. Hlavní výhodou JWT je jeho jednoduchost, bez potřeby dodatečného ukládání informací. Pro správné zabezpečení je ovšem potřeba chránit komunikaci mezi klientem a serverem pomocí šifrování, jinak hrozí jeho odposlechnutí a zneužití. [23]

6.5 Testování

Aplikace je testována pomocí jednotkových testů. Jednotkový test je test, který testuje pouze malé množství kódu v nezávislém prostředí a je rychlý a opakovatelný. Jednotkové testy nepomáhají pouze s ověřováním správné funkčnosti aplikace, ale jejich využití často vede i k lepšímu návrhu aplikace. To se děje díky tomu, že pokud kód nelze řádně otestovat jednotkovými testy, typicky to znamená velkou provázanost kódu a tedy i špatný návrh aplikace. Cílem jednotkových testů je tedy podpořit vývoj do budoucna udržitelného kódu. [24]

Kapitola 7

Zhodnocení a možný rozvoj aplikace

Vzhledem k veliké rozsáhlosti aplikace a zaměření především na analýzu a návrh aplikace, jsem z časových důvodů nestihl v prototypu aplikace implementovat všechny funkcionality definované v analýze požadavků (4). Zejména jsem nestihl implementovat zobrazování domovské stránky aplikace, výpisy psích záznamů a formuláře pro vytváření a editaci obsahu uživatelem. Prototyp aplikace v současném stavu umožňuje tyto funkcionality:

- vytváření a přihlašování uživatele,
- zobrazení uloženého obsahu na základě uživatelské preference,
- zobrazení přehledu a detailů institucí včetně filtrování,
- zobrazení přehledu a detailů psů včetně filtrování,
- zobrazení přehledu a detailů ras včetně filtrování,
- zobrazení přehledu a detailů nemocí včetně filtrování.

Ze stejného důvodu jsem nestihl vytvořit jednotkové testy pokrývající všechny třídy a testoval jsem tak tedy hlavně třídy, které jsem považoval za ty nejpodstatnější. Kódová báze je pro implementaci těchto funkcionalit a testů však připravená a tato implementace bude prvním krokem v dalším vývoji aplikace. Vhodné by bylo také důkladnější testování i jinými metodami, než jsou jednotkové testy.

Aplikace byla navržena a vyvíjena s předpokladem budoucího vylepšovaní a přidávání jejích funkcionalit. V následujícím textu představím pár nápadů pro rozšíření aplikace.

Důležitým a žádaným rozšířením portálu by mohlo být přidání evidence psích soutěží, výstav a závodů. Uživatelé by tak mohli při vytváření záznamů pro své psy rovnou přidat vazbu na takovouto konkrétní událost.

Další možné rozšíření by bylo v podobě „lajkování“ a přidávání komentářů k záznamům. Toto rozšíření by přidávalo aplikaci na interaktivitě a umožňovalo uživatelům mezi sebou více interagovat. Naneštěstí by tato funkcionalita vyžadovala větší moderaci uživatelského obsahu, aby bylo zajištěno, že se v aplikaci nebude objevovat spam.

V analýze existujících řešení některé aplikace umožňovali svým uživatelům přidávat akce a prohlížet je v kalendáři. Podobná funkcionalita by mohla být dobrým budoucím rozšířením i pro tuto aplikaci.

V neposlední řadě by bylo možné přidat řadu rozšíření z hlediska institucí. Kromě přidání dalších typů institucí jako jsou například obchody, nebo psí salóny, by mohlo být dobrým vylepšením i přidání možnosti vazby instituce na inzerát. Dále by mohli uživatelé spravující instituce, přidávat za tyto instituce příspěvky, stejně tak jako tomu je u psů. Běžný uživatel by pak mohl kromě sledování zajímavých psích profilů, také sledovat profily institucí a mít tak neustálý přehled o dění v psím světě.

Kapitola 8
Závěr

Hlavním cílem této práce bylo navrhnout webovou aplikaci pro evidenci psů a jejich rodokmenů, a dát jejich majitelům možnost s ostatními uživateli sdílet události ze života jejich mazlíčků. Dále byla součástí návrhu evidence psích institucí a nemocí, a možnost sdílení inzerátů s psím světem souvisejících. Tento návrh byl vytvořen na základě jemu předcházející analýzy a následně byl v praktické části práce implementován jeho prototyp.

V analytické části práce jsem nejdříve představil tři podobné, již existující, webové aplikace nabízející funkcionality podobné těm, které jsou součástí aplikace navržené v této bakalářské práci. Následně jsem se zabýval analýzou požadavků, ve které jsem vylíčil detail požadovaných funkcionalit a vlastností aplikace. V neposlední řadě jsem definoval uživatele, kteří aplikaci využívají a jejich očekávaný průchod aplikací.

Následovala kapitola o návrhu aplikace, ve které jsem představil architekturu a technologie, které byly použity pro serverovou a klientskou část aplikace. Zároveň jsem v této kapitole zhodnotil databázové technologie, ze kterých jsem následně pro aplikaci jednu vybral. Následně jsem navrhl databázový model, k jehož důležitým částem vytvořil jejich textový popis.

V kapitole věnující se implementaci jsem popsal své postupy při vytváření aplikace, které jsem doplnil o ukázky konkretních řešení v kódu. Dále jsem se v této kapitole věnoval řešení autentizace uživatele a popsal testovaní aplikace.

V poslední kapitole jsem se věnoval zhodnocení své výsledné práce a navrhl další postup pro její dokončení. Také jsem se zde věnoval možným rozšířením a vylepšením aplikace, které by uživatelům aplikace mohli přinést ještě lepší zážitek z užívaní aplikace.

I přesto, že aplikace nebyla v rámci této práce dokončená, osobně věřím, že tato bakalářská práce je správným prvním krokem pro vytvoření portálu pro všechny majitele psů, bez ohledu na jejich původ.

Bibliografie

1. Dog Pedigree Database. *Pedigree Online* [online]. 2024 [cit. 2024-04-02]. Dostupné z: `https://dogs.pedigreeonline.com/`.
2. O portálu. *Paper-Dog* [online]. 2024 [cit. 2024-04-15]. Dostupné z: `https://www.paper-dog.cz/o-portalu`.
3. HOOGENRAAD, Wim. Send passwords via email? A bad idea! *ITpedia* [online]. 2022 [cit. 2024-04-15]. Dostupné z: `https://en.itpedia.nl/2022/05/15/wachtwoorden-via-e-mail-verzenden-een-slecht-idee/`.
4. Co je to spam? Jak se zbavit spamu? *ESET* [online]. 2024 [cit. 2024-04-15]. Dostupné z: `https://www.eset.com/cz/spam/`.
5. O projektu PesWeb. *PesWeb* [online]. 2014 [cit. 2024-04-16]. Dostupné z: `https://www.pesweb.cz/cz/o-projektu-pesweb`.
6. MALL, Rajib. *Fundamentals of Software Engineering*. 5. vyd. Delhi, India: PHI Learning, 2018. ISBN 978-93-88028-03-5.
7. PARADKAR, Sameer. *Mastering non-functional requirements*. Birmingham, England: Packt Publishing, 2017. ISBN 9781788297899.
8. What is a REST API? *IBM* [online]. 2024 [cit. 2024-04-14]. Dostupné z: `https://www.ibm.com/topics/rest-apis`.
9. SKLENÁŘ, Vladimír. Úvod do paradigmat programovní [online]. 2007 [cit. 2024-04-14]. Dostupné z: `https://phoenix.inf.upol.cz/esf/ucebni/syspro.pdf`.
10. PROCHÁZKA, Jaroslav. Softwarové inženýrství [online]. 2009 [cit. 2024-04-14]. Dostupné z: `https://web.osu.cz/~Zacek/6swen/skriptaSWENG.pdf`.
11. What is three-tier architecture? *IBM* [online]. 2024 [cit. 2024-04-22]. Dostupné z: `https://www.ibm.com/topics/three-tier-architecture`.
12. COCCA, Germán. Different Types of APIs – SOAP vs REST vs GraphQL. *freeCodeCamp* [online]. 2023 [cit. 2024-04-23]. Dostupné z: `https://www.freecodecamp.org/news/rest-vs-graphql-apis/`.
13. MEIER, Andreas; KAUFMANN, Michael. *SQL & NoSQL databases*. 1. vyd. Wiesbaden, Germany: Springer Fachmedien, 2019. ISBN 978-3-658-24549-8.
14. JOHNSON, Ralph. Frameworks = (Components + Patterns). *Commun. ACM*. 1997, roč. 40, s. 39–42. Dostupné z DOI: `10.1145/262793.262799`.
15. KIRAN, Harsha. How Many Websites Use PHP? [PHP Usage Statistics 2024]. *techjury* [online]. 2024 [cit. 2024-05-01]. Dostupné z: `https://techjury.net/blog/php-usage-statistics/`.

16. GUTIERREZ, Felipe. *Pro spring boot 2*. 2. vyd. New York, NY: APRESS, 2018. ISBN 978-1-4842-3675-8.

17. RAVIKIRAN, A. S. What is SQLite? And When to Use It? *simplilearn* [online]. 2023 [cit. 2024-04-28]. Dostupné z: https://www.simplilearn.com/tutorials/sql-tutorial/what-is-sqlite#limitations_of_sqlite.

18. *Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others* [online]. 2023. [cit. 2024-04-28]. Dostupné z: https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/.

19. DRAKE, Mark. *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems* [online]. 2022. [cit. 2024-04-28]. Dostupné z: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems#postgresql.

20. ZOLKIFLI, Nazatul Nurlisa; NGAH, Amir; DERAMAN, Aziz. Version Control System: A Review. *Procedia Computer Science*. 2018, roč. 135, s. 408–415. ISSN 1877-0509. Dostupné z DOI: https://doi.org/10.1016/j.procs.2018.08.191. The 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life.

21. The DTO Pattern (Data Transfer Object). *Baeldung* [online]. [B.r.] [cit. 2024-05-10]. Dostupné z: https://www.baeldung.com/java-dto-pattern.

22. *The Serializer Component (Symfony Docs)* [online]. [B.r.]. [cit. 2024-05-10]. Dostupné z: https://symfony.com/doc/current/components/serializer.html.

23. AKANKSHA; CHATURVEDI, Akshay. Comparison of Different Authentication Techniques and Steps to Implement Robust JWT Authentication. In: *2022 7th International Conference on Communication and Electronics Systems (ICCES)*. 2022, s. 772–779. Dostupné z DOI: 10.1109/ICCES54183.2022.9835796.

24. KHORIKOV, Vladimir. *Unit testing: Principles, practices and patterns*. Manning Publications, 2020. ISBN 9781617296277.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `docs` ..... adresář obsahující vygenerované dokumentace kódu
- `src`
  - `impl`
    - `client` ..... zdrojové kódy implementace klienta
    - `server` ..... zdrojové kódy implementace serveru
    - `HOWTO.txt` ..... soubor obsahující postup pro spuštění
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `Stursa - thesis.pdf` ..... text práce ve formátu PDF