VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Katedra technických studiíKatedra technických studií

Informační systém pro interní potřeby veterinární kliniky

bakalářská prácebakalářská práce

Autor práce: Jan Fortelka

Vedoucí práce: Ing. Marek Musil

Jihlava 2021

Vysoká škola polytechnická Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Jan Fortelka**
Studijní program:	Elektrotechnika a informatika
Obor:	Aplikovaná informatika
Název práce:	**Informační systém pro interní potřeby veterinární kliniky**
Cíl práce:	Cílem práce je zanalyzovat požadavky veterinární kliniky paní Souchové v Pelhřimově a na základě těchto požadavků vytvořit aplikaci s odpovídajícími agendami. Práce bude realizována v prostředí .NET/C# jako desktopová aplikace.

Ing. Marek Musil
vedoucí bakalářské práce

doc. Ing. Zdeněk Horák, Ph.D.
vedoucí katedry
Katedra technických studií

Abstrakt

Práce se zabývá návrhem a implementací aplikace umožňující obsluhu veterinární kliniky. Účelem této aplikace je provádět zákazníkem požadované operace nad daty uloženými v databázi, která je součástí této aplikace. Aplikace je navržena tak, aby uchovávala všechna požadovaná data a zároveň dokázala provádět veškeré základní i pokročilé akce, jako je přidávání, mazání či úprava dat, ale také záloha celé databáze či přiřazení fotografií. Aplikace VetPel je pak určena pro uživatele bez větších technických znalostí, aby mohla být po úvodním proškolení snadno používána kýmkoli.

Klíčová slova

C#; .NET; SQL;Veterinární_ordinace; WPF; XAML

Abstract

The purpose of this work is to propose and implement an application that allows its users to control a veterinary clinic.The aplication is supposed to perform operations with data stored in local database, which is part of this aplication. Aplication Vetpel is built to keep all needed data and perform all basic and advanced actions like adding, deleting and modifiying stored data but also to back up whole database or add photos. VetPel is ment for users who doesnt have any advanced knowledge of controlling computers and after brief training anyone should be able to work with it.

Keywords

C#; .NET; SQL; Vererinary_clinic; WPF; XAML

Prohlašuji, že předložená bakalářskábakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Souhlasím s umístěním bakalářskébakalářské práce v knihovně VŠPJ a s jejím užitím k výuce nebo k vlastní vnitřní potřebě VŠPJ.

Byl/a jsem seznámen/a s tím, že na mou mé bakalářskoubakalářskoupráci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářskébakalářské práce a prohlašuji, že **souhlasím** s případným užitím mé bakalářskébakalářsképráce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářskébakalářsképráce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 26. dubna 202126. dubna 2021

…….

Podpis studenta/ky

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu, panu Ing. Marku Musilovi za cenné rady i dohled nad touto prací a především pak za projevenou trpělivost při jejím dokončení.

Obsah

Úvod ... 8
Motivace ... 9
1 Teoretická část ... 10
1.1 Současný stav ... 10
1.2 Zákaznické požadavky ... 10
1.2.1 Požadavky z pohledu designu ... 10
1.2.2 Požadavky z pohledu dat ... 11
1.2.3 Požadavky z pohledu funkčnosti ... 12
1.3 Současná řešení ... 14
1.3.1 Vetfox ... 14
1.3.2 VetPro ... 17
1.3.3 Další software pro správu veterinárních ordinací ... 20
1.4 Vlastní řešení ... 22
1.4.1 Aplikace Vetpel ... 22
1.4.2 Porovnání aplikace Vetpel s komerčními řešeními ... 23
1.4.3 Zhodnocení použití aplikace VetPel ... 25
2 Praktická část ... 26
2.1 Použité technologie ... 26
2.2 Návrh aplikace ... 32
2.2.1 Nutné prerekvizity ... 32
2.2.2 Architektura tenkých vrstev ... 33
2.2.3 Model tenký klient ... 33
2.2.4 Vetpel_Client ... 34
2.2.5 Vetpel_DataAccess ... 37
2.2.6 Vetpel_DataContract ... 39
2.2.7 Vetpel_DB ... 39
2.2.8 Vetpel_Services ... 39
2.2.9 Regulární výrazy ... 41
2.2.10 Bezpečnostní upozornění ... 41
2.3 Návrh databáze ... 42
2.3.1 Databáze pomocí LINQ2DB ... 43
2.3.2 Model databáze ... 44
2.3.3 Soubory MDF a BAK ... 45

2.3.4 Inicializace databáze ... 46
2.4 Testování ... 47
2.4.1 Zkušební data ... 48
2.4.2 Řešené problémy ... 48
2.4.3 Známé chyby ... 49
2.5 Provoz aplikace ... 50
2.5.1 Aktuální funkce ... 50
2.5.2 Budoucí rozšíření funkčnosti ... 51
2.5.3 Opravné balíčky ... 51
3 Závěr a zhodnocení výsledků ... 53
Seznam zkratek ... 54
Seznam použité literatury ... 55
Seznam obrázků ... 57
Přílohy ... 58

Úvod

Tato práce se zabývá nalezením a realizací vhodného řešení určeného k uchovávání a spravování nezbytných informací pro běžný provoz veterinární kliniky. Věnuje se v současné době celosvětově nutnému trendu digitalizace dat a možnosti snížení jak byrokratické, tak ekologické a časové zátěže kladené na pracovníky i zákazníky veterinární ordinace v Pelhřimově.

Práce představuje nový pohled na práci s daty, zcela odlišný od běžného - a v některých případech státem vyžadovaného – skladování nepřeberného a v praxi zcela nepoužitelného množství papírových dokumentů a materiálů vedoucích k nepřehlednosti a nemožnosti efektivní práce s uchovávanými daty. Prozkoumává již existující komerční řešení, která jsou v tuto chvíli dostupná na trhu, tak i možnost návrhu a následné realizace vlastní ekvivalentní aplikace, která by dokázala zajistit služby o stejném nebo podobném rozsahu a kvalitě, které jsou poskytovány v současnosti dostupnými řešeními. Podrobně analyzuje požadavky zákazníka ze všech pohledů, jež jsou bezpodmínečně nutné ke každodennímu fungování veterinární kliniky, ať už z pohledu designérského, tak pohledu funkčnosti, ale také z pohledu požadavků na uchovávání dat, budoucích modifikací a úprav, a neposlední řadě z pohledu jednoduché obsluhy a přehlednosti aplikace.

Zjištěné skutečnosti jsou následně konfrontovány jak s relevantními tržními nabídkami, tak s dovednostmi autora vytvořit nekomerční desktopovou aplikaci určenou ke stejnému účelu. Analýza současného stavu této problematiky je provedena především praktickým zkoumáním tržních nabídek, porovnáváním jejich dostupných funkcí, srovnáním cen a celkovým průnikem požadavků zákazníka s vlastnostmi nabízených produktů. Všechny tyto aplikace jsou též porovnány s aplikací „Aplikace pro interní potřeby veterinární ordinace Pelhřimov“ – dále nazývané pouze VetPel, vytvořené autorem této práce. Hlavním impulsem pro vytvoření zde prezentované aplikace VetPel je pak především předběžně provedená analýza trhu kombinovaná se znalostí současného stavu uchovávání dat ve veterinární ordinaci, které je z velké části reprezentováno stohy nepřehledných dokumentů a excelových tabulek. Díky znalosti prostředí se autor také uchyluje k závěru, že tato praxe není pouze výjimkou, nýbrž reprezentuje častý stav dokumentace a uchovávání dat nejen drobných veterinárních a zootechnických ordinací, ale také drobných živnostníků bez ohledu na oborovou příslušnost.

Motivace

Mojí motivací pro volbu této bakalářské práce byl především fakt, že patřím mezi stálé zákazníky pelhřimovské veterinární kliniky a vidím tak problémy spojené s absencí digitální administrace v praxi. V době, kdy digitalizací prochází jak státní správa, tak soukromý sektor, je vedení veškeré administrace v podobě stohů papírů a několika tabulek, vytvořených v aplikaci MS Excel, nejen nepraktické a časově náročné, ale s přihlédnutím k vzrůstající byrokracii také neekologické a dlouhodobě neudržitelné. Proto jsem se rozhodl skoncovat s těmito problémy a vytvořit desktopovou aplikaci umožňující snadnou a rychlou práci s veškerými dokumenty spojenými s výkonem veterinární praxe.

Hlavní přínos své práce pak vidím v časové úspoře, ke které uvedení této aplikace do praxe povede, dále také zpřehlednění již existujících záznamů a snadné nalezení dokumentů, jejichž hledání ve fyzické formě by zabralo několikanásobně více času. Existenci takové aplikace ocení nejen provozovatel, jenž tak ušetří čas i místo, ale i zákazníci, díky snadnému přehledu o svých zvířatech.

Cíle práce

Cílem práce je vytvořit desktopovou aplikaci sloužící k obsluze veterinární ordinace. Tato práce musí obsahovat databázi vhodnou pro uchovávání informací o zákaznících, zvířatech, medikaci, zákrocích proběhnuvších i budoucích, kalendář akcí a také grafické prostředí vhodné pro práci s touto databází. Cílem je poskytnout podrobné informace o každém zvířeti, včetně jeho fotografie, prodělaných nemocí, užívaných léků nebo podstoupených zákroků a umožnit tak snadnou a rychlou práci s daty, která nahradí velkou část dokumentace vedené v papírové formě. Kromě toho také umožňuje přesné plánování zákroků, návštěv pacientů a dalších obecných událostí probíhajících na půdě ordinace. Aplikace je tvořena tak, aby mohla být obsluhována člověkem s minimem technických znalostí.

1 Teoretická část

1.1 Současný stav

Současný stav vedení dokumentace je takový, že veškerá dokumentace je vedena buďto v papírové podobě a skladována v samostatné místnosti nebo v elektronické formě jakožto dokumenty programu Microsoft Excel. Současná podoba vedení dokumentace je neefektivní a nepřehledná. Jednotlivé dokumenty jsou rozděleny podle měsíců a v nich je třeba hledat příslušné zápisy k jednotlivým provedeným výkonům a podané medikaci, k návštěvám zákazníků a další podrobnosti o provozované praxi. Celkově je obtížné a zdlouhavé v případě potřeby najít jednotlivé záznamy.

Protože se jedná pouze o malou ordinaci s jednou lékařkou a jedním operačním sálem, jsou komerční řešení pro velké kliniky nevhodné a finančně nákladné. Proto jsem nabídl vytvoření jednoduché offline aplikace pro správu těchto dokumentů a rezervaci operačního sálu.

1.2 Zákaznické požadavky

Hlavním požadavkem bylo zajistit snadno ovladatelnou aplikaci s grafickým rozhraním, která dokáže trvale uchovávat informace o všem, co se v ordinaci děje. Jednoduchý design, nekomplikovaná obsluha, ale i snadná orientace v datech jsou prioritními body, které je nutné při návrhu dodržet. Obecně lze požadavky na tuto aplikaci rozdělit do několika kategorií, vyplývajících s rozhovoru se zákazníkem.

1.2.1 Požadavky z pohledu designu

Aplikace VetPel musí disponovat grafickým, nikoli pouze textovým rozhraním. Aplikace by měla – nikoli nutně musí – barevným designem odpovídat lékařskému prostředí, preferovány jsou tedy bílá a zelenomodrá barva s tradičním černým textem. Požadavky na font či velikost písma nebyly nijak podrobně specifikovány, jejich volba je tedy závislá čistě na výsledném celkovém dojmu a vhodnosti užití dle posouzení autora. Design má být pokud možno jednoduchý a jednotný, zaměřující se primárně na funkčnost, nikoli na vzhlednost. Preferována je však také snadná přehlednost obrazovky a rozlišení jednotlivých prvků, které se na ní nacházejí. Hlavní menu musí sdružovat všechny hlavní položky specifikované v požadavcích

na uchovávaná data a jejich primární funkce, které jsou následně rozčleněny na další podfunkce vztahující se k jednotlivým položkám.

Titulní strana aplikace pak má být opatřena interaktivním kalendářem, který umožňuje zobrazovat naplánované události na uživatelem zvolený den. Tato funkce zajišťuje okamžitý přehled o denním rozvrhu ihned při spuštění aplikace.

Aplikace také musí obsahovat možnost rychlého vyhledávání z titulní strany a to jak lidí, tak zvířat.

1.2.2 Požadavky z pohledu dat

Aplikace VetPel musí uchovávat relativně velké množství dat u několika hlavních subjektů.

Prvním, o kom je potřeba uchovávat data, je zákazník. Zákazník se registruje při první návštěvě veterinární ordinace a je o něm třeba evidovat následující údaje. Jméno a příjmení, v případě právnické osoby pak její název. Jednoznačný identifikátor, který správně určí právě jednoho zákazníka. Tímto identifikátorem bude rodné číslo, které je pro každého člověka s Českým občanstvím unikátní. V případě právnických osob pak bude namísto rodného čísla použito jejich IČO, podle kterého lze právnickou osobu spolehlivě identifikovat – IČO je ze své podstaty unikátní řetězec, který se navíc neshoduje s tvarem rodného čísla a není je tak možné zaměnit. U zákazníka je také možné nepovinně evidovat jeho telefonní číslo, případně email, kvůli budoucí komunikaci, plánování ošetření či prohlídek nebo kvůli uskutečnění nákupů zboží pro zvířata.

Další nedílnou součástí aplikace jsou pak samotná zvířata. U nich je povinně nutné uchovávat jméno zvířete, jeho druh, datum narození – které však v mnoha případech nemusí být zákazníkovi přesně známo, případně i číslo nebo kód čipu, pokud něčím takovým zvíře disponuje, nebo pokud jeho čipování ukládá zákon. U zvířete je také nepovinně evidována jeho medikace, seznam proběhnuvších i budoucích vyšetření a zákroků, fotografie zvířete umožňující identifikaci v případě ztráty nebo nálezu neznámého zvířete a také recepty a seznam vydaných opiátů, pokud existují. Zde je nutné upozornit, že vydání opiátů do rukou zákazníka je extrémně vzácné a recepty s modrým pruhem značící opiáty musí být evidovány v jejich originální formě po zákonem stanovenou lhůtu. Uchovávání jejich čísel v systému je tak spíše formalita umožňující snadnější orientaci v dokumentech než reálné uchovávání dat.

Poslední a do počtu položek největší hlavní položkou jsou pak samotné události spjaté s chodem ordinace. Jedná se o prohlídky, zákroky, plánované operace, čipování zvěře, statistické výzkumy a vyšetření. Tyto události jsou generovány pomocí kalendáře a uchovávány v databázi spolu s dalšími daty.

Kromě těchto hlavních položek jsou v databázi ještě uchovávány druhy zvířat a také druhy zákroků, které je možné vykonat.

1.2.3 Požadavky z pohledu funkčnosti

Aplikace VetPel musí být schopna pracovat s uchovávanými daty a provádět s nimi většinu základních i některých pokročilých operací.

U zákazníků je nutné mít možnost zaregistrovat nového zákazníka, upravit veškerá data u stávajícího zákazníka a smazat existujícího zákazníka. U zákazníka může dojít k změně povinných i nepovinných údajů, včetně jména i příjmení. Tato skutečnost je tedy reflektována možností změnit i povinné údaje, nicméně se nepředpokládá, že by tato možnost byla využívána ve větší míře. U každého zákazníka je veden seznam jeho vlastněných zvířat, která do tohoto seznamu mohou být volně přidávána, ale i odebírána. Údaje o zákazníkovi mohou být také exportovány do PDF.

Aplikace také musí nabídnout seznam zákazníků, řazených abecedně podle příjmení a následně v rámci příjmení také abecedně podle jména. Seznam zákazníků pak musí umožňovat vyhledávání podle jména, příjmení nebo rodného čísla. Ze seznamu zákazníků se poté musí být možné dostat na profil vybraného zákazníka.

U zvířat, stejně jako u zákazníků, existuje seznam zvířat obsahující všechna zvířata obsažená v databázi. Tato zvířata bude možné vyhledat podle jména, čísla čipu nebo zobrazit pouze vybraný druh zvířete. U jednotlivých zvířat musí jít zobrazit jejich profil přímo ze seznamu zvířat. Zvíře je nutné mít možnost zaregistrovat, upravit jeho údaje a zvíře také smazat. U zvířete musí být možné přidat či změnit jeho fotografii – samotné odstranění fotografie není nutné, je potřeba pouze v případě, je-li mazáno celé zvíře, v kterémžto případě se smaže spolu s celým profilem. Zvířeti bude také možné přidat recept na medikamenty, přidat či odebrat události týkající se zvířete, jako jsou prohlídky, operace nebo další nespecifikované zákroky. Zvíře musí být schopné v databázi existovat i bez majitele a to z několika důvodů – evidována jsou i nalezená či divoká zvířata, jejichž majitel není znám a které buďto čekají na předání do

útulku či jsou odchycena za statistickými účely, jako je například četnost vztekliny mezi liškami, nebo čipování divokého ptactva za účelem pozorování tažnosti. Profil zvířete je také možné vyexportovat do PDF pro další práci s daty či kvůli upozornění na pohřešování.

U aplikace VetPel je také nutné zhotovit kalendář, který bude evidovat události na jednotlivé dny. Kalendář bude zobrazen na titulní straně aplikace a bude mu věnována samostatná položka v menu. Na titulní straně bude možné zobrazovat jednotlivé dny a události k těmto dnům vztažené. U každé události pak půjde rozkliknout její detail. V položce menu „kalendář" pak bude možné přidat jednotlivé události, upravit je, případně i smazat. Veškeré události naplánované na daný den jde také hromadně vyexportovat do PDF a například vytisknout, pro lepší přehlednost. U jednotlivých událostí pak musí být možné napsat určitý popis, zvolit typ události, nastavit v jaké datum a čas se daná událost odehrává, může-li současně s ní probíhat další událost a přiřadit zvíře, kterého se daná událost týká. Bude možné také přidat do události majitele zvířete, nebude to ovšem povinnost.

Jedním z minoritních požadavků je také možnost napojení na obchodní aplikaci. Tento požadavek bude realizován prostým tlačítkem, které spustí soubor jemu zadaný v nastavení podle cesty k cíli.

U aplikace bude nutné provádět určitá nastavení, k čemuž bude sloužit stejnojmenná položka v menu. Bude nutné přidávat či odebírat druhy zvířat, stejně tak přidávat či odebírat druhy zákroků, například různé druhy operací, vyšetření provedených pomocí různých přístrojů nebo například zdravotní prohlídky. Bude také nutné nastavit cestu k obchodní aplikace a zvolit složku, do níž budou exportovány jednotlivé PDF soubory.

Kromě standardního ukládání změn v databázi po každé provedené akci je také nutné aby aplikace disponovala možností manuální zálohy a obnovy dat. K tomu poslouží položka „záloha" jejíž funkcí bude vytvořit či obnovit soubor .bak do vybrané obsahující veškerá existující data do vybrané složky a zabránit tak nechtěné ztrátě dat v případě poruchy.

Jedním z požadavků je také vypracování manuálu na obsluhu této aplikace, ten bude přiložen v samostatném souboru na disku s prací.

1.3 Současná řešení

Při hledání softwaru pro správu veterinární ordinace jsem provedl průzkum trhu, abych zjistil, zda existují i komerční varianty programů, které by bylo možné použít. Při zkoumání mě zajímalo, jaká je celková cena daného řešení, jaké funkce tato řešení nabízejí, jakou část nabízených funkcí využije a také jestli je možné případnou aplikaci rozšířit o další moduly, odborné zaškolení, či o správu softwaru danou firmou. Tyto informace jsou důležité především kvůli legislativním změnám v průběhu let, které nutí veterináře k podnikání dalších kroků, které dříve nebyly nezbytné, příkladem za všechny budiž zavedení povinného čipování psů a sním související vybavení na zavedení a identifikaci implantátů.

Během průzkumu trhu jsem nenarazil na žádný dostupný freeware, který by zvládl realizovat zákaznické požadavky byť ne v plném, tak alespoň dostačujícím rozsahu. Bezplatné řešení jsem tedy po průzkumu trhu zavrhl.

Zároveň ale některé firmy nabízejí svůj software na vyzkoušení zdarma, což může zákazníka přesvědčit o koupi, nicméně zkušební verze těchto programů jsou omezeny buďto délkou doby, po kterou je smí uživatel zkoušet zdarma nebo je uživateli ve zkušební verzi uzamčena část funkčnosti.

1.3.1 Vetfox

Domovská stránka: http://www.vetfox.cz (k 12.04.2021), společnost NOVIKO s.r.o. IČO 284 33 092 dále jen „NOVIKO“.

Společnost NOVIKO nabízí online řešení správy veterinární ordinace v podobě založení klientského účtu a následné správy přiděleného místa v jejich databázi v prohlížeči pomocí aplikace Vetfox. Zákazník nemusí stahovat ani instalovat žádnou aplikaci, veškeré změny probíhají přímo na straně Vetfoxu. Zákazník se pouze zaregistruje, vybere příslušný tarif a po zaplacení může začít používat aplikaci Vetfox.

Aplikace Vetfox nabízí po registraci také vyzkoušet tuto aplikaci na po dobu tří měsíců zdarma, respektive třiceti dnů ve verzi Premium. Bez registrace je také možné vyzkoušet demoverzi této aplikace, to je však funkčností silně omezené a jedná se opravdu pouze o reprezentativní demonstraci funkcí aplikace Vetfox.

Vetfox nabízí přehlednou tabulku návštěv zákazníků, obsluhu skladu léčiv, seznam zákazníků s jednoduchým profilem obsahujícím základní informace, možnost napojení na EET, Kalendář akcí a správu laboratorních testů. Aplikace v základu počítá se zadáváním obchodních transakcí je počítáním obratu. Není tak kompatibilní s dalším ekonomickým softwarem používaným samostatně.

Mezi výhody aplikace Vetfox patří její jednoduché a přehledné rozhraní, možnost napojení odkudkoli, kterou ocení především klienti s více pobočkami, kteří tak mohou používat jednotnou databázi a větší ordinace, které ocení možnost fakturace přímo z rozhraní.

Mezi nevýhody pak patří nutnost platit pravidelný měsíční poplatek podle zvoleného tarifu, nutnost stálého připojení k internetu, kdy v případě libovolné technické závady na připojení či na straně serveru zákazník zkrátka nemůže fungovat a také fakt, že drobné ordinace nevyužijí ani zdaleka všechny funkce této aplikace. Aplikace také neumí nahrávat fotografie zvířat, nedisponuje tak detailními profily zvířat a lidí jako aplikace VetPel a celkově je zaměřena spíše na ekonomické, než zdravotnické hledisko. Mezi další nevýhody pak patří vlastnit na email registrovaný účet online, kdy může dojít k útoku na danou stránku a klientská data tak mohou být odcizena či zneužita. Dále jsem nikde na stránkách nenalezl žádné informace o provádění záloh této databáze, z uvedených zdrojů tudíž nelze posoudit, do jaké míry jsou data chráněna před ztrátou. Další velkou nevýhodou je limitace návštěv klientů měsíčně, kdy pouze verze „PREMIUM“ nabízí neomezený počet návštěv měsíčně.

Aplikaci Vetfox bych rozhodně doporučil do velkých ordinací s více pobočkami, které jsou schopné a ochotné takový software financovat a které využijí veškeré jeho vlastnosti, za které platí. V případě použití pro malou ordinaci, která si ekonomický software řeší nezávisle, bych však hledal levnější řešení. Aplikace Vetfox navíc není desktopovou aplikací a nesplňuje tak jeden z hlavních zákaznických požadavků.

Aplikace Vetfox je poskytována ve čtyřech variantách a to „Lite“, „Pro“, „Pro plus“ a „Premium“. Jednotlivé verze se od sebe liší nejen cenou, ale také dostupnými funkcemi.

Verze Lite vyjde na 290Kč měsíčně, umožňuje spravovat nejvýše 150 klientů a zaznamenat až 200 návštěv za měsíc. Zákazníkovi nabídne 1GB prostoru na jeho data. Tato verze nedisponuje kalendářem, přístupem do laboratoře ani marketingovými nástroji.

Verze Pro stojí 490Kč měsíčně, počet klientů není nijak limitován, počet návštěv je omezen na 750 měsíčně, a zákazník může užívat až 1GB prostoru. Stejně jako ve verzi Lite, ani zde není dostupný kalendář, není dostupná správa laboratoře a ani marketingové nástroje.

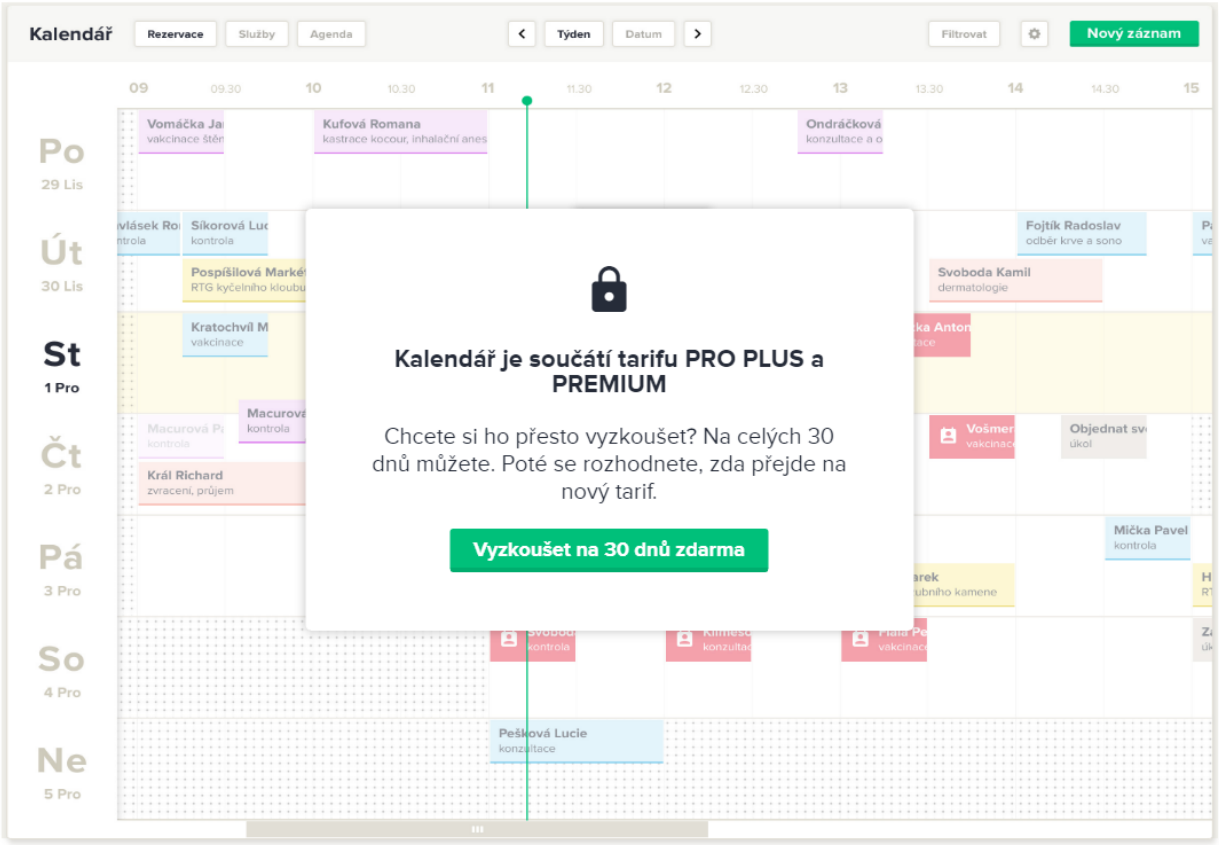
Verze Pro plus bude zákazníka stát 890Kč měsíčně, za což obdrží možnost spravovat neomezené množství klientů, až 750 návštěv měsíčně, kalendář, přístup do správy laboratoří a marketingové nástroje. Zákazník může užívat až 5GB prostoru.

Verze Premium za 1290Kč měsíčně poskytuje zákazníkovi neomezený počet klientů, neomezený počet návštěv, kalendář, správu laboratoří, rozšířené přehledy a 10GB úložiště.

Chce-li tedy zákazník používat elektronický kalendář s událostmi napojený na zbytek aplikace, musí tedy platit nejméně 890kč měsíčně. Celkové splnění zákaznických požadavků by pak vyžadovala verzi Premium, ani ta však nenabízí všechny požadované funkce a prostor. Naopak nabízí funkce jako napojení na EET nebo marketingové nástroje, které však klient nevyužije. Aplikace nabízí textový manuál na její obsluhu.

Za nadstavbových 10GB prostoru si zákazník připlatí 200Kč měsíčně, bez ohledu na zvolený tarif.

Servisní práce a zásahy ze strany provozovatele aplikace Vetfox, jsou-li nějaké požadovány, pak zákazníka vyjdou na 500Kč za každou započatou hodinu práce.



Obrázek 1 - Ukázka Vetfox

1.3.2 VetPro

Domovská stránka: http://medsoftsro.cz/vetpro-veterinarni-informacni-system/ (k 12.04.2021), Společnost MedSoft s.r.o. IČO: 037 88 270 dále jen „MedSoft“.

Společnost MedSoft nabízí desktopovou aplikaci VetPro v6.x jako plnohodnotné řešení pro drobné veterinární ordinace.

Aplikace nabízí dva měsíce užívání zdarma, poté musí klient zvolit jeden z nabízených tarifů. Užití verze zdarma není nijak technicky omezeno, zákazník může užívat veškeré funkce aplikace VetPro. Následně je možné vybrat některou z komerčních variant. Zákazník může buď zakoupit roční licenci za 1 400Kč, nebo koupit aktuální verzi programu VetPro s aktualizacemi zdarma za 4 850Kč. Součástí objednávky také může či nemusí být technická podpora společnosti MedSoft, která vyjde zákazníka na 1 500Kč ročně a není možné uhradit žádnou jednorázovou částku, která by technickou podporu zajistila po celou dobu užívání.

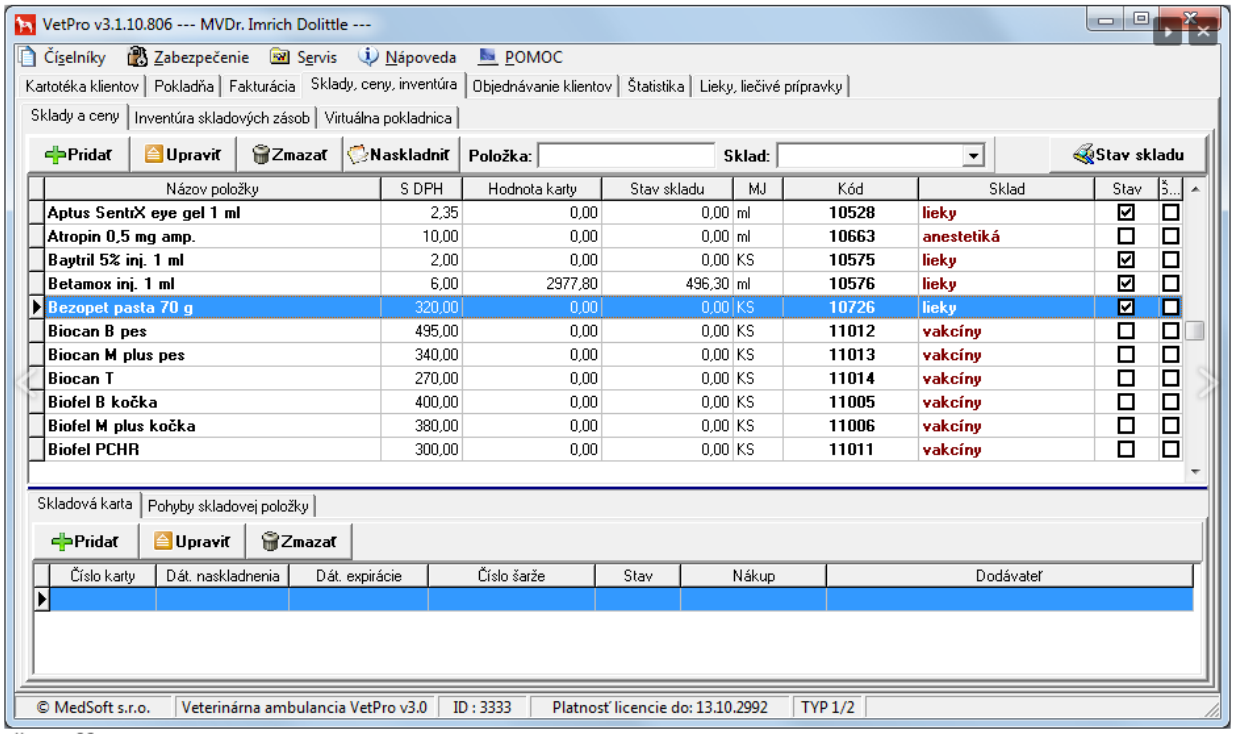
Aplikace VetPro nabízí několik hlavních funkcí. Evidenci klientů, evidenci zvířat, která lze přiřadit ke klientům, umožňuje zřídit galerii fotografií, zaznamenávat očkování a spravovat

pokladnu a sklad léčiv. VetPro také nabízí rozšíření pro odesílání klientských SMS s nastavitelnými upozorněními, tato služba pak zákazníka vyjde na 1,49Kč bez DPH za SMS, respektive 0,99Kč bez DPH za SMS, pokud si zákazník předplácí technickou podporu.¨

Mezi výhody aplikace VetPro jednoznačně patří fakt, že se jedná o aplikaci desktopovou, není tedy třeba být neustále připojen k internetu a zákazník může aplikaci spustit kdykoli. Aplikaci tak zle používat z více počítačů najednou, i pokud jsou offline. To se samozřejmě nevztahuje na odesílání SMS, které vyžadují internetové připojení. Další výhodou jsou velmi detailní možnosti nastavení vakcinace, nastavení medikace a výkonů. Aplikace je lokalizována do Českého i Slovenského jazyka. Další výhodou je lokální fotogalerie, která umožňuje identifikovat spravovaná zvířata. Aplikace VetPro je tedy vzhledem ke své ceně poměrně dostupným řešením pro začínající ordinace, které nemohou platit vysoké měsíční poplatky za jiný software.

Mezi nevýhody bych zařadil v první řadě zastaralost aplikace VetPro. Poslední aktualizace této aplikace pochází ze září roku 2017. Od té doby nelze dohledat žádné změny softwaru, není tedy možné garantovat, že software bude fungovat v souladu s nejnovějšími právními předpisy, ani že spolehlivě zajistí veškeré potřeby veterináře v roce 2021. Aplikace také nemá žádné další dostupné moduly upravující její funkčnost. Zákazník tedy zaplatí za používání věcí, z nichž část nepotřebuje, část není vůbec aktuální, a pokud narazí na problém, připlatí si dalších 1 500 Kč ročně, aby byl někdo ochoten jeho problém řešit. Aplikace sice nabízí správu EET a laboratorních výsledků, pro účely této práce jsou ovšem zbytečné. Nikde jsem nenalezl způsob, jak data z aplikace VetPro zálohovat, kromě tvrdého překopírování celé aplikace se vším všudy na záložní disk. Zákazník je tedy odkázán na důvěru v odolnost svého disku a softwaru, neboť hrozí snadná a neobnovitelná ztráta dat. Mezi nevýhody patří také zastaralý a nepřehledný design plný tabulek a všemožných nastavení, která jsou chaoticky uspořádaná a mají podobu excelových tabulek s podbarvením. Celý software pak svojí složitostí a robustností působí dojmem, jako by vypadl z dob Windows 95.

Software VetPro bych pro řešení svého problému rozhodně nezvolil, nejen že je cenově nevýhodný, ale také velmi nepřehledný a s omezenou funkčností. Software sice nabízí věci jako propojení s laboratorními formáty některých společností nebo napojení na EET, jeho kvality co se týká uchovávání uživatelského obsahu, však nejsou příliš dobré, design je zastaralý, záloha dat neexistující a ovládání zcela neintuitivní a zmatené.



Obrázek 2 - Ukázka VetPro

1.3.3 Další software pro správu veterinárních ordinací

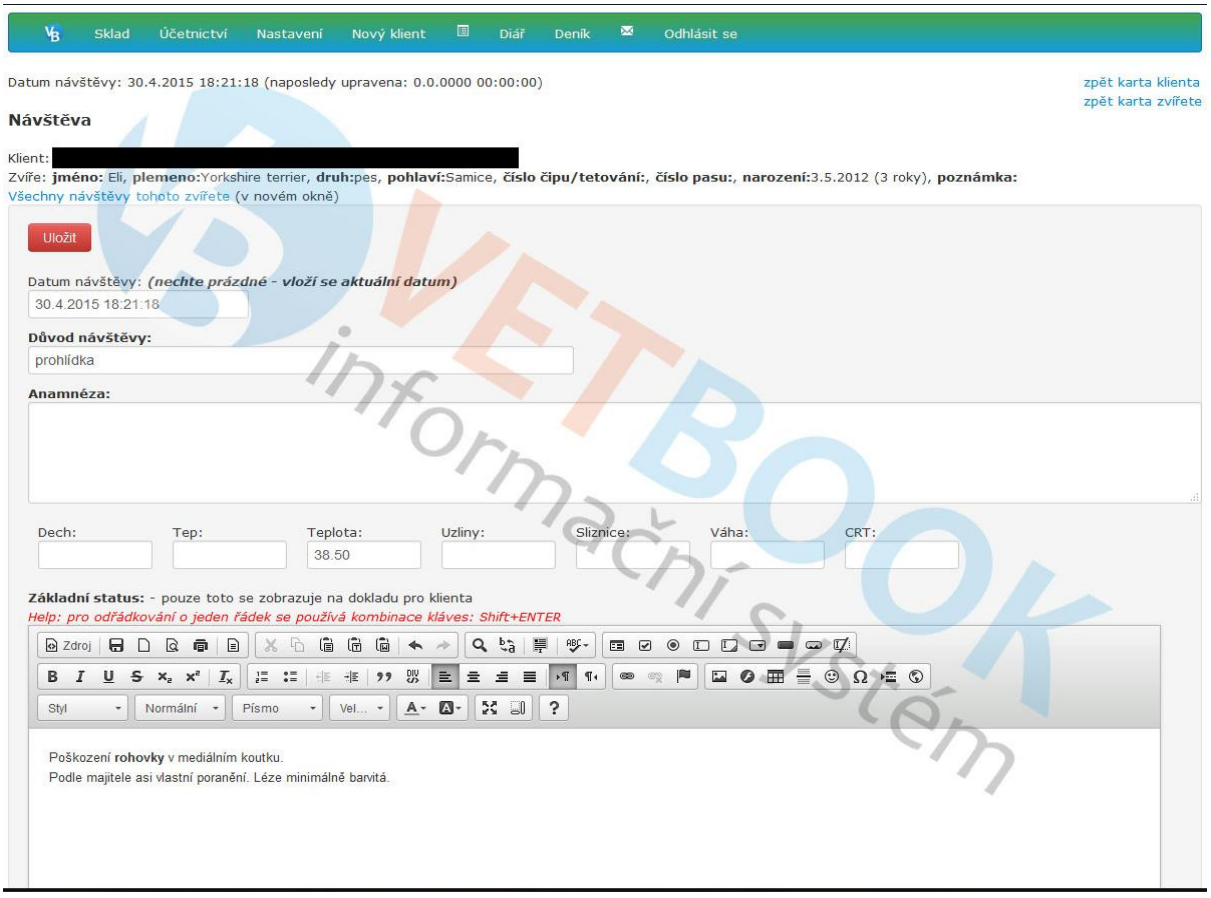
Při průzkumu trhu jsem také narazil na další firmy, které vznikly povětšinou během vytváření této práce. Některé – nikoliv všechny z nich dokonce splňují veškeré zákaznické požadavky a byly by tak vhodné pro implementaci a použití. Největší potíží těchto aplikací je však cena pohybující se v částkách, které si malá ordinace se dvěma zaměstnanci nemůže dovolit. Proto zde uvádím pouze sporadický přehled těchto softwarů, podrobná analýza jejich produktů není potřeba s přihlédnutím k cenové politice těchto společností.

Společnost BISI plus s.r.o. nabízí online řešení v podobě aplikace Veclis (https://www.veclis.cz/). Jedná se o cloudové řešení, které je schopné zajistit běh a zálohování velkého množství dat a zároveň zajišťuje prakticky všechny potřebné funkce, od evidence zákazníků a jejich zvířat, přes kalendář s možností plánování akcí a odesílání zákaznických SMS až po zaškolení personálu. Některé zákaznické požadavky sice nejsou obsaženy, ale s trochou snahy by bylo možné je zajistit pomocí dalšího nesouvisejícího freewaru. Bohužel, cena, která by poskytla adekvátní množství záznamů na měsíc je u této aplikace 3 750 Kč bez DPH měsíčně, kde navíc nejsou započítány ani ceny klientských SMS nad 100 odeslaných SMS, zaškolení pro práci s aplikací vyjde na dalších 3 000Kč, import předchozích dat na 1 500Kč a tak dále.Celkově je toto řešení cenově natolik nedostupné, že jsem jej zavrhl hned při prohlédnutí ceníku. Všechny ceny jsou uvedeny bez DPH.

Obrázek 3 - Ukázka Veclis

Veterinární informační systém Vetbook (https://www.vetbook.cz)–Informační systém Vetbook je jedno z mála komerčních řešení, které bych skutečně doporučil pro nasazení do praxe. Tato aplikace je moderní, snadná na ovládání, designově i uživatelsky přívětivá

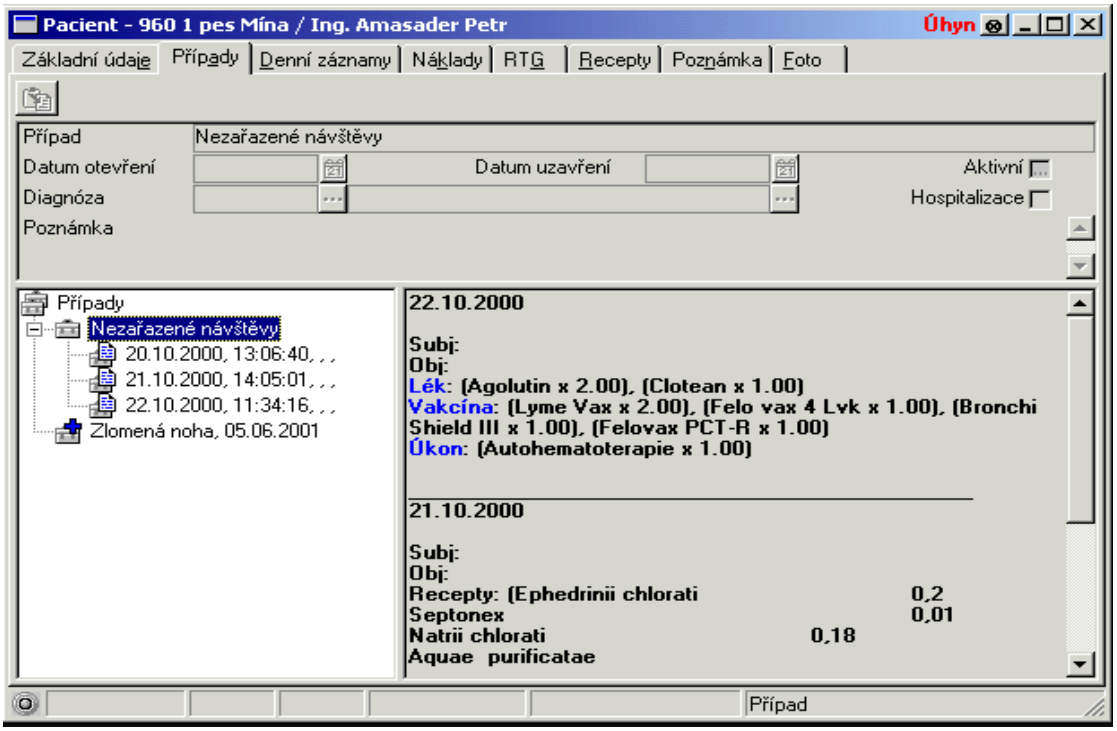
a obsahuje téměř veškeré zákazníkem specifikované požadavky, které je nutné zahrnout. Zároveň nabízí i velké množství nadstavbových vlastností, jako je podpora čteček čárových kódů, napojení na Registr petpasů ČR nebo evidence EET. Jedná se však o webovou aplikaci, která vyžaduje neustálý přístup k internetu a hlavní nevýhodou je pak cena. Dodavatel sice na svých stránkách uvádí cenu za používání 605kč s DPH měsíčně, nicméně v této ceně není zahrnuto používání rezervačního systému, které vyjde na dalších 121kč měsíčně. K dispozici má zákazník pouze 1GB prostoru, za každý další započatý 1GB pak připlácí dalších 50Kč bez DPH měsíčně. Větší databáze se tak může vyšplhat do řádů tisíců korun měsíčně, nemluvě o individuálních požadavcích, konzultacích a dalších IT službách, jejichž cena je v ceníku označena pouze jako „dohodou“. S přihlédnutím k standardním cenám IT služeb na trhu lze očekávat, že se tyto ceny budou pohybovat nejníže v řádech tisíců korun českých.



Obrázek 4 - Ukázka Vetbook

Winvet (https://www.winvet.cz/) – společnost je zastoupenaIng. Jaroslavem Matouškem a spadá pod již představenou společnost NOVIKO. Winvet představuje alternativní a pokročilé řešení k produktu Vetfox, navíc se jedná o desktopovou aplikaci.Základní cena za roční licenci se pohybuje v ceně 6 888kč bez DPH. K tomu je však třeba přičíst cenu za systémovou

podporu, bez níž se zákazník v praxi neobjede a jejíž cena je 4 728Kč na rok, servisní zásah v případě nutných změn, provádění úprav a dalších služeb se pak pohybuje v ceně 800kč za hodinu (respektive 400kč za hodinu při zaplacené systémové podpoře) + 9kč za každý kilometr cesty k zákazníkovi. Toto řešení je pro malé ordinace natolik nevýhodné, že se jedná spíše o alternativní obchodní model téže společnosti s podobnou aplikací.



Obrázek 5 - Ukázka Winvet

Výčet komerčních řešení není konečný, na trhu se nalézají i další zde neuvedená řešení, nicméně žádné z nich není vhodné nebo cenově dostupné pro řešení problému řešenému v této práci.

1.4 Vlastní řešení

Po provedení průzkumu trhu jsem se rozhodl přijít s vlastním řešením v podobě aplikace VetPel. Ta bude přizpůsobena přímo požadavkům vyplývajících z kapitoly 1.2.

1.4.1 Aplikace Vetpel

Aplikace VetPel je na míru vytvořená aplikace vzniklá pro účely této práce. Aplikace VetPel není vhodná pro obecné ani komerční užití, je vyrobena zakázkově přímo podle přání klienta a reflektuje tak pouze potřeby jedné konkrétní veterinární ordinace, nikoliv celého trhu.

Aplikace VetPel zajišťuje všechny požadavky uvedené v kapitole 1.2 a zaměřuje se na splnění těchto konkrétních požadavků. Další funkce, které komerční řešení v této oblasti nabízí, jsou ignorovány, neboť nejsou uvedeny v klientských požadavcích a je možné případně zakázkově takové funkce přidat. Aplikace bude vytvořena jako desktopová, bez nutnosti registrací, přihlašování, internetového připojení nebo odesílání dat neznámo kam a neznámo komu.

Aplikace bude uchovávat data lokálně a stejně tak vytvářet lokální zálohy, které bude moct zákazník následně uchovávat jak kdekoli ve svém počítači nebo libovolné přenosné paměti, ale také v jím zvolené cloudové službě. Aplikace tak zabrání nepřehlednému hromadění nepoužívaných a nepřehledných funkcí, umožní připojit k aplikaci software třetích stran, například EET nebo obchodní aplikaci. V konečném produktu bude kromě samotné aplikace dodána také podrobná uživatelská příručka a zdarma dostupný SQL server zajišťující funkčnost aplikace.

1.4.2 Porovnání aplikace Vetpel s komerčními řešeními

Výhodou komerčních aplikací oproti aplikaci VetPel je bezesporu výrazně rozšířená funkčnost, která spoléhá na používání jednoho softwaru na kompletní správu ordinace, včetně všech ekonomických záležitostí, jako je vedení účetnictví, vystavování EET, odesílání SMS nebo poskytnutí okamžitého servisu kdykoli na zavolání a některé z výše jmenovaných softwarů bych byl ochotný plně doporučit firmám, které jsou ochotné vydávat částky ve výši od jednotek do desítek tisíců ročně za bezproblémový chod jejich aplikace pro správu veterinární ordinace. Z obecného hlediska problematiky je zakoupení kvalitního komerčního řešení lepší, než použití aplikace VetPel, protože ta není designována pro obecné užívání všemi potenciálními zákazníky z tohoto oboru.

Naopak mezi nevýhody těchto řešení patří v první řadě cena, která se pohybuje často i v desítkách tisíc ročně za plný přístup k funkčnosti, důležité funkce skryté za „prémiové“ verze těchto programů, nedostatečný prostor pro data či omezený počet klientů v měsíci. Všechna tato omezení mají jediný účel – donutit zákazníka platit víc, a to jak za obecné odemčení dostupných funkcí, tak za případné na míru dělané moduly ale i základní servis. Financování těchto služeb je navíc často velmi nepřehledné a zakoupení jedné služby bez nějaké nadstavby ani neumožní správné užívání aplikace. Zákazník je pak zmatený a přikupuje vše, co vidí, jen aby jeho aplikace fungovala správně. V několika případech jsem také zjistil, že některé aplikace sice vytváří dojem konkurenčního prostředí, nicméně při bližším prozkoumání

jsou na sebe buďto napojené přes nějakou další firmu, nebo jsou například dceřiné společnosti téže firmy. Každá z těchto firem pak ve svých materiálech nabízí například přetažení dat z cizích softwarů za určitý poplatek, zákazník ale ve finále přechází k téže skupině firem a platí akorát další službu témuž subjektu, aniž by o tom bez zkoumání živnostenského rejstříku věděl. Komerční aplikace zároveň sice nabízí další funkce, které aplikace VetPel nenabízí, nicméně většinu těchto funkcí zákazník vůbec nevyužije a platí tak za služby, které nepotřebuje.

U žádné z dostupných desktopových aplikací jsem také nenalezl snadnou možnost zálohování veškerých dat a design většiny z nich je dnes již silně zastaralý. V konečném důsledku zákazník zaplatí tisíce a tisíce korun za služby, z nichž část nevyužije, část nefunguje podle jeho představ a poslední část si musí doplatit v prémiovém balíčku, balíčku servisních služeb nebo dalším jinak pojmenovaném lákadle na peníze. Možnost modifikovat tyto aplikace na míru pro daného klienta pak buďto není k dispozici vůbec, nebo cena takového na míru vytvořeného modulu výrazně převyšuje cenu samotné aplikace a pro zákazníka je výhodnější poohlédnout se po jiném softwaru, který zajistí například pouze tuto potřebnou funkci – například po samostatně implementované čtečce čipů.

Výhodou aplikace VetPel je konstrukce na míru. Aplikace je tvořená přesně podle zákaznických požadavků, splňuje tedy vše, co si zákazník přeje. VetPel je také díky své architektuře velmi flexibilní, co se týká úprav. V případě potřeby není žádný problém změnit design, funkčnost či způsob a obsah uchovávaných dat. Aplikace je vytvářena moderním způsobem a proto není problém její funkčnost rozšiřovat a měnit i bez zásahů do celkové konstrukce, které by mohli nějak narušit funkčnost aplikace. Zákazník tak může časem přijít i s další funkčností, kterou není problém dodat. Velkou výhodou je cena, která je prakticky nulová. SQL server lze sehnat zdarma, dokonce jeho stažení Microsoft umožňuje jako jednu z instalačních prerekvizit. Aplikace je také vytvořena zdarma v rámci této práce a kromě toho je zajištěna také bezplatná instalace a kompletní proškolení personálu pro práci s aplikací. Samotné zaškolení pak například u aplikace Veclis vyjde na 3 000kč. Zákazník také obdrží kompletní a srozumitelný manuál pro obsluhu této aplikace, a to jak v elektronické tak papírové formě. Další velkou výhodou aplikace VetPel je snadné zálohování, které je z komerčních projektů nabízeno pouze u webových aplikací. VetPel také používá místo na disku zákazníka, aplikace tedy není limitována ani prostorem, ani počtem měsíčních přístupů. Jediný limit je velikost disku, na kterém je databáze uchovávána. Aplikace VetPel je psána čistě a srozumitelně, v případě výskytu nějakého drobného problému není nikterak složité vydat opravný balíček během pár hodin.

Nevýhodou aplikace VetPel je, že je vyvíjena pouze jedním člověkem bez komerčních účelů, není tedy možné zajistit profesionální tým poskytující servis na úrovni technologických korporací s ekvivalentním vybavením a časovou flexibilitou. Vzhledem k faktu, že tento software však bude používán pouze v jedné ordinaci, navíc ve městě ve kterém sám vývojář žije, nejedná se o tak zásadní nedostatek, aby převážil výhody ve prospěch komerčních servisů, které jsou za správnou cenu ochotné přijet klidně v neděli v noci.

1.4.3 Zhodnocení použití aplikace VetPel

Po zvážení všech kladů a záporů, které se vyskytují jak na straně komerčních řešení, tak na straně aplikace VetPel jsem došel k závěru, že pro zakázkové použití na míru ve veterinární ordinaci v Pelhřimově bude lepší použít aplikaci VetPel než některé z výše jmenovaných komerčních řešení. Zákazník tak nejenže ušetří jednotky až desítky tisíc korun, ale také dostane aplikaci přesně na míru s veškerým možným servisem a úpravami, které si bude přát. Je pravdou, že jednoduchá aplikace vytvářená jedním člověkem nemusí v některých ohledech dosahovat stejných možností, jako aplikace vyvíjené velkými specializovanými firmami, nicméně v tomto případě jsou naplněna všechna přání zákazníka včetně funkcí, které standardně komerční řešení nenabízí a možné nevýhody tak výrazně převažují nad cenou a nepřizpůsobenou funkčností aplikací, které jsou nyní k dispozici na trhu.

2 Praktická část

V této kapitole jsou popsány jednotlivé části aplikace Vetpel, včetně použitých technologií, způsobu tvorby a jednotlivých komponent.

2.1 Použité technologie

Zde je výčet nejdůležitějších použitých technologií. Kromě zde zmíněných byly použity v minoritním množství i další technologie, jejich míra užití je však natolik minoritní, že je možné je pro účely této práce zanedbat.

Microsoft Visual Studio 2019

Microsoft Visual studio 2019 je vývojové prostředí od společnosti Microsoft dodávané vývojářům a umožňující vývoj různých druhů aplikací ve více jazycích. Umožňuje pracovat například v C, C++, C# ale i v Pythonu, který je pomocí těchto jazyků konstruovaný nebo v stále populárnějším objektovém jazyce Ruby, pro něž existuje do Visual Studia vlastní rozšíření. Tyto jazyky je také možné do jisté míry kombinovat. Visual Studio umožňuje vytvářet projekty od databází, přes desktopové aplikace, textové aplikace až po různé druhy skriptových projektů. Umožňuje používání knihoven, rozšíření, vytváření instalačních verzí aplikace, stahování dalších aplikací společnosti Microsoft nutných k běhu aplikací a tak dále. Microsoft Visual Studio je ideálním řešením pro účely této práce, neboť umožňuje snadno a přehledně vytvářet jednotlivé projekty a jejich funkce včetně předdefinovaných funkcí od společnosti Microsoft jako je vyhledávání nebo řazení.

Microsoft Visual Studio poskytuje z běžně dostupných vývojových prostředí nejlepší podmínky pro praktické dokončení této práce, včetně jejího grafického rozhraní, databáze, obslužných funkcí i všech dalších potřebných komponent. [1]

XAML

XAML, neboli Extensible Application Markup language je značkovací jazyk od společnosti Microsoft. Používá se k popisu Grafického uživatelského rozhraní a je založený na starším jazyce XML. Tento jazyk je často využíván v kombinace právě s níže zmíněným WPF pro tvorbu desktopových aplikací s grafickým rozhraním. XAML lze také použít pro tvorbu webových aplikací, za pomocí XBAP (XAML Browser Applications) a jedná se o preferovaný vývojářský nástroj, pokud jde o aplikace vyvíjené.[2]

WPF

Windows Presentation Foundation je knihovna tříd, která se používá pro tvorbu grafického rozhraní. Je přímým nástupcem Windows Forms, které si již od dob Windows Vista nepoužívají. Tato knihovna je součástí .NET frameworku a pomocí značkovacího jazyka XAML dokáže oddělovat vzhled a funkčnost aplikace. Takto je možné pomocí Xaml nadefinovat a nakreslit vzhled jednotlivých oken a jejich komponent, včetně tlačítek, tabulek, seznamů, popisků a dalších věcí nezbytných pro grafické rozhraní. Zároveň je také možné definovat některé logické operace, jako například které podmínky musí být splněny pro předání vstupu další vrstvě, tak jako je tomu například v této práci. WPF zajišťuje renderování uživatelského rozhraní popsaného právě jazykem XAML. WPF sjednocuje různé druhy grafik a typů uživatelských rozhraní, od vektorové grafiky, přes klasickou 2D grafiku ale umí pracovat také v prostoru s 3D modely. WPF také umožňuje vytvářet a zobrazovat animace, přechody různých grafik pomocí časovače, nebo vytvářet šablony které jsou následně aplikovány na více částí aplikace. [3]

V této práci je WPF použito právě pro tvorbu uživatelského rozhraní, jehož popis a některé logické funkce jsou umístěny v podprojektu Vetpel_Client.

C#

C# je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý společností Microsoft. C# je hojně využíván jak k tvorbě desktopových aplikací, tak aplikací webových a také k tvorbě webových stránek. Tento programovací jazyk nejlépe splňoval mé požadavky na vytvoření aplikace Vetpel, proto jsem jej vybral spolu s ostatními technologiemi pro tuto práci.

C# je moderní, snadno čitelný i strukturovatelný jazyk, který umožňuje snadno konstruovat velké kusy kódu bez nutnosti zabíhat do větších detailů jak tomu je u jazyků nízké úrovně. Při vývoji snadno poráží C i C++, na jejichž základech je zkonstruován. Spolu s jazykem Ruby je v současné době pravděpodobně nejlepším jazykem pro práci s objekty, především pak v silně objektově orientovaných projektech, což aplikace VetPel bez pochyby je. C# také umožňuje při použití dalších technologií například i věci, jako je definování tabulek v databázi. Přestože by bylo bezpochyby možné vyvíjet aplikaci VetPel na jiné platformě za použití jiných jazyků, C# je pro mě v tomto bodě jasnou volbou.[4][5]

.NET framework

.NET framework je v současnosti nejpoužívanější platforma pro osobní počítače. Vychází z původního souboru .NET technologií používaných jak pro tvorbu webových stránek (ASP.NET) tak pro tvorbu desktopových aplikací. Platforma .NET prošla za léta své existence řadou verzi, kdy jsou postupně rozšiřovány její možnosti. Pro Windows 10 je aktuální verze .NET 4.7 která je již od instalace systému jeho součástí. Umožňuje běh programů psaných v jakémkoli jazyce, který dokáže přeložit. Nejčastěji je používána pro jazyky C# a Visual Basic. Mluvíme-li o vývoji aplikací v C#, měli bychom zároveň vzpomenout i platformu .NET neboť .NET framework poskytuje všechny potřebné základní funkce k objektově orientovanému programování. Automaticky podporuje třídy, metody, vlastnosti, konstruktory, události a další věci nutné k vývoji moderních aplikací. Navíc umožňuje i částečné kombinování jazyků ve kterých je daná aplikace vyvíjena a výrazně tak usnadňuje práci programátora.

SQL

Structured Query Language je dotazovací jazyk používaný v práci s daty, často spojovaný s databázemi, ve kterých umožňuje snadnou práci.

Tento jazyk umožňuje vytvářet tabulky, upravovat je, mazat, plnit daty, zkrátka všechny základní funkce, včetně dotazů na obsahy tabulek.

Pro programy psané v C# je SQL ideální způsob tvorby databáze, navíc jsou jednoduše propojitelné například pomocí návrhového vzoru jedináček (singleton).

V této práci je databáze skutečně tvořena pomocí SQL, nicméně je zde použito pouze v rámci databáze. Jednotlivé entity jsou totiž definovány v C# a následně přeloženy do SQL pomocí níže popsané knihovny LINQ2DB

SQL je také možné zadat do skriptů a spouštět automaticky v dávkách, například kvůli automatickému nastavení serveru.[6]

LINQ2DB

LINQ2DB je knihovna určená pro práci v aplikacích psaných v C# umožňující několik zásadních věcí. Jedná se o objektové, silně typové rozhraní, které umožňuje vyhnout se konstrukcí databází i dotazů v SQL. Umožňuje specifikaci dotazů v jazyce C#, kde entity jsou specifikovány jako veřejné třídy s anotací pomocí LINQ2DB parametrů. V praxi to znamená,

že prakticky celá databáze je napsaná v C#, který je následně přeložen do SQL a pracuje v databázi bez jakéhokoli dalšího zásahu uživatele.

LINQ2DB se dále také stará o interpolaci parametrů, čímž chrání databázi před SQL injection a zvyšuje tak bezpečnost celé aplikace. Použití této knihovny tedy usnadňuje práci v objektově orientovaném programování, neboť vývojář může s jednotlivými entitami databáze zacházet jako s třídami a tímto způsobem je v celé aplikaci používat. [7]

Kalendář Microsoft

Jedná se o veřejně dostupný kalendář od společnosti Microsoft dostupný ve Visual Studiu. Podobný kalendář je pro vývojáře dostupný již od doby používání Windows forms – dříve používané knihovny tříd určené pro vytváření grafického rozhraní. Tento kalendář je předpřipravený tak, aby mohl zobrazovat dny v měsíci a přepínat mezi jednotlivými měsíci. Nic dalšího nezvládne. [8]

Ukazovat zde celý kód upraveného kalendáře by bylo příliš zdlouhavé, odkážu tedy pouze do umístění CalendarWindow.xaml.cs v projektu Vetpel_Client kde je základní struktura i s částí nadstavby umístěná a volně k prohlédnutí.

Tento defaultní kalendář není k dispozici v české jazykové variantě, první nadstavbovou prací tedy bylo celý kalendář přeložit, jak je podrobněji ukázáno v kapitole 2.2.4

Kromě toho také kalendář ve své základní podobě neumí vytvářet ani spravovat události, tato část práce je tedy také moje. Události jsou tedy vedeny jako vlastní položka databáze, u které se datum shoduje s nějakým datem v kalendáři a tam je tato událost zobrazena. Dále je vedle kalendáře separátně umístěn seznam událostí pro dané datum, protože takováto konstrukce na dva díly je snazší a přehlednější než snažit se soupis učinit součástí samotného kalendáře. Záznamy z databáze se pak zobrazují právě v tomto seznamu umístěném vpravo od kalendáře, kdy v kalendáři vyberete příslušný den, a v seznamu se zobrazí položky, kde je datum kalendáře shodné s datem událostí. Události jsou následně seřazeny podle času od půlnoci daného dne do půlnoci dalšího.

Veškeré další funkce kalendáře s ním sice přímo souvisí, ale kvůli obtížnosti provádění změn přímo v dostupném balíčku od Microsoftu jsou taktéž umístěné stranou a obsluhují data v databázi propojená s kalendářem. Události mají svůj repozitář umístěný v projektu Vetpel_DataAccess kde jsou dopodrobna specifikovány jednotlivé funkce událostí, od jejich přidání, zobrazení, změny či smazání. Definování nutných a dobrovolných vstupních dat je pak

uloženo v komponentně AppointmentRegistrationComponent.xaml.cs v projektu Vetpel_Clinet. Zde je také možnost přeskočit validaci schůzky – tedy pokud je na daný čas již naplánována jiná schůzka, nemohou se dvě schůzky standardně překrývat. Pokud je ovšem zaškrtnuto přeskočení validace – taktéž manuálně dodělávané, je možné umístit dvě události na překrývající se časy.

```
if (cb_skipvalidation.IsChecked == false)
{
    var conflictingExaminations = await DI.medicalExaminationService.ValidateAppointmentAvailability(GetAppointmentDate(), GetAppointmentEndDate(), ID);
    if (conflictingExaminations.Any())
    {
        DI.notificationProvider.NotificationEvent(Infrastructure.EventType.ToasterEvent, $"{Vetpel_Client.Resources.Common.AppointmentTimeNotAvailable} ({conflictingExaminations.First().Animal.
        return false;
    }
}

return true;
```

Obrázek 6 - Nekompletní kód přeskočení události

Funkce, které repozitář definuje jsou následně obsluhovány z projektu Vetpel_services, který se stará o realizování těchto funkcí.

BLOB – Binary Large Object

BLOB – Binary Large Object je datový typ uchovávaný v databázi bez bližší specifikace. BLOB se standardně používá dat, která neodpovídají žádnému ze základních datových typů, jako například hodnota, řetězec, či datum. Často bývá používán například pro obrázky nebo prezentace. Jeho výhodou je, že není identifikován databází. Zatímco text nebo hodnoty interpretuje databáze, BLOB dorazí do aplikace – v tomto případě do servisní vrstvy – přesně v takovém stavu v jakém byl uložen a o jeho interpretaci se stará samotná aplikace. V případě aplikace Vetpel je BLOB použit právě pro zobrazování obrázků jednotlivých zvířat. Na profilu zvířete je vyhrazené okno 400x260px, ve kterém se zobrazuje obsah BLOB položky přiřazené k danému zvířeti. Při vytvoření nového zvířete je mu přidělena prázdná BLOB položka nulové velikosti neobsahující nic. Při volbě fotografie je pak tato položka nahrazena Zvoleným obrázkem, který je uložen do databáze. Zdrojový obrázek tak není nutné uchovávat uložený v zdrojové cestě, například na ploše. Tento obrázek je pak propojen se zvířetem tak, že je odstraněn pouze v případě, je-li odstraněno samotné zvíře. V případě nahrazení jiným obrázkem je pak původní položka přepsána novou. Všechny zvolené obrázky jsou na straně aplikace také upraveny tak, aby se i při vyšším rozlišení zobrazovali zmenšené na zadané okno a zároveň zachovávali svůj přirozený poměr stran. [9]

```
using LinqToDB.Mapping;

namespace Vetpel_DB.Entities
{
    [Table(tableName: "Blob")]
    12 references | janfortelka, 74 days ago | 1 author, 2 changes
    public class Blob : BaseEntity
    {
        [Column(DataType = LinqToDB.DataType.Image)]
        3 references | janfortelka, 79 days ago | 1 author, 1 change
        public byte[] Data { get; set; }
    }
}
```

Obrázek 7 - Definování entity Blob

Návrhové vzory

Návrhové vzory představují v oblasti softwarového inženýrství sbírku obecných řešení jednotlivých problémů. Nejedná se však o žádné knihovny či kusy kódu, jsou to spíše myšlenkové konstrukty, které jsou aplikovatelné na sobě podobné problémy. Některé návrhové vzory jsou vázány na řešení specifických problémů v konkrétních jazycích, většina jich je však volně aplikovatelná v libovolném jazyce nebo alespoň ve skupině jazyků. Jedná se o různé šablony či popisy řešení problémů, které lze řešit tímto způsobem, aniž by musel autor vymýšlet celý koncept od začátku.

Návrhové vzory jsou velice oblíbené obzvlášť u objektově orientovaných projektů, kde popisují, jak mají vypadat jednotlivé interakce mezi třídami a objekty tak, aby splňovali požadované činnosti. Mezi často používané návrhové vzory patří například jedináček (singleton), sloužící k databázovému připojení. Celý program pracuje s jedním připojením k databázi místo toho, aby se při každé interakci či změně předávalo. Dalo by se říct, že singleton zajišťuje obecně přístup k jedné instanci nějaké třídy

Návrhový vzor přepravka (messenger) - messenger neboli přepravka je zvláštní druh třídy, která má tolik atributů, kolik potřebujeme přenést hodnot. Přepravka se na první pohled tváří jako normální třída, nicméně zde vstupuje do funkčnosti jeden zásadní faktor v podobě neměnných objektů. Instance přepravky jsou totiž právě neměnné objekty (immutable objects), artibuty jsou tedy veřejné konstanty a ty jsou nastaveny v konstruktoru tak, že už je dále nelze měnit. Dochází tedy k bezpečnému předání dat a zároveň lze ke konstantám snadno přistupovat, právě proto, že jsou veřejné.

Typickým použitím přepravky je situace, kdy například potřebujeme, aby nějaká metoda vracela více hodnot najednou. V takovém případě se stále vrátí jedna přepravka. Nebo když potřebujeme předat více hodnot jedním parametrem. Snadno si lze takovou situaci představit na příkladu bodu v rovině či prostoru, který má více souřadnic. Jedné instanci bodu přidělíme všechny souřadnice, následně pak ale předáváme pouze „bod“ bez souřadnic, které jsou převzaty z jeho instance. Nemusíme tak stále opakovaně zadávat jeho souřadnice, protože jsou již zabalené v přepravce tohoto bodu.

Návrhový vzor úložiště (repository pattern) – jedná se o návrhový vzor, který specifikuje repozitář nad každou entitou v databázi a zároveň definuje základní sadu metod CRUD – Create, Read, Update, Delete – neboli vytvoření, přečtení, nahrazení, smazání. Návrhový vzor úložiště výrazně usnadňuje oddělení servisní vrstvy od ostatních vrstev a umožňuje práci s daty i v momentě, kdy jsou data původně uložena jinak, než jak je s nimi dále nakládáno. Při změnách v SQL databázových tabulkách je třeba přepisovat celé kusy aplikací, které s těmito tabulkami pracují. Díky návrhovému vzoru úložiště je však možné se tomuto vyhnout, neboť s původním formátem uložených dat pracuje právě pouze úložiště. Cokoli dalšího, co uložená data používá, již komunikuje s úložištěm a není tak třeba přepisovat žádné další služby, které by měli problém se změnami databáze, protože jejich komunikace probíhá stále s úložištěm. Takto je možné například přejít z SQL na MySQL a nedělat přitom žádné zásadní změny v aplikaci. [10]

Mezi další technologie použité při tvorbě aplikace VetPel patří například Draw.io pro tvorbu databázových diagramů, balíčky a knihovny od společnosti Microsoft umožňující export do PDF, práci s databází nebo předcházení SQL injection.

2.2 Návrh aplikace

Tato část se věnuje návrhu aplikace VetPel, její struktuře, fungování jednotlivých řešení, ale také prerekvizitám nutným pro správný běh této aplikace.

2.2.1 Nutné prerekvizity

Aplikace VetPel ukládá data pomocí SQL databáze, která pro svůj běh vyžaduje funkční SQL server. Není vyžadován žádný konkrétní server, plně stačí ten, který je dodáván spolu s Microsoft Visual Studiem. Pro účely nasazení práce do praktického použití a na instalačním CD dodaném spolu s touto prací je pak použit Microsoft SQL Server 2019 express. Jedná se

o volně dostupnou verzi SQL serveru od společnosti Microsoft, plně kompatibilní s programy vytvořenými v Microsoft Visual Studiu. Tento server je možné zadarmo získat přímo ze stránek Microsoftu, není však dovoleno tento software dále modifikovat či přeprodávat. Tento software je v práci použit především z důvodu volné licence, není tak třeba platit za další program, který by byl nutný pro fungování aplikace VetPel.

2.2.2 Architektura tenkých vrstev

Kompletní návrh aplikace Vetpel není součástí jednoho projektu. Hlavním důvodem pro tuto volbu je nepřehlednost takového řešení, špatné oddělování jednotlivých vrstev aplikace a jejích funkcí, často zmatený či nedůsledně oddělený kód jednotlivých složek. Jelikož je nutné udržovat kód aplikace přehledný, funkčně čistý a jednoduchý na úpravy, jsou jednotlivé části aplikace rozděleny do pěti oddělených podprojektů, kde každé řešení obstarává jinou část funkčnosti. Je tak možné se v kódu rychle orientovat a zároveň přesně vědět, co je kde umístěné.

Jednotlivá řešení jsou strukturována tak, aby jejich rozdělení bylo intuitivní a rozdělovali jednotlivé prvky aplikace do funkčních celků. Například Vetpel_Client neobsahuje nic z databáze, Vetpel_Services zase nezajišťují grafický návrh uživatelského rozhraní. Takovéto dělení do projektů by mělo být u moderních aplikací standardem, neboť pokud například práci převezme po dokončení další vývojář nebo je předána jinému oddělení, je nutné, aby každá další osoba, která bude mít ke kódu přístup, byla schopná snadno pochopit jeho funkčnost a jednotlivé souvislosti.

2.2.3 Model tenký klient

Model „tenký klient“ je typ architektury, která předpokládá, že uživatelem viditelný klient bude na své straně provádět co nejméně operací a bude sloužit pouze pro komunikaci se servisní vrstvou a databází. Účelem tenkého klienta je zajistit přehledné, snadno pochopitelné a uživatelsky přívětivé rozhraní, které však slouží právě pouze za účelem rozhraní, maximálně pro použití některých logických funkcí a předání vstupů servisní vrstvě. V modelu tenký klient nedochází k ověření ani validaci údajů na straně klienta, k podrobným výpočtům ani k zápisům dat do databáze. O tyto věci se starají další vrstvy programu, klient slouží pouze o zjištění zajištění vstupů a předání dat do dalších vrstev.

Tento model také brání tzv. manipulaci vstupu a výpočtů, který byl v minulosti zaznamenán především v oblasti e-sportu, kde uživatelé upravovali výpočty na straně klienta tak, aby například změnili výsledné kolize objektů a odeslali tak programu pro ně výhodnější výsledky. Tím si pak zajišťovali soutěžní výhodu oproti hráčům s nepozměněnými klienty.

2.2.4 Vetpel_Client

Řešení Vetpel_Client obsahuje kompletní návrh grafického uživatelského rozhraní aplikace VetPel. Je rozděleno několika částí podle funkčnosti.

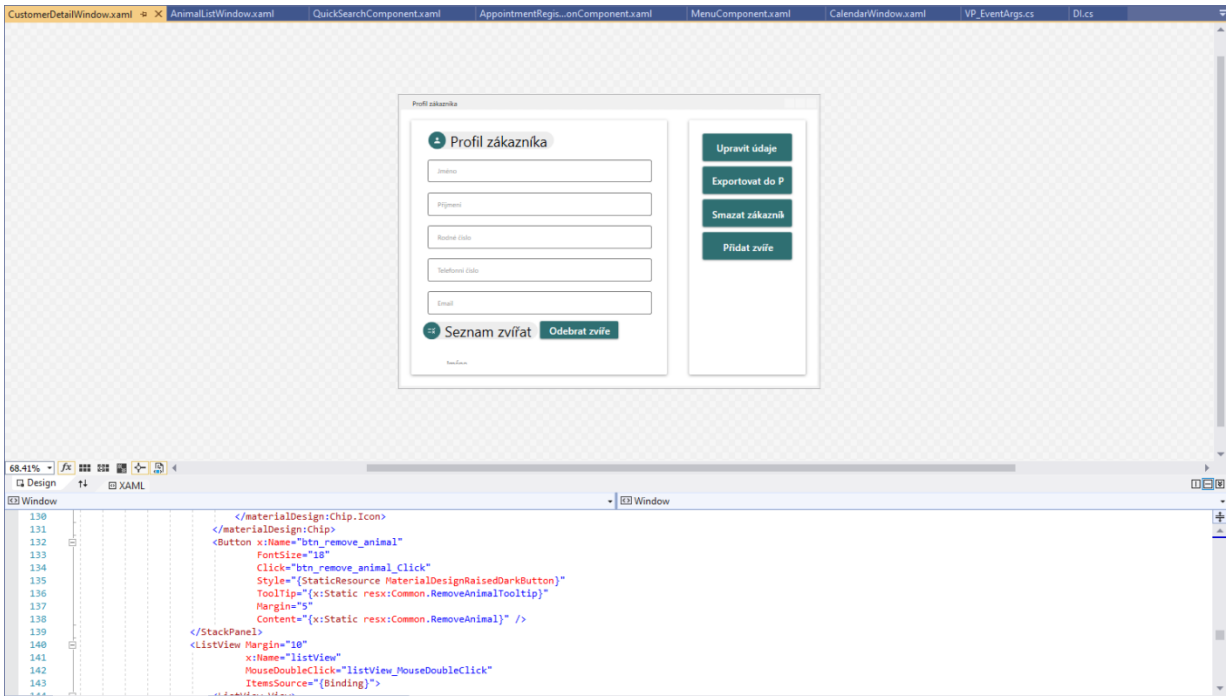
Vetpel_Client obsahuje identifikační certifikát vygenerovaný Microsoft Visual Studiem při releasu aplikace. Ten podepisuje tvůrce a vlastníka aplikace, určuje jméno počítače na kterém byla aplikace vytvořena a další náležitosti. Například „DESKTOP-8UI7U9M“ je název mého stolního počítače, na kterém byla většina aplikace vytvořena a vydán realease. „Jan_Fortelka“ je zase název mého notebooku. Certifikát tak určuje majitele aplikace a brání jejímu ukradení jiným vývojářem.

Další věcí, kterou Vetpel_Client obsahuje je App.config, který umožňuje nastavení některých prvků aplikace, jako je například čisté vytvoření databáze s předpřipravenými daty a další možnosti. Konfigurace také slouží k definování encodingu, či verze používaného .NET frameworku.

V tomto řešení také najdeme specifikaci balíčků použitých pro tvorbu aplikace VetPel. Ať už je to „linq2db“ database Access library, balíček ShowMeTheXAML nebo System.Security.Permission. Celkem obsahuje dvanáct balíčků použitých pro tvorbu této práce.

Vedle všech těchto položek pak Vetpel_Client obsahuje samotný design uživatelského rozhraní pro aplikaci VetPel. To je rozděleno do několika kategorií.

První z nich tvoří hlavní okna aplikace v položce windows. Zde najdeme všechny velké stránky umístěné v aplikaci Vetpel, ať už profil zvířete, profil zákazníka, či například kalendář. Každé okno obsahuje grafický design a kód v XAML popisující jeho parametry. [11]



Obrázek 8 - Ukázka XAML

Jako další najdeme složku resources, kde jsou umístěné překlady jednotlivých položek aplikace tak, aby se v uživatelském rozhraní zobrazovaly tyto položky v českém jazyce a nikoliv ve svém originální názvu zapsaném v kódu programu. Každý název má jednoznačný překlad, aby nedocházelo k záměnám a bylo možné aplikaci ovládat ryze českém rozhraní.

Name	Value	Comment
AnimalBirth	Datum narození:	
AnimalExamination	Přidat výkon	
AnimalIntervention	Přidat zákrok	
AnimalName	Jméno zvířete	
AnimalPrescription	Přidat recept	
AnimalProfile	Profil zvířete	
AnimalRemoveFromCustomer	Opravdu si přejete odebrat vybrané zvíře od uživatele?	
AnimalType	Druh zvířete	
AnimalTypeAdd	Přidat druh zvířete	
AnimalTypeDeleteConfirmation	Opravdu si přejete smazat vybraný typ zvířete? Tato akce může způsobit nekonzistentní data.	
AnimalTypeRequired	Položka druh zvířete je povinná	
AppointmentAnimalRequired	Vyberte zvíře, pro které vytváříte návštěvu	
AppointmentDateRequired	Položka datum návštěvy je povinná	
AppointmentEdit	Úprava prohlídky	
AppointmentEnd	Konec vyšetření	
AppointmentStart	Začátek vyšetření	
AppointmentStartRequired	Položka čas návštěvy je povinná	
AppointmentTimeInvalid	Zkontrolujte zaátek a konec schůzky	
AppointmentTimeNotAvailable	V zadaném rozmezí je naplánovaná další schůzka	
AppointmentType	Typ vyšetření	
AppointmentTypeRequired	Položka typ návštěvy je povinná	

Obrázek 9 - Překlad aplikace

Třetí položkou v projektu Vetpel_Client je pak jeho samotná infrastruktura, která umožňuje obsluhu celého klienta, umožňuje čtení repozitářů a běh jednotlivých událostí v rozhraní, jako jsou například upozornění na nevyplněnou položku – samotná validace vyplnění položky sice proběhne v servisní vrstvě, nicméně grafické upozornění na chybu je generováno na straně klienta.

Poslední částí projektu Vetpel_Client jsou pak komponenty jednotlivých obrazovek, tedy vyskakovací a ukotvená okna ve kterých je prováděna nějaká akce. Jedná se například o registraci nového zvířete, která probíhá přímo z hlavní strany bez změny okna, přidání jednotlivých akcí do kalendáře, nebo vytvoření nového typu prohlídky. Je zde také umístěna komponenta rychlého vyhledávání. Kromě toho zde najdeme také několik logických operací, jako je například určení, které položky jsou při registraci zákazníka či zvířete povinné a které je možno ponechat prázdné. Na obrázku níže lze snadno identifikovat, které podmínky musíme splnit, abychom mohli zvíře zaregistrovat.

```
private bool validateForm()
{
    if (string.IsNullOrWhiteSpace(input_name.Text))
    {
        DI.notificationProvider.NotificationEvent(Infrastructure.EventType.ToasterEvent, Vetpel_Client.Resources.Common.NameRequired);
        return false;
    }
    if (string.IsNullOrWhiteSpace(input_identification.Text))
    {
        DI.notificationProvider.NotificationEvent(Infrastructure.EventType.ToasterEvent, Vetpel_Client.Resources.Common.IdentificationRequired);
        return false;
    }
    if (dp_birth.SelectedDate == null)
    {
        DI.notificationProvider.NotificationEvent(Infrastructure.EventType.ToasterEvent, Vetpel_Client.Resources.Common.BirthDateRequired);
        return false;
    }
    if (dp_birth.SelectedDate > DateTime.Now)
    {
        DI.notificationProvider.NotificationEvent(Infrastructure.EventType.ToasterEvent, Vetpel_Client.Resources.Common.BirthDateisInFuture);
        return false;
    }
    if (cb_animal_types.SelectedItem == null)
    {
        DI.notificationProvider.NotificationEvent(Infrastructure.EventType.ToasterEvent, Vetpel_Client.Resources.Common.AnimalTypeRequired);
        return false;
    }
    return true;
}
```

Obrázek 10 - Povinné atributy zvířete

Z tohoto obrázku jasně vyplývá, že abychom mohli zvíře zaregistrovat, musí mít vyplněné jméno, musí mít vyplněnou identifikaci neboli číslo čipu (tato položka je povinná pouze dočasně, ve finální verzi bude tato povinnost odstraněna), musí mít vyplněné datum narození, datum narození nesmí být v budoucnosti, a druh zvířete nesmí zůstat prázdný.

Podobné postupy jsou zavedené také při registraci zákazníka nebo dojednání prohlídky. Podmínky pro jejich fungování je možné nalézt v příslušných komponentách.

2.2.5 Vetpel_DataAccess

Pomocí návrhového vzoru úložiště jsou v tomto podprojektu vytvořeny repozitáře jednotlivých entit uložených v databázi. Každý repozitář je uložen zvlášť, tak aby nedocházelo k míchání kódu a všechny entity a jejich základní funkce byly dostatečně odděleny. Již na první pohled je zřejmé, který repositář k čemu slouží. [12]

Máme zde AnimalRepository, který definuje operace nad zvířaty, včetně zobrazení jejich kompletního seznamu, přidání, odebrání nebo nahrazení informací.

```
public async Task<List<ReadAnimal>> GetAll()
{
    using (var db = new Database())
    {
        var query = from a in db.Animal
                    from c in db.Customer.LeftJoin(c => c.ID == a.CustomerId).DefaultIfEmpty()
                    from t in db.AnimalTypes.LeftJoin(t => t.ID == a.Type).DefaultIfEmpty()
                    where a.IsDeleted == false
                    orderby a.Name ascending
                    select new ReadAnimal
                    {
                        ID = a.ID,
                        Identification = a.Identification,
                        BlobId = a.BlobId,
                        TypeId = a.Type,
                        Name = a.Name,
                        CustomerId = c.ID,
                        BirthDate = a.BirthDate,
                        Type = t.Name,
                        Customer = new Vetpel_DataContract.Models.Customer.ReadCustomer
                        {
                            Name = c.Name,
                            BirthNumber = c.BirthNumber,
                            Email = c.Email,
                            ID = c.ID,
                            Phone = c.Phone,
                            Surname = c.Surname,
                            Fullname = c.Name + " " + c.Surname
                        }
                    };
        return await query.ToListAsync();
    }
}
```

Obrázek 11 - Zobrazení seznamu zvířat

AnimalTypesRepository – definuje operace nad typy zvířat

BackupRepository – definuje funkce umožňující vytvoření a nahrání zálohy již existující databáze.

```
public void Backup(string path)
{
    using (var db = new Database())
    {
        var command = db.Connection.CreateCommand();
        command.CommandType = System.Data.CommandType.Text;
        command.CommandText = $"backup database ["
            + Path.GetFullPath(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, @"..\..\..\Vetpel_DB\Vetpel_DB.mdf"))
            + "] to disk='"+ Path.GetFullPath(Path.Combine(path, "Database.bak")) + "' WITH FORMAT";
        command.ExecuteNonQuery();
    }
}

2 references
public void Restore(string path)
{
    using (var db = new Database())
    {
        var command = db.Connection.CreateCommand();
        command.CommandType = System.Data.CommandType.Text;
        command.CommandText = $"USE master; ALTER DATABASE ["
            + Path.GetFullPath(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, @"..\..\..\Vetpel_DB\Vetpel_DB.mdf"))
            + "] SET SINGLE_USER WITH ROLLBACK IMMEDIATE; RESTORE DATABASE ["
            + Path.GetFullPath(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, @"..\..\..\Vetpel_DB\Vetpel_DB.mdf"))
            + "] FROM DISK = '" + Path.GetFullPath(path) + "' WITH FILE = 1, NOUNLOAD, REPLACE, STATS = 10";
        command.ExecuteNonQuery();
    }
}
```

Obrázek 12 - Vytvoření a nahrání zálohy

BlobRepository – zajišťuje funkce pro obrázky

CustomerRepository – definuje funkce týkající se zákazníka

ExaminationTypesRepository – repozitář věnovaný typům prohlídek

MedicalExaminationRepository – repozitář definující operace nad kalendářem

```
public async Task<int> Create(ReadMedicalExamination data)
{
    using (var db = new Database())
    {
        var id = await db.InsertWithInt32IdentityAsync(new MedicalExamination
        {
            AppointmentDate = data.AppointmentDate,
            AppointmentEndDate = data.AppointmentEndDate,
            Description = data.Description,
            Type = data.TypeId,
            AnimalId = data.Animal.ID
        });
        return id;
    }
}
```

Obrázek 13 - Vytvoření události v kalendáři

PrescriptionRepository – definuje operace s recepty

SettingsRepository – definuje funkce pro hledání cest k ukládání PDF a k obchodní aplikaci

Klient vrstva následně předává veškerá uživatelem zadaná data servisní vrstvě, která je dále zpracovává.

2.2.6 Vetpel_DataContract

Projekt Vetpel_DataContract slouží ke specifikaci datových modelů a obsahuje návrhový vzor přepravka. Jsou zde uloženy veřejné třídy pro přepravu jednotlivých parametrů v přepravce, jako například vytvoření nebo čtení prohlídek, zákazníků a zvířat. DataContract zajišťuje samotnou přepravu dat mezi rozhraním a databází. [13]

2.2.7 Vetpel_DB

Podprojekt Vetopel_DB obsahuje samotnou databázi aplikace VetPel. Obsahuje jak MDF soubor, ve kterém jsou uložena data v SQL. Dále také obsahuje LDF log soubor, který si ukládá proběhlé operace v databázi a pomáhá při jejím ukládání a zálohování. Tento soubor je kritický pro správné fungování databáze.

Ve Vetpel_DB jsou uloženy všechny entity potřebné pro fungování databáze, všechny definované v C# pomocí LINQ2DB. Tento projekt tedy obsahuje tabulky Animal, AnimalTypes, BaseEntity, Blob, Customer, ExaminationTypes, MedicalExamination, Prescriptions a Settings. Všechny tyto tabulky jsou důsledně popsané v kapitole 2.3 a proto jim v této části nebude věnována další pozornost.

Dále také obsahuje celou infrastrukturu databáze sloužící pro její inicializaci a nastavení, tedy položky Database, DatabaseInit a LinqToDbSettings. Funkce těchto položek je podrobně rozepsána v kapitole 2.3.4 a 2.3.5 kde se nalézá i ukázka jejich kódu. Tato infrastruktura zajišťuje, že není nutné při instalaci spouštět žádné další nastavovací skripty, ručně zadávat tabulky do serveru ani provádět další složité úpravy. To výrazně usnadňuje instalaci a celkové zprovoznění aplikace uživatelem.

2.2.8 Vetpel_Services

Vetpel_Sevices, neboli servisní vrstva, se stará o provádění operací definovaných v jednotlivých repozitářích, tak, aby docházelo ke správné komunikaci mezi uživatelským rozhraním a databází.

Má na starost například validaci dat, tedy je-li telefonní číslo skutečně zadáno ve tvaru telefonního čísla, což je v této práci ošetřeno pomocí regulárního výrazu. To zajišťuje validator_service.cs, kde jsou tyto výrazy definovány, a to jak validace telefonního čísla, tak validace emailu.

V AnimalService najdeme služby provádějící definované operace se zvířaty. Služba přebírá operace definované v repozitáři AnimalRepository a zajišťuje jejich provedení, včetně dalších operací, jako je vyhledávání nebo změna obrázku zvířete, která se stará i o odstranění původního obrázku, tak aby se neuchovávalo více fotografií zvířete.

V CustomerService jsou zase služby spjaté se zákazníky, včetně přiřazení zvířete zákazníkovi, odebrání zvířete zákazníkovi, ale i kód zajišťující, že je-li odebrán celý zákazník, jsou odebrány i jeho zvířata, prohlídky a recepty, aby v databázi nezbývala zbytečná data.

MedicalExaminationService zajišťuje běh funkcí dodělávaných pro microsoft kalendář a umožňuje přidávání, mazání nebo změnu jednotlivých prohlídek v kalendáři. Stejně tak se stará o zobrazení zaznamenaných událostí v listu událostí vedle kalendáře podle zvoleného data v kalendáři. Události jsou seřazené v daný den podle času.

```
public async Task<List<ReadMedicalExamination>> GetAll()
{
    using (var db = new Database())
    {
        var query = from me in db.MedicalExamination
                    join a in db.Animal on me.AnimalId equals a.ID
                    from c in db.Customer.LeftJoin(c => c.ID == a.CustomerId).DefaultIfEmpty()
                    join at in db.AnimalTypes on a.Type equals at.ID
                    join et in db.ExaminationTypes on me.Type equals et.ID
                    where me.IsDeleted == false && a.IsDeleted == false
                    orderby me.AppointmentDate ascending
                    select new ReadMedicalExamination
                    {
                        ID = me.ID,
                        AppointmentDate = me.AppointmentDate,
                        AppointmentEndDate = me.AppointmentEndDate,
                        Type = et.Name,
                        TypeId = me.Type,
                        Description = me.Description,
                        Animal = new Vetpel_DataContract.Models.Animal.ReadAnimal
                        {
                            Name = a.Name,
                            Type = at.Name,
                            TypeId = at.ID,
                            CustomerId = c.ID,
                            ID = a.ID,

                            Customer = new Vetpel_DataContract.Models.Customer.ReadCustomer
                            {
                                Name = c.Name,
                                Fullname = $"{c.Name} {c.Surname}"
                            }
                        }
                    };

        return await query.ToListAsync();
    }
}
```

Obrázek 14 - List událostí vedle kalendáře

PdfExportService má na starost vytváření a zápasit do PDF souboru. Je zde definováno, jaký bude mít vyexportovaný soubor název, podle toho, zda je exportován profil zvířete, profil zákazníka, nebo události na daný den. Dále se tato služba stará o vytvoření PDF stránky ve formátu A4, jeho uložení, a o následný zápis dat.

PrescriptionService pak vytváří a aktualizuje předpisy pro jednotlivá zvířata.

Servisní vrstva tak připravuje data pro uživatelské rozhraní, zpracovává data z repozitářů a připravuje data pro repozitáře.

2.2.9 Regulární výrazy

Regulární výraz je řetězec určující povolené, požadované či naopak zakázané znaky a jejich kombinace v textu, který je regulárnímu výrazu podroben. Slouží například pro filtrování, vyhledávání či kontrolu tvaru jiných řetězců.

V aplikaci VetPel jsou regulární výrazy použity pro dva účely – prvním z nich je ověření telefonního čísla zákazníka, které musí být ve tvaru právě devíti číslic. Při registraci či změně údajů o zákazníkovi je zadaný řetězec přezkoumán, vyhovuje-li regulárnímu výrazu a v případě že ne, je zadavatel upozorněn, že zadané telefonní číslo neodpovídá zvolenému formátu. Pokud je telefonní číslo ve shodě s pravidly stanovenými regulárním výrazem, číslo je úspěšně změněno.

Druhým příkladem použití regulárních výrazů v aplikaci VetPel je automatická kontrola emailu, který musí být zadán ve tvaru „text@text.text”. Tato kontrola je na rozdíl od kontroly telefonního čísla zajištěna automatickým výrazem nabízeným společností Microsoft k ověřování emailů.

Tyto regulární výrazy je možné nalézt v „ValidatorService.cs“ v projektu Vetpel_Services.

2.2.10 Bezpečnostní upozornění

Aplikace Vetpel je vybavena validací mazání dat, při každém smazaném zvířeti, události nebo zákazníkovi je tedy nutné potvrdit, že si uživatel skutečně přeje smazat danou položku z databáze. To zajišťuje ochranu proti náhodným výmazům záznamů, které je třeba udržovat po delší dobu. Uživatel je také upozorněn, pokud zadává neexistující formát telefonního čísla, nevyplnil některou z povinných položek nebo se snaží zadat neplatná data.

2.3 Návrh databáze

Při návrhu databáze bylo potřeba dbát na všechna požadovaná data a vytvořit model, který by umožnil jejich efektivní uchovávání a zároveň zabránil tvrdému propojení dat „navždy“ u jednotlivých položek, aby bylo možné je snadno editovat a měnit spojení mezi nimi. Například u zvířete může snadno dojít ke změně majitele, je tedy vhodné, aby zvíře nebylo navázáno na existenci konkrétního zákazníka. Stejně tak vyšetření musí být funkční samo o sobě, pro případ, že by například zákazník přišel s jiným zvířetem. Pak by bylo nevhodné mít událost napojenou na konkrétní zvíře tak, aby nešla změnit. Je proto vhodné vytvořit událost, která se následně propojí se zvířetem a je možné jí snadno místo něj propojit s jiným zvířetem.

V databázi také mohou existovat zvířata bez majitele, je tedy nutné aby existence majitele nebyla nutnou podmínkou existence zvířete. Proto je v profilu zákazníka možné připojit k němu nějaké zvíře a zase jej odebrat.

Databáze obsahuje následující tabulky:

BaseEntity – základní rodičovská entita, ke které se odkazují všechny další. Slouží k zavedení entit jako celku, v diagramu ani ve funkcích nijak nefiguruje. Jedná se o takový můstek mezi entitami zavedenými v C# a SQL tabulkami pro potřeby knihovny LINQ2DB která disponuje základními atributy jako Datum vytvoření, informací jestli je na ní napojená entita smazaná,

a ID. Z BaseEntity ostatní třídy atributy, které mají společné.

Customer – zákazník – uchovává informace o zákazníkovi, jeho jméno, příjmení, rodné číslo/ičo, telefonní číslo a email.

Animal – zvíře – U zvířete jsou uchovávána data o jeho jméně, druhu, datu narození, identifikační číslo čipu, datum úmrtí, ID připojeného zákazníka, BLOB ID, které zajišťuje propojení s fotografií a medicaments – seznam užívané medikace.

Settings – nastavení – Tato tabulka obsahuje pouze dvě položky – nastavení cesty k obchodní aplikaci a nastavení cesty pro export PDF. Ostatní položky nabídky nastavení jsou umístěny v samostatných tabulkách

MedicalExamination – zdravotní prohlídka – jedná se o jednotlivé události v kalendáři, které jsou spojené s konkrétním zvířetem. Tato tabulka uchovává popis dané události, ID zvířete, které se jí účastní, Typ prohlídky, počáteční a koncový čas a datum prohlídky a také validaci

zajišťující, že se dvě události nepřekrývají ve stejný čas. Validaci je možné manuálně zrušit a umožnit tak kolizi událostí ve stejný čas.

Prescription – recept – uchovává údaje o jednotlivých receptech vydaných na dané zvíře. Obsahuje ID zvířete, číslo a případný popis receptu a validaci pro uchovávání po dobu pěti let.

AnimalTypes – druhy zvířat – Obsahuje dostupný seznam druhů zvířat. Ty je možné přidávat a ubírat v položce nastavení. Druh zvířete je volen jak při registraci zvířete, tak při vytváření události v kalendáři.

ExaminationTypes – druhy vyšetření – umožňuje snadnější kategorizaci vyšetření a snadnější vytváření výkazů, díky uchovanému záznamu o tom, co se při dané události dělo. Druhem vyšetření může být například odběr krve, běžná prohlídka, kastrace nebo vyšetření na ultrazvuku. Typ vyšetření je povinnou položkou při vytváření jednotlivých událostí.

Blob – Tato tabulka uchovává fotografie zvířat. Fotografie je propojena se zvířetem již při jeho vytvoření.

2.3.1 Databáze pomocí LINQ2DB

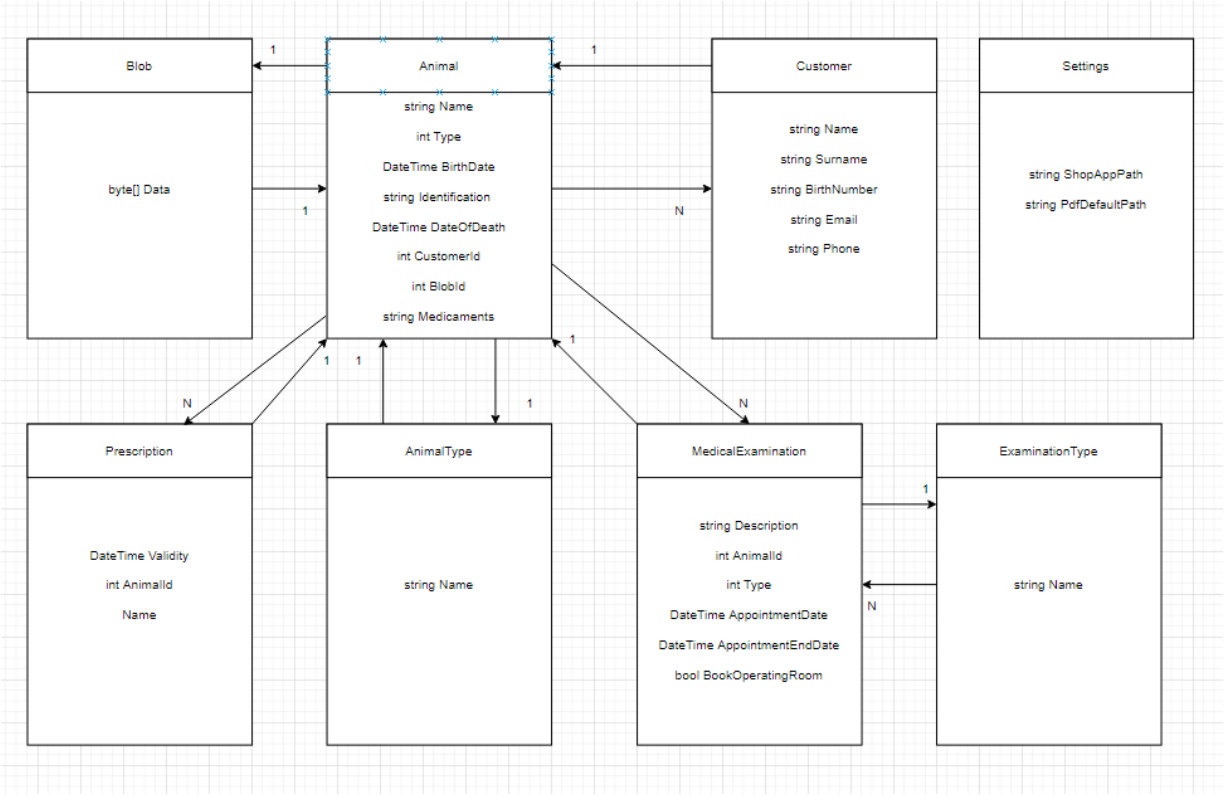
Díky využití knihovny LINQ2DB není databáze v zdrojovém kódu VetPel specifikována pomocí SQL, nýbrž pomocí C#. Knihovna pak sama převádí tabulky a funkce specifikované v C# do SQL. To zajišťuje snadné a srozumitelné tvoření databáze bez nutnosti používat dlouhé a nepřehledné selecty, joiny a další věci.

```
namespace Vetpel_DB.Entities
{
    [Table(tableName: "Animal")]
    13 references
    public class Animal : BaseEntity
    {
        [Column]
        15 references
        public string Name { get; set; }

        [Column]
        16 references
        public int Type { get; set; }

        [Column]
        12 references
        public DateTime BirthDate { get; set; }

        [Column]
        12 references
        public string Identification { get; set; }

        [Column]
        2 references
        public DateTime? DateOfDeath { get; set; }

        [Column]
        13 references
        public int? CustomerId { get; set; }

        [Column]
        4 references
        public int BlobId { get; set; }

        [Column]
        3 references
        public string Medicaments { get; set; }
    }
}
```

Obrázek 15 - Entita zvíře definovaná v C#

2.3.2 Model databáze

Model je vytvořen pomocí Draw.io – draw.io je internetová služba umožňující tvorbu různých druhů diagramů online. K dispozici je na internetové adrese draw.io Tento diagram neobsahuje entitu BaseEntity, která funguje pouze v rámci databáze a s každou další entitou je propojena tak, že jí poskytuje atributy pro fungování a informace o ní.



Obrázek 16 - Diagram databáze

2.3.3 Soubory MDF a BAK

Databáze je udržována v lokálním souboru typu .MDF umístěném v cestě C:/vetpel. Tento soubor je součástí instalačního balíčku a je nutné jej překopírovat do tohoto umístění. V budoucí verzi je počítáno s automatickým generováním cesty k souboru, v současné verzi práce je však potřeba udělat toto ručně.

Soubor .BAK je zcela běžně používán k vytváření záloh dokumentů a jejich plnému obnovení. V informačních technologiích je používán například programem AutoCAD, který slouží na průmyslových školách pro výuku.

Při zvolení možnosti „záloha“ v hlavním menu aplikace VetPel vybere uživatel cílovou cestu pro zálohování databáze. Aplikace VetPel následně zabalí celý obsah databáze do jednoho .BAK souboru, který vegeneruje ve zvolené cestě. Tento soubor je možné libovolně přesouvat, například jej zálohovat pomocí cloudové služby nebo umístit na externí médium, aby v případě poškození pevného disku nedošlo ke ztrátě dat.

Při obnově zálohy není nutné, aby byl soubor .BAK umístěn ve stejné cestě, do které byl uložen. Nemusí mít ani stejný název, stačí zachovat příponu. Své zálohy tak může uživatel

například označit datem pořízení a obnovit tak databázi z libovolného časového bodu. Stačí zvolit cestu k požadovanému .BAK souboru a data současně uložená v databázi budou nahrazena těmi ze zvoleného souboru.

2.3.4 Inicializace databáze

Samotná inicializace databáze při instalaci a prvním spuštění probíhá díky kódu obsaženému v projektu Vetpel_DB v Databaseinit.cs. Zde se při prvním spuštění vytvoří veškeré tabulky a náležitosti, po instalaci tedy není potřeba vytvářet na serveru žádné tabulky ani zadávat žádná nastavení. Díky LINQ2DB se také vyhneme zadávání SQL příkazů a můžeme pracovat rovnou v C#. Inicializace databáze tedy vypadá následovně:

```
private void InitializeDB(Database db)
{
    var dbSchema = db.DataProvider.GetSchemaProvider().GetSchema(db);
    if (!Tables.All(x => dbSchema.Tables.Select(y => y.TableName).Contains(x)))
    {
        dbSchema.Tables.ForEach(delegate (TableSchema tableSchema)
        {
            if (tableSchema.TableName != "database_firewall_rules")
            {
                db.DropTable<object>(tableSchema.TableName);
            }
        });

        db.CreateTable<Customer>();
        db.CreateTable<Animal>();
        db.CreateTable<Settings>();
        db.CreateTable<MedicalExamination>();
        db.CreateTable<Prescription>();
        db.CreateTable<AnimalTypes>();
        db.CreateTable<ExaminationTypes>();
        db.CreateTable<Blob>();
    }
}
```

Obrázek 17 - Inicializace databáze

Kromě toho se také spustí nastavení LINQ2DB které zajišťuje spojení mezi SQL databází a zadáním v C#. Toto nastavení také pomáhá s automatickým napojením na přítomný server, na kterém tyto tabulky vytvoří. Uživatel tak nemusí dělat nic, pouze zajistit přítomnost MDF souboru a LDF logu v C:/Vetpel překopírováním z instalační složky. Toto nastavení je možné najít v LinqToDbSettings.cs

```
public class LinqToDbSettings : ILinqToDBSettings
{
    0 references | janfortelka, 85 days ago | 1 author, 1 change
    public IEnumerable<IDataProviderSettings> DataProviders => Enumerable.Empty<IDataProviderSettings>();

    0 references | janfortelka, 85 days ago | 1 author, 1 change
    public string DefaultConfiguration => "SqlServer";
    0 references | janfortelka, 85 days ago | 1 author, 1 change
    public string DefaultDataProvider => "SqlServer";

    1 reference | janfortelka, 85 days ago | 1 author, 1 change
    public LinqToDbSettings()
    {
    }

    0 references | janfortelka, 79 days ago | 1 author, 2 changes
    public IEnumerable<IConnectionStringSettings> ConnectionStrings
    {

        get
        {
            yield return
                new ConnectionStringSettings
                {
                    Name = "Northwind",
                    ProviderName = "SqlServer",
                    ConnectionString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\VetPel\\Vetpel_DB.mdf" +
                    "; Integrated Security = True"

                };
        }
    }
}
```

Obrázek 18 - Konfigurace připojení databáze

2.4 Testování

Aplikace Vepel je pro testování vybavena dvěma módy, které je možné nastavit v App.config. pomocí RecreateDB.

Je-li toto nastavení zapnuté na na hodnotu „true“ bude aplikace vždy při novém spuštění obnovena do výchozí podoby a obsahovat bude pouze předehraná data. Tento režim slouží k testování funkčnosti, neukládají se v něm změny provedené v databázi a každý start aplikace je zcela nová inicializace databáze. Toto nastavení umožňuje snadné testování se stále stejnými existujícími daty.

Nastavení RecreateDB na hodnotu „false“ pak zajistí, že bude vždy docházet k uložení každé dílčí změny v databázi a ta bude po ukončení aplikace uložena. Toto je defaultní nastavení pro instalační balíček a je to také nastavení, na kterém běží odevzdaná verze aplikace VetPel.

Prvotní testování aplikace probíhalo vždy s novými daty, neboť v raných fázích vývoje docházelo k častým chybám, které bylo třeba odladit a zaplňovat vždy databázi manuálně od začátku by bylo časově velmi neefektivní.

Výstavba aplikace probíhala směrem od vytvoření základního klienta, který obsahoval pouze grafický návrh a nedisponoval žádnými funkcemi. Následně došlo k vytvoření databáze

a jejímu naplnění daty. V této fázi jsem začal přidávat jednotlivé funkčnosti aplikace Vetpel a jednu po druhé odlaďovat. Prvně bylo nutné zobrazovat zákazníky a zvířata. Vytvořil jsem tedy seznam zákazníků i zvířat a u obojího zavedl abecední třídění. V této chvíli ještě nebylo možné jednotlivé položky propojovat, existovaly pouze samostatně. Následně jsem vytvořil profily uživatelů i zvířat, protože již sice existovaly v databázi, nicméně jenom jako položky s jednotlivými vlastnostmi, které nebylo možné do detailů zobrazit. Poté jsem přidal jednotlivé funkce jako export do PDF, nahrávání fotografií, přidávání receptů a další. V této chvíli bylo možné implementovat přidělení zvířete k zákazníkovi a jeho odebrání. Nakonec přišel kalendář a úpravy do takové míry, aby bylo možné přidávat jednotlivé události navázané na zvířata i lidi a odladění aplikace.

Aplikace byla důkladně testována během každé z těchto instancí a byly odstraňovány všechny známé chyby, které v tu chvíli odstranit šly. V současné chvíli aplikace disponuje několika kosmetickými nedostatky, které však nelimitují její funkčnost.

2.4.1 Zkušební data

Aplikace VetPel je vybavena předgenerovanými zkušebními daty umístěnými v projektu VetPel_DB v DatabaseInit.cs. Tato data zajišťují, že nebude vygenerována zcela prázdná databáze, kterou bude nutné pokaždé naplnit. Databáze se tak při prvním spuštění naplní několika předpřipravenými zákazníky, několika zvířaty, typy zvířat, typy vyšetření, ale také předpřipravenými událostmi. Tyto předgenerovaná data je možné při praktické implementaci snadno odmazat, nicméně pro účely testování je praktické ponechat alespoň nějaká výchozí data. Tato data nereprezentují žádné reálné osoby ani jejich zvířata a jsou přidána čistě za účelem testování aplikace.

2.4.2 Řešené problémy

V průběhu vývoje i testování jsem narazil na nepřeberné množství problémů, které bylo potřeba odstranit. Zde je výčet těch, které považuji z hlediska svého oboru za zajímavé.

Vývoj funkcí kalendáře Microsoft – Kalendář od společnosti Microsoft je dostupný pouze v základní variantě zobrazující jednotlivé dny, týdny a měsíce v roce. Veškeré rozšiřující funkce, od vytváření událostí, přes překlady do Českého jazyka až po zavedení dobrovolné validace kolize událostí jsem musel dotvořit manuálně. Následně bylo nutné tento kalendář také propojit s databází a vytvořit možnost přidávat vyšetření přímo z profilu zvířete.

Připojení fotografií zvířete - Uchovávání fotografií daného zvířete a využití SQL Binary Large Object (BLOB) se ukázalo být velkou výzvou spojenou hned s několika problémy. V první řadě se fotografie zvířat automaticky zvětšovaly do plné velikosti, místo přizpůsobení vyhrazenému oknu 400x260px. Tento problém se ukázal být způsobený nesprávně provedenou implementací nahrání dat, kdy aplikace nedokázala přizpůsobit obraz.

Zajištění komunikace mezi klientem a databází – Jelikož aplikace pracuje právě na modelu tenký klient, neprobíhají žádná ověření na straně klienta. Bylo tedy nutné vytvořit celou separátní servisní vrstvu, umístěnou mezi databázi a klienta, která obstarává obojí.

Zajištění nadstavby kalendáře Microsoft. Kalendář fungující pro WPF je pouze základní konstrukce umožňující přepínání mezi jednotlivými měsíci. Veškeré funkce přesahující tuto základní funkčnost bylo nutné dodělat.

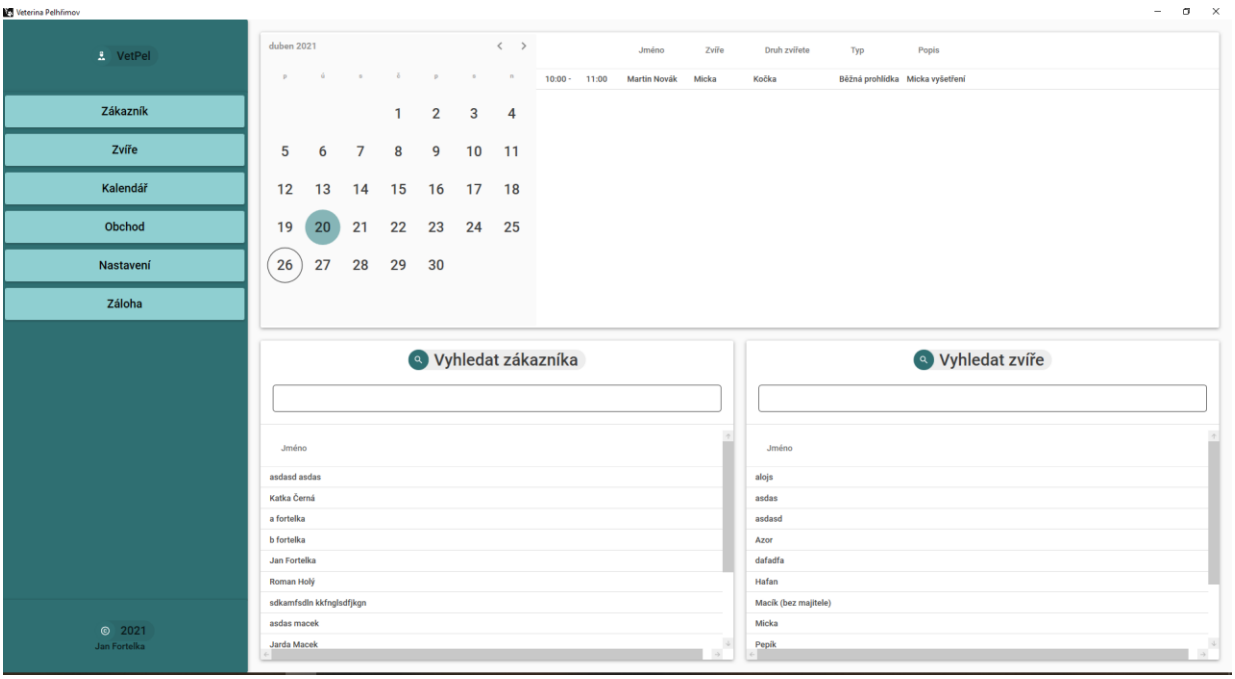
2.4.3 Známé chyby

Mezi známé chyby této aplikace patří neodstraněná závada při instalaci, kdy instalační program není z neznámého důvodu schopný správně rozbalit soubor MDF a LDF log při automatické volbě umístění během instalace. Tato chyba byla dočasně odstraněna zvolením manuální cesty k těmto dvěma souborům, konkrétně do „C:/Vetpel“, kam je po instalaci nutné překopírovat oba zmíněné přiložené soubory. Samotný instalátor při tvorbě automatické cesty hledal oba soubory kdesi v roamingu v datech uživatele, přičemž tuto cestu nebylo ani bez chybového loggeru možné dohledat. Takto instalace sice vyžaduje jeden krok navíc, nicméně nejedná se o krok, který by běžný uživatel nezvládl, nebo který by aplikaci nějak zásadně omezoval na funkčnosti. Plánem do budoucna je tuto chybu odstranit a umožnit správné rozbalení obou souborů z instalačního balíčku.

Mezi další známou chybu patří také fakt, že kalendář poskytování společností Microsoft nesahá nekonečně do minulosti. Pokud se tedy uživatel rozhodně záměrně zadávat stará data, se kterými neumí kalendář Microsoft pracovat, dojde k selhání aplikace. Tato chyba nelze jednoduše odstranit, problém je na straně poskytnutého kalendáře. V něm jde manuálně listovat zpět i do nevalidních datumů, omezit tedy rok narození nějakým nejnižším číslem tedy není možné, protože uživatel se i tam může prolistovat k dávnému datu, které nelze použít. Případů takovéto chyby bude ovšem zanedbatelné množství, zvířat narozených před začátkem Microsoft kalendáře příliš nebude a v případě potřeby jim lze přidělit první validní datum.

2.5 Provoz aplikace

K 22.04.2021 běží aplikace VetPel v aktuální verzi 1.0.1. určené k odevzdání. V této verzi již fungují všechny požadované funkce a jsou odladěny všechny zásadní chyby, které vývoj aplikace VetPel provázely. Součástí této aplikace je také uživatelská příručka s kompletním návodem k obsluze umístěná na CD spolu s aplikací VetPel.



Obrázek 19 - Titulní strana aplikace VetPel

2.5.1 Aktuální funkce

V aktuální verzi jsou zprovozněny všechny zákazníkem definované funkce obsažené v kapitole 1.2 – Zákaznické požadavky.

Aplikaci je v současné době již možné nainstalovat podle přiložených instrukcí a využívat následující funkce:

Přidávat zákazníky, upravovat údaje o zákaznících a odebírat zákazníky

Přidávat zvířata, měnit údaje o zvířatech, odebírat zvířata a přidělovat je jednotlivým zákazníkům. Přidat či změnit fotografii zvířete, přidat zvířeti prohlídku či recept

Exportovat do PDF profil zákazníka, profil zvířete a kalendář na aktuální den

Vyhledávat mezi zákazníky i zvířaty

Přidávat, upravovat a odebírat události do kalendáře, včetně přeskočení jejich validace

Zálohovat i nahrát zálohu aktuální databáze

Nastavit druhy zvířat, druhy prohlídek, zvolit cestu pro export PDF a zvolit cestu k obchodní aplikaci

2.5.2 Budoucí rozšíření funkčnosti

Odevzdávaná verze aplikace VetPel není finální verzí a nereprezentuje konečný produkt. Jedná se o verzi 1.0.1, která opravuje chyby původní verze 1.0, například nemožnost přidat více nových zákazníků při jednom zapnutí aplikace či automatické zmenšení zvoleného obrázku zvířete.

V budoucnu je plánováno rozšíření a aktualizování funkcí aplikace VetPel tak, aby odpovídala vždy aktuálním zákonným požadavkům na uchovávání soukromých dat a lékařských záznamů.

V plánu je možnost vytisknout předvygenerovaný dokument při registraci zákazníka, na kterém již budou vyplněné údaje zadané do systému a který bude představovat informovaný souhlas o poskytnutí a správě osobních údajů zákazníka, jeho právech a povinnostech ale i informovat o podmínkách služby.

Dalším velkým přídavkem bude možnost identifikace elektronických čipů v souvislosti s povinným čipováním psů na území ČR. Tato implementace by však dalece překonala rozsah a náročnost bakalářské práce a v této verzi není implementována. Jedná se o identifikaci elektronického kódu z mikročipu zalitého do silikonu a umístěného na krk pod kůži zvířete.

Aplikace bude později doplněna o instalační wizard, který umožní volbu cesty k instalaci i volbu cesty k databázovému souboru.

Implementace dalších funkcí je závislá na budoucích požadavcích a není v tuto chvíli známá.

2.5.3 Opravné balíčky

V době odevzdání této práce je k dispozici opravný balíček verze 1.0.1, který je součástí této bakalářské práce. Následující opravné balíčky, které se věnují jakýmkoli dalším nalezeným chybám či nedostatkům budou následovat až v době po odevzdání této práce a proto je zde jen strohý výčet několika dosud nalezených nedostatků:

Nemožnost zadat předvolbu u telefonního čísla

Automatická validace emailu z neznámého důvodu občas povolí email bez zadání domény prvního řádu

Některé tabulky upozorňující na nevhodně zadaná či nezadaná data vypisují nesprávná upozornění

3 Závěr a zhodnocení výsledků

Praktické výsledky této práce bych osobně zhodnotil kladně. I přes opakované odložení a nesnáze při vývoji je nyní aplikace VetPel ve funkčním a použitelném stavu, umožňuje vykonávat všechny požadované funkce a je zbavena všech chyb které bránily její funkčnosti.

Vývoj aplikace VetPel mi přinesl spoustu nových a užitečných dovedností a znalostí, výrazně rozšířil mé obzory na poli vývoje desktopových aplikací a v teoretické části i o analytické a marketingové znalosti.

Aplikace je již nyní připravena k nasazení do praxe, které proběhne někdy ke konci roku 2021 po plánované rekonstrukci veterinární ordinace. Tato práce tedy splnila všechny účely své existence, od účelů učebních až po účely praktické. Přestože jsem musel několikrát měnit použité technologie i celý původně zamýšlený koncept práce, troufám si usuzovat, že tato práce je nejen přehledně a moderně zpracována jak po stránce kódu, kde je rozdělena do přehledných projektů oddělujících jednotlivé funkčnosti a umožňujících snadné budoucí úpravy, tak po stránce designu a funkčnosti, kde každá přítomná funkce má své opodstatnění. Žádná funkce není bezpředmětná, žádný kód není samoúčelný. Některé komponenty sice byly použity jako již nabízené, například vyhledávání nebo základní kalendář Microsoft, nicméně i na těchto komponentách jsem odvedl velký kus práce při jejich úpravách pro potřeby veterinární ordinace a při rozšíření jejich funkčnosti.

Při implementaci jsem musel opustit původní koncept klienta připojujícího se na databázi umístěnou online i koncept webové aplikace. Zde prezentované řešení je tak až třetí v pořadí, které jsem vyzkoušel a které se ukázalo být velmi efektivním. Ano, aplikace se potýká s několika drobnými problémy jako dosud neodhalená příčina špatného rozbalování MDF souboru, nicméně tyto chyby nejsou nic, co by nešlo poměrně snadno manuálně obejít a v budoucnu opravit.

Povedlo se mi tedy dosáhnout hlavního cíle této práce a nalézt efektivní nástroj pro správu veterinární kliniky? Ano, a to jak mezi komerčními řešeními, která byla vyškrtnuta z důvodů uvedených výše, tak především v podobě aplikace VetPel, která svou funkčností může konkurovat produktům, jejichž cena se často pohybuje až v desítkách tisíc korun.

Seznam zkratek

BLOB – Binary Large Object

IČO – Identifikační číslo osoby

LDF – LIN Description File

MDF – Media Descriptor File

PDF – Portable Document Format

SQL – Structured Quary Language

WPF – Windows Presentation Foundation

XAML – Extensible Application Markup Language

Seznam použité literatury

[1] ŠÍMA, František a VILÍMEK, David. Visual Studio.NET praktické programování krok za krokem. ISBN 978-80-247-6213-5 [online] [Obecný popis Visual Studia] Dostupné z: https://books.google.cz/books?id=2oy7ajFS3aIC&printsec=frontcover&key=AIzaSyDlPfI89JdFhWBVsMVsavVo6aNh057xlTc#v=onepage&q&f=false

[2] JAMES, Buddy a LALONDE Lori - PRO XAML with C# - [online] [Postupy pro grafický design] Dostupné z: https://oiipdf.com/download/1515

[3] UNDERWOOD, Carry. WPF for Universities [online] [Obecné principy použití WPF] Dostupné z: https://bisquare.kaist.ac.kr/svimpw93mow5/18-miguel-batz-1/read-9781533008237-wpf-for-universities.pdf

[4] VIRIUS, Miroslav. C# 2010 Hotová řešení. Brno: Computer Press, 2012. ISBN: 978-80-251-3730-7.

[5] MAREŠ, Amadeo. 1001 tipů a triků pro C# 2010. Computer Press, 2011. ISBN: 978-80-251-3250-0.

[6] MOLINARO, Anthony. SQL kuchařka programátora. Computer Press, 2009. ISBN: 978-80-251-2617-2.

[7]Linq2db manuál. [online] Dostupné z: https://linq2db.github.io/articles/get-started/install/index.html

[8] Microsoft kalendář. [online] dostupné z: https://docs.microsoft.com/cs-cz/dotnet/desktop/wpf/controls/calendar?view=netframeworkdesktop-4.8

[9] BUYYA Rajkumar, VECCHIOLA Christian, SELVI S Thamari. Mastering Cloud Computing. 2013. [online] Dostupné z: https://books.google.cz/books?id=QQxRAgAAQBAJ&printsec=frontcover&hl=cs&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[10] PECINOVSKÝ Rudolf. Návrhové vzory: 33 vzorových postupů pro objektové programování. Brno: Computer Press, ISBN: 978-80-251-1582-4.

[11] Návod společnosti Microsoft pro užití XAML. [online] Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/xaml/xaml-basics/get-started-with-xaml?tabs=windows

[12] Vzor úložiště. [online] Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design

[13] KUDVENKAT. WCF DataContract and DataMember. [online] Dostupné z: https://www.youtube.com/watch?v=RPgTKzSGcKY&ab_channel=kudvenkat

[14] Jak implementovat návrhový vzor Úložiště. [online]Dostupné z: https://cze.small-business-tracker.com/how-implement-repository-design-pattern-c-689459

[15]VOBORNÍK, Petr. Základy programování v C# [online] Dostupné z: https://www.youtube.com/playlist?list=PLxTqV9i8bnb8tanSMrno74A4tvOF7eIqT

[16]ANGELSIX. WPF UI programming (C#). [online] Dostupné z: https://www.youtube.com/playlist?list=PLrW43fNmjaQVYF4zgsD0oL9Iv6u23PI6M

Seznam obrázků

Obrázek 1 - Ukázka Vetfox 17
Obrázek 2 - Ukázka VetPro 19
Obrázek 3 - Ukázka Veclis 20
Obrázek 4 - Ukázka Vetbook 21
Obrázek 5 - Ukázka Winvet 22
Obrázek 6 - Nekompletní kód přeskočení události 30
Obrázek 7 - Definování entity Blob 31
Obrázek 8 - Ukázka XAML 35
Obrázek 9 - Překlad aplikace 35
Obrázek 10 - Povinné atributy zvířete 36
Obrázek 11 - Zobrazení seznamu zvířat 37
Obrázek 12 - Vytvoření a nahrání zálohy 38
Obrázek 13 - Vytvoření události v kalendáři 38
Obrázek 14 - List událostí vedle kalendáře 40
Obrázek 15 - Entita zvíře definovaná v C# 44
Obrázek 16 - Diagram databáze 45
Obrázek 17 - Inicializace databáze 46
Obrázek 18 - Konfigurace připojení databáze 47
Obrázek 19 - Titulní strana aplikace VetPel 50

Přílohy

Příloha A – CD s bakalářskou prací, zdrojovým kódem, instalačním balíčkem a uživatelskou příručkou pro ovládání aplikace VetPel. Instalační balíček také obsahuje textový dokument „CTI_ME.txt“ s instalačními pokyny. Toto CD je k dispozici pouze u tištěné verze této práce. Součástí CD je také volně dostupná verze SQL serveru „Microsoft SQL express 2019“.