ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

Katedra biomedicínské informatiky

Veterinární informační systém s využitím zdravotnického standardu HL7® FHIR®

Veterinary information system using the HL7® FHIR® medical standard

Bakalářská práce

Studijní program:	Biomedicínská a klinická technika
Studijní obor:	Biomedicínská informatika

Autor bakalářské práce:	MVDr. Anna Huslarová
Vedoucí bakalářské práce:	doc. Mgr. Radim Krupička, Ph.D

Kladno 2023

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci s názvem „Veterinární informační systém s využitím zdravotnického standardu HL7® FHIR®“ vypracovala samostatně a použila k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k diplomové práci.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V Kladně 17.5.2023 …..……..……...

MVDr. Anna Huslarová

PODĚKOVÁNÍ

Ráda bych poděkovala svému vedoucímu práce, docentu Radimu Krupičkovi, za rady při vyhotovování projektu a pohotovou komunikaci. Dále bych chtěla poděkovat inženýru Liboru Seidlovi, jehož přednášky byly inspirací pro zvolení tématu práce.

ABSTRAKT

Veterinární informační systém s využitím zdravotnického standardu HL7® FHIR®

Cílem bakalářské práce bylo vytvořením aplikace pro objednávání klientů na veterinární pracoviště za použití mezinárodního standardu HL7 FHIR pro předávání zdravotnických dat. Po analýze současných systémů a požadavků na systém jsem přistoupila k tvorbě webové aplikace za využití open-source serveru HAPI FHIR a knihovny React pro vytvoření uživatelského rozhraní. Hlavní důraz byl kladen na optimální uživatelské rozhraní z hlediska rychlosti a jednoduchosti práce se systémem v rušném provozu veterinárních praxí. Část práce se také zabývá využití systému Docker pro nasazení aplikace do provozu a přístupu k ní. V závěru popisuji vytvořenou aplikaci a hodnotím přínosy a nevýhody využití zdravotnického standardu FHIR.

Klíčová slova

Objednávací systém, veterinární, FHIR, UI/UX, React

ABSTRACT

Veterinary information system using the HL7® FHIR® medical standard

The aim of this bachelor's thesis was to create an application for scheduling appointments for clients at veterinary clinics using the international standard HL7 FHIR for healthcare data exchange. After analyzing current systems and system requirements, I proceeded to develop a web application using the open-source server HAPI FHIR and the React library for creating the user interface. The main emphasis was placed on optimizing the user interface for speed and ease of use in busy veterinary practice environments. A portion of the work also addressed the utilization of Docker for deploying the application and accessing it. In the conclusion, I describe the developed application and evaluate the benefits and drawbacks of utilizing the FHIR healthcare standard.

Keywords

Booking system, veterinary, FHIR, UI/UX, React

Obsah

Seznam symbolů a zkratek

Seznam zkratek

Zkratka	Význam
API	Application programming interface
DICOM	Digital Imaging and Communications in Medicine
CC0	Creative Commons Zero
CDA	Clinical Document Architecture
CRUD	Create, Read, Update, Delete
HTML	HyperText Markup Language
IT	Informační technologie
JSX	JavaScript Syntax Extension
FHIR®	Fast Healthcare Interoperability Resources
HL7®	Health Level Seven
REST	Representational State Transfer
RTG	Rentgenografie
VETUNI	Veterinární univerzita Brno
USG	Ultrasonografie

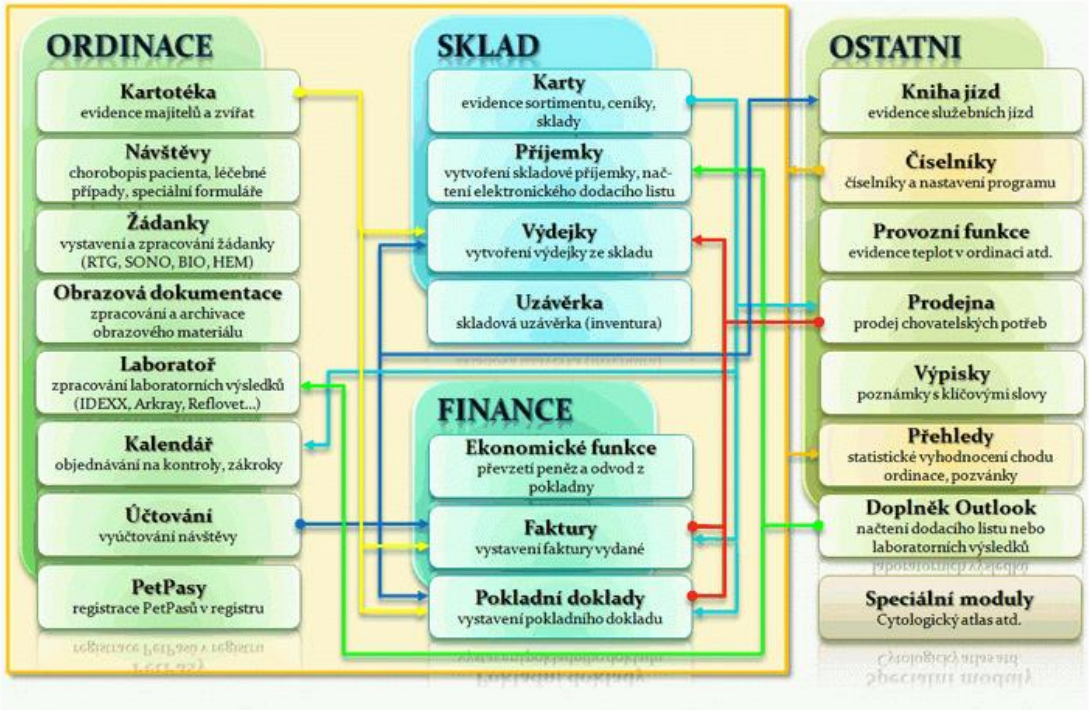
1 Úvod

V dnešní době se ve velké míře setkáváme s automatizací a digitalizací různých procesů. Je to už tak zaběhnutá praxe, že pokud nenásledujeme současné trendy může nás to znevýhodňovat oproti konkurenci. Existuje mnoho dostupných řešení různých problémů, ale občas jsou požadavky natolik specifické, že je potřeba vytvořit systémy na míru. Vzhledem k mé zkušenosti z oblasti veterinární praxe jsem se rozhodla zabývat problematikou digitalizace objednávání pacientů ve veterinární praxi. Jedná se o odvětví, které je na současném trhu řešeno, ale ne příliš s ohledem na efektivnost z hlediska práce na pracovišti. Zaznamenávání objednávek je problematika mnoha odvětví, nejen veterinárního, a existuje mnoho nástrojů, které tuto problematiku řeší, v konečném důsledku ale efektivita práce je občas vyšší, než při používání tradičního způsobu tužky a papíru. Je to velká škoda vzhledem k tomu, kolik nám toho digitalizace může nabídnou. Rozhodla jsem se proto navrhnout systém, který by byl uživatelsky přívětivý, orientovaný na konkrétní požadavky veterinární domény a jeho využití by bylo efektivnější než použití papírového diáře nebo veřejně dostupných online kalendářů. Kromě tohoto cíle, jsem ještě zjišťovala, zda lze pro takovýto systém využít standart pro předávání zdravotnických dat HL7® FHIR® a jaké výhody nebo nevýhody to přináší.

V první části práce se zabývám přehledem současných dostupných řešení pro objednávání klientů. Krátce také představím zmíněný standard HL7® FHIR®. Dále rozebírám, jaké jsou požadavky na systém objednávání veterinárních pacientů. Následuje konkrétní návrh aplikace společně s výběrem programových prostředků, které jsem pro tento případ navrhla. Po přípravné fázi jsem přistoupila k vlastní tvorbě aplikace a vytvoření uživatelského manuálu. V závěru diskutuji, jakých poznatků jsem během práce dosáhla, a zda se mi podařilo dojít vytyčených cílů.

2 Přehled současného stavu

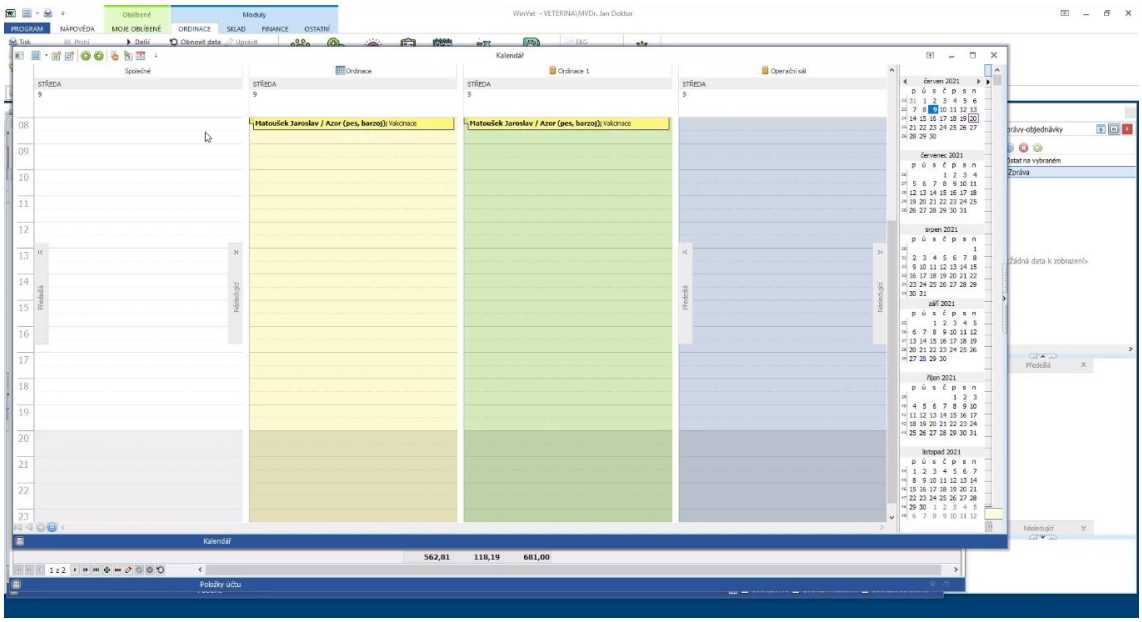
Veterinární informační systémy jsou dnes již běžnou součástí veterinárních nemocnic i menších ordinací. Zaměřují se především na zjednodušení administrace a zaznamenávání lékařských záznamů o pacientech. Poskytují informace o zásobách léčiv a zdravotnického materiálu a tím následně zjednodušují jejich objednávání a předcházení výpadku v zásobách. Dále nám zjednodušují vedení účetnictví a samotné účtování úkonů klientům, což je jednou ze stěžejních podmínek vedení úspěšné praxe. Umožňují přímou komunikaci s dalšími subjekty jako je Komora veterinárních lékařů pro registraci pasů zvířat v zájmových chovech (tzv. petpasů) nebo hlášení Státní veterinární správě počtů zvířat, která poranila člověka. V neposlední řadě uchovávají záznamy o laboratorních výsledcích a vyšetřeních pacientů. Jmenovaný výčet zdaleka není vyčerpávající, ale slouží k nastínění problematiky veterinárních informačních systémů a jejich komplexity, co se do množství dat a interakcí týče (Obrázek 2-1).



Obrázek 2-1: Příklad množství modulů v jednom z nejpoužívanějším systému veterinární praxe v ČR WinVet [1]

2.1 Veterinární informační systémy v České republice

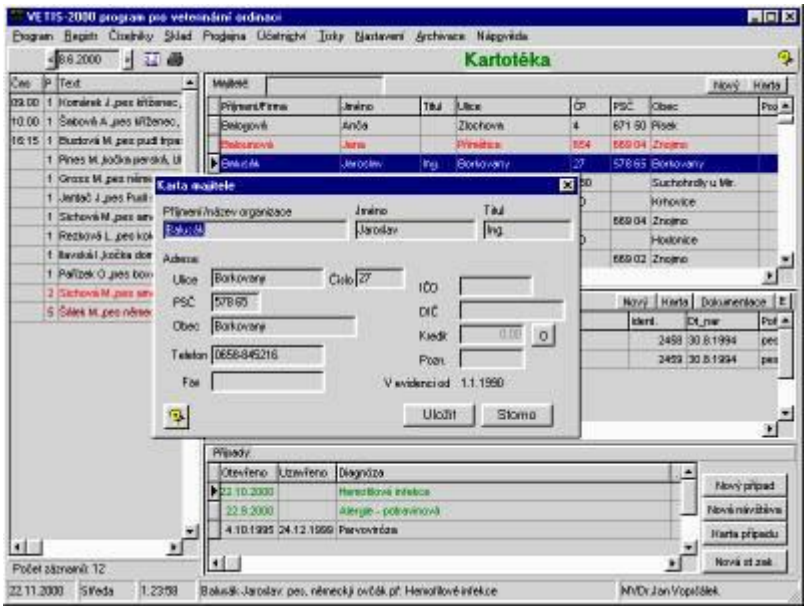
V České republice pokrývají potřeby veterinárních ordinací především dva informační veterinární systémy. Prvním z nich je WinVet (Obrázek 2-2), jehož počátky sahají až do roku 1992. Nicméně základ programu, jak ho známe dnes, vznikl v roce 1997. Tento systém je zároveň provozován na Fakultě veterinárního lékařství VETUNI, takže studenti se již při studiu dostanou do kontaktu s tímto systémem. Systém WinVet je velmi komplexní, což ho i přes snahu vývojářů přiblížit se uživatelskému rozhraní systému Windows, dělá složitý a neintuitivní. Zároveň i systém osvěty a dostupnosti návodů k používání neodpovídají dnešním moderním trendům. Internetové stránky propagující program sice disponují uživatelskou příručkou ke stažení, ale velká část lékařů nemá při práci čas k jejímu prostudování, a tak se většina funkcionalit, které nejsou na první pohled zřejmé, předává slovně mezi kolegy. Věřím, že k určitému zlepšení přispělo zveřejnění záznamu z webináře představení nové verze programu roku 2021. Jedná se o takzvaný on-premise software, kdy program běží na hardwaru vlastněným klinikou, a to jak v desktopové verzi, tak i ve verzi server a client [1].



Obrázek 2-2: Screenshot ze systému WinVet [1]

Druhým systémem je Vetis office (Obrázek 4-1). Mezi českými veterináři má také dlouholetou tradici, nicméně trpí stejnými neduhy jako předešlý systém WinVet. Uživatelská příručka není dostupná online, ale je součástí poskytnutého softwaru. Výhodou je, že systém Vetis office je nabízen na dva měsíce zdarma v plné verzi pro vyzkoušení. I pro tuto demo verzi ale musíte vyplnit objednávku s uvedením kontaktních údajů. Jedná se také o velmi komplexní systém, a i když mnohé funkcionality jsou obdobné jako v systému WinVet, přechod z jednoho systému na druhý vyžaduje určitou

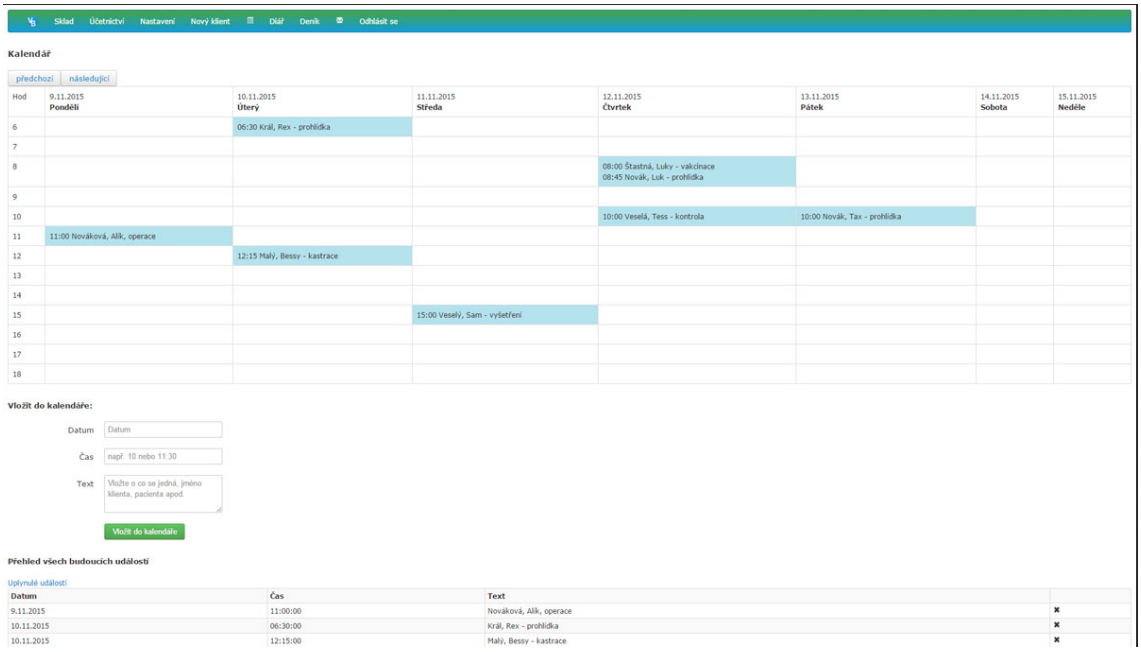
dobu k přivyknutí si na jiné uživatelské rozhraní. Systém Vetis office je také dostupný pouze on-premise v desktopové nebo v síťové verzi [2].



Obrázek 2-3_ Screenshot ze systému Vetis office [2]

Jedním z modernějších softwarů je program Vetbook (Obrázek 2-4). Tento systém má již desetiletou historii a byl jeden z prvních veterinárních informačních softwarů, které není nutné instalovat v ordinaci, ale je dostupný online a veškerá data ukládá na cloudu. Veterinárním lékařům tak odpadá správa tohoto systému, jako je zálohování dat nebo instalace nových aktualizací, což v praxi u jiných systémů bývá často opomíjeno. Je to také zajímavé řešení pro veterináře, kteří se pohybují v terénu a mohou tak svou agendu vést na mobilních zařízeních prostřednictví mobilní sítě. Nabízí to i možnost přístupu klientů k systému a umožnění online objednávání návštěv samotnými klienty, což je však u tohoto konkrétního systému samostatně zpoplatněno. Ačkoliv se zdá, že takováto architektura má spoustu předností oproti dříve zmíněným systémům on-premise, mnoho veterinárních lékařů se jí spíše vyhýbá především z důvodu, že svá data odevzdávají jiné firmě. Vetbook na svých stránkách inzeruje jako výhodu oproti ostatním veterinárním informačním systémům, že svá data svěřujete výhradně IT firmě, a ne veterinárním lékařům, kteří mohou představovat konkurenci. Jiné informační systémy naopak skutečnost, že jsou vyvíjeny veterinárními lékaři, uvádějí jako svou přednost, protože tak mohou lépe odpovídat požadavkům samotných veterinárních praxí. Stejně jako systém Vetis office i Vetbook nabízí bezplatnou zkušební verzi, kterou je nutné si online objednat. Dostupnost služeb je závislá na kvalitě internetového připojení, ale samotné služby by měly být dostupné nonstop a o jakékoliv výpadky se stará poskytovatel systému Vetbook a jeho samotná architektura se snaží těmto výpadkům předcházet. Když například nebude nějaký modul dostupný, ostatní funkcionalita by měla být zachována [3]. Určitý problém by mohl představovat přenos velkých dat (např. RTG snímků), nicméně se stejnou problematikou se potýkají i on-premis systémy, proto

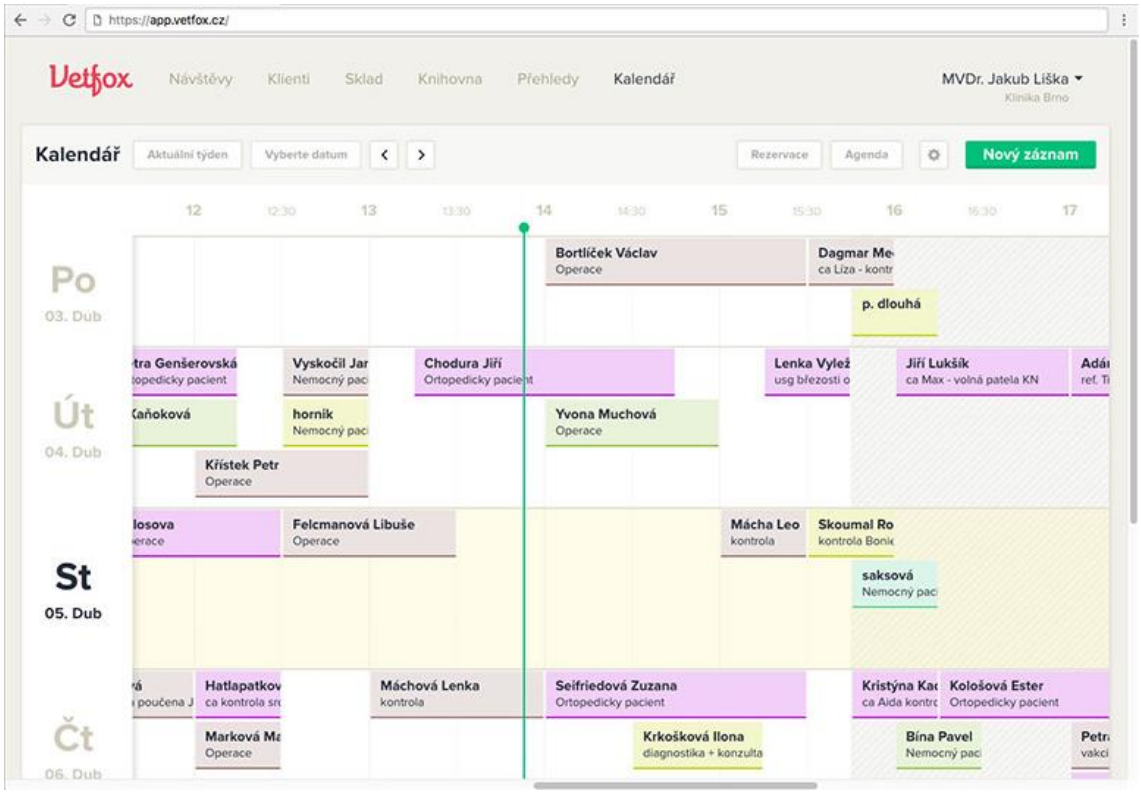
většinou zůstávají RTG snímky nebo USG záznamy uložené v systému, který poskytuje dodavatel dané zobrazovací techniky a nejsou propojeny se záznamy pacientů v informačních systémech. Za zmínku i stojí systém zprávy licencí. U systému Vetbook se platí měsíční poplatek nezávislý na počtu uživatelů a při překročení velikosti úložiště 1 GB se měsíční poplatek zvyšuje [3]. Jiné systémy účtují podle počtu uživatelů zaregistrovaných v systému nebo podle počtu nainstalovaných stanic. V případě, účtování podle uživatelů tak nastávají situace při zaměstnání nových kolegů, kdy licence pro ně je zpoplatněná a někteří zaměstnavatelé nezajistí ať už z důvodů finančních v rámci zkušební doby nebo z opomenutí v dostatečném předstihu přístup nových kolegů do systému. Ve výsledku, tak ze začátku noví veterinární lékaři zapisují do informačního systému pod přihlašovacími údaji kolegy nebo má klinika samostatný účet, který není přiřazen konkrétní osobě a využívají ho například hromadně sestry nebo právě nově najatí lékaři bez vlastního přístupu. Správnost takového přístupu je v režii samotných veterinárních praxí, nicméně systém zpoplatnění veterinárních systémů v této problematice hraje svoji roli.



Obrázek 2-4: Screenshot ze systému Vetbook [3]

Dalším veterinárním informačním systémem založeném na cloudu je Vetfox (Obrázek 2-5). Je vlastněn stejnou firmou jako systém WinVet, čímž firma pokrývá poptávku jak po systémech on-premise tak i po cloudových systémech. Na internetových stránkách je dostupné online demo s omezenou funkcionalitou a po bezplatné registraci je možné si systém vyzkoušet v plné verzi na 3 měsíce zdarma. Jako jediný z českých veterinárních informačních systémů má dostupné video tutoriály jednotlivých úkonů. Vzhledem ke cloudovému řešení má stejné výhody i nevýhody jako předchozí systém Vetbook [4]. Díky dostupné demo verzi, je možné zhodnotit jednoduchost používání. Vzhledově se velmi liší oproti systémům WinVet a Vetis office, které sázejí na podobnost

s operačním systémem Windows. Systém Vetfox uživatelským rozhraním odpovídá dnešním moderním webovým aplikacím a je výrazně jednoduší, což může být plus pro mladší veterinární lékaře, ale naopak starší veterináři zvyklí pracovat v desktopových softwarech mohou mít se systémem ze začátku problém. Důraz je především kladen na finanční management veterinární praxe. Co se týče záznamů o samotném pacientovi, kromě identifikačních údajů daného pacienta a klienta, je dostupné pouze pole pro textový denní záznam z návštěvy a připojení souborů k návštěvě, ale už nám neumožní přehledné sledování vývoje stavu pacienta v čase prostřednictvím naměřených tělesných hodnot nebo laboratorních výsledků. Je otázka nakolik je to nevýhodou, protože jiné systémy vyžadují vyplnění hodnot do konkrétních formulářů, aby mohly tuto funkcionalitu nabídnou, což je ale časově náročné a mnoho veterinárních lékařů stejně přistupuje k tomu, že dané hodnoty vyplní do textového pole denního záznamu, pro zjištění vývoje stavu je potřeba si jednotlivé záznamy přečíst a není možné využívat automatizace. Záleží na jednotlivých pracovištích, jak jsou zvyklé dané funkcionality využívat. Z hlediska většího zaměření aplikace na finanční problematiku veterinární praxe se jedná o chytrý marketingový tah, protože zisk je pro veterinární ordinace důležitější než systém zaznamenávání údajů o pacientech. Samozřejmě, že kvalita veterinární péče a úspěšnost léčby mají na zisk vliv, ale pokud chceme systém udělat co nejjednodušší a intuitivní je právě účetní modul důležitou součástí, která by neměla být opomíjena.



Obrázek 2-5: Screenshot ze systému Vetfox [4]

Okrajově bych chtěla zmínit ještě dva veterinární informační systémy, a to systém VetSoftware a VetPro. Jedná se o desktopové aplikace, které je nutné instalovat a v České republice je jejich využití velice nízké. Své zastoupení mají především na Slovensku, nicméně jsou dostupné i v české lokalizaci. Poslední aktualizace softwaru VetPro je z roku 2017 [5]. V případě VetSoftwaru je poslední aktualizace z roku 2022 [6].

2.1.1 Modul objednávání klientů

Všechny výše zmíněné veterinární informační systémy obsahují modul, který se stará o zaznamenávání naplánovaných návštěv klientů neboli objednávací systém. Řešení, která nabízejí, jsou často pro většinu uživatelů složité, nepřehledné a celkově uživatelsky nepřívětivé nebo funkčně nedostačující a zaznamenávání návštěvy časově náročné. Obzvláště se to týká starších systémů instalovaných přímo v ordinacích. Následkem toho veterinární provozy přistupují k alternativním řešením. V ideálním případě si nechají zhotovit systém přímo na míru jejich provozu, mnohem častěji se však uchylují k volně dostupným aplikacím, které jsou zdarma (např. Google kalendář). Najdou se i takové pracoviště, které pracují pouze s papírem a tužkou. V případě ordinace s jedním lékařem není takové řešení až tak problematické, ale setkala jsem se s ním i na klinice o několika lékařích ordinující v tu samou dobu. V praxi tento systém fungoval tak, že každé ráno recepční okopírovala list z papírového diáře pro konkrétního lékaře a pokud během dne se nějaký klient doobjednal, musela to danému lékaři oznámit a do jeho kopie dopsat.

V případě systémů, které jsou poskytovány bez poplatků, jako je Google kalendář, stojí za zmínku rozpor s všeobecnou nevolí veterinárních klinik posílat svá data do cloudu, ale využívání kalendáře, kam zaznamenávají jména klientů někdy i společně s jejich kontaktem a časem, kdy navštíví jejich zařízení, tak problematické nevidí. Z hlediska GDPR si myslím, že údaje o majitelích zvířecích pacientů jsou citlivější než informace o samotných zvířecích pacientech.

Pro každou kliniku je komplikované přejít z jednoho veterinárního informačního systému na druhý. Jednotliví poskytovatelé systémů nabízejí migraci původní databáze do nových systémů, ale i přesto tento úkon nějaký čas zabere a funkcionalita jednoho systému nemusí naplno odpovídat funkcionalitě druhého systému. Navíc je často takováto změna doprovázena nespokojeností zaměstnanců, kteří jsou zvyklí pracovat s původním nástrojem. Místo toho v případě objednávání si pracoviště radši zvolí nepoužívat funkcionalitu, kterou nabízí jejich informační systém a jen pro tento konkrétní problém využívají jiné nástroje. Přitom právě kvalitní a uživatelsky přívětivý objednávací systém může mít pozitivní dopad na počet pacientů, které praxe zvládne ošetřit, a tím i zvýšit její příjmy. Objednávání pacientů poskytuje několik výhod. Kromě zmíněného efektivního využití pracovní doby, jsme schopni zkrátit dobu, kterou klienti s pacienty stráví v čekárně, a současně s tím i snížit počet klientů a pacientů v čekárně v jeden okamžik. V případě veterinárních čekáren je to výrazný benefit zvláště, pokud se jedná o agresivní pacienty nebo když se v čekárně sejdou jedinci, kteří běžně představují lovce

a kořist. Zvýšíme tím nejen welfare pacientů, ale zajisté i majitelů, kteří je doprovázejí. V neposlední řadě je systém pro objednávání přínosem i pro samotné zaměstnance veterinárních ordinací. Možnost redukovat počet pacientů, kteří najednou čekají na ošetření, jistě uvítá každý, kdo takovou situaci zažil. V případě více akutních případů nás to nutí k rozhodnutí, který ošetřit dříve, a i když se nejedná o akutní pacienty, v dnešní době, kdy čas jsou peníze, se nálada klientů zhoršuje s dobou čekání. Bohužel přes nesporné výhody objednávání se neobjednaným pacientům nevyhneme. Může jít o akutní případy, které nelze objednat nebo o klienty, kteří nevědí, že je možné se objednat anebo prostě nechtějí, či to nepovažují za důležité.

Za zmínku stojí i objednávací systémy, kdy se klient může sám objednat nejčastěji prostřednictvím webové aplikace. I takovéto nadstavby nabízejí současné veterinární systémy. Vhodnost jejich implementace záleží na vytíženosti ordinace a systému organizace pracovní doby. Pokud máme spíše více klientů, než které dokážeme ošetřit, nevidím možnost objednávání se samotnými klienty jako vhodnou. V případě takového systému musíme dopředu navrhnout jaký časový úsek si chceme pro jednotlivé návštěvy vyčlenit bez znalosti problému nebo klienta a pacienta, kterých se návštěva bude týkat. To může vést k tomu, že bude časová dotace příliš velká a my ošetříme méně pacientů, než bychom zvládli nebo naopak časová dotace bude nedostatečná a v čekárně budou vznikat fronty s nespokojenými klienty. Do takového systému se samozřejmě dá nadefinovat několik typických úkonů, které naše ordinace provádí a pro každý úkon specifikovat jinou časovou dotaci. V případě specialisty, který například má vyhrazenou pracovní dobu, kdy bude dělat jen vyšetření srdce, které časově většinou zaberou obdobný časový úsek, může být takovýto systém přínosem. Většina veterinárních pracovišť, ale přijímá velmi rozmanitou škálu veterinárních problémů a často jen díky osobní zkušenosti víme, kolik na daného pacienta budeme potřebovat času. Je rozdíl mezi psím pacientem, který se k nám hrne už od dveří a nechá se sebou dělat cokoliv a kočičím pacientem, kde jen vyndání z přepravky zabere více jak deset minut. Online objednávání je v určitých případech pohodlnější způsob pro klienty, ale nikdy zcela nenahradí objednávací systém obsluhovaný zaměstnanci veterinárního pracoviště.

Z výše uvedených důvodů si myslím, že na českém trhu chybí kvalitní objednávací systém pro veterinární kliniky, který by byl uživatelsky přívětivý a práce s ním by pro zaměstnance nebyla příliš časově náročná. Přece jen je rozdíl mezi tím, kdy si chceme zaznamenat pár schůzek za den do diáře, případně na to máme sekretářku, a tím kdy recepční zaznamenává několik desítek objednávek za den a k tomu přijímá nově příchozí pacienty a zároveň zpracovává vyúčtování u ošetřených pacientů. V horším případě zmíněné úkony řeší veterinární lékař sám, což s jedním pacientem za hodinu v čekárně není problém, ale ve špičce to může být obtížné i v malých ordinacích.

2.2 Využití komunikačních standardů ve veterinární medicíně

Žádný ze zmíněných veterinárních informačních systémů nevyužívá jakýkoliv standard pro zdravotnické záznamy (výjimka je standard DICOM pro obrazovou dokumentaci). Veškerá komunikace se třetími stranami jako jsou laboratorní přístroje, Komora veterinárních lékařů nebo orgány státní správy používají svoje proprietární protokoly. Software Vetfox jako jediný na svých internetových stránkách popisuje své API, kterým je možné se se systémem propojit [7]. U ostatních systémů nejsou žádná API veřejně dostupná.

Na rozdíl od humánní medicíny není veterinární medicína příliš svázaná legislativou a veškerá povinná hlášení probíhají papírově, emailem nebo přes rozhraní té dané instituce pod kterou problematika spadá. Dokonce i v případě eReceptů mají veterinární lékaři výjimku a vydávají recepty v papírové podobě, kde stačí uvést, že léčivý přípravek je určen pro veterinární užití. Na rozdíl od toho humánní medicína potřebuje přinejmenším komunikovat se zdravotními pojišťovnami, aby jim proplatily jednotlivé úkony a s tím souvisí vykazování zdravotní péče a používání stejné nomenklatury, aby si jednotlivé systémy mezi sebou rozuměly. V okamžiku, kdy neexistují systémy, které by používaly uznávaný standard není ani pro poskytovatele informačních veterinárních systému rentabilní takový standart do svých systémů implementovat. Proto přetrvává praxe, kdy jsou pro jednotlivé koncové zařízení jako jsou laboratorní přístroje a veterinární databáze vyvíjeny vlastní komunikační protokoly, které jsou následně implementovány do veterinárních informačních systémů, pokud je po takovém řešení poptávka. V případě RTG a USG se i ve veterinární medicíně stejně jako v humánní využívá standard DICOM, nicméně žádanky pro dané vyšetření jsou v rámci veterinární informačních systémů také řešeny individuálně. Často veterinární informační systémy umožňují přenášení dat z a od jiných zařízení, ale toto spojení není nastaveno, takže pracovníci veterinární praxe přepisují informace ručně ze systému do systému.

2.3 Standard HL7® FHIR®

Standard FHIR byl vybrán pro jeho širokou využitelnost a vyzrálost. V České republice není příliš rozšířen, protože existuje český zdravotnický standard DASTA, kterému tuzemské firmy vytvářející zdravotnické aplikace dávají přednost. Nicméně standart DASTA není určený pro veterinární medicínu. FHIR je standard spravovaný organizací Health Level Seven. Organizace HL7 byla založena roku 1987 [8], takže se zdravotnickými daty má již mnohaletou zkušenost, a i když FHIR je posledním vydaným standardem, není to nevyzrálý projekt, který by nebyl vhodný k produkčnímu nasazení. Aktuálně je publikovaná již jeho 5 verze označovaná jako R5.

Standart FHIR je mezinárodní a využívá se především ve Spojených státech. Je aplikovatelný jak v běžném zdravotnictví, tak i ve veterinárních praxích a nabízí možnost

rozšíření pro úpravu ke specifickým účelům. Tento standart navazuje na již dříve vydané a léty odzkoušené (HL7 v2, HL7 v3 a CDA) a zároveň ve své architektuře zahrnuje moderní webové standardy. Na rozdíl od svých předchůdců, kteří se zaměřovali především na standardizaci formátu předávaných dat, FHIR jde dál a popisuje celý framework, jak pracovat se zdravotními daty. Rozděluje jednotlivé typy informací do takzvaných zdrojů, v originále označované jako Resources (např. Patient, Practitioner nebo Appointment). Dále definuje vztahy mezi těmito zdroji a umožňuje tak vytvořit různé struktury aplikovatelné na zdravotnická data. Tento systém se podobá objektovému programování a je navržen pro jednoduchou implementaci. Snaží se popsat 80 % procesů a dat ve zdravotnictví a ve zbylých 20 % nechává prostor pro rozšíření ve specifických případech. Standard sám o sobě popisuje struktury, vztahy mezi nimi a způsoby komunikace. Nechává však na programátorech, jaký jazyk a technologie použijí [9].

Nespornou výhodou standardu FHIR je, že oproti svým předchůdcům je vydán pod volnou licencí Creative Commons (konkrétně CC0) [10]. Umožnilo to vývoj mnoha open-source projektů, mezi které patří servery podporující standard FHIR. Některé servery jsou přímo nasazené v živém prostředí a je možné je rovnou začít používat pro testování nově vznikajících aplikací nebo je možné nainstalovat si vlastní instanci takového serveru a používat ji jen pro své účely [11]. Kromě těchto otevřených projektů existují i komerční varianty velkých společností jako je například Microsoft Azure [12] nebo Amazon Web Services [13], kteří nabízejí správu FHIR Resources jako službu. Dále existuje velké množství knihoven dostupných v různých programovacích jazycích, které usnadňují komunikaci a nasazení FHIR standartu v aplikacích [14]. Především ve Spojených státech, kde je tento standard hojně využíván, umožnily velké společnosti vyvíjející nemocniční informační systém (např. Cerner nebo Epic) ostatním vývojářům přístup k jejich serverům s testovacími daty a mohou tak vznikat aplikace třetích stran, které budou přímo zabudovatelné do jejich informačních systémů. K tomuto účelu existuje knihovna SMART on FHIR, díky níž se jednotlivé aplikace autorizují u konkrétního serveru a podle předem stanovených pravidel od něj dostanou pacientská data s kterými lze pracovat. Mohou tak vnikat aplikace pro pacienty, kteří si ze systému jejich nemocnice zobrazí data jen o své osobě nebo například aplikace, která bude pro lékaře přímo přístupná z jejich nemocničního informačního systému a bude například přehledněji vizualizovat data o pacientech (názorné je to na ukázkové aplikaci zobrazující růstovou křivku u dětí) [15]. Díky tomuto otevřenému systému je snadné získávat zkušenosti z jiných implementací a neopakovat stejné chyby.

Existují i generátory testovacích dat, které vám vytvoří velké množství nejen pacientů, ale i navázaných úkonů na ně a vše mezi sebou má logickou návaznost [14]. Bohužel tyto generátory vytváření pouze humánní zdravotnická data, takže ve veterinární sféře nejsou bez úprav použitelná.

V neposlední řadě stojí i za zmínku, že standard FHIR má kolem sebe velmi živou komunitu vývojářů, kteří ochotně odpovídají na dotazy nováčků [16]. Krom toho

vyvinula i nástroje, které pomáhají při vývoji a usnadňují vytváření návrhů samotného řešení pro implementaci standartu FHIR. Jako příklad bych uvedla webovou aplikaci clinFHIR Launcher, kde si v přehledném grafickém rozhraní můžete zorientovat jaké struktury vám standard FHIR nabízí a jak ho použít pro konkrétní aplikaci. Dále si tu můžete vytvořit testovací data a přímo je nahrát na veřejné testovací servery, ke kterým následně může přistupovat vaše aplikace. Kromě toho, pokud již máte vytvořená data, můžete si zde validovat jejich struktury [17]. Kromě těchto aplikací komunita pořádá Connectathons, kde si práci se standardem můžete zkusit v okruhu podobně zainteresovaných lidí. Jsou veřejně dostupná i videa, kde lidé, kteří vyvíjí tento standard nebo mají už dlouholeté zkušenosti s jeho používáním, debatují a popisují použití tohoto standardu v různých případech a s konkrétními dostupnými nástroji na trhu.

3 Cíle práce

Cílem práce je vytvořit jednoduchý objednávací systém, který bude vytvářet objednávky pomocí intuitivního zadávání, dále dané objednávky zobrazí a umožní je editovat. Kromě samotných objednávek umožní i zadat směny lékařů a mazat je. Důraz by měl být kladen na přehlednost, jednoduchost a intuitivnost ovládání. Řešení by mělo být možné nasadit jak přímo na serveru veterinární praxe, tak také prostřednictvím cloudových služeb. V práci budou popsány možnosti využití standardu FHIR pro objednávání pacientů ve veterinární medicíně.

3.1 Uživatelské rozhraní

Systém bude obsahovat grafické uživatelské rozhraní, které bude zprostředkovávat práci s uloženými daty objednávek. Uživatelské rozhraní bude odpovídat současným trendům a tím zajistí intuitivnost ovládání.

Systém by měl umožňovat zadat, co nejvíce informací o plánované návštěvě, zároveň ale je striktně nevyžadovat, aby místo usnadnění práce, práci nekomplikoval. Jako povinná pole by měl ukládat jméno klienta nebo popis, čas a datum objednání. Další pole by měla být volitelná, a to například popis zdravotního problému, jméno lékaře, ke kterému je klient objednán, identifikace pacienta nebo ordinace, kam je pacient objednán v případě pracoviště s více ordinacemi v rámci jedné praxe. Veškeré již zadané údaje by si systém měl pamatovat a nabízet je při dalším zadávání pro urychlení práce.

Přehled objednávek by měl být koncipován jako celodenní s časovou osou, kde bude zvýrazněn aktuální čas. Musí umožňovat současné zobrazení více ordinací nebo operačních sálů, aby bylo na první pohled zřejmé, jak je celá klinika vytížena. Jednotlivé objednávky budou barevně odlišeny podle lékařů, kteří je budou mít na starost.

Zobrazení objednávek by mělo umožňovat vybrání konkrétní objednávky a nabídnout její smazání.

3.2 Správa dat

Data budou ukládána na serveru ve standardu FHIR. K tomuto bude vybráno open-source řešení a v práci bude popsána jeho využitá funkcionalita. Bude zhodnoceno, zda je vhodné pro toto řešení používat jakýkoliv standard a zda je k tomu standard vhodný.

4 Návrh aplikace

Po prostudování dostupných řešení objednávacích systémů pro veterinární praxe byla navržena aplikace, která by vyhovovala současným softwarovým trendům a zároveň běžným a nejčastějším požadavkům veterinárních pracovišť. Navržená aplikace pracuje se standardem FHIR a implementuje jeho použití ve veterinární medicíně.

4.1 Požadavky na navrhovaný systém plynoucí ze současného stavu

Vzhledem k tomu, že veterinární informační systémy jsou velmi komplexní, což vede k mnoha problémům zmíněných výše, ráda bych se zaměřila pouze na problematiku objednávání klientů veterinárního pracoviště. Díky tomuto přístupu může vzniknout aplikace, která bude použitelná současně s běžnými veterinárními systémy a vykryje nedostatky, které tyto systémy mohou mít. Zároveň by měla používat definovaný standard, díky kterému data aplikace budou přenositelná a případně dostupná pro komunikaci s veterinárními systémy. Hlavní důraz by měl být kladen na uživatelskou přívětivost a rychlost práce s aplikací, před komplexností ukládaných dat.

4.1.1 Uživatelé

Mezi uživatele, kteří budou se systémem pracovat, patří recepční, provozní, veterinární sestry a veterinární lékaři. Pro všechny uživatele platí, že systém by měl být, co nejvíce intuitivní, aby čas na naučení se s ním pracovat byl co nejkratší, v ideálním případě nulový.

Recepční

Recepční zaznamenávají nejčastěji telefonické objednávky. První informaci, kterou pro nalezení vhodného termínu potřebuje je důvod návštěvy a druh zvířete. To hraje roli v rozhodnutí, kolik času na úkon bude potřeba a který lékař úkon může vykonat. Případně nás bude zajímat i pohlaví pacienta, kterého budeme ošetřovat (např. kastrace samce a samice se časově velmi liší). Tyto informace stačí, aby systém navrhl vhodné termíny a až následně by se recepční měla doptat na nacionále klienta a pacienta a kontakt. Recepční se málokdy dostanou do ordinací a je pro ně těžké stanovit, jaké časové rozmezí mají k objednávce rezervovat, obzvláště pokud na daném pracovišti pracují krátce. Často dostávají i protichůdné informace, kdy z jedné strany je snaha ošetřit co nejvíce pacientů a z druhé strany od veterinárního lékaře, který k diagnostice potřebuje dostatek času, aby se pacientovi mohl patřičně věnovat. Bylo by proto vhodné, aby recepční měli k dispozici přehled jednotlivých možných úkonů a kolik času je k nim potřeba. Další informací, kterou se recepční musí naučit je, kteří lékaři vykonávají určité úkony (např. specializované kardiologické vyšetření neprovádějí všichni lékaři na pracovišti). I tuto

informaci by měl systém pro objednávání znát a recepční pomáhat prostřednictvím napovídání. Časově náročné může být i zaznamenání konkrétní objednávky, pokud klient má složitější jméno nebo je mu hůře po telefonu rozumět. Určitě pro takového klienta bude pracoviště vypadat mnohem seriózněji, pokud se ho při každé objednávce nebudeme desetkrát dotazovat na jméno a systém si jméno bude pamatovat. Zároveň by samotné zadávání objednávky mělo probíhat rychleji, než když ji zapíšeme do papírového diáře, protože poté přínos takového systému výrazně upadá. Úplně krajní případ je, když zadání do systému trvá déle než zápis do papírového diáře, což se může stát v okamžiku, kdy systém po nás vyžaduje velké množství informací, aby objednávku uložil (jméno klienta, jméno pacienta, kontakt, důvod návštěvy, aj.). V praxi (obzvláště těch menších) úplně stačí zápis typu „14:00 Nováková“, protože paní Nováková už čtrnáct dní dochází s kočkou na převaz kousné rány, kdy kontroly probíhají každý druhý až třetí den. V případě, že bychom měli systém, kam bychom pokaždé museli zaznamenat „14:00 - 14:20 Nováková Jana, Fe Micka, tel.: 999 999 999, převaz kousné rány“, bude takový systém paradoxně na obtíž. Na druhou stranu, pokud v ordinaci ordinují dva veterinární lékaři a naše paní Nováková čtrnáct dní chodila k jednomu a teď je z jakéhokoliv důvodu objednaná k druhému, bude pouhá informace „14:00 Nováková“ nedostatečná a lékař nebude tušit, co ho čeká až do okamžiku, než klient vejde do ordinace, protože paní Novákových v systému pravděpodobně bude několik. Následně se bude muset paní Novákové vyptávat, co všechno se odehrálo, nebo několik zdlouhavých minut nechat paní Novákovou sedět a pročítat si zápisy od kolegy, aby se uvedl do obrazu. V případě, že klient sedí již v ordinaci, to také nevypadá příliš profesionálně a jistě by pro lékaře bylo příjemnější mít možnost si nastudovat kartu v soukromí a s předstihem. V případě kousné rány možná není tak dokonalé seznámení s historií pacienta potřeba, případně stačí, když nám to klient sám vylíčí, ale pokud se jedná o komplikovaný rozvleklý případ, je to věc jiná. Když to shrneme, bylo by vhodné, aby systém ukládal co nejvíce informací, ale zároveň jejich zadávání nezabíralo příliš mnoho času. K tomu nám může dopomoci tzv. našeptávač, kdy už jednou zadané údaje si bude pamatovat a v případě jejich opakovaného zadávání nám je nabídne. V našem ukázkovém příkladě by recepční v okamžiku, kdy se paní Nováková bude chtít podruhé objednat na kontrolu, zadá pouze „Nováková“ a systém jí automaticky nabídne všechny paní Novákové a jejich mazlíčky seřazené v pořadí, kdy u nás byli naposledy, a bude stačit pouze vybrat o kterou paní Novákovou se jedná a zbytek informací si systém doplní.

Provozní

Systém nemůže zcela plně fungovat bez informace, kdy, který lékař slouží, a v jaké ambulanci a je tudíž možné se k němu objednat. Je potřeba, aby měl přehled o rozpisu směn, o který se nejčastěji stará provozní veterinárního pracoviště nebo jiná pověřená osoba. Zároveň si ale systém neklade za cíl být plnohodnotným plánovačem směn, kdy sám navrhuje rozpisy služeb podle zadaných kritérií. Systém by měl pouze umožnit zadat již předem naplánované směny a případně při změně nebo špatném zadání je i smazat.

Veterinární sestra

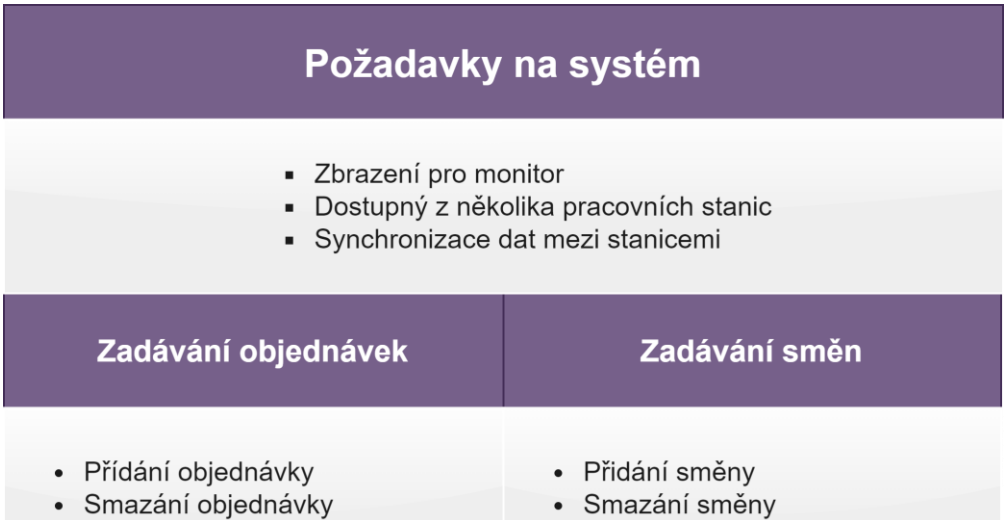
Veterinární sestry budou systém využívat především pro zobrazení agendy aktuální pracovní doby a pro orientaci, co mají na daný úkol lékaři připravit k ošetření. Využijí tedy zobrazení systému objednávek. Je potřeba, aby měli přehled o všech ordinacích, kde se aktuálně ordinuje, protože se může stát, a zpravidla se stává, že sestra asistuje více doktorům v rámci jedné služby. Ve výjimečných případech sestra zastane i pozici recepční, proto by měla mít v systému přinejmenším stejná práva.

Veterinární lékař

Veterinární lékař potřebuje mít přehled o objednaných pacientech pro efektivní plánování pracovní doby, tudíž je pro něj důležité přehledné zobrazení. Potřebuje vidět naplánované objednávky pro daný den s vyznačeným aktuálním časem. Dále také sám objednává zejména klienty, kteří jsou aktuálně v ordinaci a chtějí objednat na kontrolu. Vzhledem k tomu, že to není hlavní náplní jeho práce, ale zároveň on sám dokáže nejlépe zhodnotit, kolik času mu daný úkon zabere a v jakém rozsahu, či co konkrétně je potřeba udělat, je vhodnější, aby některé objednávky zprostředkoval sám. Když se vrátíme k našemu příkladu uvedeného ve specifikaci provádění objednávky recepční, v případě lékaře, který objednává klienta sám k sobě, je informace o času a jménu klienta zcela dostatečná a systém by měl tento způsob zaznamenání objednávky umožnit. Více objednávek v jeden čas by systém měl povolit, ale upozornit na kolizi.

4.2 Analýza požadavků

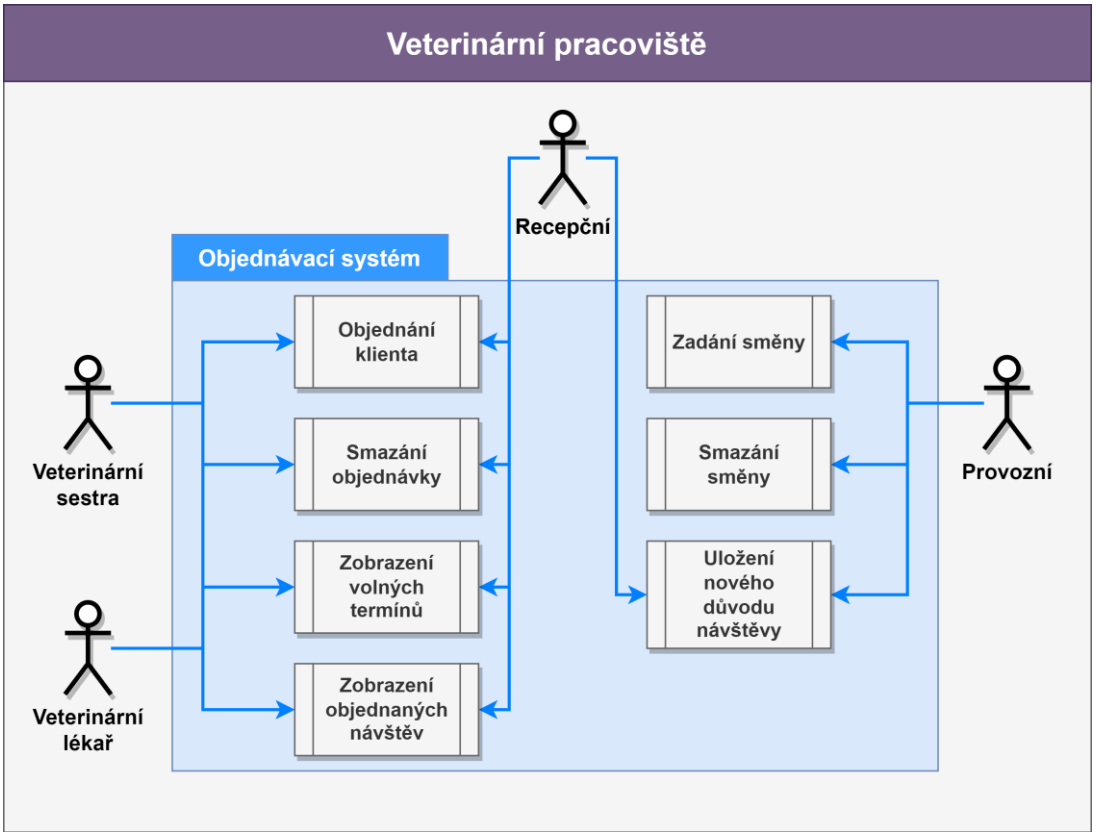
Požadavky na systém (Obrázek 4-1) byly sesbírány díky osobní zkušenosti při práci na čtyřech veterinárních klinikách v České republice v roli veterinární sestry, veterinárního lékaře, a i jen nezúčastněného pozorovatele. Díky tomuto přístupu byl získán široký přehled o potřebách, nedostatcích a rozmanitému využití objednávacích



Obrázek 4-1: Seznam požadavků na objednávací systém

systému ve veterinární medicíně. Na základě těchto poznatků jsem vyhodnotila, že praktické řešení této problematiky je na veterinárních klinikách nedostatečné a často opomíjené. Tvoří se tak nevyhovující pracovní podmínky pro zaměstnance, kdy je na jedné straně od managementu pracoviště, a i samotných klientů vyvíjen nátlak na efektivitu práce, ale zároveň zaměstnancům není poskytnut dostatečně kvalitní nástroj, který by tento aspekt jejich práce zjednodušil.

Konkrétní požadavky na systém jednotlivých uživatelů byly již popsány v kapitole 4.1, proto zde uvedu přehled požadavků na systém jako celku (Obrázek 4-2).



Obrázek 4-2: Požadavky na systém

4.2.1 Obecné požadavky na systém

Systém budou používat pouze zaměstnanci veterinárního pracoviště, nebude umožněn přístup jejím klientům. Nicméně i tak je potřeba, aby byla data sdílena mezi několika počítači v rámci pracoviště – na recepci, v ordinaci, v laboratoři a podobně. Data by měla zůstat uložená na pracovišti a musí být synchronizována v reálném čase. K systému se bude přistupovat pomocí desktopových počítačů nebo notebooků, nejčastěji s nainstalovaným operačním systémem Windows z čehož vyplývá, že zobrazení dat by mělo být optimalizováno pro počítačové obrazovky a využít co nejvíce jejího prostoru. Ovládání systému bude pomocí klávesnice a myši, dotykové obrazovky nebývají na veterinárních pracovištích dostupné, nicméně mělo by být možné systém obsluhovat i z dotykových zařízení jako jsou například tablety. Zobrazení i přes to by mělo být

prioritně navržené pro monitory. Jako nadstandartní požadavek by mohl být systém přístupný i prostřednictvím internetu z osobních zařízení zaměstnanců i mimo pracovní dobu. Tuto funkcionalitu není v této fázi nutné implementovat, ale při návrhu systému je potřeba myslet na toto možné rozšíření.

Co se týče funkcionalit systému dělí se na dvě části. Jednak činnosti spojené s objednávání návštěv klientů veterinárních pracovišť a potom druhou část spojenou s organizací směn lékařů.

4.2.2 Objednávání klientů

Hlavní funkcionalitou systému je objednávání, kde je kladen důraz na optimální uživatelské rozhraní. Cílem je co nejrychlejší vytvoření objednávky. Zároveň však musí obsahovat všechny nutné údaje, které se u jednotlivých případů mohou lišit. Uložená objednávka by měla obsahovat nebo aspoň měla mít možnost uložit informace o důvodu návštěvy, času návštěvy, délce trvání návštěvy, ošetřujícího lékaře, jakou místnost je potřeba zarezervovat, jméno objednaného klienta, kontakt na něj, jméno, druh a věk pacienta. Jediné dva povinné údaje z tohoto výčtu jsou datum a čas a místnost k rezervaci. Díku tomu bude možné pomocí těchto objednávek evidovat i nutná sdělení personálu, například o nedostupnosti nějaké místnosti.

Datum a čas objednání bude možné zadat jednak pomocí výběru konkrétního data z kalendáře nebo přímo zapsání data do textového pole. Zároveň bude možné vybrat datum a čas ze zobrazení jednotlivých objednávek, kdy dojde i k automatickému vyplnění místnosti a lékaře, které budou patrné z rozvržených směn.

Pole pro důvod návštěvy, místnost, lékaře, jméno klienta a jméno pacienta by měly mít možnost našeptávače, kdy systém nabídne uživateli dříve zadané fráze pro rychlejší zadání. V případě klienta by na základě tohoto výběru měl být schopen automaticky doplnit i kontakt na klienta, jméno pacienta, datum narození pacienta, jeho věk a jakého je druhu. Dále by pole pro věk pacienta mělo být generováno podle data narození, aby věk zůstával vždy aktuální jak v době zadávání, tak i prohlížení záznamu, nicméně často se stává, že přesné datum narození není známé. V takovém případě se do systému zadá odhadovaný věk a systém si uloží vypočítané datum narození, aby při následné návštěvě se věk aktualizoval a nezůstala u pacienta hodnota původně odhadnutého věku. Možnosti k zadání druhu zvířete by měly být předem dané, aby nomenklatura byla unifikovaná. Při každém uložení návštěvy, kdy pacient nebyl zadán z předem nabídnutých možností by systém měl uložit i pacienta pro případnou budoucí referenci. Je potřeba i myslet na případ, kdy se objednává klient, který je již systému známý, ale s novým nebo jiným pacientem/zvířetem. V takovém případě uživatel vybere uloženého klienta a upraví hodnoty o pacientovi. Systém následně uloží informaci jako nového pacienta.

V případě našeptávače důvodů návštěvy by systém měl evidovat předdefinované důvody návštěvy, které by k sobě měly mít přiřazeny lékaře a místnosti, které daný úkon

mohou provádět, pokud zde existuje nějaké omezení, a také by měly mít informaci o běžné délce takovéto návštěvy. Díky tomu, se při výběru důvodu návštěvy automaticky doplní i doba trvání, místnost a lékař, pokud je volba jednoznačná, pokud ne, tak se výběr lékaře a místnosti vyfiltruje na doporučené. Stálé by ale tato funkcionalita měla plnit pouze formu doporučení, musí být zachována možnost zadat jako důvod návštěvy jakékoliv slovní spojení, vybrat jakéhokoliv lékaře, místnost nebo dobu trvání.

Výběr data by měl být omezen pouze na výběr dat v budoucnosti, nemělo by být umožněno zadat návštěvu v minulosti. Výběr místnosti a lékaře by měl být umožněn pouze z předdefinovaných možností, aby nebylo možné objednat klienta do neexistující místnosti nebo k neexistujícímu lékaři.

Objednávky musí být možné i mazat například při chybném zadání nějakého údaje nebo při omluvě klienta. Ideálně pří výběru objednané návštěvy ve zobrazení kalendáře by měla být možnost objednávku smazat, musí tak být provedeno přes potvrzovací zprávu, aby nedocházelo ke smazání návštěvy omylem.

Objednané návštěvy je potřeba přehledně zobrazit na denní časové linii, kde bude zvýrazněn aktuální čas. V případě volného místa k objednání návštěvy bude naznačeno, který lékař v danou dobu ordinuje, v jaké ordinaci, aby bylo zřejmé kam a ke komu je možné návštěvu objednat. Bude možné najednou zobrazit všechny místnosti pracoviště v rámci jednoho dne. Kromě aktuálního data bude možné zobrazovat jak dny v minulosti, tak dny v budoucnosti. Přepínání mezi dny bude jednak umožněno pomocí rychlé volby předchozí a následující den, a také pomocí výběru konkrétního data v kalendáři. Časový rozsah bude nastaven podle požadavků pracoviště (nonstop provoz versus osmihodinová pracovní doba). U každé objednané návštěvy bude barevně označeno, ke kterému lékaři patří a podle umístění v kalendáři bude patrné, do jaké místnosti je objednaná. Při zobrazení v kalendáři budou vybrány jen nezbytně nutné údaje, aby se na jednu obrazovku vešlo co nejvíce informací. Těmito údaji budou příjmení klienta, jméno pacienta a důvod návštěvy. Datum a čas návštěvy bude patrný z umístění v kalendáři a doba trvání návštěvy bude patrná z grafického zobrazení návštěvy.

Use Case 1: Objednávání

Popis: Sestra/doktor/recepční potřebuje klientovi rezervovat termín jeho návštěvy.

Aktér: Sestra/doktor/recepční

Spouštěč: Klient chce objednat

Hlavní scénář:

1. A: Klient veterinární ambulance se chce telefonicky objednat. Aktér stiskne tlačítko pro vytvoření nové objednávky.

   S: Otevře dialog pro vytvoření objednávky.

2. A: Aktér začne vyplňovat pole důvod návštěvy.

S: Nabídne aktérovi list s předem evidovanými důvody, které obsahují zadaný řetězec písmen. Systém nabízí položky dynamicky, tzn. přidá-li aktér další písmeno, systém znovu vyfiltruje důvody – našeptávač.

3. A: Vybere si z nabídnutého listu nebo zadá novou hodnotu a volbu potvrdí.

   S: Zobrazí vybranou volbu. Pokud byl vybrán předem definovaný důvod návštěvy předvyplní pole doby trvání návštěvy podle vybraného důvodu.

4. A: Vybere termín, který vyhovuje klientovi pomocí kalendáře nebo vyplní datum a čas do textového pole.

   S: Přepne zobrazení objednávek na vybraný termín.

5. A: Klikne na možnost výběru doktora nebo rovnou začne vyplňovat jméno doktora.

   S: Nabídne doktora, který může daný úkon vykonat a odpovídá zadanému řetězci. Ostatní doktory také zobrazí, ale budou buď barevně odlišení nebo skrytí a zobrazitelní až po kliknutí na tlačítko.

6. A: Klikne na možnost výběru místnosti nebo rovnou začne vyplňovat jméno místnosti.
   S: Nabídne místnost, kde se může návštěva odehrát. Ostatní místnosti jsou také zobrazeny, ale nějakým způsobem odlišeny od doporučených.

7. A: Aktér začne vyplňovat jméno klienta nebo pacienta.

   S: Nabídne aktérovi list se jmény klientů nebo pacientů, které odpovídají zadanému řetězci, přidá-li aktér písmeno, systém znovu vyfiltruje jména – našeptávač.

8. A: Vybere z listu klienta/pacienta a výběr potvrdí.

   S: Doplní chybějící položky:

   - Jméno klienta
   - Kontakt (email, telefon)
   - Jméno pacienta
   - Datum narození nebo věk pacienta (vypočítavá se z data narození, aby bylo vždy aktuální)
   - Druh pacienta

9. A: Zkontroluje všechny položky na formuláři a pokud s nimi souhlasí formulář odešle.

   S: Zadané informace uloží.

Rozšíření:

2.a. Neznámý Důvod návštěvy

1. A: Zadá Důvod návštěvy, který neodpovídá žádnému uloženému v systému.

   S: Zobrazí možnost pro uložení nového důvodu návštěvy (Use Case 3: Uložení nového důvodu návštěvy).

2. A: Vybere možnost pro vytvoření nového důvodu návštěvy.

4.a. Výběr termínu ze zobrazení objednávek

1. A: Vybere volnou časovou buňku z denního zobrazení naplánovaných objednávek.

   S: Doplní do formuláře pro vytvoření nové návštěvy informace z vybrané časové buňky, a to datum a čas návštěvy, lékaře a místnost. Dobu trvání návštěvy stanoví na délku vybrané časové buňky.

8.a. Zadaný klient/pacient není v databázi

1. A: Doplní pole specifikující pacienta:
   a. Jméno klienta
   b. Kontakt (telefon, email)
   c. Jméno pacienta
   d. Druh zvířete
   e. Datum narození pacienta nebo věk

   S: Uloží informace do databáze pro budoucí použití a pokračuje krokem 8.

8.b. Zadaný klient je v databázi, ale pacient ne

1. A: Vybere nabídnutého klienta.
   S: Vyplní zbylé informace o klientovi a o pacientovi
2. A: Změní informace o pacientovi a objednávku uloží.
   S: Systém uloží nového pacienta.

Následná podmínka: Zobrazení nové objednávky v kalendáři.

Use Case 2: Smazání uložené návštěvy

Popis: Aktér potřebuje smazat uloženou návštěvu

Aktér: Zaměstnanec

Spouštěč: Kliknutí na objednanou návštěvu.

Hlavní scénář:

1. A: Chce smazat objednanou návštěvu a klikne na ni v denním zobrazení.

S: Otevře podrobnosti o návštěvě společně s tlačítkem s možností jejího smazání.

2. A: Klikne na tlačítku pro smazání.
   S: Zeptá se, jestli chce aktér návštěvu skutečně smazat.
3. A: Potvrdí smazání.
   S: Smaže objednanou návštěvu a časové pole uvolní.

Use Case 3: Uložení nového důvodu návštěvy

Popis: Aktér potřebuje uložit nový důvod návštěvy

Aktér: Sestra/doktor/recepční

Spouštěč: Stisknutí volby nový důvod návštěvy.

Hlavní scénář:

1. A: Chce uložit důvod návštěvy pro příští rychlé použití.
   S: Otevře formulář pro bližší specifikaci důvodu návštěvy.
2. A: Vyplní název důvodu návštěvy a klikne na možnost přidání lékaře pro tento důvod návštěvy nebo začne vyplňovat jméno (pokud mohou úkon vykonat všichni lékaře ponechá volbu prázdnou):
   S: Nabídne dostupné lékaře na klinice a umožní výběr více lékařů najednou.
3. Klikne na možnost přidání místnosti pro tento důvod návštěvy nebo začne vyplňovat jméno místnosti (pokud může být úkon proveden ve všech ordinacích ponechá volbu prázdnou).
   S: Nabídne dostupné místnosti na klinice.
4. A: Zadá obvyklou dobu trvání a klikne na tlačítko pro uložení důvodu.
   S: Důvod návštěvy uloží a zadané hodnoty doplní do formuláře pro ukládání nové návštěvy.

4.2.3 Zadávání směn

Aby bylo možné v přehledu objednaných návštěv zobrazit i přehled služeb lékařů je potřeba mít možnost je do systému zadat. Zadávání směn by mělo probíhat v jiné části aplikace, aby náhodou při zadávání objednávek nedocházelo omylem ke změně směny. Zároveň směny nejčastěji zadávají provozní veterinárního pracoviště, které nepotřebují mít přístup k objednávaným návštěvám a naopak lékaři, sestry a recepční, kteří návštěvy objednávají nepotřebují mít přístup k zadávání nových směn do systému, proto je vhodné tyto dvě funkcionality oddělit. Při zadávání nové směny je potřeba určit kterého lékaře se týkají, jaké místnosti se týkají a časový rozsah služby. Vzhledem k tomu že se může jednat i o služby noční, které jsou v rozsahu dvou dnů, je potřeba jak u začátku, tak i u konci směny udávat nejen čas, ale i datum. Před uložením směny je potřeba ověřit, zda daná místnost není již v tento časový úsek obsazená. Při zadávání nové směny je nutné mít možnost vybrat pouze z předdefinovaných místností a lékařů, aby nedocházelo k ukládání směn k neexistujícím lékařům nebo místnostem. Směny je možné zadat pouze

do budoucna. Jednotlivé směny se budou automaticky propisovat do zobrazení objednávek.

Při chybném zadání směny je potřeba mít možnosti ji i smazat. Vybrání chybně zadané směny by mělo probíhat skrze grafické rozhraní, kde budou jednotlivé směny přehledně zobrazeny. U každé zobrazené směny by mělo být patrné jakého lékaře se týká, v jaké místnosti bude probíhat a v jakém časovém horizontu bude probíhat. Přehled směn by měl být ideálně zobrazen v co největším časovém horizontu, ideálně jednoho měsíce.

Use Case 4: Přidání směny

Popis: Rozvržení pracovní doby zaměstnanců a místností.

Aktér: provozní

Spouštěč: Vybere možnost přidání směny.

Hlavní scénář:

1. A: Vybere možnost přidání směny.

   S: Zobrazí dialog pro přidání směny.

2. A: Začne psát jméno lékaře, kterého se směna týká.

   S: Nabídne výběr z dostupných lékařů, kteří odpovídají zadanému řetězci.

3. A: Začne psát jméno místnosti, které se směna týká.

   S: Nabídne výběr z dostupných místností, které odpovídají zadanému řetězci.

4. A: Zadá časové rozmezí, pro které chce plánovat služby a směnu uloží.

   S: Zkontroluje, zda jsou všechna pole zadaná a ověří, zda v daný časový úsek je místnost volná. Pokud není kolize směnu uloží.

Rozšíření:

4.a. Kolize služeb

1. S: Systém najde kolizi s ukládanou směnou. Kolizi ohlásí aktérovu a směnu neuloží.

2. A: Upraví danou směnu a změnu potvrdí.

   S: Zkontroluje nově zadanou směnu a pokud není kolize, uloží ji.

Následná podmínka: Vytvoření směny a její zobrazení v přehledu směn a přehledu objednaných návštěv.

Use Case 5: Smazání uložené směny

Popis: Aktér potřebuje smazat uloženou směnu

Aktér: Provozní

Spouštěč: Kliknutí na směnu v přehledu směn.

Hlavní scénář:

1. A: Chce smazat uloženou směnu a klikne ne ni v přehledu směn.
   S: Smaže směnu společně se všemi jejími sloty pro návštěvy.

4.3 Funkční specifikace

Vzhledem k potřebě přistupovat k datům z několika pracovních stanic je nutné v systému implementovat backend s datovým úložištěm, který bude na vybrané pracovní stanici, serveru nebo hostovaný na cloudu a následně aplikaci frontendu, která bude dostupná vzdáleně. Na backendu se bude pracovat výlučně se standardem FHIR, kdežto aplikace frontendu bude mít svoje vlastní datové modely uzpůsobené k zobrazení.

Ve funkční specifikaci je podrobně rozebráno, s jakými datovými modely bude systém pracovat a jaké vztahy mezi nimi budou existovat. Vzhledem k integraci standardu FHIR, v jakém se data budou ukládat do backendu je nutné datové modely samotné aplikace spravující frontend propojit s odpovídajícími Resource standardu FHIR.

4.3.1 Datové modely

Z analýzy požadavků vyplynulo, že je potřeba aby systém pracoval s několika datovými modely, a to modelem objednávky, pacienta, směny, důvod návštěvy, lékaře a místnosti. Modely lékaře a místnosti jsou pevně dané a neměly by jít v aplikaci změnit, ale pouze z nich číst. Budou se zadávat při inicializaci systému. Modely objednávky, pacienta a směny jsou interaktivní a bude je možné v aplikaci vytvářet, mazat a načítat.

Model lékaře bude obsahovat informaci o jméně lékaře, jeho identifikátor v systému, iniciály pro zobrazení směny a o jaký typ datového modelu se jedná – doktor.

Model místnosti bude obdobný. Bude specifikovat jméno místnosti, její identifikátor v systému, jméno, jaké se má zobrazovat v kalendář, a o jaký typ datového modelu se jedná – místnost.

Model důvodu návštěvy bude pracovat s předchozími dvěma modely, kdy bude mít uloženo jaký lékař může danou návštěvu řešit a v jaké místnosti se může řešit, pokud nebude tato specifikace nutná, zůstanou tato pole prázdná. Dále bude možné navolit, jak dlouho návštěva s tímto důvodem obvykle trvá. Povinným polem bude označení důvodu návštěvy a systém automaticky vygeneruje unikátní identifikátor pro tento důvod návštěvy.

Model směny bude ukládat informaci jakého lékaře a jaké místnosti se směna týká, od kdy do kdy je naplánovaná. Také bude obsahovat unikátní identifikátor.

Model pacienta bude pracovat s informacemi o jménu majitele, kontaktu majitele (bude možné uložit jakýkoliv kontakt – mail, telefon a podobně), jména pacienta, jeho data narození a informace o jeho druhu.

Model objednávky bude pracovat s několika předchozími modely a to lékařem, ke kterému je návštěva objednaná, místnost, do jaké je objednaná, pacient, kterého se týká, důvod návštěvy, o který se jedná, v tomto případě je nutné mít možnost uložit pouze text, pokud objekt s konkrétním důvodem návštěvy v systému neexistuje. Bude možné zadat předpokládanou dobu trvání návštěvy, která slouží k vyblokování daného úseku v kalendáři.

Interně bude systém ještě pracovat s modelem nejkratšího bloku pro možné objednání návštěvy. Každá směna bude rozdělena do těchto bloků. Tyto bloky budou sloužit k uložení informace, zda směna má v daný okamžik volné místo k objednání návštěvy. Tento model bude použit pouze na backendu a frontend se bude pouze dotazovat, zda je blok volný.)

4.3.2 Funkční spojení mezi modely standardem FHIR

Model návštěvy je propojen s modely lékaře, místnosti, důvodu návštěvy a pacientem. Z modelů lékaře, místnosti a pacienta si ukládá kompletní kopie těchto dat, aby v případě, kdy dojde k přepisu původních zdrojových dat, u návštěvy zůstaly informace z okamžiku, kdy byla návštěva vytvořena. V případě důvodu návštěvy se u návštěvy ukládá pouze popis důvodu návštěvy zbylé informace jsou využitelné pouze před uložením návštěvy pro našeptávání doporučených hodnot.

Model důvodu návštěvy je propojen s modely lékaře a místnosti, aby bylo možné zadat pouze hodnoty uložené v systému, v tomto případě se s důvodem návštěvy ukládá pouze identifikátor dané instance modelu lékaře nebo místnosti.

Model směny pracuje s modely lékaře a místnosti, zda zase ukládá kompletní informace o instancích v době uložení směny, aby směna nesla informace z daného okamžiku.

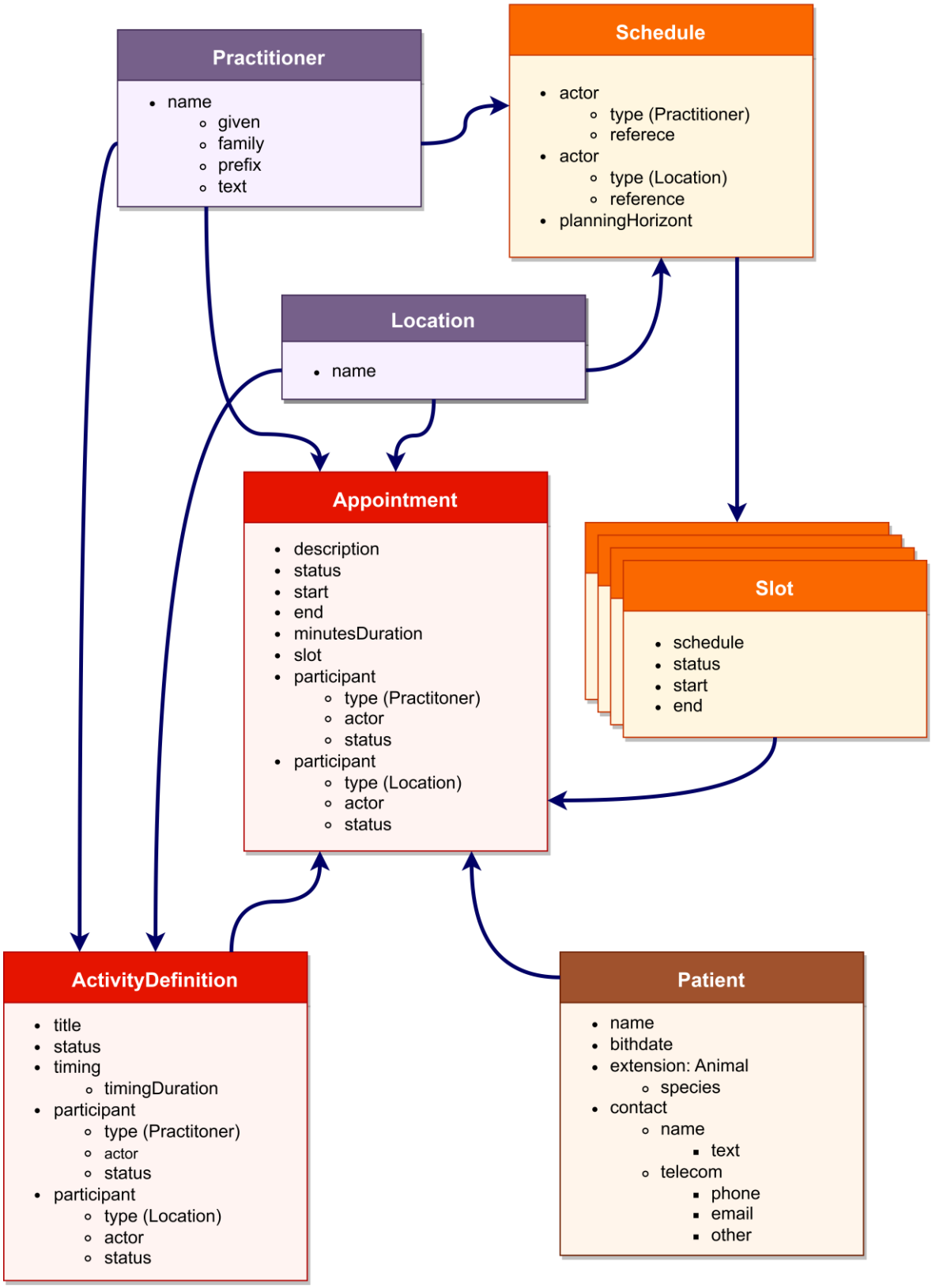
Zbylé modely slouží pouze jako zdroj dat sami v sobě neobsahují odkaz na jiné modely.

Mapování na standard FHIR

Ze standardu FHIR byly pro využití v systému vybrány tyto typy Resource:

- Appointment [18]
- Schedule [19]
- Slot [20]
- ActivityDefinition [21]
- Patient [22]
- Location [23]
- Practitioner [24]

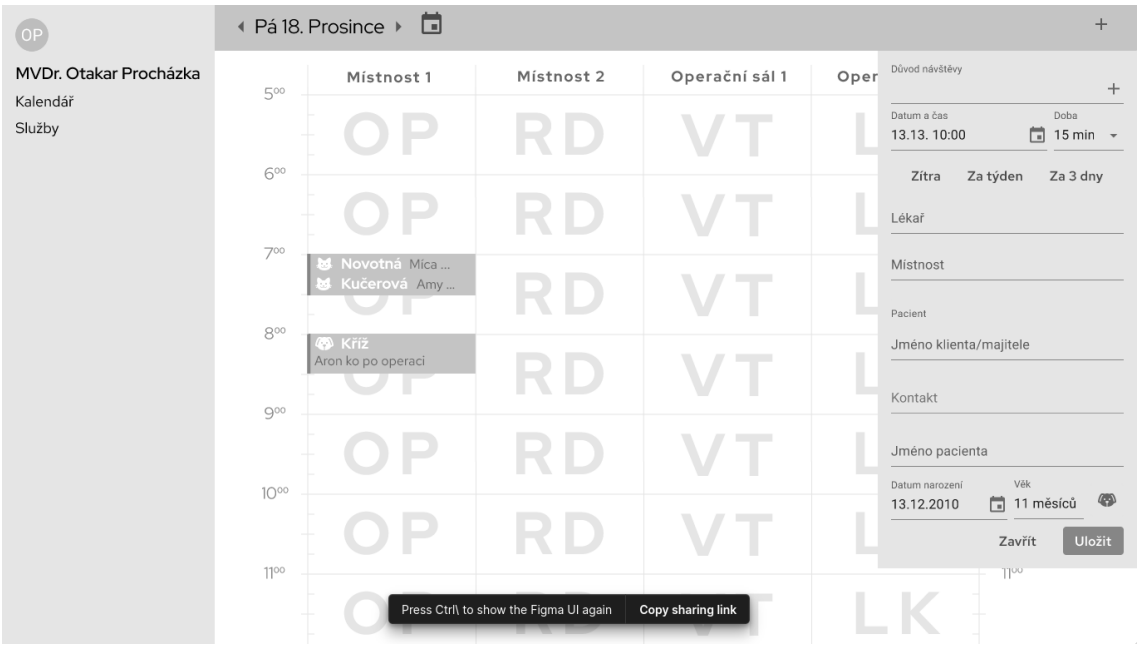
U každého Resource je uveden odkaz na jeho popis společně se seznamem, které atributy je možné u tohoto Resource ukládat a které jsou povinné. Zde jsou uvedeny pouze ty, které budou využité pro systém objednávání (Obrázek 4-3).



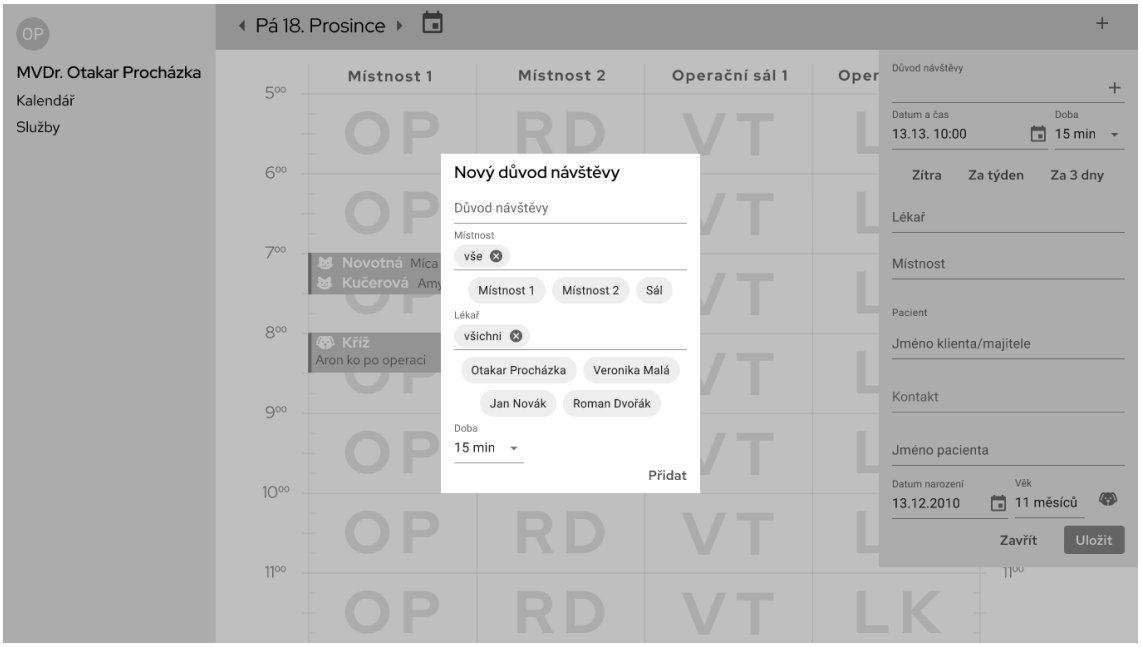
Obrázek 4-3: Využité typy FHIR Resource s jejich atributy a funkčním propojením

4.3.3 Návrh grafického rozhraní

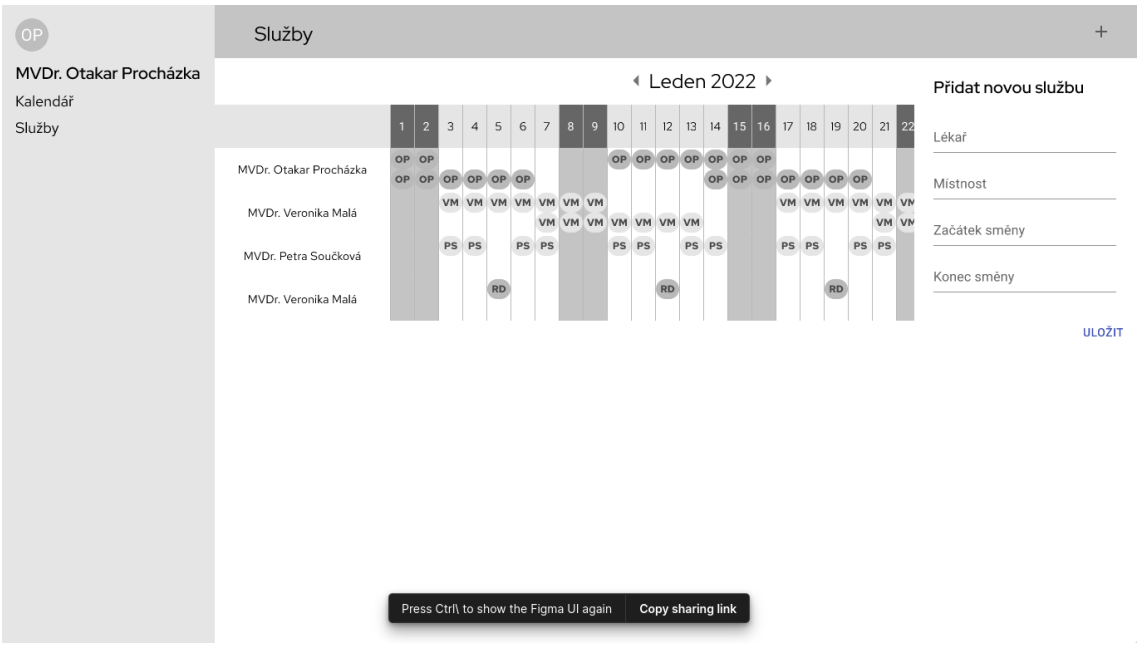
Na základě zpracovaných požadavků na systém (Obrázek 4-1) jsem vypracovala návrh aplikace tzv. wireframe. Návrh slouží k rozvržení jednotlivých funkcionalit systému a ujištění se, že žádný požadavek nebude opominut. Návrh si neklade za cíl přesně navrhnout grafické zpracování, jedná se jen o hrubou kostru jednotlivých prvků systému (Obrázek 4-4, Obrázek 4-5, Obrázek 4-6).



Obrázek 4-4: Přehled objednávek společně s formulářem pro zadání nové objednávky



Obrázek 4-5: Formulář pro zadání nového důvodu návštěvy

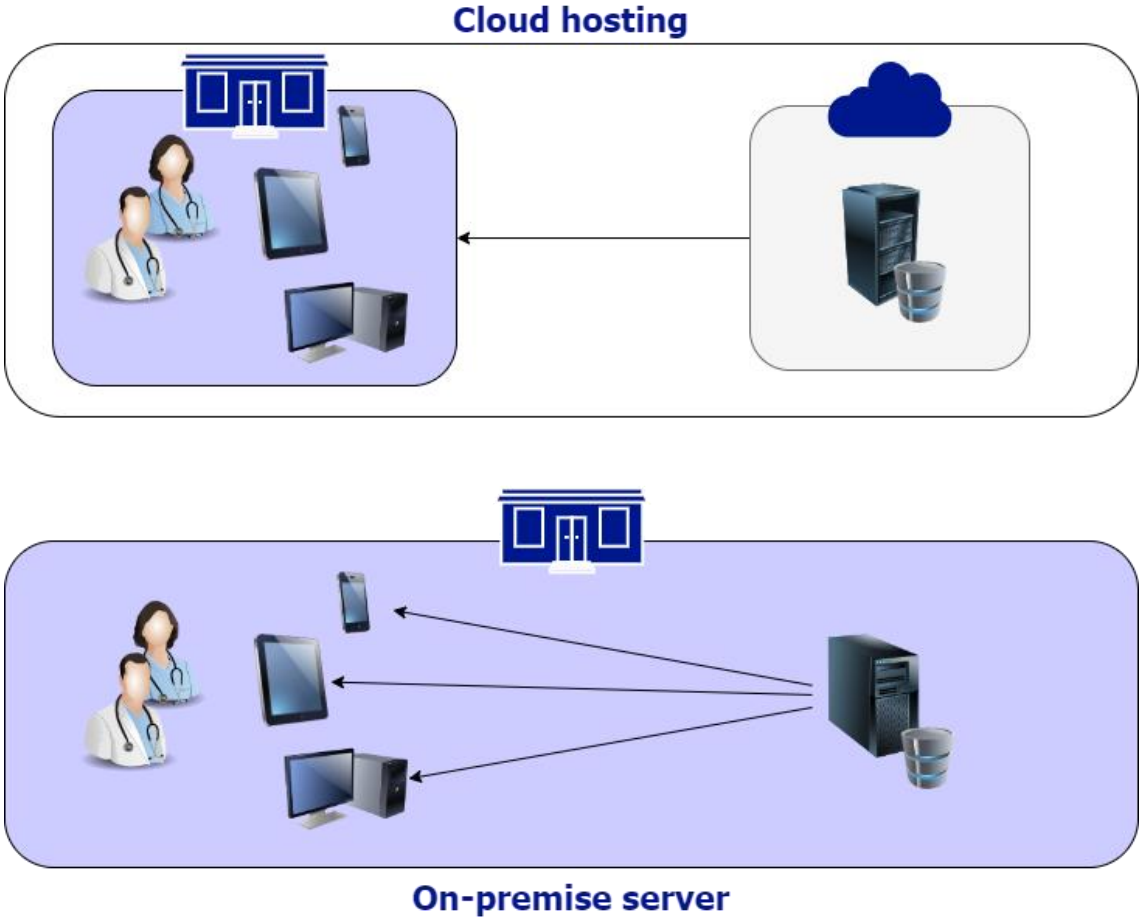


Obrázek 4-6: Přehled směn společně s formulářem pro zadání nové směny

Wireframe byl zpracován pomocí nástroje Figma. Tento nástroj umožňuje vytvořit interaktivní prototyp, ve kterém lze velmi dobře nasimulovat výsledné chování aplikace. Návrh objednávkového systému je dostupný na adrese https://www.figma.com/file/NzIC7pb8luZvBtAvZt9Izz/vetCal-bakalarska-prace?type=design&node-id=122%3A1093&t=aGiTQSTTcCK8PYfi-1

4.4 Technická specifikace

Jako vhodné řešení objednávacího systému byla navrhnuta architektura webové aplikace. Díky tomuto způsobu jsme schopni aplikaci nasadit jak v cloudovém prostředí a klienti ji mohou ovládat přes internet, ale zároveň je i možné ji nasadit na servery spravované konkrétní klinikou a přistupovat k ní prostřednictvím vlastního intranetu (Obrázek 4-7). Veterinární pracoviště se tak samo může rozhodnout, zda je pro něj důležitější mít absolutní kontrolu nad daty a ukládat je tzv. on-premise nebo by raději svěřila správu dat dodavateli cloudové platformy, který za ni může řešit problémy s výpadky služby nebo zálohování dat. Zároveň samotné škálování aplikace v cloudu je jednodušší než na vlastních serverech, kdy se musíme starat i o hardware a doplňovat jeho kapacitu v případě potřeby. Aplikace bude univerzální i z toho hlediska, že nebude vázaná na konkrétní cloudové nebo hostingové prostředí. Oboje řešení umožňuje nastavení přístupu odkudkoliv, nejen z počítačů na klinice.



Obrázek 4-7: Schéma možnosti nasazení

Webové aplikace jsou specifické tím, že samotný program běží na serveru a uživatelé se k němu připojují pomocí webové prohlížeče nainstalovaném na desktopu, tabletu nebo chytrém telefonu. Díky tomuto systému provozu, je nutné aplikaci nainstalovat pouze na serveru, ale jednotlivé klientské stanice už se nastavovat nemusí, rozšiřování počtu míst,

odkud má být aplikace dostupná, není problém. Druhou velkou výhodou je, že aplikace není přístupná pouze ze zařízení na pracovišti, ale zaměstnanci se k ní dostanou i na svých osobních zařízeních.

Typické workflow webové aplikace vypadá takto:

1. Uživatel prostřednictvím webového prohlížeče odešle požadavek.
2. Daný požadavek se přenese skrze internet nebo intranet v závislosti na umístění serveru.
3. Webový server přijme požadavek a předá ho aplikačnímu serveru.
4. Aplikační server požadavek zpracuje a podle jeho povahy se případně doptá na data z databáze.
5. Aplikační server předá odpověď webovému serveru.
6. Webový server odešle odpověď skrz internet/intranet.
7. Webový prohlížeč zobrazí odpověď uživateli v uživatelském rozhraní.

Struktura webových aplikací se dá rozdělit na frontend a backendu, kdy backend se stará o správu dat na serveru a frontend tvoří uživatelské rozhraní, které pracuje s daty na backendu. Po průzkumu současných dostupných technologií jsem pro tvorbu backendu, který bude odpovídat standardu FHIR vybrala open-source server HAPI FHIR, který je dostupný pod licencí Apache Software License 2.0 a zprostředkovává standardizované API. Frontend bude napsán v JavaScriptu pomocí knihovny React, která nabízí širokou škálu předdefinovaných funkcionalit a mnoho dalších kompatibilních knihoven jako je Redux pro správu dat v aplikaci a Material UI, která dodává grafické interaktivní prvky pro aplikaci. Díky kvalitně popsanému standardu FHIR a jeho komunikaci pomocí REST nakonec nebyla zvolena žádná knihovna pro práci s FHIR Resource a o vše se stará aplikace sama. Deployment aplikace bude řešen pomocí technologie Docker.

4.4.1 HAPI FHIR server

HAPI FHIR [25] je otevřený a plně interoperabilní server pro správu a výměnu dat ve formátu FHIR. HAPI FHIR server poskytuje prostředí pro uložení, vyhledávání, správu a výměnu zdravotnických dat pomocí FHIR API. Je napsán v jazyce Java a díky jeho otevřenému kódu je možné ho upravovat dle svých potřeb. Nicméně je dostupné hotové řešení, které se jen nainstaluje a je připravené k použití. Já jsem si pro implementaci vybrala řešení, které je dostupné jako Docker image a dá se nainstalovat a spustit pomocí Dockeru.

HAPI FHIR server umožňuje spravovat a ukládat informace o pacientech, včetně jejich kontaktních informací a veškerých informací spojených s objednáváním návštěv. Server poskytuje pokročilé vyhledávací funkce, které umožňují filtrovat a hledat zdravotnická data podle různých kritérií, jako jsou identifikátory pacientů, směny podle lékařů nebo místností, či volná místa pro objednání návštěvy. Díky podpoře FHIR standardu je navržen tak, aby byl plně interoperabilní s ostatními systémy založenými na

FHIR. Podporuje standardní FHIR API a datové modely, což umožňuje snadnou výměnu dat mezi různými systémy. HAPI FHIR server podporuje verzování dat, což umožňuje sledovat změny v zdravotnických datech a uchovávat historii úprav.

4.4.2 React

React je populární JavaScriptová knihovna pro tvorbu uživatelských rozhraní. Byla vyvinuta společností Facebook a používá se pro vývoj moderních webových a mobilních aplikací. React se zaměřuje na deklarativní a efektivní vytváření uživatelských rozhraní pomocí komponent.

Hlavní myšlenkou Reactu je rozdělit uživatelské rozhraní do samostatných a znovupoužitelných komponent. Každá komponenta reprezentuje část uživatelského rozhraní, která má vlastní logiku a zobrazovací pravidla. Komponenty mohou být vnořeny do sebe, což umožňuje vytvářet hierarchii a složité struktury.

React využívá virtuální DOM (Document Object Model), což je abstraktní reprezentace skutečného DOM v paměti. Při změně stavu komponenty React porovnává virtuální DOM s reálným DOM a provádí pouze nezbytné aktualizace, což zvyšuje efektivitu a výkon aplikace.

Další významnou vlastností Reactu je jednosměrný tok dat. Data jsou předávána komponentám z tak zvaných parent komponent a nejsou měnitelná z children komponent [26].

React je také často kombinován s dalšími knihovnami a nástroji, jako je například Redux pro správu stavu aplikace nebo React Router pro správu navigace v jednostránkových aplikacích. Obě zmíněné knihovny budou využity i v objednávkovém systému.

4.4.3 Redux

Redux je populární JavaScriptová knihovna pro správu stavu aplikace. Redux se stejně jako React řídí konceptem jednosměrného toku dat, což znamená, že data v aplikaci cestují jedním směrem od jednoho místa k druhému. Hlavními stavebními kameny Reduxu jsou:

Store: Store je centralizovaný kontejner pro stav aplikace. Uchovává všechny data, která jsou potřebná pro správnou funkčnost aplikace. Store je jediným místem, kde lze změnit stav aplikace.

Actions: Akce jsou objekty, které popisují nějakou událost nebo požadavek na změnu stavu aplikace. Každá akce musí obsahovat typ (typicky vyjádřený řetězcem) a další potřebné data pro provedení změny stavu.

Reducers: Reducers jsou čisté funkce, které přijímají aktuální stav aplikace a akci a vracejí nový stav. Reducers jsou zodpovědné za provedení skutečné změny stavu na základě akce.

Dispatching: Dispatching je proces odesílání akce do Reduxu. Akce se odesílají pomocí funkce dispatch, která je volána ve správném kontextu aplikace.

Subscribers: Redux umožňuje registrovat subscribers, kteří se zaregistrují na změny stavu a jsou automaticky upozorněni, když dojde ke změně. To umožňuje aktualizovat uživatelské rozhraní podle nového stavu.

Redux usnadňuje správu stavu aplikace tím, že odděluje logiku stavu od komponent, což zlepšuje přehlednost a škálovatelnost. Díky jednoznačnému toku dat je také snazší ladit, testovat a udržovat aplikaci. Zároveň existují developerské nástroje pro prohlížeče, které přehledně zobrazují, jaká data jsou ve store a jak probíhal jejich tok v čase [27].

Další knihovnou, která funguje nad Reduxem a předkládá doporučené použití knihovny Redux s knihovnou React je Redux Toolkit. Jako nadstavbu nabízí standardizovanou komunikaci s API externích serverů, v našem případě využitelné pro komunikaci se serverem HAPI FHIR. Nabízí následnou nadstavbu nad Reduxem:

Slice: Slice je koncept, který umožňuje definovat reducers, akce a selektory pomocí jednoduché syntaxe. Slice je samostatný modul, který zahrnuje všechny potřebné definice pro práci s konkrétní částí stavu aplikace.

createSlice: createSlice je funkce, která automaticky generuje akce, reducers a selektory pro daný Slice.

configureStore: configureStore je funkce, která vytváří Reduxový store s výchozí konfigurací. Tato funkce usnadňuje inicializaci Reduxu a poskytuje některé doporučené výchozí nastavení.

Immer: Redux Toolkit implicitně používá knihovnu Immer, která zjednodušuje práci s neměnitelným stavem aplikace. Immer umožňuje provádět změny na stavu pomocí běžných mutačních operací, přestože ve skutečnosti vytváří novou neměnitelnou kopii stavu.

Redux Toolkit usnadňuje a zjednodušuje vývoj s Reduxem, eliminuje opakující se a zdlouhavý kód a přináší s sebou několik vylepšení a konvencí pro psaní Reduxových aplikací. Je navržen tak, aby urychlil vývoj a snížil množství příručního kódu potřebného pro správu stavu aplikace [28].

4.4.4 Material UI

Material-UI je open-source knihovna komponent pro React, která implementuje vizuální designový systém nazvaný Material Design. Material Design je designový jazyk vyvinutý společností Google pro tvorbu moderních a atraktivních uživatelských rozhraní

na různých platformách. Vzhledem k jeho širokému rozšíření je mnoho uživatelů navyklých na tyto komponenty a práce s nimi jim přijde intuitivnější než s individuálně vytvořenými komponenty.

Material-UI poskytuje širokou škálu předem navržených a stylizovaných komponent, které mohou být použity pro vytváření responzivního a přitažlivého uživatelského rozhraní. Knihovna nabízí komponenty jako tlačítka, formulářové prvky, navigační panely, tabulky, modální okna, dialogy, snackbar atd.

Hlavní vlastnosti a výhody Material-UI zahrnují:

Material Design: Material-UI dodržuje principy a vzhled Material Designu. Komponenty jsou navrženy tak, aby měly jednotný a moderní vzhled, včetně stínování, vrstevnatosti, barev a animací.

Responzivita: Komponenty Material-UI jsou plně responzivní a lze je snadno přizpůsobit různým zařízením a velikostem obrazovky.

Rozšiřitelnost: Material-UI poskytuje možnost rozšíření komponent díky flexibilnímu API a možnosti vytvářet vlastní stylizované komponenty.

Dobrá dokumentace: Material-UI má podrobnou dokumentaci s příklady a ukázkami, které pomáhají vývojářům rychle se seznámit s knihovnou a efektivně ji používat.

Téma a stylizace: Material-UI umožňuje snadno upravit vzhled komponent pomocí možnosti vlastního nastavení tématu, barev a stylů [29].

4.4.5 Docker

Docker je platforma pro kontejnerizaci, která umožňuje balení, distribuci a spouštění aplikací a jejich závislostí do izolovaných prostředí nazývaných kontejnery. Docker umožňuje snadnou a konzistentní nasazování aplikací napříč různými prostředími, jako jsou vývojové, testovací a produkční prostředí.

Kontejnery jsou samostatné provozní prostředí, které obsahuje všechny potřebné komponenty pro spuštění aplikace, včetně kódu, běhového prostředí a závislostí. Kontejnery izolují aplikaci od hostitelského operačního systému a poskytují konzistentní prostředí pro spuštění aplikace na různých platformách.

Dockerfile je textový soubor, který obsahuje instrukce pro sestavení Docker kontejneru. Pomocí Dockerfile definujete, jaké závislosti a konfigurace jsou potřebné pro vaši aplikaci. Soubor obsahuje příkazy jako kopírování souborů, instalace softwaru a konfigurace.

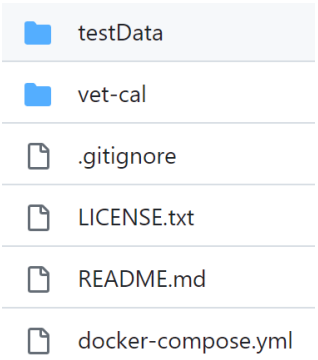
Docker image je spustitelný balíček, který obsahuje všechny potřebné komponenty pro spuštění aplikace. Obraz je vytvořen na základě Dockerfile a může být nasazen do kontejneru.

Registry jsou repositáře pro ukládání a sdílení Docker image. Veřejným registrem je například Docker Hub, kde se nachází i volně dostupný Docker image pro HAPI FHIR server.

Pomocí Dockeru lze snadno sestavit Docker image aplikace pomocí Dockerfile, následně spustit kontejner na základě tohoto obrazu a aplikaci nasadit na libovolném systému, který podporuje Docker. Docker poskytuje jednotný a konzistentní způsob distribuce aplikací, což usnadňuje nasazování a škálování.

5 Implementace

Zdrojový kód aplikace je dostupný skrze veřejný repositář na serveru GitHub: https://github.com/vetsystem/vetCalendar nebo je přiložený jako komprimovaná složka ZIP k této práci. Kód je publikovaný pod licencí MIT. Pro spuštění aplikace je zapotřebí buď rozbalení ZIP složky nebo naklonování repositáře pomocí verzovacího programu Git. Výsledkem je lokálně stažený projekt (Obrázek 5-1) s následující strukturou:



Obrázek 5-1:Struktura projektu

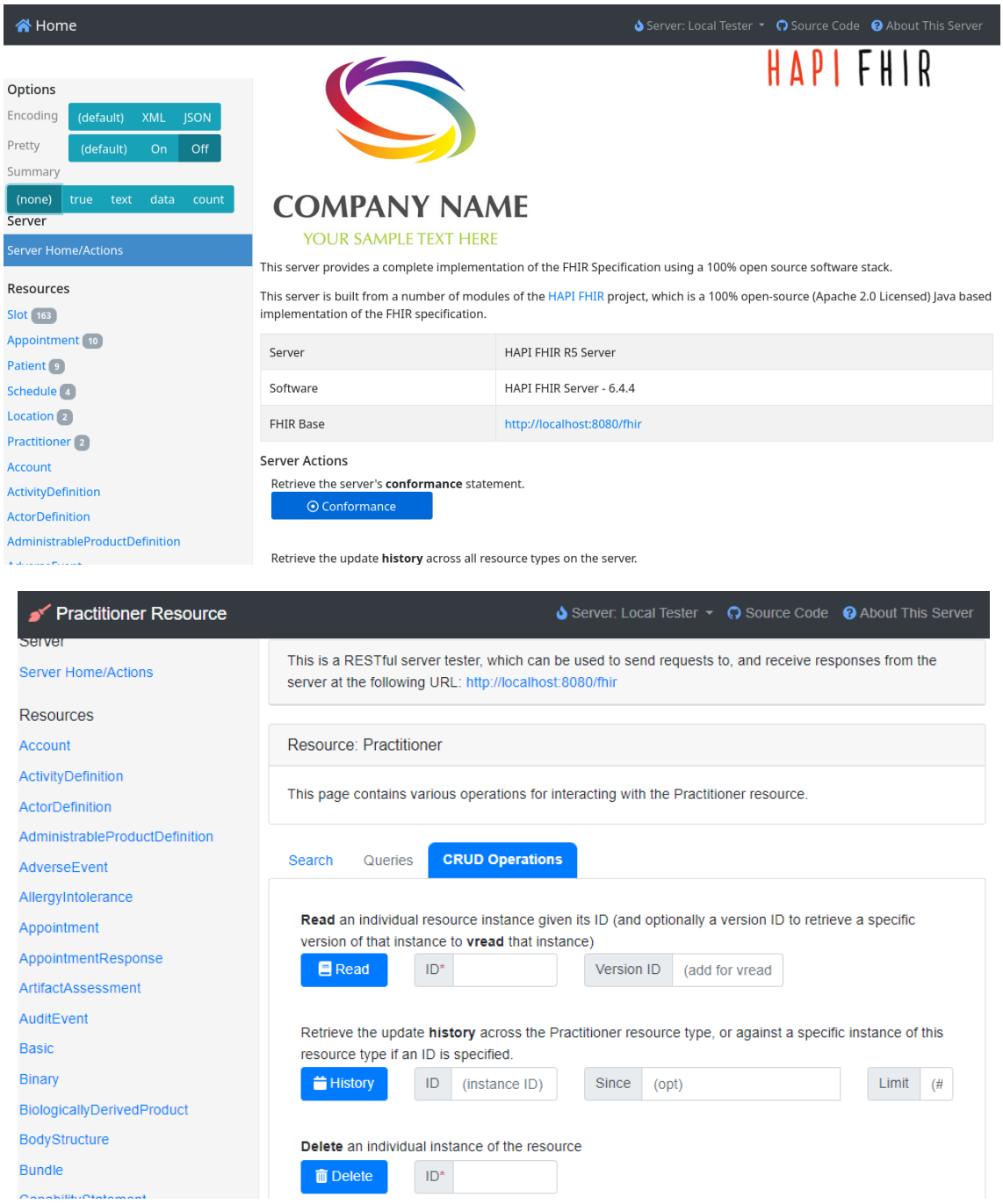
Aplikace je rozdělena do tří stavebních bloků:

- HAPI FHIR server (backendu)
- Testovací/inicializační data
- React aplikace pro objednávání (frontend)

Tyto stavební bloky jsou uzavřeny do Docker image a je vytvořen docker-compose.yml soubor pro jejich jednoduché nasazení v jakémkoliv prostředí. K lokálnímu spuštění systému je proto nutné mít nainstalovaný Docker Desktop [30] (nebo jeho části Docker Engine a Docker Compose plugin). Systém se spustí z adresáře vetCalendar pomocí příkazu z terminálu `docker compose up`. První spuštění bude trvat několik minut, protože Docker musí stáhnout a sestavit všechny součásti systému, jakékoliv další spuštění již nebude tak časově náročné, pokud nedojde k vymazaní Docker kontejnerů jednotlivých součástí systému.

Konfigurační soubor docker-compose.yml říká Dockeru aby z veřejného repositáře pro Docker image stáhl obraz pro FHIR HAPI server a spustil ho v samostatném kontejneru na adrese http:// localhost:8080. Dále spouští sestavený Docker kontejner, který obsahuje testovací data a script pro nahrání dat na server. Tento script si hlídá okamžik, kdy bude server dostupný a následně data nahraje. Jako poslední se spouští React aplikace na adrese http:// localhost:3000, která zobrazí uživatelské rozhraní.

Tyto jednotlivé Docker image lze spustit i samostatně, nicméně je potřeba pamatovat na to, že uživatelské rozhraní pro svůj běh potřebuje mít spuštěný server a image s testovacími daty je nastaven pro spuštění pomocí příkazu `docker compose up` společně se serverem, jinak nenajde adresu serveru a data nenahraje. Nicméně lze využít soubor s daty testData/VetScheduling-Overall.json pro jeho editaci a samostatné nahrání pomocí příkazu curl nebo webového rozhraní HAPI FHIR serveru (Obrázek 5-2). Zde je nutné ho nahrát jako FHIR Resource Bundle, který slouží k hromadnému nahrání dat. Případně lze pomocí tohoto rozhraní data nahrávat jednotlivě, podle své potřeby.



Obrázek 5-2: Webové rozhraní serveru HAPI FHIR

Webové rozhraní serveru HAPI FHIR je používané také k nahrání FHIR Resource, které se v uživatelském rozhraní pouze zobrazují ale needitují. Jsou to entity lékařů a místností. Je na ně nahlíženo jako na pevně dané konstanty, které by se v programu neměly měnit, pouze se zadají před spuštěním. Je k tomu využíváno CRUD operací CREATE, UPDATE a DELETE a jako obsah požadavku se posílá JSON (Obrázek 5-3) odpovídající standardu FHIR. Kromě těchto metod nabízí toto webové rozhraní i validaci FHIR Resource. Serveru se poskytne JSON s Resource a on nám vrátí informaci, zda se jedná o položku, která odpovídá standardu FHIR a je možné ji nahrát na server.

```
{
      "resourceType": "Practitioner",
      "name": [
        {
          "given": ["Jan"],
          "family": "Novotný",
          "text": "MVDr. Jan Novotný",
          "prefix": ["MVDr."]
        }
      ]
    },
    "request": {
      "method": "POST",
      "url": "Practitioner",
      "ifNoneExist": "/Practitioner?family=Novotn%C3%BD&given=Jan"
    }
  },
```

Obrázek 5-3: Příklad CREATE požadavku pro vytvoření lékaře

Kromě vestavěného webového rozhraní lze ke CRUD operacím využívat jakéhokoli CRUD klienta jako je například aplikace Postman. Výhodou těchto aplikací je, že nám umožňuje ukládat dotazy a odpovědi pro opakovatelné použití, upravuje formát JSON do čitelnější podoby a zjednodušuje práci s vyhledávacími parametry url adres.

Data jsou na serveru ukládána pouze po dobu existence Docker kontejneru, po jeho smazání jsou smazán i data, protože jsou ukládána uvnitř kontejneru. Pro další vývoj aplikace by bylo vhodné ukládat data mimo kontejner, aby byla nezávislá na jeho životnosti, zároveň by ale také bylo vhodnější využít samostatný kontejner s databází pro případné rozšíření. Aktuální prototyp využívá databáze H2, která je součástí HAPI FHIR kontejneru.

Aplikace obsluhující frontend je napsána pomocí knihovny React. React využívá především JavaScript a JSX, aby do základní kostry HTML stránky vložil dynamické prvky. HTML tak tvoří velmi malou část výsledné aplikace. Díky této knihovně je možné aplikaci rozdělit na menší stavební bloky, takzvané komponenty a případně je využívat

na více místech, pokud se opakují. Cílem je co nejpodrobnější rozdělení, aby každá komponenta měla za úkol jeden jediný úkol. React [26] nabízí dva přístupy k psaní komponent, a to klasické komponenty typu třídy a potom funkční komponenty (Obrázek 5-4). V aplikaci byly použity výhradně funkční komponenty, protože se jedná o doporučené použití v nových aplikacích. Většina komponentů je uložena ve složce vet-cal/src/components (o výjimkách později) a dále dělena podle umístění v uživatelském rozhraní. Název každé komponenty začíná velkým písmenem, což je specifikace Reactu a vrací nám grafické zobrazení prvku pomocí JSX syntaxe.

```
8   export default function ShiftsView() {
9     return (
10      <Box
11        component="main"
12        sx={{ flexGrow: 1, bgcolor: "background.default", p: 3 }}
13      >
14        <Toolbar />
15        <ShiftTable />
16      </Box>
17    );
18  }
```

Obrázek 5-4: Příklad funkčního React komponentu

Z důvodů snadného a intuitivního ovládaní jsem vybrala knihovnu Material UI [29]. Obsahuje hotové komponenty s ovládacími či zobrazovacími prvky. Jsou to prvky, které využívá i Google, proto je velké množství lidí zvyklé ji používat. Jedná se prvky jako jsou tlačítka, textová pole, našeptávače nebo avatary. Knihovna Material UI řeší především vizuální stránku aplikace ke které patří i správa použitých barev a fontů, slouží k tomu soubor theme.js, kde jsou dané proměnné nakonfigurované a v aplikaci jsou pak používány tyto proměnné.

Knihovna Material UI nedisponuje vhodným zobrazením kalendáře pro zaznamenávání návštěv, proto jsem použila jiné řešení, a to knihovnu React Scheduler Component. Jako jedna z mála je nezpoplatněná a řeší problematiku zobrazení kalendáře s více zdroji, v tomto případě potřebuji v rámci jednoho dne zobrazit návštěvy alokované v různých místnostech nebo u různých lékařů. Nicméně i přesto plně nevyhovovala zamýšlenému účelu. Konkrétně formulář pro zadávání nové návštěvy se zobrazuje jako modální dialog, čímž znemožňuje interakci s kalendářem během zadávání nové návštěvy. Proto jsem tuto knihovnu doupravila a zdrojový kód obsahuje komprimovanou upravenou verzi vet-cal/dependencies/aldabil-react-scheduler-2.7.9_1.tgz, která je instalovaná

společně s ostatmími knihovnami během instalace React aplikace pomocí příkazu `npm install` nebo `npm ci`.

O datové modely se na straně frontendu stará knihovna Redux [27]. Díky Reduxu je možné vytvořit takzvaný Store, kde lze ukládat data, která jsou pak přístupná kdekoliv z aplikace. Nadstavbou této knihovny je Redux Toolkit [28], který popisuje best practices použití Reduxu. Stará se i o stahování a nahrávání dat na backend, cachování těchto dat a jejich normalizaci. V aplikaci je soubor /app/store.js, který inicializuje redux a soubor /api/fhirApi.js, kde je nastavení pro připojení na Fhir server. Dále má každý datový model svůj Slice soubor (např. doctorsSlice) ve kterém jsou specifikovány CRUD metody a struktura datového modelu. V případě, že se jedná o datový model, který nepracuje s beckendem, řeši soubor Slice pouze metody ve vztahu k frontendu. Společně se Slice souborem je ve složde i React komponenta, které k s datovým modelem přímo pracuje. Všechny tyto zmíněné soubory jsou pak ve složce features, která sdružuje prvky, které pracují s daty. Konkrétně se jedná o tyto modely:

Modely mapované na FHIR Resource

- appReasons (id, type, label, room, doctor, duration)
- appointments (event_id, title, start, end, assignee, doctorId, pacientId)
- doctors (id, type, resourceType, name, initials, color)
- patients (name, id, bd, species, owner, contact)
- rooms (id, type, resourceType, name, assignee)
- shifts (id, doctor, room, start, end)

Modely využité pouze na frontendu

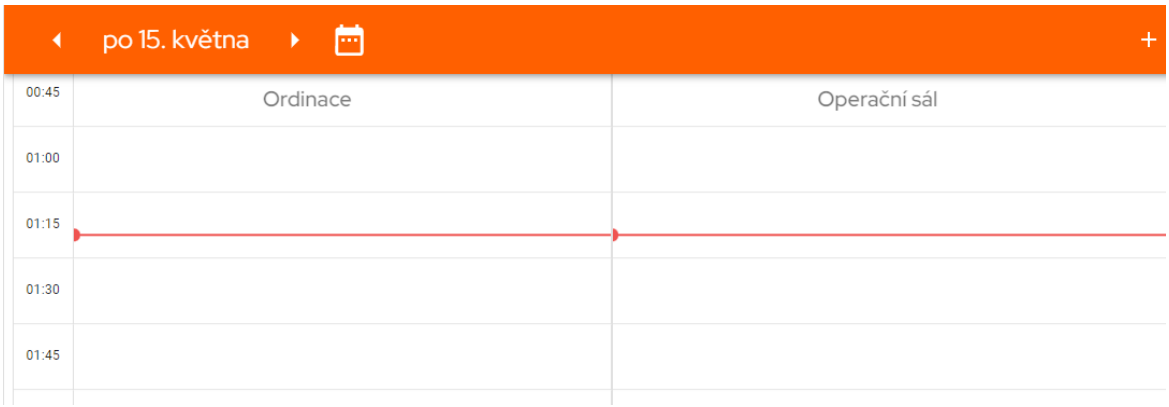
- calendar (selectedDate)
- window (height)

Modely určené pro frontend obsahující pouze pár proměnných, ukládajících informaci o aktuálním zobrazení.

Poslední částí aplikace je složka utils, která shrnuje různé pomocné funkce pro práci s daty v aplikaci, mimo jiné i funkce pro transformaci FHIR Bundle a Resource na datové modely aplikace a naopak. I přes pestrou nabídku knihoven pro práci s FHIR na klientovi, jsem se nakonec rozhodla pro toto vlastní řešení, protože systém komunikace aplikace s API FHIR serveru byla mnohem lépe řešena knihovou Redux Toolkit, která sice není uzpůsobena Resource FHIR, ale kvalitně zpracovaná dokumentace tohoto standardu mi umožnila tento nedostatek vyřešit vlastním kódem.

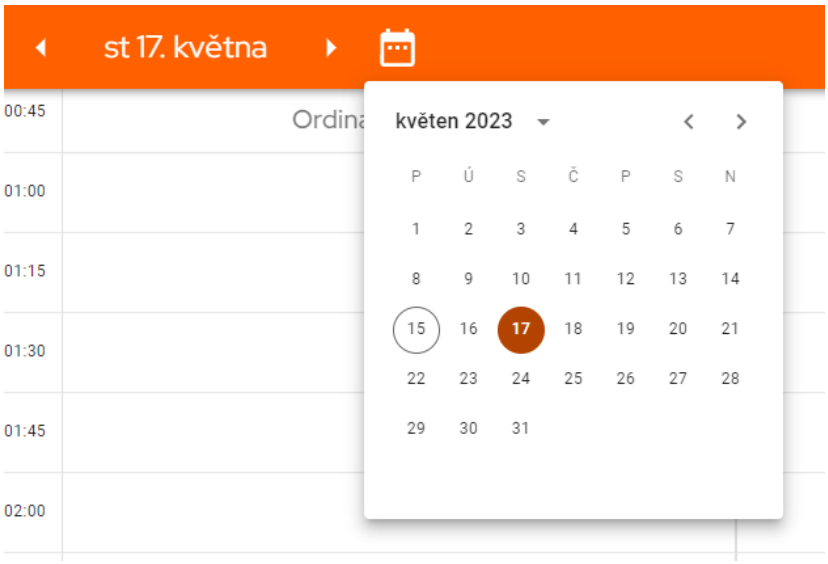
6 Uživatelská dokumentace

Uživatelské prostředí je zpracováno tak, aby bylo co nejintuitivnější a nebylo potřeba se se systémem učit před jeho použitím. Pro práci s aplikací je potřeba mít nainstalovaný webový prohlížeč. Aplikace by měla fungovat v prohlížečích Chrome, Firefox a Edge. Po otevření prohlížeče se zadá do řádku pro adresu http://localhost:3000. Otevře se okno s lištou ukazující aktuální datum. Pod ním bude běhat indikátor, že se nahrávají data. V okamžiku, kdy budou data dostupná zobrazí se vertikální denní časová osa se sloupci pro jednotlivé místnosti. Červená linie bude ukazovat na aktuální čas (Obrázek 6-1).



Obrázek 6-1: Zobrazení aktuálního dne

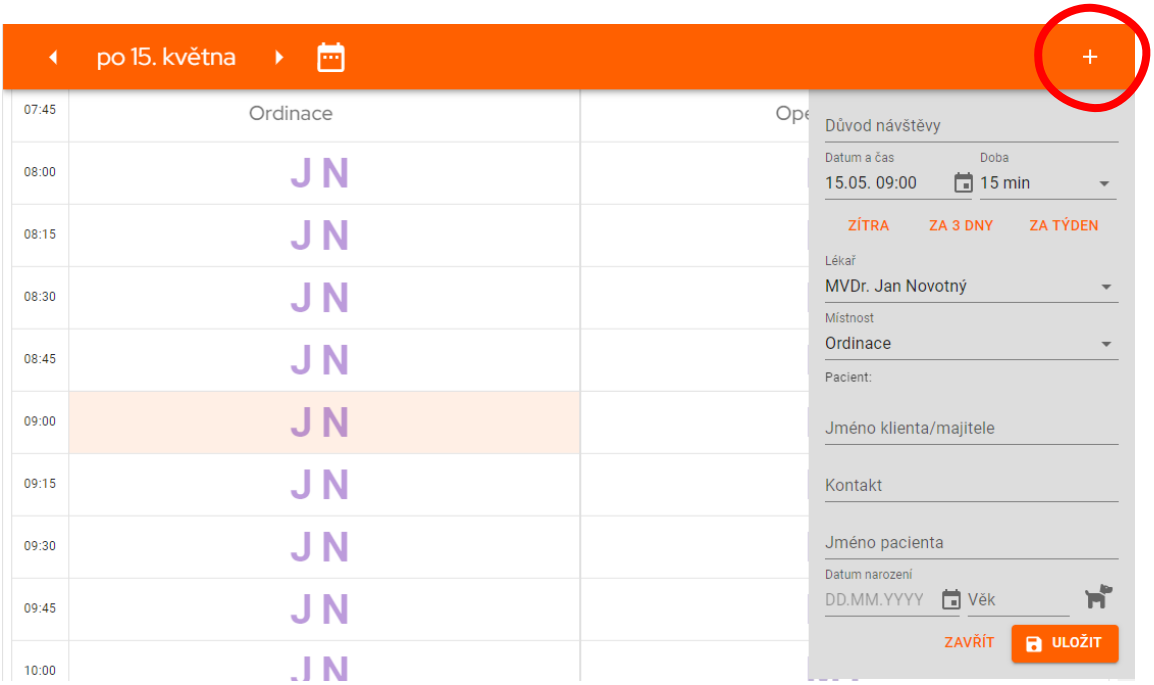
Na horní liště ukazující aktuální datum najdeme i dvě ikony se šipkami pro přepínání mezi předchozím a následujícím dnem. Napravo od aktuálního data a šipek je ikona s kalendářem. Po kliknutí na ni se otevře kalendář s měsíčním zobrazením (Obrázek 6-2), kde můžeme vybrat jakékoliv datum. Aktuální den zůstane vždy zakroužkovaný a vybraný den bude barevně odlišen.



Obrázek 6-2: Vybrání data k zobrazení

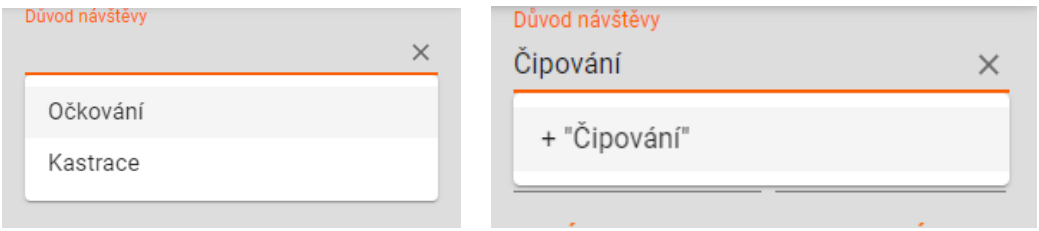
6.1 Správa návštěv

V pravém horním rohu obrazovky je ikona plus pro otevření formuláře k uložení nové návštěvy (Obrázek 6-3). Stejně tak se formulář otevře, pokud klikneme na volné časové políčko v denní zobrazení. Při najetím na něj myší se vysvítí červeně. V případ výběru políčka v časové ose se předvyplní známé hodnoty ve formuláři, jako jsou datum a čas, místnost a lékař (pokud má někdo v tomto čase rozepsanou službu). Vypsané služby se v kalendáři zobrazí pomocí barevných iniciál lékaře v příslušném časovém políčku. V případě kliknutí na jiné časové políčko při otevřeném formuláři pro novou návštěvu se zmíněné hodnoty zase upraví.



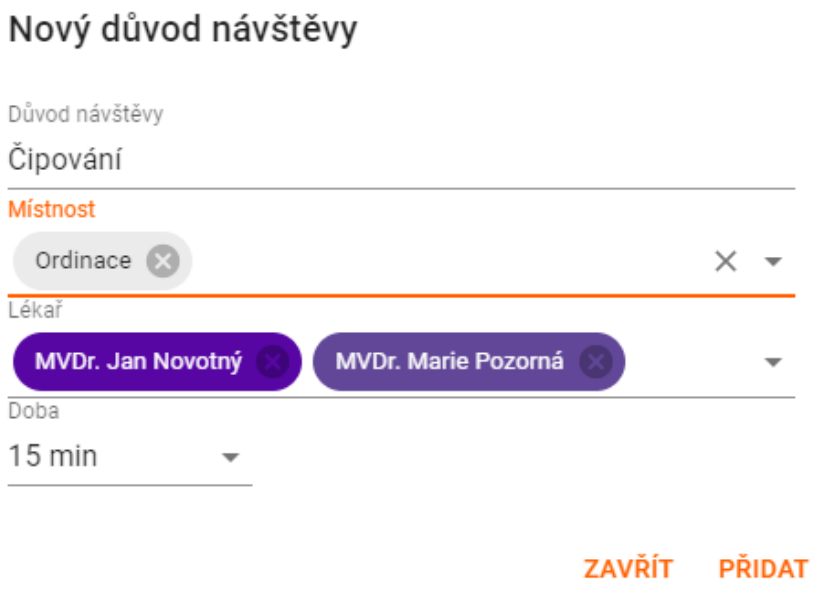
Obrázek 6-3: Otevření formuláře pro přidání návštěvy

Při kliknutí na textové pole Důvod návštěvy se rozbalí dostupná nabídka uložených důvodů. Postupně, jak budeme do pole psát bude se nabídka filtrovat. Pokud zadáme hodnotu, která v systému není uložená, nabídne nám danou hodnotu přidat mezi známé důvody návštěv. Pokud nechceme hodnotu ukládat, ale chceme ji ponechat pro danou návštěvu, klikneme na další pole. Pokud chceme hodnotu uložit klikneme na nabízenou volbu se znaménkem + (Obrázek 6-4).



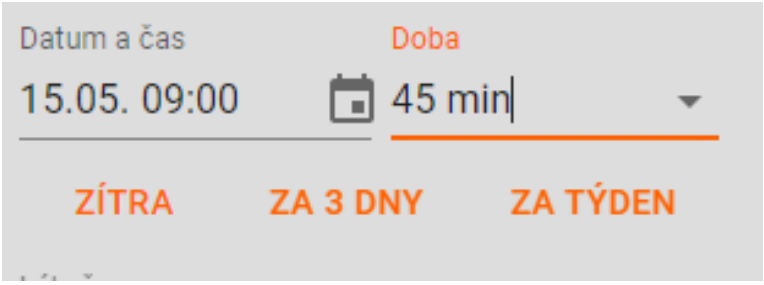
Obrázek 6-4:Přidání důvodu návštěvy

Po vybrání volby uložení nového důvodu návštěvy se otevře formulář pro vyplnění podrobností (Obrázek 6-5). Povinné pole je označení důvodu návštěvy. Další pole jsou volitelná. U výběru místnosti a lékaře lze vybrat více možností. Pokud je známa obvyklá doba trvání, vybere se i doba a důvod návštěvy se uloží tlačítkem Přidat. Pokud nechceme důvod návštěvy uložit stiskneme Zavřít.



Obrázek 6-5: Formulář pro nový důvod návštěvy

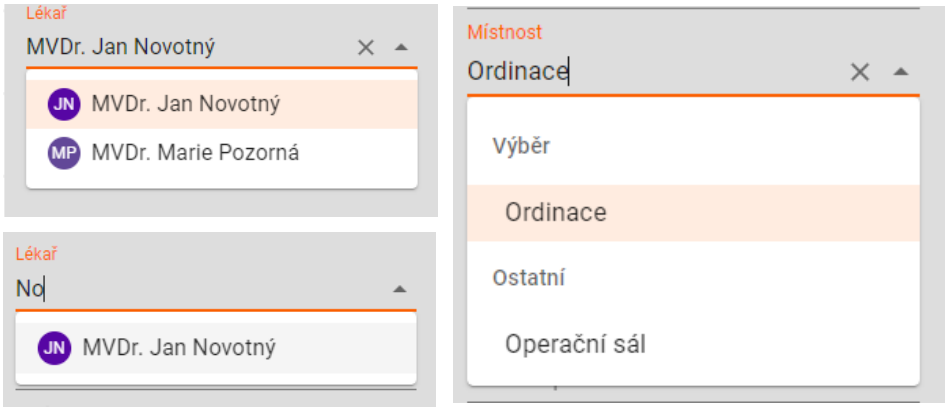
Vrátíme se k formuláři pro uložení návštěvy. Vyplníme datum, čas a dobu (Obrázek 6-6), pokud už není předvyplněno tím, že jsme formulář otevřeli kliknutím na volné časové políčko v přehledu návštěv. Jsou tu dostupné tři rychlé volby: zítra, za tři dny, za týden. Všechny možnosti jsou počítané od dnešního data. Po vybrání rychlé volby pravděpodobně bude ještě zapotřebí doplnit čas.



Obrázek 6-6: Výběr času a doby návštěvy

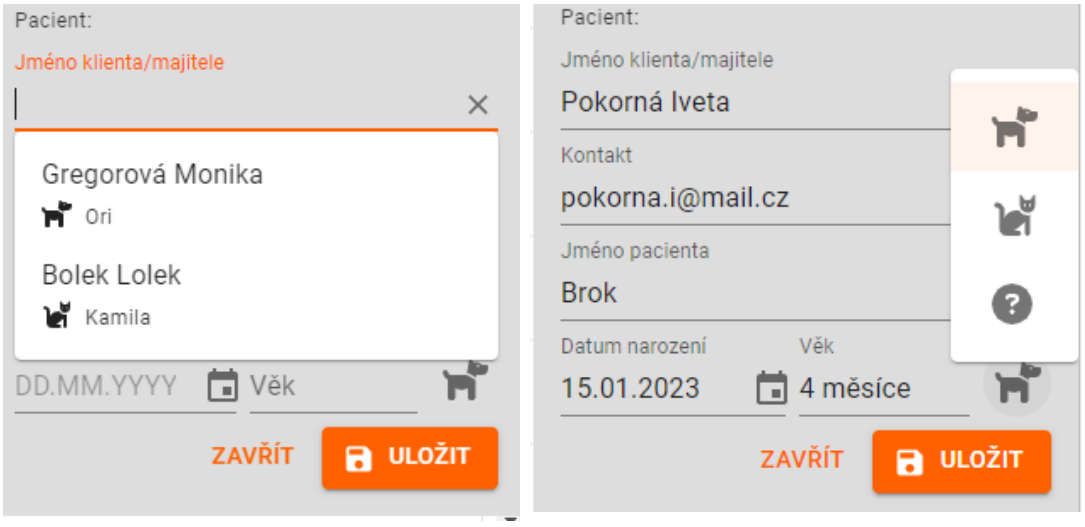
Dále vybereme lékaře a místnost, pokud již nejsou předvyplněné. V případě, že jsme vybrali uložený důvod návštěvy, který může vykonávat pouze specifický lékař, nebo ho lze provést jen v konkrétní místnosti, jsou tyto volby v nabídce oddělené (Obrázek 6-7).

V případě lékaře a místnosti lze vybrat pouze z předem definovaných voleb, nicméně je možné je vybírat psaním do textových polí, kdy se nám nabídka filtruje podle zadaného textu.



Obrázek 6-7: Výběr lékaře a místnosti

Poslední položkou je vyplnění informací o pacientovi. Pokud jsme už někdy pacienta objednávali, bude přítomný v nabídce, která se filtruje podle zadaného textu (Obrázek 6-8). Tato nabídka se otevře pří kliknutí na pole Jméno klienta/majitele nebo na pole Jméno pacienta. Při výběru známého pacienta se nám všechny pole předvyplní. Pokud u nás majitel byl, ale teď se chce objednat s jiným mazlíčkem, vybereme ze seznamu jeho jméno s původním zvířetem a informace o pacientovi přepíšeme. Pokud je klient i pacient nový, musíme údaje vyplnit ručně. Pokud známe přesné datum narození zvířete vyplníme ho. Pokud víme jen přibližný věk, zadáme hodnotu věku a přibližné datum narození se dopočítá. Informaci o druhu pacienta vybíráme z rozeviratelného seznamu po kliknutí na ikonu zvířete.

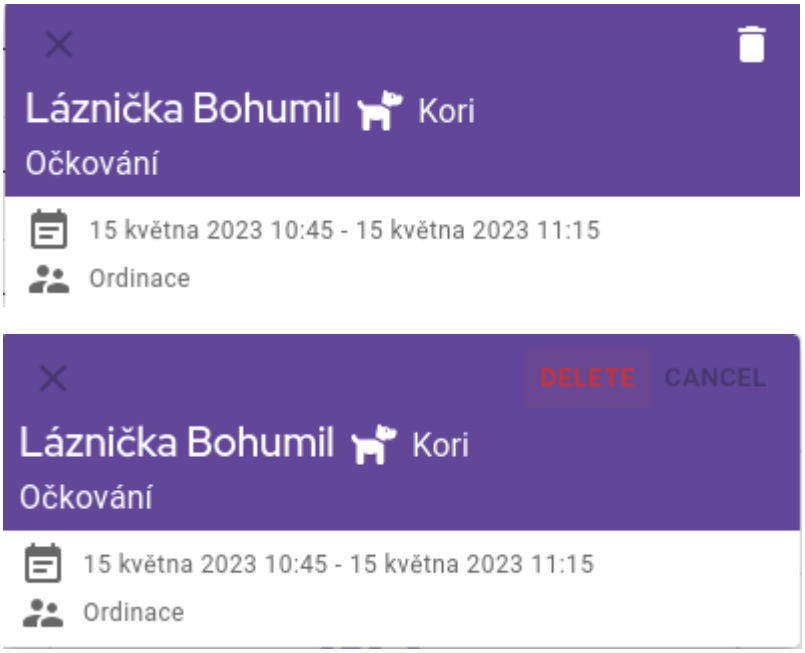


Obrázek 6-8: Zadání informací o pacientovi

Návštěvu uložíme kliknutím na tlačítko uložit nebo ji zahodíme kliknutím na tlačítko zavřít. Po uložení návštěvy se nám zobrazí v kalendáři v daný časový interval ve vybrané místnosti u konkrétního lékaře (Obrázek 6-9). Pokud pro daný časový úsek není vypsaná směna, návštěva se i přesto uloží.

Obrázek 6-9: Zobrazení návštěvy v kalendáři

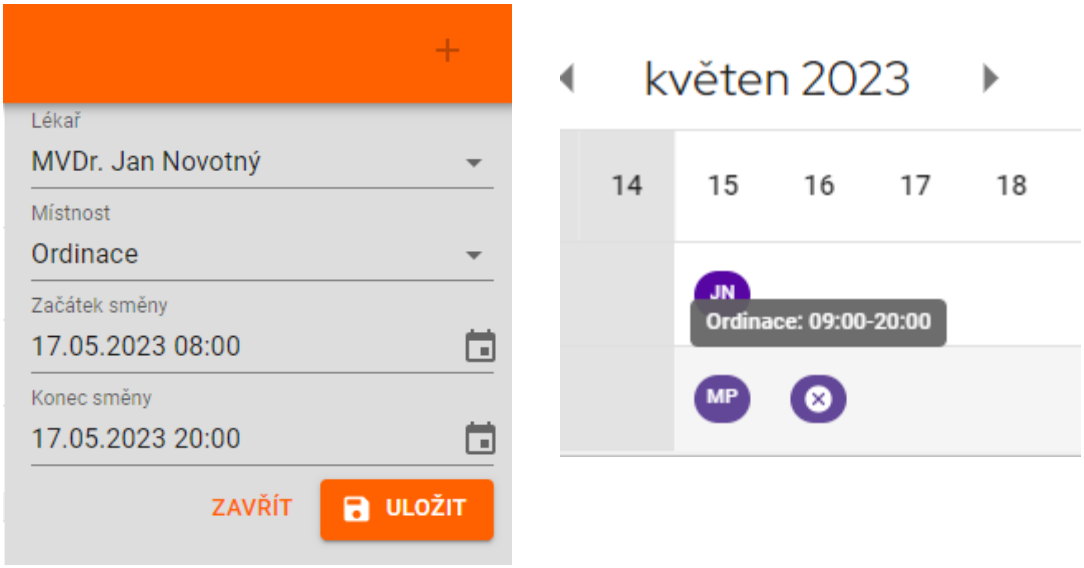
Pokud chceme vybranou návštěvy smazat, klikneme na ni v kalendáři. Otevře se nám okno s podrobnostmi a tlačítkem s ikonou odpadkového koše (Obrázek 6-10). Po kliknutí na tuto ikonu se zobrazí dvě tlačítka. Jedno pro potvrzení smazání, druhé pro zrušení požadavku.



Obrázek 6-10: Smazání návštěvy

6.2 Správa směn

Pro přidání směny je potřeba se přepnout na záložku směny. Odkaz najdete v levém panelu. Zde opět v pravém horním rohu je ikona plus, která otevře formulář pro přidání nové směny. Všechny údaje je potřeba vyplnit. Jméno lékaře a místnosti lze vybrat pouze z nabídky. Datum začátku a konce směny je nutné vybrat v budoucnosti. Pro uložení klikněte na tlačítko uložit. Pro zavření formuláře na tlačítko zavřít. Po uložení směny se hodnoty vynulují a po znovu načtení stránky bude směna viditelná na přehledu směn. Při najetí na konkrétní směnu se zobrazí zpráva s podrobnostmi (Obrázek 6-11).



Obrázek 6-11: Správa směn

Pro smazání směny stačí kliknout na křížek, který se objeví při najetí myši na konkrétní směnu. Dále je možné v přehledu směn přepínat mezi daty. Ovládání je podobné jako u přepínání mezi aktuálním dnem v zobrazení návštěv.

7 Výsledky

Po prostudování dostupných možností v oblasti aplikací pro objednávání klientů ve veterinární doméně a analýze požadavků na takovouto aplikaci jsem vytvořila funkční a technickou specifikaci, která by odpovídala současným trendům a pokryla nedostatky se kterými se potýká dostupné řešení. Součástí technické specifikace bylo vybrání vhodných technologií pro vývoj aplikace, která by mohla být hostovaná na serveru na klinice, ale přístupná z několika pracovních stanic. Dále bylo navrženo uživatelské rozhraní s důrazem na intuitivnost a jednoduchost ovládání. Jako doplnění studované problematiky jsem prozkoumala dostupné standardy pro předávání zdravotnických dat. Pro daný účel vyšel jako jediný vhodný standard HL7 FHIR. Nastudovala jsem jeho dokumentaci a navrhla konkrétní implementaci v aplikaci.

Po přípravné fázi jsem vytvořila prototyp objednávkového systému napsaného v jazyce JavaScript s použitím knihoven React, Redux a Material UI. Pro implementaci standardu FHIR jsem vybrala volně dostupný open-source server HAPI FHIR. Pro snadné nasazení systému jsou jednotlivé části systému uložené do Docker image a dohromady spustitelné jako jeden celek pomocí Docker Compose. Zdrojový kód je publikovaný na veřejném úložišti GitHub pod licencí MIT a zároveň přiložen k této práci.

Aplikace nebyla podrobena testování a ani není navržena pro vysokou zátěž. Nicméně ani se nepředpokládala, že by měla více jak pár desítek uživatelů celkem, vzhledem k tomu, že je cílena k použití na veterinárních pracovištích samotnými zaměstnanci. Další fází projektu bude najít vhodné pracoviště, které by prototyp otestovalo a zhodnotilo, zda má smysl v jeho vývoji pokračovat.

8 Diskuse

Cílem práce bylo vytvořit aplikaci pro objednávání klientů na veterinární pracoviště a zhodnotit použití standardu HL7 FHIR. Vývoj aplikace prošel standardním postupem vývoje softwaru, kdy nejdříve byly analyzovány požadavky na systém v kontextu současně dostupných řešení. Následně byla vyhotovena funkční a technická specifikace a vyhotoven návrh uživatelského rozhraní. Výsledkem je prototyp aplikace, který je připraven k testování uživateli.

Domnívám se, že aspekt použití standardu FHIR je nejspornějším bodem celé práce. Jedním z cílů bylo zhodnotit, zda tento standard vyhovuje použití ve veterinární doméně, konkrétně v tomto případě pro objednávání klientů. Z tohoto hlediska zcela vyhovuje a během průzkumu dostupných řešení jsem nenarazila na alternativu, která by byla v tomto případě využitelná. Na druhou stranu ale vyvstává otázka, zda je nutné jakýkoliv standard pro předávání zdravotnických dat využít. Vzhledem k tomu, že v aktuálním kontextu není potřeba, aby aplikace předávala jakákoliv data jinému systému, tak je jednoznačná odpověď, že standard nepotřebuje. Na druhou stranu aplikace je navržena tak, aby ukládala základní informace o pacientech, nabízí se tak myšlenka přímého napojení na systémy managementu praxe a případného přímého otevírání karet pacientů z objednávkového systému nebo na tak zvaný systém čekárny, který některé dostupné systémy mají. Evidují pacienty, kteří dorazili na pracoviště a čekají na ošetření. I v tomto případě je však standard FHIR aktuálně nepoužitelný, protože současné systémy ho nevyužívají. Když se oprostím od konkrétního případu využití pro objednávání a podívám se na jeho obecné použití ve veterinární medicíně není odpověď už tak jednoznačná. Jediným případem, kdy je standart FHIR spjatý s veterinární medicínou je využití Evropskou lékovou agenturou v jejich SPOR API (substance, product, organisation and referential application programming interface) [31]. Tato agentura se stará nejen o humánní léčivé přípravky, ale i o veterinární. Do budoucna je tedy možné, že legislativně bude požadováno, aby i veterinární medicína byla schopna předávat data ve zdravotnickém standardu. Vzhledem k tomu, že i české zdravotnictví se rozhodlo vydat cestou standardu FHIR [32], je možné že jeho využití ve veterinární medicíně bude aktuální. Jako příklad mě napadá používání eReceptů, kde mají dnes veterinární lékaři výjimku a používají stále papírové verze receptů, což se ale do budoucna může změnit.

Druhý aspekt použití standardu FHIR je, že nabízí hotové open-source řešení, díky němuž jsem se mohla zaměřit pouze na vývoj uživatelského rozhraní a správu dat přenechat serveru HAPI FHIR. Zprvu by se mohlo zdát, že tím se vývoj aplikace zjednodušil. Vzhledem ale ke komplexnosti standardu v kontrastu s nepříliš složitou datovou strukturou objednávání si myslím, že standard celé řešení spíše zkomplikoval a po zkušenosti s jeho implementací bych se asi příště rozhodla ho nevyužít. V případě

komplexnějších datových struktur, jaké jsou třeba u systémů managementu praxí bych ale už jeho využití zvážila.

Vrátím-li se zpátky ke konkrétní vytvořené aplikaci, tak cíl práce byl splněn. Nicméně aplikace není ve stavu, kdy by mohla být spuštěna v ostrém provozu. Největší důraz byl kladen na intuitivnost ovládání. Úspěch tohoto cíle lze hodnotit až o odzkoušení aplikace co největším počtem uživatelů a zároveň je potřeba aplikaci podrobit provozu s více daty a více uživateli najednou. Domnívám se, že následně zcela jistě bude potřebovat některé její části upravit, aby byla v praxi použitelná.

9 Závěr

Výsledkem práce je prototyp webové aplikace pro objednávání pacientů s implementací standardu HL7 FHIR. Cílem bylo zaměřit se na optimální uživatelské rozhraní, kde bude možné vytvářet a mazat objednané návštěvy a také vytvářet a mazat služby jednotlivých lékařů. Zadání dat do systému mělo být co nejrychlejší, aby práci s ním omezilo na co nejkratší dobu. Aplikace měla být navržena tak, aby data byla dostupná z několika pracovních stanic na pracovišti. Všechny zmíněné cíle se podařilo splnit.

Aplikace je napsána v jazyce JavaScript a deployment je řešen pomocí Docker kontejnerů. Je možné tak aplikaci nasadit v cloudovém prostředí, tak i přímo na serverech vlastněných klinikou. Jedinou podmínkou je přítomnost Dockeru. Kód je publikován ve veřejném repositáři na platformě GitHub pod licencí MIT.

Dále je potřeba nechat aplikaci otestovat více uživateli a zjistit, zda bude bezchybně běžet i s více daty. Použití standardu HL7 FHIR bylo v konečném důsledku zhodnoceno jako nadbytečné. Aplikace mohla být implementována i bez něj, jen by se musela doplnit o vlastní server a datové úložiště.

Seznam použité literatury

[1] *WinVet* [online]. Brno, 2023 [cit. 2023-04-08]. Dostupné z: https://www.winvet.cz/

[2] *VETIS office* [online]. [cit. 2023-04-08]. Dostupné z: http://www.vetis.cz/

[3] *Vetbook* [online]. c2012-2023 [cit. 2023-04-08]. Dostupné z: https://www.vetbook.cz/

[4] *Vetfox* [online]. Brno, c2014-2023 [cit. 2023-04-08]. Dostupné z: http://www.vetfox.cz/

[5] VetPro – veterinární informační systém. In: *Medsoft veterinární informační systém* [online]. 2015 [cit. 2023-04-08]. Dostupné z: https://medsoftsro.cz/vetpro-veterinarni-informacni-system/vetpro-veterinarni-software/

[6] *Vetsoftware* [online]. [cit. 2023-04-08]. Dostupné z: https://vetsoftware.eu/

[7] Propojte se s Vetfoxem. In: *Vetfox* [online]. c2014-2023 [cit. 2023-04-08]. Dostupné z: http://www.vetfox.cz/api

[8] About HL7. In: *HL7 International* [online]. Ann Arbor (Michigan), c2007-2023 [cit. 2023-04-09]. Dostupné z: http://www.hl7.org/about/index.cfm?ref=nav

[9] *HL7® FHIR® Release 5* [online]. 2023 [cit. 2023-04-08]. Dostupné z: https://www.hl7.org/fhir/

[10] License and Legal Terms. In: *HL7® FHIR® Release 5* [online]. 2023 [cit. 2023-04-09]. Dostupné z: https://www.hl7.org/fhir/license.html

[11] Public Web Servers. In: *FHIR Product Family* [online]. [cit. 2023-04-09]. Dostupné z: https://confluence.hl7.org/display/FHIR/Public+Test+Servers

[12] Azure Health Data Services. In: *Azure* [online]. 2023 [cit. 2023-04-09]. Dostupné z: https://azure.microsoft.com/en-us/products/health-data-services/

[13] Amazon HealthLake. In: *AWS* [online]. 2023 [cit. 2023-04-09]. Dostupné z: https://aws.amazon.com/healthlake/

[14] *Open Source Implementations* [online]. In: . 2023 [cit. 2023-04-09]. Dostupné z: https://confluence.hl7.org/display/FHIR/Open+Source+Implementations

[15] *SMART* [online]. Boston, 2020 [cit. 2023-04-09]. Dostupné z: http://docs.smarthealthit.org/

[16] *FHIR Community chat* [online]. [cit. 2023-04-09]. Dostupné z: https://chat.fhir.org/

[17] *ClinFHIR Launcher* [online]. [cit. 2023-04-09]. Dostupné z: http://clinfhir.com/

[18] Appointment. In: *HL7® FHIR® Release 5* [online]. 2023 [cit. 2023-04-08]. Dostupné z: https://www.hl7.org/fhir/appointment.html

[19] Schedule. In: *HL7® FHIR® Release 5* [online]. 2023 [cit. 2023-04-08]. Dostupné z: https://www.hl7.org/fhir/schedule.html

[20] Slot. In: *HL7® FHIR® Release 5* [online]. 2023 [cit. 2023-04-08]. Dostupné z: https://www.hl7.org/fhir/slot.html

[21] ActivityDefinition. In: *HL7® FHIR® Release 5* [online]. 2023 [cit. 2023-04-08]. Dostupné z: https://www.hl7.org/fhir/activitydefinition.html

[22] Patient. In: *HL7® FHIR® Release 5* [online]. 2023 [cit. 2023-04-08]. Dostupné z: http://hl7.org/fhir/patient.html

[23] Location. In: *HL7® FHIR® Release 5* [online]. 2023 [cit. 2023-04-08]. Dostupné z: http://hl7.org/fhir/location.html

[24] Practitioner. In: *HL7® FHIR® Release 5* [online]. 2023 [cit. 2023-04-08]. Dostupné z: https://www.hl7.org/fhir/practitioner.html

[25] *HAPI FHIR* [online]. 2023 [cit. 2023-05-13]. Dostupné z: https://hapifhir.io/

[26] *React* [online]. 2023 [cit. 2023-05-13]. Dostupné z: https://react.dev/

[27] *Redux* [online]. c2015-2023 [cit. 2023-05-13]. Dostupné z: https://redux.js.org/

[28] *Redux Toolkit* [online]. In: . c2015-2023 [cit. 2023-05-13].

[29] *MUI* [online]. 2023 [cit. 2023-05-13]. Dostupné z: https://mui.com/

[30] *Docker Desktop* [online]. 2023 [cit. 2023-05-14]. Dostupné z: https://www.docker.com/products/docker-desktop/

[31] EU Implementation Guide (Vet EU IG) on veterinary medicines product data in the Union Product Database. In: *European Medicines Agency (EMA)* [online]. Amsterdam, c1995-2023 [cit. 2023-05-16]. Dostupné z: https://www.ema.europa.eu/en/documents/regulatory-procedural-guideline/eu-implementation-guide-ig-veterinary-medicines-product-data-union-product-database-chapter-5_en.pdf

[32] EHealth Network přijímá HL7 FHIR pro výměnu zdravotních informací v EU pro nové případy použití. In: *Národní centrum elektronizace zdravotnictví* [online]. 2020 [cit. 2023-05-16]. Dostupné z: https://ncez.mzcr.cz/index.php/cs/aktuality/ehealth-network-prijima-hl7-fhir-pro-vymenu-zdravotnich-informaci-v-eu-pro-nove-pripady

Příloha A: Obsah přiloženého ZIP souboru

- Repositář se zdrojovými kódy systému