Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Aplikace pro veterinární ambulance

Bakalářská práce

Ondřej Doktor, DiS.

Školitel: PhDr. Miloš Prokýšek, Ph.D.

České Budějovice 2018

Doktor, O., 2018: Aplikace pro veterinární ambulance. [Application software for veterinary clinics. Bc. Thesis, in Czech.] – 54 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Cílem práce je vytvořit aplikaci s webovým rozhraním, která bude poskytovat řešení pro základní administrativu malé veterinární ambulance, bude umožňovat vedení kartotéky pacientů, vystavování účetních dokladů, bude napojena na EET a podporovat DPH. Aplikace má nahradit stávající software, který již nesplňuje současné technické a legislativní požadavky, přičemž má zachovat přiměřenou zpětnou kompatibilitu z hlediska workflow. Aplikace bude mutlitenantní. Tato bakalářská práce se zabývá analýzou, návrhem, implementací a testováním této aplikace.

Abstract

The goal is to create a software application for basic adniministration of a small veterinary clinic. The software application will be able to manage patient card index, issue accounting documents, will be connected to the EET system (electronic records of sales) and will support VAT. This new software application will replace currently used solution which is not up to the contemporary technical and legal requirements, but should retain reasonable backwards compatibility of the workflow. The software application will be mutlitenant. This thesis documents the process of analysing, designing and testing this software application.

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Ve Českých Budějovicích dne 18. 4. 2018

Ondřej Doktor, DiS.

Poděkování

Za odborné rady, skvělou podporu a trpělivost děkuji svému školiteli, PhDr. Miloši Prokýškovi, Ph.D. Dále bych rád poděkoval svému otci, MVDr. Romanu Doktorovi, za praktické poznatky z oblasti veterinární praxe, které značně pomohly při analýze a návrhu softwarové aplikace.

Obsah

1 Úvod

Bakalářská práce se zabývá tvorbou multitenantní[1] aplikace pro administrativní vedení malé veterinární ambulance napojené na EET. Cílovou skupinou této aplikace jsou veterináři–jednotlivci, kteří nedisponují zázemím velké kliniky a mnohdy vzhledem ke svému profesnímu věku nechtějí investovat do složitého software, který by jim navíc narušil stávající pracovní postupy. Aplikace je primárně vyvíjena pro potřeby veterinární ambulance MVDr. Romana Doktora, se kterým byl vývoj průběžně konzultován, a který posloužil jako modelový project owner (klient).

Cílem práce je vytvořit multitenantní aplikaci s webovým rozhraním, která bude poskytovat řešení pro základní administrativu malé veterinární ambulance, bude umožňovat vedení kartotéky pacientů, vystavování účetních dokladů, bude napojena na EET a podporovat DPH. Jednoduché a dostupné řešení, bez zbytečných funkcí, avšak v souladu s nejnovějšími technickými požadavky. To vše dostupné v počítači, tabletu i mobilu — kdykoliv a kdekoliv bude potřeba.

[1] Mutlitenantí aplikace dokáže obsluhovat pomocí jedné instance více tzv. tenantů. Tenant představuje skupinu uživatelů, kteří sdílejí společný přístup k aplikaci a mají izolovaný datový kontext od ostatních tenantů. Tenanti, v případě této aplikace, reprezentují jednotlivé veterinární ambulance.

2 Analýza

2.1 Současný stav klienta a situace na trhu

Cílovou skupinou této aplikace jsou veterináři–jednotlivci, kteří buď nepoužívají žádný specializovaný software, nebo používají software zastaralý. Modelový klient dosud používá zastaralý software vyvinutý v 90. letech na platformě *PC-FAND*, který pro svůj běh vyžaduje *MS-DOS* a je prakticky neaktualizovatelný. Zvyšující se nároky a často se měnící legislativa, v současnosti zejména pak povinnost zapojit se do *elektronické evidence tržeb* (*EET*), však tohoto veterináře nutí ke změnám.

V současnosti dostupná tuzemská řešení obvykle disponují velkým množstvím funkcí a jsou schopna pracovat s velmi širokou škálou dat — od triviálních věcí až po práci s RTG snímky. To je však činí nepřehlednými pro provoz malé veterinární praxe, která není vybavena hi-tech přístroji a provádí pouze běžné úkony. Jedním příkladem za všechny je rozšířený a robustní software *WinVet*[1] od společnosti *HENRY SCHEIN s.r.o.*, která je zároveň distributorem léčiv[1]. Pro srovnání je možné podívat se do zahraničí, např. na software *VetPort*[3], avšak takové řešení z pochopitelných důvodů bohužel nelze v českém prostředí použít.

Nová aplikace má plnit zejména tyto úkoly, v rozsahu původní aplikace:

- Vést kartotéku pacientů a majitelů, evidovat jejich návštěvy (zjištěné anamnézy, stanovené diagnózy, provedené úkony, podané léky).
- Vyúčtovat cenu za poskytnuté zboží a služby.
- Uchovávat číselníky[2] plemen, diagnóz, úkonů a léků.
- Evidovat statistiky prodejů, minimálně v podobě obratu (součtu jednotlivých tržeb) rozlišenou alespoň do úrovně jednotlivých ošetřujících lékařů a jednotlivých kalendářních měsíců.

[1] Pro ilustraci: manuál k tomuto software má přes 260 stran[2].

[2] Termín *číselník* je zde mírně zavádějící, protože příslušné entity obvykle uchovávají více atributů než jen *id* a *název*. Pro zjednodušení je však budeme v této práci nazývat číselníky.

2.1.1 Nedostatky původní aplikace

Původní aplikace má zejména tyto nedostatky, které by měla nová aplikace vyřešit:

- Legislativní nedostatky:
  - Neumí pracovat s DPH.
  - Neumí vytvořit daňový doklad, ani evidovat údaje v potřebném rozsahu.
  - Neumí vytvořit EET doklad a odeslat platbu.
  - Neumí evidovat údaje v rozsahu potřebném pro vystavení *petpasu*[3].
- Funkční nedostatky:
  - Nedokáže evidovat u jednoho majitele více zvířat.
  - Neumí evidovat doplňující kontaktní údaje na majitele (telefon, e-mail).
  - Neumí zpracovat „anonymní návštěvy" (jde např. o pultový prodej krmiv a jiných doplňků, či hromadnou vakcinaci)
  - Má problém s některými znaky české abecedy.
  - Používaná datová struktura má omezenou kapacitu pro počet návštěv jednoho zvířete.
  - Nehlídá integritu uložených dat, změny v číselnících ovlivní i záznamy pořízené v minulosti.
  - Neumožňuje práci v síti, synchronizaci dat s jinými zařízeními ani jinou podobnou funkci.
  - Nulové zabezpečení.
- Technické nedostatky:
  - Velmi komplikované nasazení kvůli závislosti na MS-DOS, z dosud podporovaných operačních systémů a zařízení lze nativně spustit pouze na PC s *Windows 7* verze 32bit, nepodporuje jakékoliv mobilní zařízení.
  - Nefunguje tisk na současných tiskárnách.
  - Data se nedají jednoduše importovat nebo exportovat.
  - Není dostupný zdrojový kód, aplikace není aktualizovatelná.
  - Není multitenantní a neumožňuje centrální správu.

[3] Tzv. *petpas* (přesněji *Identifikační doklad pro zvíře v zájmovém chovu*) je speciální doklad, který je nutný mj. pro vycestování zvířete do zahraničí[4].

2.1.2 Přednosti původní aplikace

Původní aplikace má ale i jisté kladné vlastnosti z hlediska její použitelnosti, které by měli být pokud možno zachovány:

Keyboard-only ovládání — I když není na první pohled intuitivní, vede ovládání pomocí klávesnice k rychlejšímu zadávání údajů. Nevýhodou je však dlouhá křivka učení a potřeba naučit se nazpaměť určité sekvence kláves.

Zadávání identifikačními kódy — Z dnešního pohledu zvláštní vlastností aplikace založené na *PC-FAND* je možnost zadávat hodnoty určitých polí přímo pomocí identifikačního kódu položky z číselníku, který je s polem propojen. V našem případě se používají celočíselné kódy o třech nebo pěti číslicích. Při znalosti kódu je tedy možné velmi rychle zadat požadovanou hodnotu, aniž by bylo nutné ruční vyhledávání v číselníku na základě dalších kritérií. V kombinaci s keyboard-only ovládáním pak může zkušený operátor pracovat velice rychle a efektivně. Veterináři používající původní aplikaci navíc již často používané kódy znají nazpaměť a diktují je asistentům přímo (Pro ilustraci, takto může vypadat pokyn k zapsání návštěvy pacienta, který se dostavil na přeočkování vztekliny kombinovanou vakcínou DHPPi+LR, který si navíc ještě vzal tablety na odčervení: „Pacient čtyřičtyřišest, diagnóza třicetdvojka, úkony dvacetdvojka, léky třitřišestka dvakrát, pětistovka 5 tablet, všechno”.).

Přehledné zobrazení — Vše se vejde na jednu obrazovku, v aplikaci prakticky neexistuje scrollování. Tento princip nelze samozřejmě zcela zachovat, avšak je nutné zejména kartu pacienta pokud možno zobrazit v rámci viewportu[4].

Z výše uvedeného je patrné, že pro úspěšné nasazení a dosažení kladných ohlasů je nutné tyto vlastnosti původní aplikace přenést do aplikace nové.

2.2 Legislativní požadavky

Činnost veterinárního lékaře je legislativně upravena mnoha zákonnými a podzákonnými předpisy. Ačkoliv právní problematika není předmětem této práce, je nutné se seznámit s některými legislativními požadavky, které se dotýkají fungování vyvíjené aplikace, aby tato pracovala v souladu s nimi. V tomto kontextu nás zajímají především náležitosti vystavovaných dokladů, práce s DPH a pravidla fungování EET. Popsané skutečnosti jsou univerzální a vztahují se i na jiné podnikající osoby.

[4]Viewport je termín, který označuje obrazovku zařízení nebo okno prohlížeče.

2.2.1 Vystavované doklady a jejich náležitosti

Doklad o prodeji

Základním dokladem, který jsou povinni vystavovat na žádost kupujícího všichni podnikatelé je tzv. **doklad o prodeji zboží a o poskytnutí služby**, který upravuje § 31 odst. 14 zákona č. 455/1991 Sb. živnostenský zákon. Jeho smyslem je poskytnout kupujícímu informaci o tom, kdo, kde, kdy, co a za kolik mu prodal. Obdobný doklad je vyžadován též na základě § 16 zákona č. 634/1992 Sb. o ochraně spotřebitele (zde nazývaný jako *doklad o zakoupení výrobku nebo o poskytnutí služby*).

Daňový doklad

Pokud je prodávající zároveň plátcem DPH, vztahují se na něj dále povinnosti zákona č. 235/2004 Sb. o dani z přidané hodnoty. Ten mj. ukládá prodávajícímu vystavit tzv. **daňový doklad** — podnikajícím fyzickým osobám[5] na požádání, jakýmkoliv právnickým osobám vždy. V případě, kdy celková částka vč. DPH nepřekročí 10.000 Kč, je možné vystavit tzv. **zjednodušený daňový doklad** (lidově *paragon*), v opačném případě je nutné vystavit daňový doklad se všemi náležitostmi (lidově *fakturu*). Hlavním rozdílem oproti zjednodušenému daňovému dokladu je zejména povinnost uvést fakturační údaje zákazníka (vč. IČO a DIČ) a rozepsat podrobně výpočet DPH.

Situaci navíc komplikuje skutečnost, že daňové doklady na částku větší nebo rovnou 10.000 Kč rovněž vstupují jako samostatné položky do tzv. kontrolního hlášení na základě § 101c a násl. zákona č. 235/2004 Sb. o dani z přidané hodnoty. Proto je nutné je evidovat a předávat k účetnímu zpracování samostatně. Ostatní případné tržby ordinace, tedy všechny od nepodnikajících fyzických osob a od podnikajících do 10.000 Kč vstupují do kontrolního hlášení souhrnně, proto postačí dodávat účetní pouze měsíční součty. (podrobnější vysvětlení viz Časté dotazy a odpovědi na webu Finanční správy [5]).

Aplikace nemá za cíl vést účetnictví či daňovou evidenci ambulance, naopak má respektovat jakékoliv řešení, které veterinární ambulance používá — ať už jsou faktury vystavovány ručně, nebo pomocí specializovaného software. Veterinární lékaři mohou rovněž vystavovat daňové doklady i jiným subjektům než návštěvníkům ambulance (např. zemědělským podnikům při ošetřování tzv. velkých zvířat, příp. chovatelským stanicím), proto by všechny účetní případy stejně nebylo možné aplikací pokrýt a ani

[5] Přesněji *fyzickým osobám povinným k dani.*

by to nebylo účelné, protože další podnikatelské aktivity provozovatele ambulance mohou být různorodé.

V malé ambulantní praxi jsou však návštěvy pacientů, kteří se nachází v obchodním majetku podnikatelů, spíše vzácné. Spolupráce s jinými podnikateli je obvykle upravena zvláštními smlouvami a fakturace probíhá z provozního hlediska ve velkoobchodním režimu (B2B). Z toho důvodu je možné celý problém daňových dokladů zjednodušit takto:

- Při částce do 10.000 Kč bude vždy vystaven zjednodušený daňový doklad
- Při částce nad 10.000 Kč včetně:
  - Bude vystaven doklad o prodeji s dovětkem „TOTO NENÍ DAŇOVÝ DOKLAD".
  - Pokud je kupujícím právnická osoba nebo podnikající fyzická osoba, která o to požádá, vystaví uživatel řádný daňový doklad obvyklým způsobem dle zvyklostí své ambulance. Využíváme přitom skutečnosti, že daňový doklad nemusí být vystaven ihned, ale do 15 dnů od zaplacení nebo poskytnutí plnění[6].
- Na dokladech bude nad rámec zákonných povinností uvedena u každé položky jednotková cena celkem, sazba DPH a základ DPH. (Jde o běžnou zvyklost, usnadňuje to práci účetním kupujícího a poskytuje podklad pro vystavení řádného daňového dokladu v situaci v předchozím bodě).

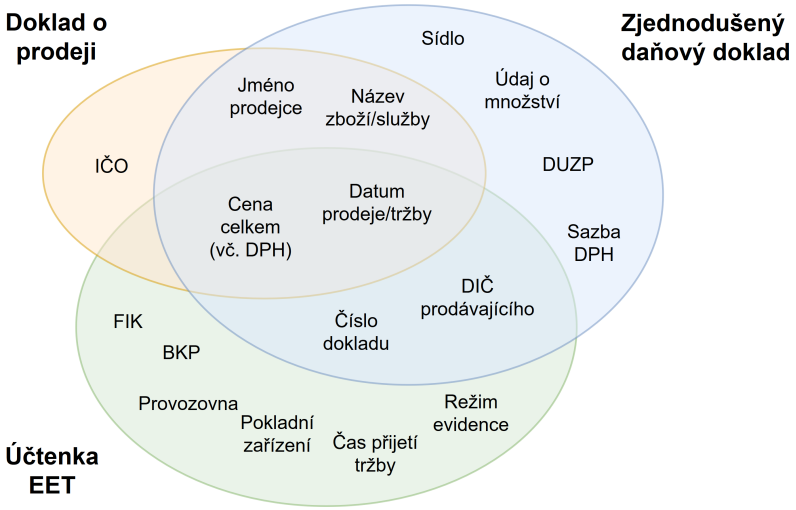
Účtenka EET

Novinkou v podnikatelském světě je Elektronická evidence tržeb, která se vztahuje také na veterinární lékaře. Na základě § 18 zákona č. 112/2016 Sb. o evidenci tržeb, je podnikatel povinen nejpozději při uskutečnění evidované tržby vystavit tzv. **EET účtenku** tomu, od koho evidovaná tržba plyne, v našem případě kupujícímu. Náležitosti účtenky jsou podrobně popsány v § 20 zákona č. 112/2016 Sb. o evidenci tržeb a jsou úzce spojeny s mechanismem fungování EET, který je podrobněji popsán dále v textu.

Další obecné požadavky na doklady

Vystavované doklady mohou být podle potřeby použity pro účely daňové evidence nebo účetnictví. Zde platí obecné pravidlo, že účetní jednotky jsou povinny vést

[6] § 28 odst. 4 zákona č. 235/2004 Sb. o dani z přidané hodnoty



Obrázek 2.1: Přehled náležitostí jednotlivých dokladů

účetnictví správné, úplné, průkazné, srozumitelné, přehledné a způsobem zaručujícím trvalost účetních záznamů. Z našeho pohledu to znamená především nutnost doklady číslovat v nepřerušené řadě a tisknout je na trvanlivý materiál (i když tyto povinnosti legislativa takto přesně nedefinuje, dovozujeme je z běžné a ostatně i logické účetní praxe).

Rovněž je třeba vzít v úvahu, že ve vystavených dokladech (a nota bene v zaplacených) nelze dělat dodatečné opravy či úpravy. U daňových dokladů (faktur) se tato situace řeší vydáním tzv. opravného daňového dokladu (lidově *storno faktura* nebo *dobropis*). Z hlediska EET je dodatečná oprava zaevidované tržby rovněž neproveditelná, jedinou možností je zaevidování záporné tržby: *„Vrací-li se evidovaná tržba nebo provádí-li se její opravy, použijí se ustanovení týkající se evidence tržeb obdobně s tím rozdílem, že je tato tržba evidována jako záporná."*[6]. V případě chyby v zaplacené účtence je nutné vytvořit samostatnou storno účtenku se zápornou částkou.

Shrnutí

Náležitosti jednotlivých dokladů přehledně zobrazuje diagram na Obrázku 2.1. Je patrné, že požadavky na údaje se u jednotlivých typů dokladů částečně překrývají. Všechny výše popsané doklady je možné sloučit do jednoho fyzického dokladu, který bude obsahovat veškeré náležitosti — *pro účely této práce budeme dále v textu termínem* ***účtenka*** *nazývat právě tento agregovaný doklad.* Číslování dokladů je v našem

případě také možné sloučit a používat stejnou číselnou řadu jak pro zjednodušené daňové doklady, tak pro potřeby EET. Podmínkou přitom je, aby význam údajů uvedených na účtence byl čitelný a jednoznačně identifikovatelný [7].

2.2.2 EET

Elektronickou evidenci tržeb upravuje zákon č. 112/2016 Sb. o evidenci tržeb. Na základě něj je podnikatel povinen nejpozději v okamžiku uskutečnění evidované tržby zaslat datovou zprávou údaje o této evidované tržbě správci daně a vystavit účtenku zákazníkovi [7]. Veterináři spadají do tzv. „třetí fáze" EET a jsou povinni evidovat tržby od 1.3.2018 [8]. Situace je samozřejmě složitější, veškeré podmínky a další informace ohledně EET jsou k dispozici na portálu *Etržby.cz* provozovaném Finanční správou ČR [9].

Před zahájením evidence tržeb podnikatel požádá Finanční správu o autentizační údaje, které mu umožní přihlášení do webové aplikace *Elektronická evidence tržeb*. Po přihlášení si podnikatel v aplikaci *Elektronická evidence tržeb* zaeviduje své provozovny a vygeneruje certifikát, který poté nainstaluje do svého pokladního zařízení [10] — s tímto certifikátem bude pracovat vyvíjená aplikace. Certifikát slouží k elektronickému podepisování datové zprávy, kterou podnikatel zasílá správci daně údaje o evidované tržbě.

Po úspěšném odeslání datové zprávy s údaji o evidované tržbě odešle systém Finanční správy zpět potvrzení, jehož součástí je tzv. **fiskální identifikační kód** (FIK), který je nutné uchovat a umístit na účtenku. Při výpadku spojení je možné zaslat údaje o evidované tržbě dodatečně do 48 hodin, na účtence se pak místo FIK vytiskne tzv. **podpisový kód poplatníka** (PKP). V každém případě se na účtence musí objevit tzv. **bezpečnostní kód poplatníka** (BKP).

Proces, který je stručně popsán výše, odpovídá evidenci v tzv. **běžném režimu** (on-line). EET umožňuje také evidovat tržby v tzv. **zjednodušeném režimu** (off-line), který je zřízen zejména pro evidenci tržeb na místech bez připojení k internetu (např. na palubě dopravních prostředků). O možnost použít evidenci tržeb ve zjednodušeného režimu se ale musí individuálně žádat[7], proto jej v našem případě (nota bene u aplikace s webovým rozhraním) zcela pomineme. Údaj o použitém režimu je nicméně nutné na účtence vždy uvést.

Technické podrobnosti ohledně komunikace se servery správce daně jsou podrobněji popsány dále v textu a vychází z oficiální dokumentace [11].

[7] § 11 zákona č. 112/2016 Sb. o evidenci tržeb

2.2.3 Práce s DPH

Daň z přidané hodnoty je nepřímou daní, kterou jsou podnikatelé – plátci DPH, povinni účtovat svým zákazníkům a odvádět Finančnímu úřadu. Z našeho pohledu je důležité, že podnikatel může i nemusí být plátce DPH. Pokud plátce DPH není, níže popsané povinnosti se na něj nevztahují. Kritéria, na základě kterých plátcovství vzniká a zaniká nejsou předmětem této práce, vyvíjená aplikace ale musí počítat s oběma eventualitami.

V současnosti existují tři sazby DPH: základní (21 %), první snížená (15 %) a druhá snížená (10 %). Výčet zboží a služeb, které spadají do snížených sazeb je definován v přílohách k zákonu č. 235/2004 Sb. o dani z přidané hodnoty.

Veterinární služby nespadají do snížených sazeb, proto při jejich poskytování účtuje plátce 21% DPH.

U zboží / léčiv se účtuje rozdílná sazba DPH v různých situacích — v případě spotřeby při poskytování služby (např. odebereme 10ml léku z velké lahvičky) je sazba 21%, při samostatném prodeji (prodáme celé, nenačaté balení, tak jak je) pak podle druhu zboží buď 10%, 15% nebo 21%. Z toho důvodu je nutné u zboží evidovat dvě sazby pro obě popsané situace, a dále rozlišovat zda se zboží prodává jako součást služby (návštěva pacienta) nebo samostatně (prodej zboží).

2.2.4 Petpasy

Někteří veterinární lékaři jsou oprávněni vydávat tzv. petpasy, přesněji *identifikační doklady pro zvířata v zájmovém chovu*. Tento doklad slouží mj. jako nutný doklad při vycestování zvířete do zahraniční. I když vystavovaní petpasů není přímým úkolem vyvíjené aplikace, měla by tato umožnit evidovat u pacientů a jejich majitelů všechny údaje, které jsou nutné pro vystavení petpasu. Pravidla pro vydávání petpasů, vč. rozsahu zpracovávaných údajů a odkazů na příslušnou legislativu, jsou přehledně zpracovány v dokumentu *Pasové minimum pro praktické veterinární lékaře*, který vydala Komora veterinárních lékařů ČR [4].

2.2.5 GDPR

Další legislativní novinkou je Nařízení Evropského parlamentu a Rady (EU) 2016/679 (označované zkratkou GDPR). Toto nařízení v rámci EU nastavuje obecná pravidla pro ochranu osobních údajů a týká se všech, kteří je zpracovávají. Týká se tedy i veterináře, který se stává tzv. správcem osobních údajů (zvířata samotná nejsou subjektem práva ochrany osobních údajů).

Z hlediska požadavku na integritu a důvěrnost je třeba zajistit dostatečné zabezpečení aplikace. Z programátorského hlediska musí aplikace bezpečně uchovávat data (opatření předcházející různým útokům, zejména pak ošetření všech vstupů a výstupů), bezpečné přihlašování a důsledné oddělení datových kontextů pro jednotlivé tenanty. Stejně důsledně je třeba zajistit bezpečnou komunikaci s ostatními systémy (zejména nasadit HTTPS). Problematika zabezpečení výkonného prostředí (tedy nastavení serveru), ve kterém bude aplikace spouštěna je však již mimo rozsah této práce a spadá do gesce systémového administrátora, nikoliv programátora.

Povinnost zajistit zákonný důvod zpracování má správce údajů (veterinář) a nikoliv zpracovatel (my, jako tvůrci nebo provozovatelé aplikace). Obecně platí, že v případě pokud dochází ke zpracování, které je nezbytné pro splnění smlouvy, není souhlas vyžadován. Stejně tak se souhlas nevyžaduje pokud je zpracování nutné pro splnění jiných právních povinností správce. To se týká i údajů majitelů v rozsahu vytvářené aplikace. Problematické by mohlo být pouze uchování e-mailu a telefonního čísla. Pokud tyto údaje majitel veterináři dobrovolně sdělil pro účely plnění smlouvy, není souhlas vyžadován. Pokud ale tyto údaje veterinář využije k provádění marketingových kampaní, musí si již souhlas zajistit. To je však již v kompetenci veterináře, protože aplikace neobsahuje funkce pro provádění marketingové činnosti. Podrobnější vysvětlení viz rešerše Komory veterinárních lékařů ČR [12].

2.3 Funkční požadavky

1. Správa tenantů:
   a) Rozlišovat kontext tenanta přihlášeného uživatele.
   b) Umožnit administrátorům přepínat kontext tenanta (přihlašovat se jeho jménem).
   c) Umožnit administrátorovi správu tenantů - CRUDL[8] uživatele.
2. Správa uživatelů:
   a) Přihlásit uživatele.
   b) Odhlásit uživatele.
   c) Změnit heslo uživatele.
   d) Rozlišovat úrovně uživatelských oprávnění.

[8] CRUDL — zkratka pro množinu základních operací: vytvořit (CReate), upravit (Update), smazat (Delete), procházet (List)

e) CRUDL uživatele.

3. Operace s číselníky:
   a) CRUDL diagnóz.
   b) CRUDL úkonů.
   c) CRUDL léků.
   d) CRUDL sazeb DPH.
   e) CRUDL druhů zvířat.
   f) CRUDL majitelů.
   g) CRUDL pacientů.
4. Spravovat návštěvy pacientů:
   a) Vyhledat pacienta podle zadaných kritérií.
   b) Procházet (zobrazovat) návštěvy.
   c) Otevřít (vytvořit) novou návštěvu (anamnéza, diagnózy, úkony, léky a jejich množství, poznámka).
   d) Vrátit se k neuzavřené návštěvě.
   e) Uzavřít návštěvu.
   f) Připravit účtenku na základě uzavřené návštěvy.
   g) Stornovat návštěvu v případě chybně zadaných dat.
   h) Body b) – g) provádět také bez znalosti konkrétního pacienta (anonymní návštěva).
5. Prodávat zboží:
   a) Otevřít nový prodej zboží (léky a jejich množství).
   b) Uzavřít prodej zboží a připravit na jeho základě účtenku.
   c) Stornovat prodej v případě chybně zadaných dat.
6. Spravovat účtenky:
   a) Procházet účtenky.
   b) Zaplatit připravenou účtenku.
   c) Fiskalizovat účtenku v EET.

   d) Tisknout účtenku na pokladní tiskárnu.

   e) Stornovat zaplacenou účtenku.

7. Generovat statistiky a účetní podklady:

   a) Zobrazovat a tisknout měsíční součty tržeb.

8. Rozhraní a workflow:

   a) Zachovat kompatibilitu s kódy pacientů, diagnóz, úkonů a léků z původního systému (možnost provést import).

   b) Vizuálně upozorňovat na neuzavřené návštěvy, nezaplacené účtenky, nefiskalizované účtenky a jiné stavy.

   c) Při placení účtenky zobrazit pomůcku pro výpočet částky k vrácení.

   d) Udržet rozhraní přehledné, zobrazovat maximum informací v rámci viewportu, minimalizovat scrollování.

   e) Umožnit ovládání nejčastěji používaných funkcí také pomocí klávesnice.

2.4 Nefunkční požadavky

1. Aplikace s webovým rozhraním.

2. Data uchovávána primárně v databázovém serveru.

3. Možnost používat knihovny třetích stran výhradně s těmito licencemi: *MIT*, *BSD*, *CC0* a *Public domain*.

4. Rozhraní optimalizované pouze pro aktuálně podporované systémy a prohlížeče.

5. Optimalizace pro mobilní zařízení (tablety, mobily).

6. Napojení na systém EET, fiskalizace synchronní a transakční, podpora pouze on-line režimu.

7. Spolupráce více uživatelů zároveň, izolace datového kontextu tenantů.

2.5 Případy užití

Na základě výše popsaných skutečností a konzultací s modelovým project ownerem byly sestaveny případy užití. Vzhledem k velkému rozsahu je dokumentace jednotlivých případů užití připojena formou elektronické přílohy na přiloženém CD, viz Příloha B. Pro přehlednost byly související případy užití seskupeny do několika částí:

- Přihlašování, viz Obrázek 2.2.
- Uživatelé a tenanti, viz Obrázek 2.3.
- Pacienti, viz Obrázek 2.4.
- Návštěvy a prodej, viz Obrázek 2.5.
- Reporting, viz Obrázek 2.6.
- Účtenky, viz Obrázek 2.7.
- Správa seznamů, viz Obrázek 2.8.

Diagramy zachycují na straně uživatelů našeho systému několik aktérů - *Hosta*, *Lékaře*, *Majitele ambulance* a *Administrátora*. Aktéři *Lékař* a *Majitel ambulance* spouští případy užití vždy v kontextu svého tenanta. *Administrátor* může pro sebe kontext tenanta přepnout a vystupovat pak vůči systému jako člen uživatelské skupiny tenanta. Jedna osoba může vystupovat v roli více aktérů – např. jako *Lékař* a zároveň *Majitel ambulance* v případě veterináře-jednotlivce.

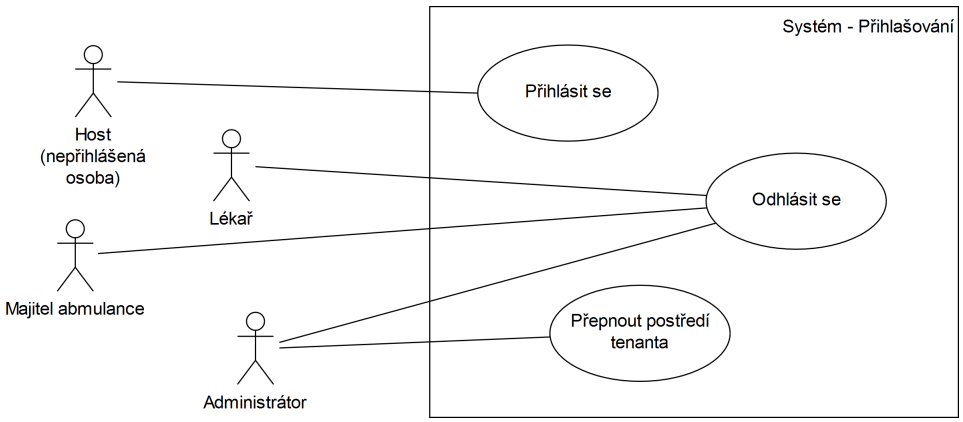
Kromě uživatelů zachycuje diagram další aktéry, kteří představují externí systémy:

EET Speciálním aktérem je systém EET, který vstupuje do případu užití *Fiskalizovat účten*ku. Jde o externí systém provozovaný Finanční správou ČR, jeho rozhraní je podrobněji popsáno v části 3.1.4 této práce.

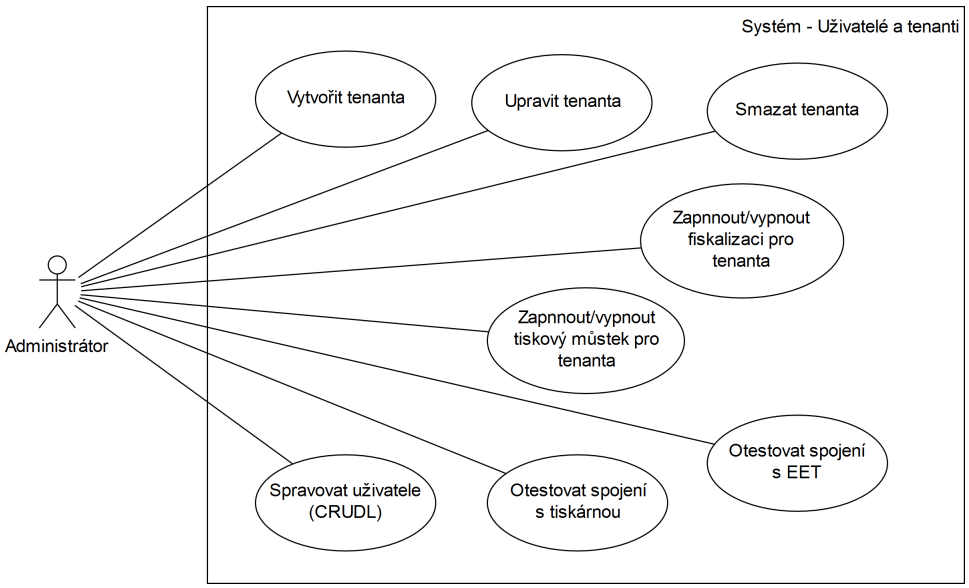
Tiskový můstek Pro tisk na pokladní tiskárnu je využíván tiskový můstek, zajišťovaný službou třetí strany *PrintNode*.

Plánovač úloh Slouží k pravidelnému spouštění úloh. Plánovač aktivuje případ užití *Hromadná fiskalizace účtenek*. Tím se ošetřuje podmínka EET, že při poruše pokladního zařízení (nefunkčním spojení) musí být údaje o evidované tržbě odeslány do EET do 48 hodin od realizace tržby.

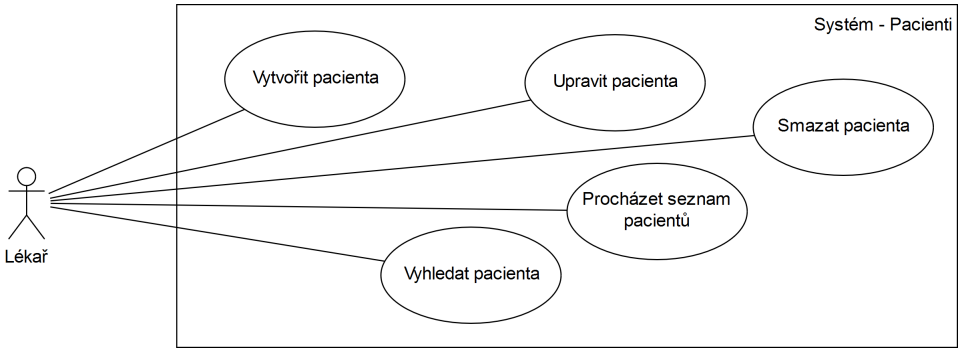
Některé případy užití zahrnují části jiných, tento vztah je značen přerušovanou čarou se stereotypem *«include»*. Případy užití, které svou povahou představují správu seznamu typu CRUDL jsou pro zjednodušení spojeny do jednoho případu užití. Tyto sdružené případy se však liší v primárních aktérech a vstupních a výstupních podmínkách.



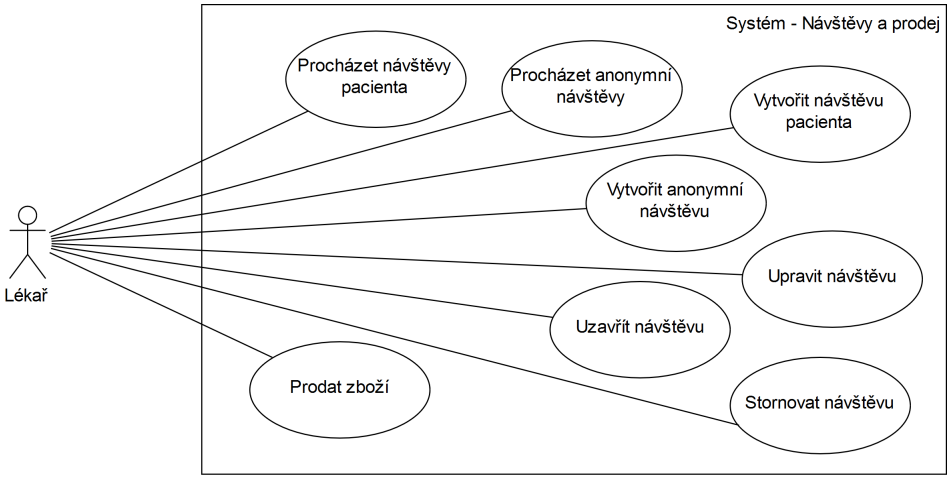
Obrázek 2.2: Diagram případů užití — část Přihlašování



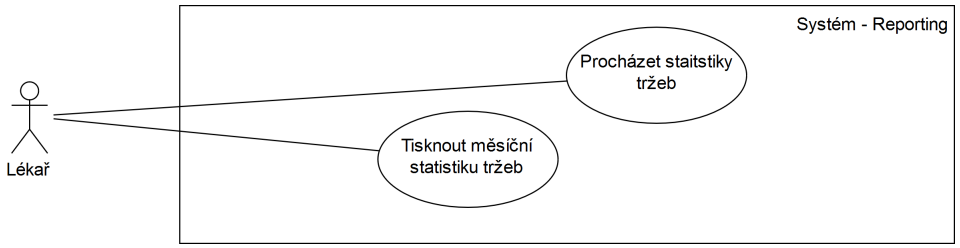
Obrázek 2.3: Diagram případů užití — část Uživatelé a tenanti



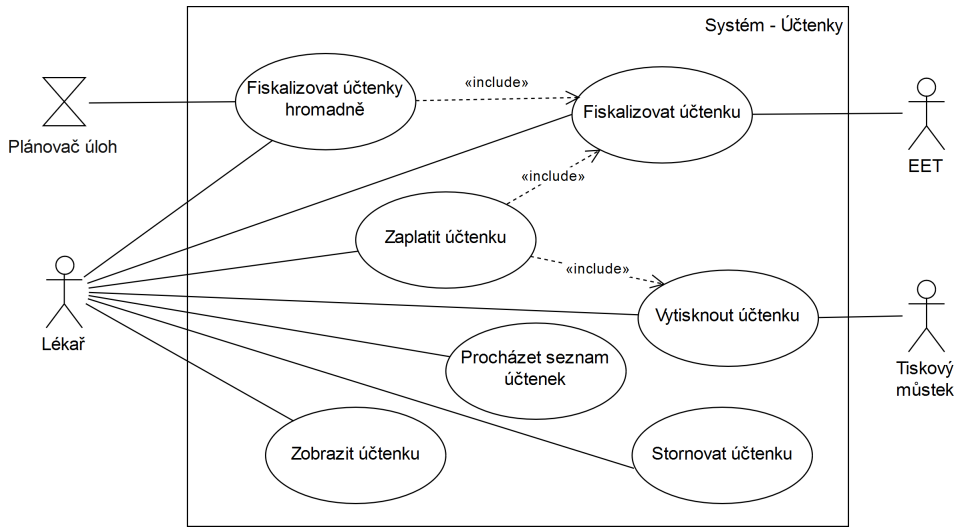
Obrázek 2.4: Diagram případů užití — část Pacienti



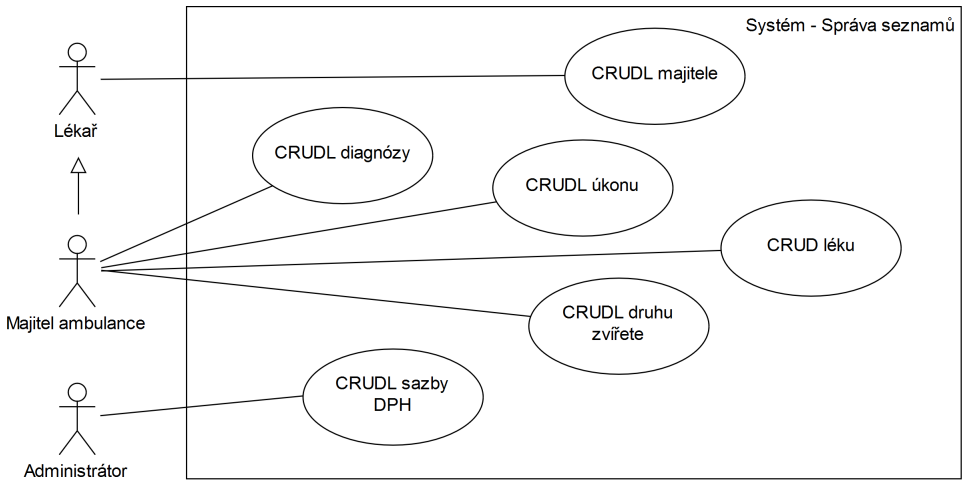
Obrázek 2.5: Diagram případů užití — část Návštěvy a prodej



Obrázek 2.6: Diagram případů užití — část Reporting



Obrázek 2.7: Diagram případů užití — část Účtenky



Obrázek 2.8: Diagram případů užití — část Správa seznamů

2.6 Analytické entity

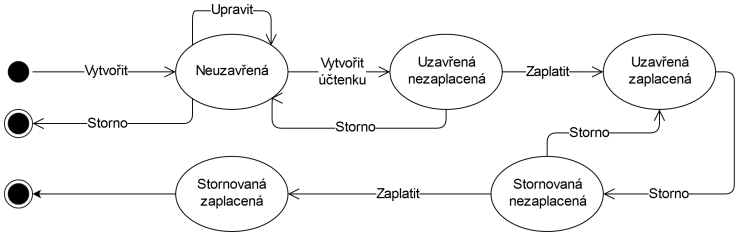
Na základě funkčních požadavků a případů užití byly identifikovány tyto entity, se kterými aplikace pracuje:

- Tenant
- Uživatel
- Sazba DPH
- Entity závislé na tenantovi:
  - Majitel
  - Pacient
  - Druh zvířete
  - Návštěva
  - Diagnóza
  - Úkon
  - Lék
  - Účtenka

2.6.1 Životní cyklus návštěvy

Na základě výše uvedených zjištění je nyní možné obecně abstrahovat následující:

- Návštěva pacienta a účtenka musí být oddělené entity — účtenka totiž může existovat i samostatně v případě „pultového prodeje" zboží.
- Údaje na účtence musí být za každých okolností neměnné (např. je neovlivní změna ceníku, změna sazeb DPH, změna způsobu výpočtu apod.). Pokud je účtenka spojena s návštěvou, nelze po vystavení účtenky návštěvu měnit.
- Účtenku je do zaplacení možné smazat (měnit), po zaplacení již nikoliv.
- Zaplacením je účtence přiřazeno jedinečné číslo dokladu a je fiskalizována v EET, pokud má příslušný tenant tuto službu aktivní.
- Musí být uchována informace o proběhlé fiskalizaci i pokud byla neúspěšná (pro pozdější další pokusy). Neúspěšná fiskalizace nebrání zaplacení účtenky a dalšímu postupu.



Obrázek 2.9: Životní cyklus návštěvy pacienta

- V případě zjištění chyby na zaplacené účtence se oprava provede vytvořením storno účtenky se zápornou hodnotou (nebudou se opravovat jednotlivé položky, stornuje se celý doklad a poté se příp. vytvoří nový, se správnými údaji).
- Musí být uchována vazba mezi stornovanou a stornující účtenkou, aby nedošlo k nekonzistentnímu stavu a bylo možné korektně navazovat akce.
- Zaplacenou storno účtenku již nelze stornovat, aby se předešlo zacyklení.

Z těchto podmínek byl sestaven životní cyklus návštěvy, který zachycuje stavový diagram na Obrázku 2.9. Přechod mezi jednotlivými stavy musí být transakční, aby se v případě chyby systém navrátil do předchozího konzistentního stavu.

2.6.2 Stavy účtenky

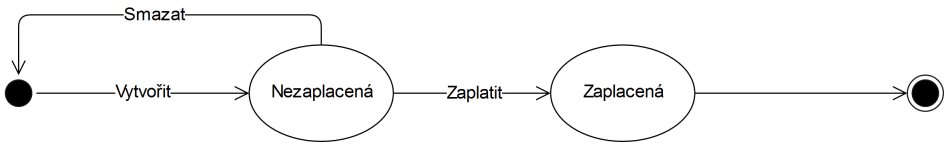
Na základě výše popsaného životního cyklu vyplývají i pro účtenku jisté stavové vlastnosti, které je možné vyhodnotit na základě hodnot jejích dat a vazeb na jiné entity. Tyto stavy a přechody mezi nimi jsou popsány na Obrázcích 2.10 až 2.13.

Z hlediska platby je určující, zda bylo účtence přiděleno číslo z dokladové řady, které se nastavuje při platbě. Nezaplacenou účtenku lze smazat, zaplacenou již nikoliv.

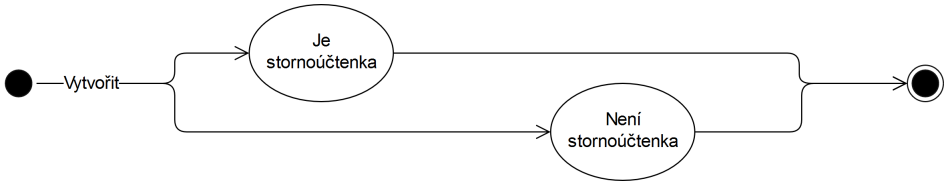
Z hlediska povahy dokladu zda účtenka slouží ke stornování jiné účtenky. Stornoúčtenka má kromě této vazby také všechny položky se zápornou cenou. Informace o povaze dokladu je známa již při vytváření účtenky.

Z hlediska vztahu k jiným dokladům rozlišujeme, zda byla účtenka stornována nebo ne. Jde o inverzi vztahu popsaného v předchozím bodě.

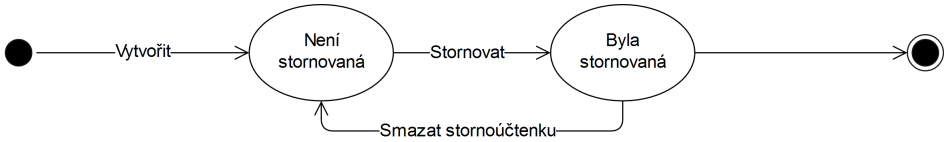
Z hlediska EET rozlišujeme účtenku nefiskalizovanou, neúspěšně fiskalizovanou (byl vygenerován kód BKP a PKP, ale není znám FIK) a fiskalizovanou (známe FIK). Význam těchto kódů je vysvětlen v části 2.2.2 této práce.



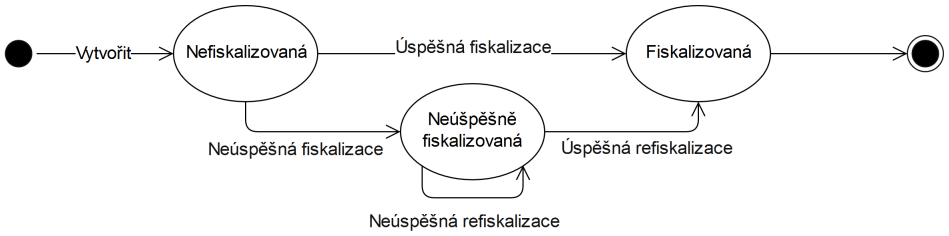
Obrázek 2.10: Stavový diagram stavů účtenky z hlediska platby.



Obrázek 2.11: Stavový diagram stavů účtenky z hlediska povahy dokladu.



Obrázek 2.12: Stavový diagram stavů účtenky z hlediska vztahu k jiným dokladům.



Obrázek 2.13: Stavový diagram stavů účtenky z hlediska EET.

3 Návrh a implementace

3.1 Technologie

Aplikace je zhotovena jako multitentatní PHP webová aplikace s responsivním frontendem a integrací pro tiskový můstek. Volba tohoto řešení oproti klasickému desktopovému byla provedena především z těchto důvodů:

Multitenance U tzv. multitenantní aplikace obsluhuje jedna instance zároveň více odběratelů (tenantů), kteří mají oddělené prostředí. Díky možnosti udržovat v chodu pouze jednu instanci se výrazně zjednodušuje správa celého řešení. Aplikace je tak vhodná pro provozování obchodním modelem Software as a Service (SaaS).

Snadný vývoj a široké možnosti nasazení PHP aplikace jsou lehké a snadno nasaditelné v podobě hostované (sdílený hosting, virtuální server, *MS Azure*, atd.), na dedikovaném nebo managed serveru nebo jako self-hosted on-premise řešení.

Fungování out-of-the-box Odpadá potřeba složité konfigurace na straně uživatelů. Pro provoz stačí zařízení s moderním webovým prohlížečem a internetové přípojení + nainstalovaný tiskový můstek.

Kompatibilita Jedna aplikace může být obsluhována jak z počítače, tak z tabletu, příp. i z mobilu. Jak v kanceláři, tak v terénu. Aplikace nebude závislá na výrobci klientského zařízení.

Škálovatelnost Do budoucna je možné snadno nasadit novou funkcionalitu a provést úpravy, pokud dojde ke změně legislativních požadavků nebo technologickým změnám na straně EET. Díky multitenaci je však stále možné jednotlivá rozšíření nasazovat postupně (např. i formou příplatkové služby).

Cílem práce není „znovu vynalézat kolo", naopak – provést vývoj co možná nejefektivněji. Z toho důvodu se při řešení počítá s použitím dostupných knihoven třetích stran, zejména pro integraci služeb externích systémů. Vzhledem ke komerčnímu

potenciálu jsou akceptovány pouze knihovny pod licencí BSD, MIT, Apache 2.0, OFL, CC (CC0, ostatní výjimečně podle typu). Zásadně je třeba se vyvarovat knihoven pod "rakovinotvornými"[1] licencemi z rodiny GNU (GPL, AGPL, LGPL ve všech variantách)[14].

Otázka, zda vnášet či nevnášet do našich projektů cizí kód, není předmětem této práce. Autor na ni proto rovnou odpovídá: Ano vnášet, pokud je tento kód dostupný pod výše uvedenými licencemi a pokud je dostatečně otestován a používán (a tudíž funkčně prověřen) komunitou.

3.1.1 Server-side technologie

Pro server-side část je použit jazyk PHP, aplikace je vyvinuta na funkční úrovni verze PHP 5.6 s dopřednou kompatibilitou pro veze 7.0 a 7.1. Verze 5.6 byla zvolena pro maximalizaci kompatibility (společné minimum všech použitých knihoven), avšak autor se v kódu záměrně vyhnul konstrukcím, které by omezili kompatibilitu s vyššími verzemi. Nic tak nebrání tomu provozovat aplikaci v prostředí s verzí PHP řady 7 a využít tak nezanedbatelný nárůst výkonu, který vyšší verze přináší [15]. Aplikace je připravena pro nasazení na server *Apache*.

Aplikace je založena na *Nette* frameworku verze 2.4 (poslední stabilní verze) a používá související knihovny.

Pro databázové operace je použit databázový server *MySQL,* funkční úroveň verze >=5.5. Synchronizaci mezi objekty aplikace a jejich reprezentací v databázi zajišťuje ORM[2] framework *Doctrine 2* [16], integrovaný pomocí rozšiřující knihovny *Kdyby/Doctrine* [17].

Stručně o Nette frameworku

Nette je sada modulárních komponent, které dohromady tvoří framework pro snadný a rychlý vývoj PHP aplikací. Je intenzivně používán (dle dostupných dat je třetím nejpoužívanějším ve světě [18]) a disponuje dobrou dokumentací [19] a aktivní uživatelskou komunitou [20].

Aplikace založené na *Nette* používají softwarovou architekturu MVC[3], jejímž smyslem je oddělit vzhled od aplikační logiky. Kontroléry jsou v *Nette* realizovány pomocí

[1] „Linux (tedy dílo pod GNU GPL, pozn. aut.) is a cancer that attaches itself in an intellectual property sense to everything it touches." – Steve Balmer, 1. června 2001, Chicago Sun Times[13].
[2] ORM – Object-relational mapping (objektově relační zobrazení).
[3] MVC – Model-View-Controller (model - pohled - kontrolér)

tzv. *presenterů*, podrobné informace o jejich životním cyklu viz dokumentace [21]. Pro realizaci pohledů používá *Nette* vlastní šablonovací systém *Latte* [22]. Model může být realizován různým způsobem, dle povahy aplikace. V našem konkrétním případě reprezentuje datový model kolekce ORM a jejich repozitářů, aplikační model pak sada vlastních služeb a fasád, které realizují požadavky presenterů. Pro integraci všech komponent aplikace se používá princip *dependency injection*, který realizuje DI kontejner *Nette*, viz dokumentace [23].

3.1.2 Client-side technologie

Pro client-side technologie z rodiny *HTML5* (*HTML, CSS3, JavaS*cript), je využit front-end toolkit *Bootstrap* verze 3 a knihovna *jQuery* verze 3.3. Stylopis je napsán v jazyce preprocesoru Less [24].

Kompilaci všech frontendových souborů má na starosti knihovna *WebLoader* [25], a to včetně sjednocení, minifikace[4] a překladu LESS souborů na CSS.

3.1.3 Tiskový můstek

Pro komunikaci s pokladní tiskárnou je použit tiskový můstek třetí strany, služba *PrintNode*. Tiskový můstek je nezbytný pro zasílání přímých instrukcí pokladní tiskárně [26], které nelze zajistit tiskem z prohlížeče ani službami typu *Google Cloud Print* [27]. Služba je placená (v přepočtu 3 USD/měsíc/počítač [28]), pro vývoj a testování byla využita zkušební doba zdarma.

Komunikace s *PrintNode* probíhá pomocí REST API, ke kterému *PrintNode* nabízí vlastní oficiální PHP knihovnu, což výrazně usnadňuje integraci této služby do aplikace.

3.1.4 Komunikace s EET

Obecný mechanismus Elektronické evidence tržeb a definice pojmů, které zavádí je vysvětlen v části 2.2.2 této práce. Z technického hlediska funguje komunikace se servery Finanční zprávy na principu výměny datových zpráv pomocí protokolu SOAP[5] přes rozhraní HTTPS (bez autentizace klientským certifikátem). Datová zpráva je SOAP XML dokument, který obsahuje specifikované údaje. Formát a struktura datových zpráv a rozhraní služby je formálně popsáno formou WSDL[6],

[4] Minifikace je proces odstranění prázdných řádek, komentářů a provedení dalších optimalizací zdrojového kódu.
[5] SOAP – Simple Object Access Protocol
[6] WSDL – Web Services Description Language

viz [29]. Obsah odesílaných datových zpráv je elektronicky podepsán dle standardu X.509 (PKI) pomocí soukromého klíče odesílatele. Každý poplatník (v případě naší aplikace tenant) získá od Finanční správy soukromý klíč v podobě PKCS#12 certifikátu chráněného heslem – tento certifikát společně s příslušným heslem aplikace uchovává pro každého tenanta a používá jej k podpisu výše popsaných datových zpráv. Způsob podpisu datových zpráv a veškeré další detaily jsou podrobně popsány v technické specifikaci vydané správcem daně [11].

Pro implementaci komunikace s EET neexistuje oficiální knihovna pro žádný jazyk – způsob implementace je zcela ponechán na subjektech soukromé sféry. Finanční správa na druhou stranu ale nijak neomezuje vývojáře aplikací. EET řešení je oprávněn vyvíjet kdokoliv a není k tomu potřeba žádná forma certifikace. Odpovědnost za hlášení tržeb do EET nese poplatník a je jen na něm, jakého dodavatele si vybere [30].

Pro komunikaci s EET existuje několik neoficiálních knihoven. Jednou z nich je *slevomat/eet-client* [31], kterou tvoří vývojáři firmy Slevomat.cz, s.r.o., která ji dává k dispozici jako open source. Tato knihovna je otestovaná [32] a hojně používaná[7]. Komunikace s EET je do aplikace integrována prostřednictvím této knihovny.

3.1.5 Správa závislostí a použité knihovny

Pro správu závislostí je použit PHP package manager *Composer [34].* Ten zajišťuje snadnou instalaci knihoven třetích stran a jejich vlastních závislostí. Všechny závislosti aplikace jsou definovány v konfiguračním souboru composeru, viz Výpis 3.1. Veškeré informace k jednotlivým balíčkům je možné nalézt v repozitáři *packagist.org* [33].

3.2 Metodika vývoje a použité nástroje

Pro vývoj byla zvolena metodika *FDD*[8]. Práce na aplikaci pobíhala po jednotlivých funkcích (*features*) a jejich souborech (*feature sets*). Funkcím odpovídají jednotlivé případy užití, funkčním souborům pak skupiny případů užití, tak jak jsou popsány v sekci 2.5 této práce.

[7] V době psaní práce přes 3 tisíce stažení za měsíc, dle informací z repositáře *packagist.org [33]*
[8] *Feature driven development*, stručné shrnutí od doktorky Buchalcevové viz [35].

```
{
  "name": "drml/vet-system",
  "description": "VetSystem - simple and easy veterinary practice
      management",
  "type": "project",
  "license": "<proprietary> All rights reserved",
  "require": {
    "php": ">=5.6.0",
    "nette/nette": "^2.4",
    "nette/utils": "2.4",
    "janmarek/webloader": "dev-master",
    "oyejorge/less.php": "v1.7.0.5",
    "components/bootstrap": "^3.3",
    "components/webfontloader": "^1.6",
    "nextras/forms": "^1.6",
    "joseki/webloader-filters": "^1.0",
    "afarkas/html5shiv": "3.7.3",
    "kdyby/doctrine": "v3.1.2",
    "ipub/visual-paginator": "v1.0.3",
    "kdyby/forms-replicator": "@dev",
    "drmonty/chosen": "1.6.1",
    "slevomat/eet-client": "1.0.2",
    "guzzlehttp/guzzle": "^6.2",
    "printnode/printnode-php": "2.0.0-rc1",
    "myclabs/php-enum": "1.5.2"
  },
  "require-dev": {
    "arachne/codeception": "v0.7.7"
  },
  "minimum-stability": "stable"
}
```

Výpis 3.1: Výpis souboru *composer.json*

Pro vývoj bylo použito IDE *PhpStorm*, jehož pokročilé funkce umožňují kontrolu kvality kódu a mj. provádění inspekcí, které jsou součástí zvolené metodiky *FDD*. K verzování kódu byl použit *Git*. Pro správu závislostí je použit package manager *Composer*. Stylopis je generován CSS preprocesorem *Less*. Jazykem pro identifikátory i komentáře je angličtina.

3.2.1 Programátorská dokumentace

Struktury zdrojového kódu jsou vhodně okomentovány, všechny struktury souborů PHP dle standardu *PHPDoc* [36]. Na základě těchto komentářových bloků byla pro celý projekt vygenerována přehledná programátorská dokumentace nástrojem *phpDocumentor* [37]. Výstupem tohoto nástroje je kolekce HTML dokumentů, které umožňují rychlé a uživatelsky přívětivé procházení všech PHP součástí aplikace. Vzhledem velkému rozsahu (přes 280 položek) je tento výstup přiložen jako elektronická příloha na přiloženém CD, viz Příloha B.

3.2.2 Statistiky

Statistiky aplikace byly vygenerovány nástrojem *PhpMetrics* [38]. Ten kromě základních statistik, umožňuje sledovat také další ukazatele, jako např. cyklomatickou složitost nebo Halsteadovy metriky velikosti programu. Tyto pokročilé metriky mohou napovědět, které elementy by bylo vhodné např. dále dekoponovat. Musíme je však brát s rezervou – např. entitní třída `App\Model\Entities\Patient` byla označena jako příliš komplexní, i když představuje jen datovou obálku s několika triviálními pomocnými metodami. Souhrnné statistiky aplikace jsou k dispozici ve Výpisu 3.2. Kompletní výstup je přiložen jako elektronická příloha na přiloženém CD, viz Příloha B.

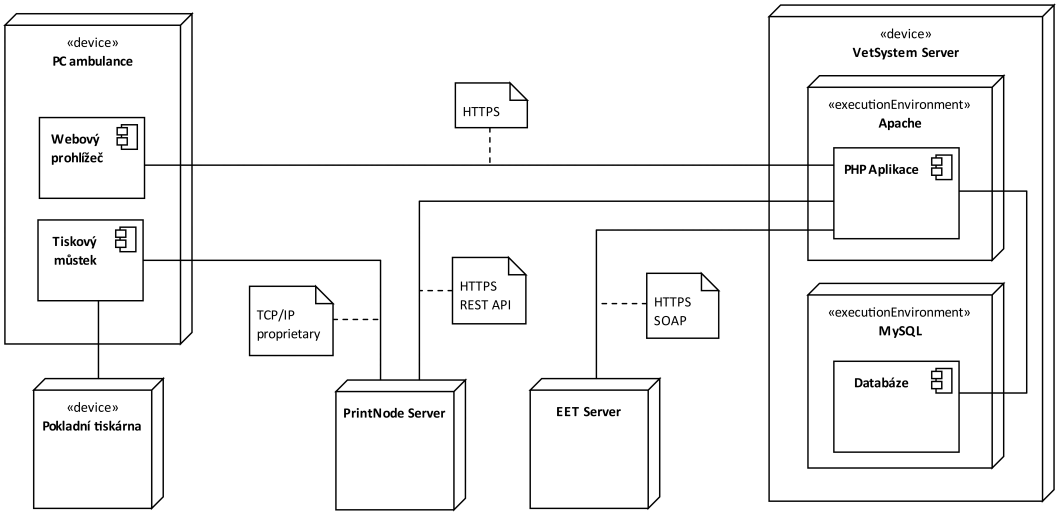
```
LOC
    Lines of code                               6552
    Logical lines of code                       3463
    Comment lines of code                       3089
    Average volume                              367.5
    Average comment weight                      43.63
    Average intelligent content                 43.63
    Logical lines of code by class              42
    Logical lines of code by method             14

Object oriented programming
    Classes                                     82
    Interface                                   2
    Methods                                     240
    Methods by class                            2.93
    Lack of cohesion of methods                 1.13

Coupling
    Average afferent coupling                   1.8
    Average efferent coupling                   3.27
    Average instability                         0.7
    Depth of Inheritance Tree                   1.38

Complexity
    Average Cyclomatic complexity by class      3.17
    Average Relative system complexity          131.87
    Average Difficulty                          5.54

Bugs
    Average bugs by class                       0.12
    Average defects by class (Kan)              0.4

Violations
    Critical                                    0
    Error                                       2
    Warning                                     9
    Information                                 0
```

Výpis 3.2: Souhrnné statistiky vygenerované nástrojem *PhpMetrics*

Tento nástroj nicméně nepostihuje ostatní typy souborů, které jsou nedílnou součástí aplikace – zejména šablony a stylopisy. Tyto další soubory jsou v celkovém rozsahu přes 4 tisíce řádek (včetně komentářů a prázdných řádků), viz Obrázek 3.1.

Extension	Count	Size SUM	Size MIN	Size MAX	Size AVG	Lines	Lines MIN	Lines MAX	Lines AVG
latte *(LATTE files)*	42x	81kB	0kB	12kB	1kB	2509	10	318	59
less *(LESS files)*	2x	11kB	3kB	7kB	5kB	649	189	460	324
sql *(SQL files)*	6x	18kB	0kB	12kB	3kB	425	1	281	70
js *(JS files)*	4x	9kB	0kB	5kB	2kB	317	8	145	79
neon *(NEON files)*	7x	6kB	0kB	2kB	0kB	222	11	99	31
htaccess *(HTACCESS files)*	6x	2kB	0kB	1kB	0kB	68	2	45	11
json *(JSON files)*	1x	0kB	0kB	0kB	0kB	30	30	30	30
md *(MD files)*	1x	0kB	0kB	0kB	0kB	22	22	22	22
yml *(YML files)*	1x	0kB	0kB	0kB	0kB	10	10	10	10
ico *(ICO files)*	1x	285kB	285kB	285kB	285kB	3	3	3	3
txt *(Text files)*	1x	0kB	0kB	0kB	0kB	0	0	0	0
Total:	**72x**	**416k**	**291k**	**329k**	**301k**	**4255**	**286**	**1413**	**639**

Obrázek 3.1: Statistika ostatních souborů (kromě PHP).



Obrázek 3.2: Schéma produkčního nasazení.

3.3 Schéma produkčního nasazení

Na Obrázku 3.2 je zachycen diagram nasazení aplikace. Aplikace samotná je spouštěna v prostředí webového serveru *Apache* (PHP aplikace mají krátký životní cyklus a spouštějí se při každém HTTP dotazu) s nakonfigurovanými moduly a rozšířeními (např. *cron* pro spouštění naplánovaných úloh přes konzolové rozhraní). Aplikace komunikuje přes interní rozhraní systému s databázovým serverem. Veškerá komunikace s vnějším světem probíhá přes standardní HTTPS, s externími službami přes jejich příslušné rozhraní. S tiskárnou veterinární ambulance komunikuje prostřednictvím služby *PrintNode*, která tiskárnu instruuje přes tiskový můstek s proprietárním síťovým rozhraním.

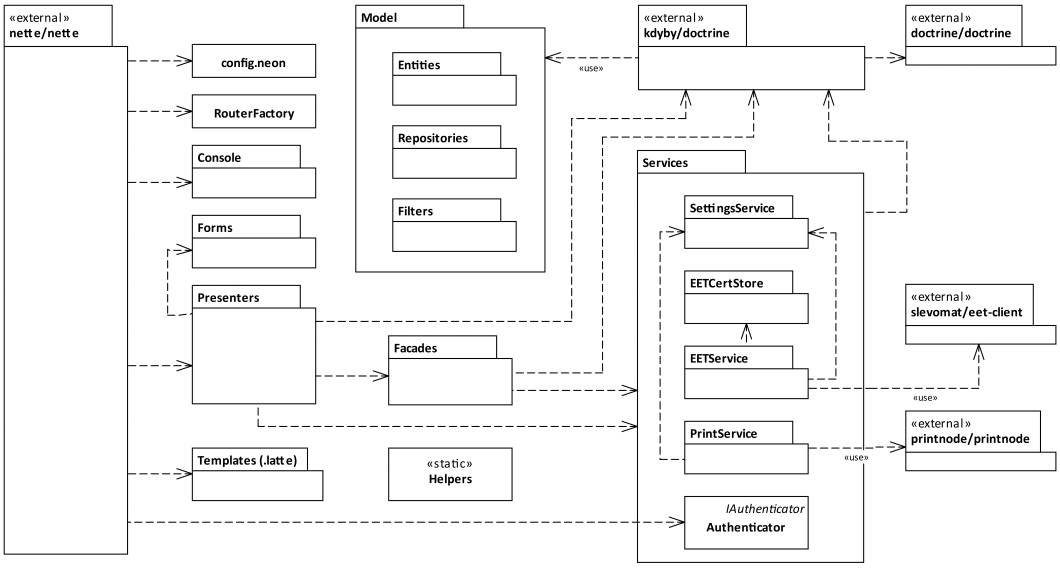
3.4 Struktura aplikace

Na následujících stranách je velmi stručně popsána struktura aplikace a okomentován účel vybraných komponent. Kompletní detailní dokumentace, včetně popisu tříd, metod a jejich rozhraní, je realizována formou HTML reportu vygenerovaného na základě *PHPDoc* komentářových bloků a je k dispozici jako elektronická příloha na přiloženém CD.

3.4.1 Metamodel komponent aplikace

Aplikace je rozdělena na několik komponent, jejichž struktura vychází z MVC architektury a zvyklostí *Nette* frameworku, na kterém je aplikace postavena. Komponentám odpovídají jmenné prostory jazyka PHP (tzv. *namespace*), do kterých jsou organizovány jednotlivé třídy. Diagram metamodelu komponent je zachycen na Obrázku 3.3. Přerušovanou čarou jsou v diagramu znázorněny závislosti (uspokojované především pomocí dependency injection kontejneru *Nette* framewokru), stereotypem *«use»* jsou zobecněny nepřímé složitější závislosti. (např. ORM knihovna *Doctrine* používá entity, repozitáře a filtry na základě sady různých mechanismů, zejména pomocí analýzy anotací s následným generování dočasných proxy objektů, se kterými pak poté reálně pracuje). Elementy knihoven třetích stran jsou označeny stereotypem *«external».* Vztahy dědičnosti jsou záměrně omezeny rámcem jednoho namespace, proto se na tomto diagramu nevyskytují, uvidíme je ale dále v diagramech jednotlivých komponent.

Všechny komponenty orchestruje knihovní balík `nette/nette`, který na základě konfigurace (`config.neon`) a předpisu routování (`RouterFactory`) zpracovává příchozí požadavky od webového serveru, podle nich volá příslušné metody jednotlivých presenterů, výsledky jejich činnosti renderuje pomocí *Latte* šablon a odpověď vrací zpět webovému serveru k dalšímu zpracování. Při realizaci pohledů používají presentery vykreslitelné komponenty, zejména pak formuláře (instance `Nette/Form`), jejichž továrny jsou sdruženy do jmenného prostoru `Forms`. Presentery přistupují ke službám aplikace buď přímo (při realizaci jednodušší funkcionality), nebo přes fasády, které agregují rozhraní ostatních komponent aplikace a realizují složitější prvky aplikační logiky. Služby ze jmenného prostoru `Services` zajišťují především spojení s externími systémy, k čemuž používají příslušné knihovny. Speciální službou je `Authenticator`, který realizuje přihlašování uživatelů, služba pro autorizaci (`Nette/Permissions`) je dynamicky instancována a konfigurována přímo na základě konfiguračního souboru (`config.neon`). Posledním prvkem na diagramu je statická



Obrázek 3.3: Metamodel komponent aplikace.

třída `Helpers`, která obsahuje sadu globálně dostupných pomocných metod, např. pro správné formátování peněžních částek a podobně.

3.4.2 Komponenty aplikace podrobně

V následujícím textu jsou stručně popsány vybrané komponenty aplikace. Diagramy tříd a závislostí vybraných komponent jsou z důvodu rozsahu připojeny v Příloze A této práce.

Presentery – `\App\Presenters`

V hierarchii presenterů na Obrázku A.1 se nachází několik abstraktních předků, které realizují společné chování svých potomků:

`BasePresenter` se stará o zavedení frontendových prvků (CSS, JS) pomocí knihovny `WebLoader`, přebírá službu nastavení aplikace `SettingService` a prostřednictvím součásti *Nette* frameworku `HttpRequest` získává z hlavičky požadavku *HTTP referer* (používá metoda `userCameFrom()`, která umožňuje dalším součástem zjišťovat tok procházení uživatele aplikací).

`AuthenticatedBasePresenter` provádí kontrolu přihlášení a oprávnění. Presentery `HomepagePresenter` a `SignPresenter`, které přihlášení nevyžadují proto dědí od `BasePresenter`.

`TenantAwareBasePresenter` představuje předka presenterů, které pro svůj chod nutně potřebují znát informaci o aktivním tenantovi. `TenantAwareBasePresenter` tuto skutečnost kontroluje a v případě potřeby umožní administrátorům přesměrování na pohled pro přepnutí kontextu tenanta.

`DialBasePresenter` sdružuje společnou funkcionalitu pro presentery, které realizují CRUDL operace číselníků. Tento abstraktní předek je konfigurovatelný pomocí parametrů konstruktoru.

Abstraktní presentery své závislosti záměrně nezískávají přes konstruktor, ale pomocí anotace `@inject`, která je specifická pro *Nette* framework. Tento způsob je vhodný pouze u předků presenterů, protože zjednodušuje správu závislostí jejich potomků. Poruší se tím však zapouzdření. Koncové presentery již závislosti získávají standardním způsobem přes konstruktor. Více k tématu viz dokumentace *Nette* [23].

Speciálním presenterem, který stojí mimo vztahy dědičnosti je `ErrorPresenter`, který zpracovává chybové stavy aplikace do lidsky čitelné podoby.

Fasády – `\App\Model\Facades`

Diagram fasád je znázorněn na Obrázku A.2. Všechny fasády pracují s databází, resp. entity managerem ORM vrstvy a se službou `SettingsService`. `InvoiceFacade` nadto používá ještě služby `EETService` pro fiskalizaci účtenek.

Služby – `\App\Services`

Na Obrázku A.3 je znázorněn jmenný prostor `\App\Services` včetně jeho podprostorů.

`PrintService` interně používá dva interface. `IPrinterDefinition` zobecňuje rozhraní pro definice, kterými se popisují technické parametry tiskárny – jaké kombinace řídících znaků tiskárna vyžaduje k provedení daných činností (formátování, otevření pokladny, odtržení papíru, self-test, apod.). Konkrétní implementací tohoto rozhraní je třída `CashinoPTPII`, která popisuje parametry stejnojmenné tiskárny. `IPrintRenderer` představuje rozhraní tříd, jejichž úkolem je převést objekt do řetězce znaků vhodných k odeslání na pokladní tiskárnu. K tomu používají zmíněné definice technický parametrů. Renderery implementující rozhraní `IPrintRenderer` poté používá třída `PrintService` k plnění svých úkolů, která v případě chyby tisku emituje vlastní typ výjimky `PrinterException`.

Služba `EETSErvice` používá ostatní služby a stejně jako `PrintService` instancuje třídy externích knihoven. Tato služba emituje dva druhy výjimek: `EETException`

pro chyby zpracování a fatální chyby EET a `PermisiveEETException` pro průchozí varování EET, které nevyžadují přerušení navazujících procesů.

`EETCertStore` poskytuje službu pro ukládání a správu EET certifikátů jednotlivých tenantů. Ukládá je na zabezpečené místo ve filesystému.

Služba `SettingService` poskytuje informace o konfiguraci systému pro zadaného tenanta. Informace vrací v instancích třídy `Settings`.

Entity – `\App\Model\Entities`

Datové třídy jmenného prostoru z diagramu na Obrázku A.4 představují databázové entity, ze kterých poté *Doctrine* nástrojem *orm:schema-tool* generuje strukturu databáze. Databázové vlastnosti jsou definovány pomocí speciálních anotací, viz dokumentace [16]. Pro generalizaci používají entity jednak klasickou dědičnost, jednak užití tzv. *trait* [39], které v tomto případě vkládají logiku umělého identifikátoru a usnadňují přístup k atributům entit. Abstraktní třída `TenantAwareEntity` neslouží pouze pro zobecnění cizího klíče pro vazbu potomků na tenanta, ale zejména jako označení entit, na které se aplikují filtry ve třídě `\App\Model\Filters\TenantFilter` – zde se zajišťuje oddělení kontextu jednotlivých tenantů. Pro reprezentaci hodnot výčtových atributů entity `Patient` slouží třídy se stereotypem *«abstract,enum»*[9].

Speciální pomocnou třídou je `InvoiceSums`, která nepředstavuje entitu, ale objekt účetních součtů účtenek. Jde o datovou třídu jejíž instance vytváří metoda `Invoice::sumUp()`.

Repozitáře – `\App\Model\Repositories`

Repozitáře na Obrázku A.5 obsahují metody realizující speciální databázové dotazy a zpřístupňují je ostatním součástem systému přes jednoduché rozhraní. Protože dědí od knihovního předka, jsou v současném stavu do jisté míry prázdné a připravené tak pro další rozšíření.

Formuláře – `\App\Forms`

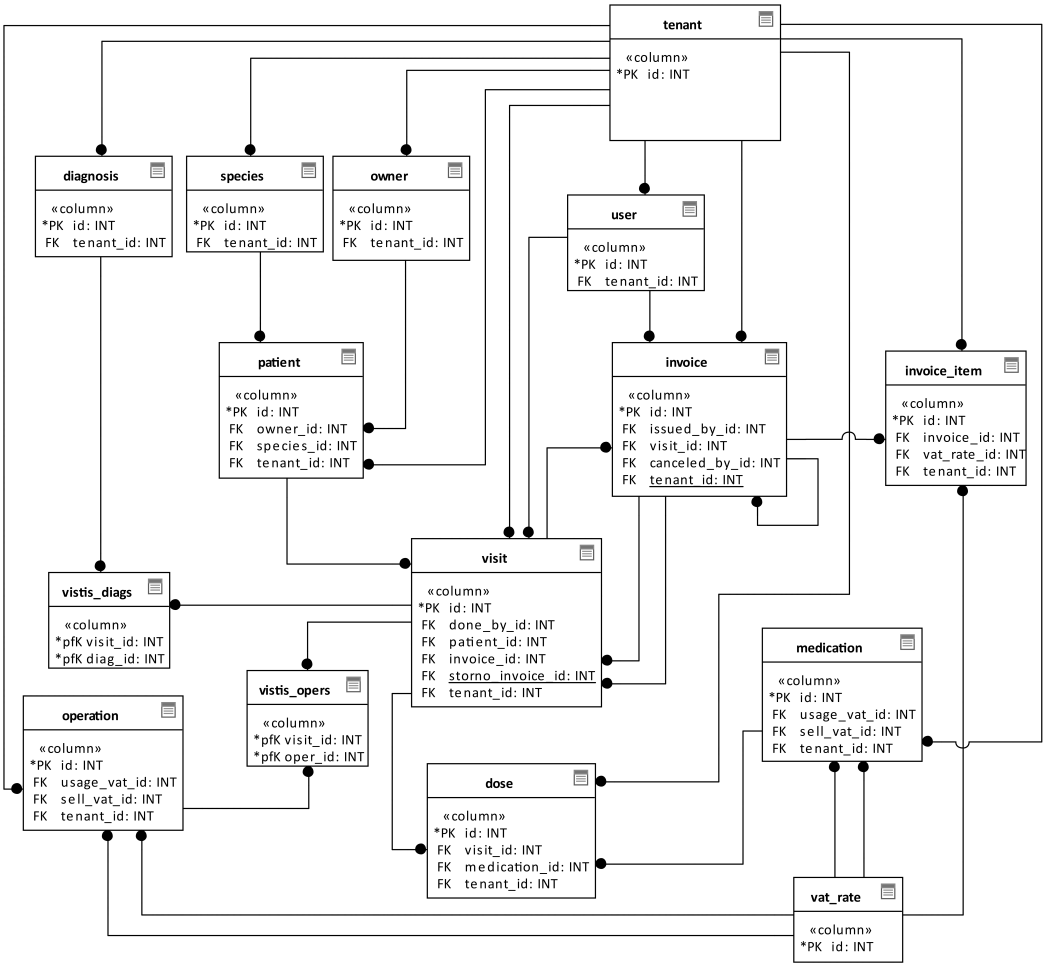
Jmenný porstor na Obrázku A.6 uchovává továrničky na komponenty formulářů – instancí `Nette\Form`. Zde se definují i jejich validační pravidla a naplňují se zde daty např. elementy *<select>*, proto některé továrničky používají entity manager ORM vrstvy. Třída `SignInFormFactory` obsluhuje přihlašovací formulář, proto potřebuje

[9] Pro upřesnění: V PHP neexistuje přímá jazyková konstrukce typu *enum*, tyto třídy chování výčtového typu pouze do jisté míry emulují.

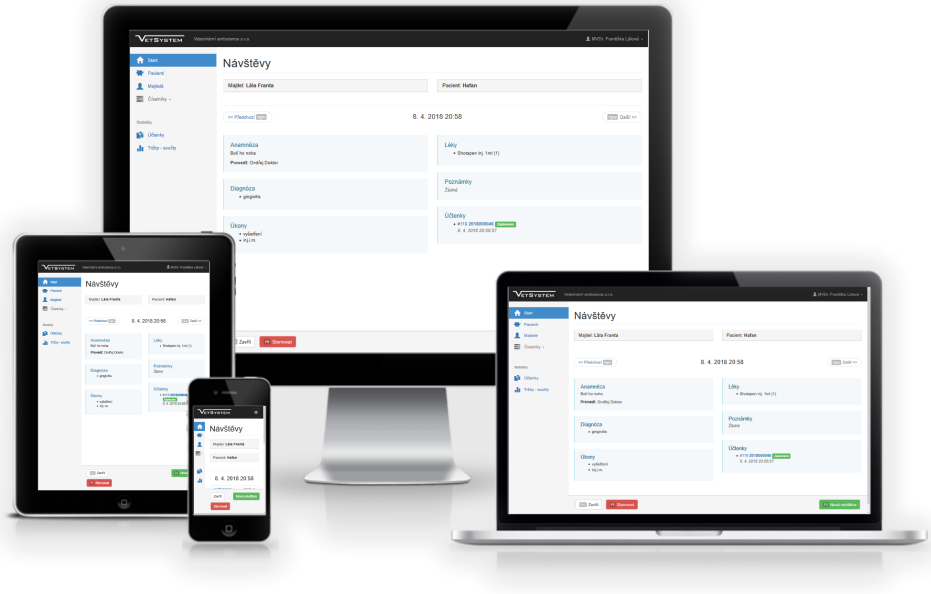
autentizační službu *Nette*. Všechny továrničky dědí od společného předka, který zajišťuje korektní zpracování chybových stavů vytvářených formulářů.

3.5 Model relační databáze

Na Obrázku 3.4 je zachycen zjednodušený model relační databáze. Zobrazeny jsou pouze klíčové atributy. Podtržené atributy jsou součástí unikátních klíčů, které zde nejsou zobrazeny. Kompletní diagram je přiložen jako elektronická příloha na CD.



Obrázek 3.4: Zjednodušený model relační databáze.



Obrázek 3.5: Ukázka webového rozhraní v různých zařízeních – karta pacienta.

3.6 Webové rozhraní a frontend

Pro webové rozhraní je jako základ použit frontendový toolkit *Bootstrap 3*, který dále rozšiřují vlastní stylopisy. Logiku frontendu dále rozšiřují skripty v jazyce *JavaScript* používající knihovnu *jQuery*, které zajišťují dynamické chování frontendu, ochranu před náhodným zavřením formulářů s neuloženými daty a zejména integraci ovládání pomocí klávesnice, které emuluje rozhraní klasické desktopové aplikace (vč. funkčních kláves). Pro zvýšení uživatelského komfortu jsou HTML elementy `<select>` a `<select multiple>` rozšířeny o vyhledávání pomocí knihovny *chosen*. Kompatibilitu ve starších verzích prohlížečů rozšiřuje polyfill knihovny *html5shiv*.

Zdrojové soubory aplikace pro frontend se nachází ve složce `app/assets` odkud je čte knihovna PHP `WebLoader`. Ta provede kompilaci preprocesoru *Less*, spojí je se zdrojovými kódy knihoven ze složky `vendor` a minifikované verze zapisuje do adresáře `www/webtemp`, odkud jsou přístupné pro webový prohlížeč. Rekompilace probíhá automaticky při detekci změn ve zdrojových souborech.

Webové rozhraní si můžete sami vyzkoušet na `https://vetsystem.ondrejdoktor.cz`, kde je připraveno veřejné demo.

4 Testování

Aplikace byla otestována akceptačními testy dle případů užití. Akceptační testy se navrhují z pohledu zákazníka a představují v podstatě přejímací zkoušku software. Pro testování byl použit nástroj *Codeception* [40], který je velmi vhodný pro testování webových aplikací jakékoliv platformy. Tento nástroj provádí akceptační testy s pomocí tzv. *php browseru*, který do jisté míry emuluje chování webového prohlížeče. V konfiguračním souboru testovací sady vývojář zadá URL adresu aplikace a testuje. Akceptační tester „nevidí" přímo do aplikace, zajímá ho pouze její vnější chování – stejně jako je tomu při provádění testů reálnou osobou. Díky tomu je možné zachytit i chyby, které by standardní jednotkové testování neodhalilo (např. chyby v konfiguraci prostředí).

Rozhraní pro zápis testů nástroje *Codeception* bylo dobře navrženo a v maximální míře odpovídá přirozenému jazyku, kterým se obvykle specifikují testovací scénáře. Ve Výpisu 4.1 je ukázka testu, který ověří úspěšné přihlášení uživatele – lékaře (bude přesměrován na startovací pohled ale neuvidí odkaz do administrace). Předpis testu byl pro přehlednost zkrácen.

```
<?php
$I->wantTo("přihlásit se k systému jako lékař");
$I->amOnPage('/sign/in');
$I->see("Přihlásit se",'h1');
$I->seeElement('form#frm-signInForm');
$I->fillField('username', 'franta');
$I->fillField('password', 'lala');
$I->click("Přihlásit se");
$I->seeInCurrentUrl('patient');
$I->see('Start', 'h1');
$I->see('Vyhledat pacienta');
$I->dontSeeLink("Uživatelé");
```

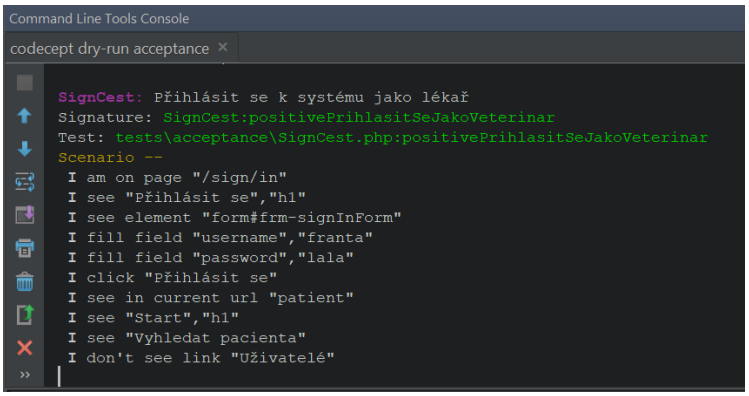
Výpis 4.1: Ukázka zápisu testovacího scénáře v nástroji *Codeception*

I když *Codeception* přímo nepracuje s kódem testované aplikace, může se napojit na databázi pomocí vlastních prostředků. Díky tomu je možné nejen ověřit obsah databáze (např. voláním `$I->seeInDatabase()` po zápisové operaci), ale především automaticky připravovat oddělenou databázi s testovacími daty před každým testem

nebo sadou testů. Tento přístup byl použit i při testování této aplikace. *Codeception* při přístupu v dotazech předává speciální *cookie*, kterou instruuje aplikaci, aby zavedla konfigurační soubor `app/config/config.test.neon` (viz `app/bootstrap.php`). Zde jsou alternativní nastavení pro provádění testů – přístup k testovací databázi, úložiště certifikátů apod.

Codeception lze také dobře integrovat s použitým IDE *PhpStorm*. Ukázka lidsky čitelného testovacího scénáře, který byl v prostředí tohoto IDE vygenerován z Výpisu 4.1 je zachycena na screenshotu na Obrázku 4.1. Provádění samotných testů je možné automaticky sekvencovat a spouštět je všechny v jedná dávce.

Aplikace byla otestována výše popsaným způsobem a všechny testy proběhly v pořádku.



Obrázek 4.1: Ukázka testovacího scénáře vygenerovaného z předpisu na Výpisu 4.1.

5 Závěr

Výsledkem této práce je funkční aplikace, která umožňuje administrativní vedení malých veterinárních ambulancí v souladu se současnými technickými a legislativními požadavky. Vystavuje platné účetní doklady a je napojena na systém EET. Aplikace je multitenantní, čímž výrazně zjednodušuje nasazení a správu, přičemž bezpečně odděluje datové kontexty jednotlivých tenantů.

Aplikace poskytuje responsivní webové rozhraní, které bylo navrženo s důrazem na jednoduchost a snadné ovládání. Pro uživatele je důležité zejména zmíněné ovládání klávesnicí, které se snaží dodržet způsob ovládání nahrazovaných aplikací vyvinutých na platformě PC-FAND. Tím je dosažen dílčí cíl zachovat přiměřenou zpětnou kompatibilitu z hlediska workflow.

Vývoj probíhal dle zvolené metodiky FDD s iteracemi po jednotlivých funkčních celcích v moderním vývojovém prostředí, které integruje nástroje pro verzování kódu a kontrolu jeho kvality pomocí inspekcí. Dostupná je kompletní dokumentace. Uživatelská dokumentace je realizována formou videonávodu, který je umístěn na přiloženém CD.

Aplikace pokrývá všechny identifikované případy užití, je otestovaná akceptačními testy a připravená na produkční nasazení s obchodním modelem SaaS.

Demo aplikace je nasazeno a veřejně dostupné na webovém severu, kde si může práci s aplikací kdokoliv vyzkoušet: `http://vetsystem.ondrejdoktor.cz/`

Aplikace je připravena pro implementaci dalších funkcí, zejména pak podpory širší škály pokladních zařízení, řešení skladového hospodářství, napojení na API dodavatelů léčiv či rozhraní pro registraci petpasů Komory veterinárních lékařů ČR. Díky multitenaci je však možné jednotlivá rozšíření nasazovat postupně (např. i formou příplatkové služby).

Na základě výše uvedených skutečností je možné konstatovat, že cíle práce bylo dosaženo v plném rozsahu.

Odkazy a literatura

[1] WinVet [online]. Brno: *HENRY SCHEIN*, 2017 [cit. 2017-10-10]. Dostupné z: http://www.winvet.cz/

[2] SHÁNĚL, Pavel, Jaroslav ROČEK a Martin KOUDELA. Uživatelská příručka programu WinVet 2011. In: *WinVet* [online]. Brno: Noviko, 2011 [cit. 2017-10-10]. Dostupné z: http://winvet.cz/images/stories/Download/Manual/WinVet_2011.pdf

[3] VETport [online]. Milford, OH: *VETPORT*, 2017 [cit. 2017-10-10]. Dostupné z: https://www.vetport.com/web.html

[4] GRYM, Martin. Pasové minimum pro praktické veterinární lékaře [online]. Praha: Komora veterinárních lékařů ČR, 2015 [cit. 2018-02-18]. Dostupné z: http://www3.vetkom.cz/content/showPage/pasove-minimum-pro-prakticke-veterinarni-lekare-779

[5] Časté dotazy a odpovědi. *Finanční správa ČR* [online]. Praha, 2018 [cit. 2018-02-14]. Dostupné z: http://www.financnisprava.cz/cs/dane/dane/dan-z-pridane-hodnoty/kontrolni-hlaseni-DPH/dotazy-a-odpovedi#IV

[6] ČESKO. § 7 odst. 1 zákona č. 112/2016 Sb., o evidenci tržeb. In: *Zákony pro lidi.cz* [online]. © AION CS 2010-2018 [cit. 2018-02-17]. Dostupné z: https://www.zakonyprolidi.cz/cs/2016-112#p7-1

[7] Způsoby evidence a účtenka. *Etržby.cz* [online]. Praha: Finanční správa ČR, 2018 [cit. 2018-02-14]. Dostupné z: http://www.etrzby.cz/cs/zpusoby-evidence-a-uctenka

[8] Odkdy evidovat tržby: Fáze postupného náběhu evidence tržeb dle klasifikace NACE. *Etržby.cz* [online]. Praha: Finanční správa ČR, 2016 [cit. 2018-02-18]. Dostupné z: http://www.etrzby.cz/cs/odkdy-evidovat-trzby

[9] Kdo a jaké tržby eviduje. *Etržby.cz* [online]. Praha: Finanční správa ČR, 2016 [cit. 2018-02-18]. Dostupné z: http://www.etrzby.cz/cs/kdo-a-jake-trzby-eviduje

[10] Základní kroky k evidenci tržeb. *Etržby.cz* [online]. Praha: Finanční správa ČR, 2017 [cit. 2018-02-18]. Dostupné z: http://www.etrzby.cz/cs/zakladni-kroky-k-evidenci-trzeb

[11] *Formát a struktura údajů o evidované tržbě* [online]. Verze 3.1.1. Praha: Finanční správa ČR, 2016 [cit. 2018-02-18]. Dostupné z: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

[12] LÁNÍK, Zbyněk. Návod pro nakládání s osobními údaji na veterinárním pracovišti. In: *Komora veterinárních lékařů České republiky* [online]. Brno: Komora veterinárních lékařů České republiky, 2018, 6.4.2018 [cit. 2018-01-16]. Dostupné z: https://www.vetkom.cz/category/rubrika-pro-veterinare/rubrika-legislativa-pro-veterinare/ochrana-osobnich-udaju-gdpr/

[13] Microsoft CEO takes launch break with the Sun-Times. *Chicago Sun Times* [online]. Digital Chicago, 2001 [cit. 2017-11-18]. Dostupné z: https://web.archive.org/web/20011108013601/http://www.suntimes.com/output/tech/cst-fin-micro01.html

[14] DOKTOR, Ondřej. *Díla třetích stran z pohledu webdesignéra*. Světlá nad Sázavou, 2015. Absolventská práce. Akademie – Vyšší odborná škola, Gymnázium a Střední odborná škola uměleckoprůmyslová Světlá nad Sázavou.

[15] Turbocharging the Web with PHP7. *Zend the PHP Company* [online]. Louisville, CO 80027: Zend Technologies, 2015 [cit. 2018-01-05]. Dostupné z: http://www.zend.com/en/resources/php7_infographic

[16] *Object Relational Mapper* [online]. Doctrine Project, 2018 [cit. 2018-01-10]. Dostupné z: https://www.doctrine-project.org/projects/orm.html

[17] Kdyby/Doctrine. *GitHub* [online]. GitHub, Inc., 2018 [cit. 2018-01-10]. Dostupné z: https://github.com/Kdyby/Doctrine

[18] The Best PHP Framework for 2015: SitePoint Survey Results. In: *SitePoint* [online]. Collingwood VIC 3066, Australia: SitePoint Pty., 2015, 30.3.2015 [cit. 2018-01-12]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

[19] Seznámení s Nette Frameworkem. *Nette Framework* [online]. Nette Foundation, 2018 [cit. 2018-02-04]. Dostupné z: https://doc.nette.org/cs/2.4/getting-started

[20] *Nette Forum* [online]. Nette Foundation, 2018 [cit. 2018-04-16]. Dostupné z: https://forum.nette.org

[21] MVC aplikace & presentery. *Nette Framework* [online]. Nette Foundation, 2018 [cit. 2018-02-04]. Dostupné z: https://doc.nette.org/cs/2.4/presenters

[22] *Latte* [online]. Nette Foundation, 2018 [cit. 2018-02-04]. Dostupné z: https://latte.nette.org/

[23] Dependency Injection. *Nette Framework* [online]. Nette Foundation, 2018 [cit. 2018-02-04]. Dostupné z: https://doc.nette.org/cs/2.4/dependency-injection

[24] *Less.js: Getting started* [online]. Berlin, Germany: Less team, 2018 [cit. 2018-02-05]. Dostupné z: http://lesscss.org/

[25] WebLoader. *GitHub* [online]. GitHub, Inc., 2018 [cit. 2018-02-05]. Dostupné z: https://github.com/janmarek/WebLoader

[26] What is Raw Printing?. *PrintNode: Cloud Printing, Remote Printing For Web Applications* [online]. London, United Kingdom: PrintNode, 2018 [cit. 2018-02-13]. Dostupné z: https://www.printnode.com/docs/what-is-raw-printing/

[27] PrintNode Vs. Google Cloud Print. *PrintNode: Cloud Printing, Remote Printing For Web Applications* [online]. London, United Kingdom: PrintNode, 2018 [cit.2018-02-13]. Dostupné z: https://www.printnode.com/docs/printnode-vs-google/

[28] Cloud printing at a great price. *PrintNode: Cloud Printing, Remote Printing For Web Applications* [online]. London, United Kingdom: PrintNode, 2018 [cit. 2018-02-13]. Dostupné z: https://www.printnode.com/pricing/

[29] EETServiceSOAP.wsdl. *Etržby.cz* [online]. Praha: Finanční správa ČR, 2016 [cit. 2018-02-18]. Dostupné z: http://www.etrzby.cz/assets/cs/prilohy/EETServiceSOAP.wsdl

[30] Základní informace pro vývojáře. *Etržby.cz* [online]. Praha: Finanční správa ČR, 2016 [cit. 2018-02-18]. Dostupné z: http://www.etrzby.cz/cs/zakladni-informace-pro-vyvojare

[31] Client for EET - Elektronická evidence tržeb. *GitHub* [online]. GitHub, Inc., 2018 [cit. 2018-03-01]. Dostupné z: https://github.com/slevomat/eet-client

[32] Code Quality Summary - slevomat/eet-client. In: *Scrutinizer*: Your platform for software quality management [online]. Kassel, Germany: Scrutinizer, 2018 [cit. 2018-03-01]. Dostupné z: https://scrutinizer-ci.com/g/slevomat/eet-client/?branch=master

[33] *Getting Started: The PHP Package Repository* [online]. Berlin, Germany: Private Packagist UG, 2018 [cit. 2018-03-01]. Dostupné z: https://packagist.org/

[34] *Composer*: Dependency Manager for PHP [online]. 2018 [cit. 2018-03-12]. Dostupné z: https://getcomposer.org/

[35] BUCHALCEVOVÁ, Alena. Metodika feature-driven development neopouští modelování a procesy, a přesto přináší výhody agilního vývoje [online]. Katedra informačních technologií VŠE Praha, , 6 [cit. 2017-11-18]. Dostupné z: http://nb.vse.cz/~buchalc/clanky/tsw2005.pdf

[36] PSR-5: PHPDoc. *GitHub* [online]. GitHub, Inc., 2018 [cit. 2018-04-12]. Dostupné z: https://github.com/php-fig/fig-standards/blob/master/proposed/phpdoc.md

[37] *PhpDocumentor*: Analyzes your code to create great documentation [online]. 2018 [cit. 2018-04-12]. Dostupné z: https://www.phpdoc.org/

[38] *PhpMetrics*: *Static analysis for PHP - by Jean-François Lépine* [online]. 2018 [cit. 2018-04-12]. Dostupné z: http://www.phpmetrics.org

[39] Traits. In: *The PHP Manual* [online]. The PHP Group, 2018 [cit. 2018-04-12]. Dostupné z: http://php.net/manual/en/language.oop5.traits.php

[40] *Codeception* [online]. Krakow, Poland: Optimum Solutions Sp. z o.o., 2018 [cit. 2018-04-12]. Dostupné z: https://codeception.com/

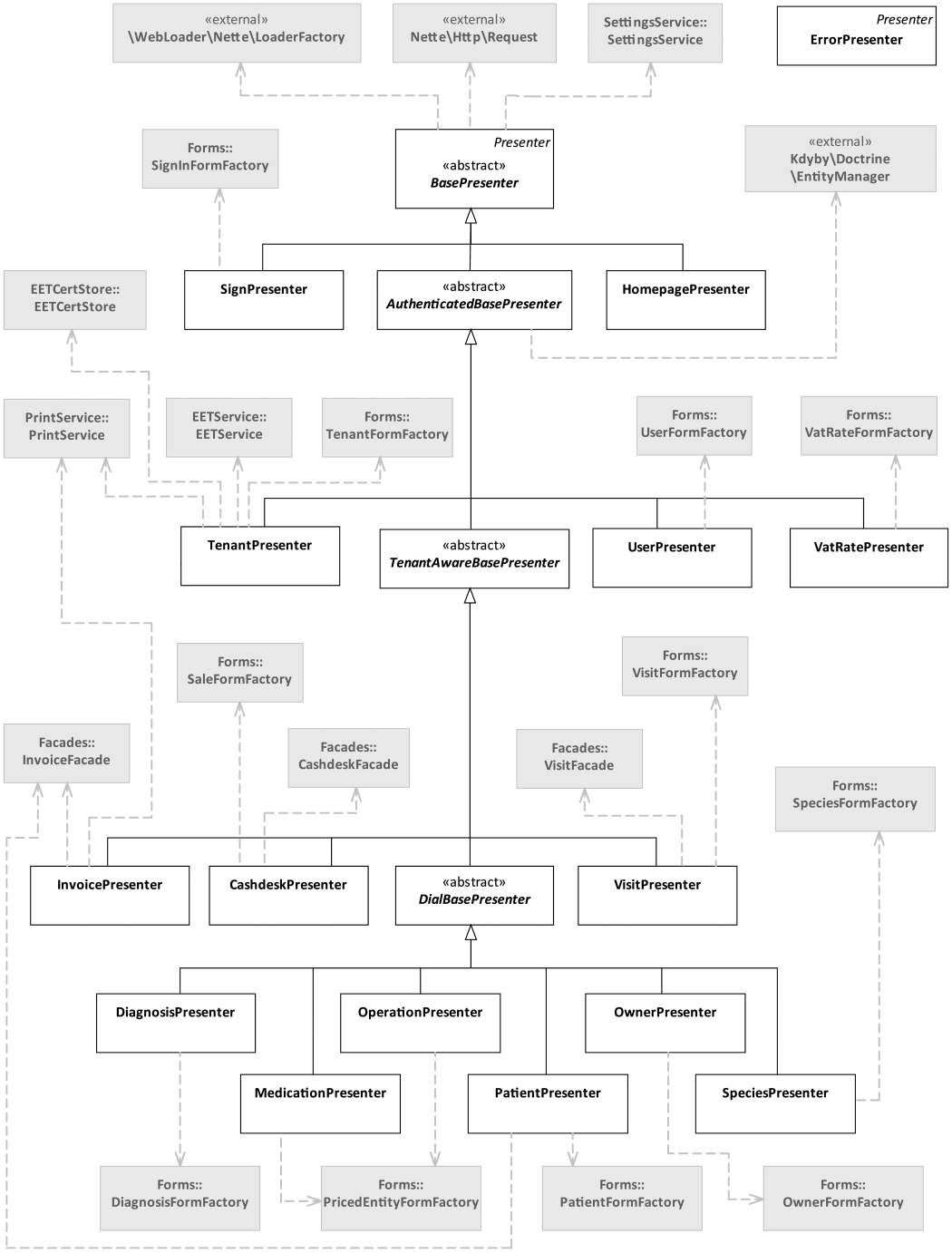
Legislativa

- Zákon č. 455/1991 Sb. živnostenský zákon
- Zákon č. 634/1992 Sb. o ochraně spotřebitele
- Zákon č. 235/2004 Sb. o dani z přidané hodnoty
- Zákon č. 112/2016 Sb. o evidenci tržeb
- Nařízení Evropského parlamentu a Rady (EU) 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES (obecné nařízení o ochraně osobních údajů, označované zkratkou GDPR)

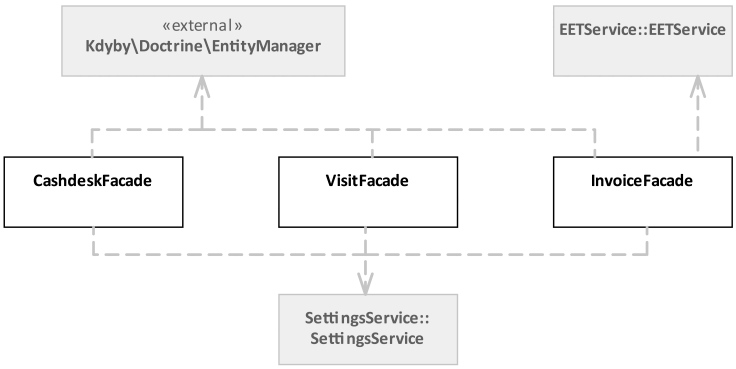
Seznam příloh

- Příloha A – Diagramy tříd a závislostí
- Příloha B – Obsah přiloženého CD

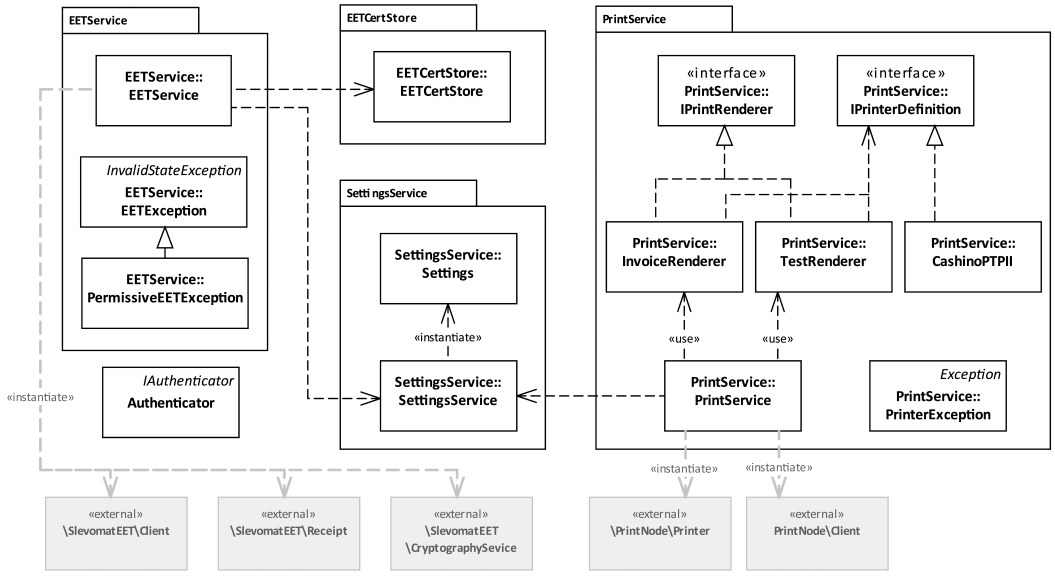
Příloha A – Diagramy tříd a závislostí



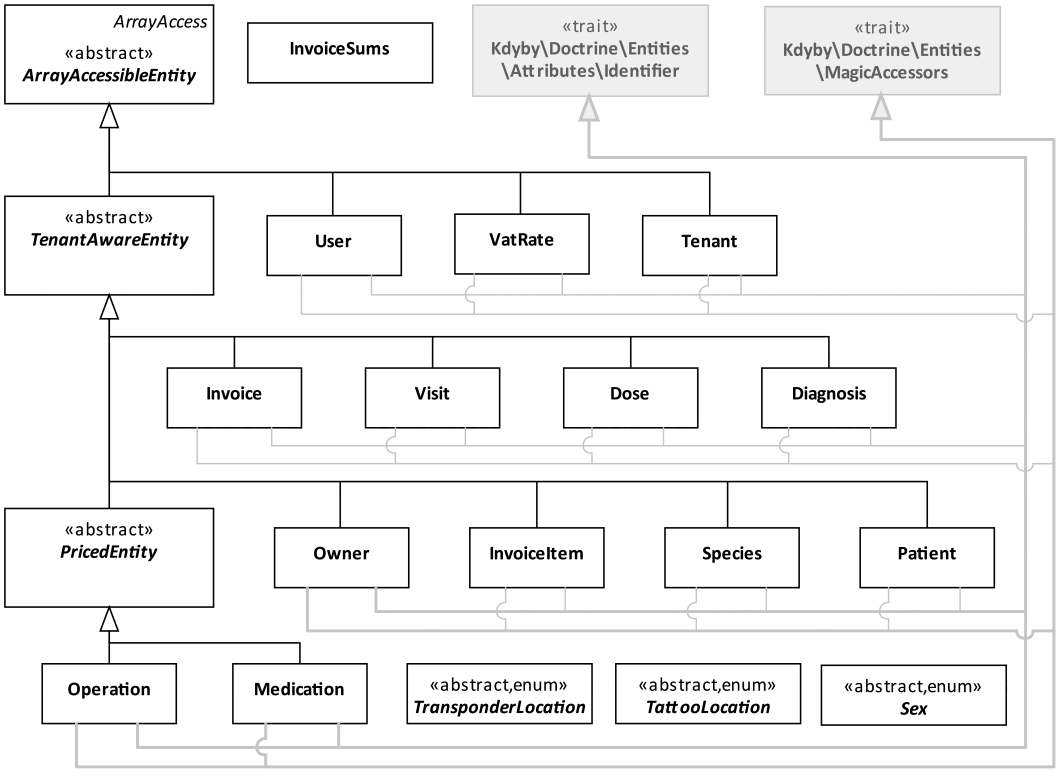
Obrázek A.1: Diagram tříd a závislostí - namespace `\App\Presenters`. Elementy mimo namespace jsou vysázeny zašedle. Závislosti vkládané pomocí DI kontejneru jsou zobrazeny přerušovanou čarou, pouze do první úrovně.



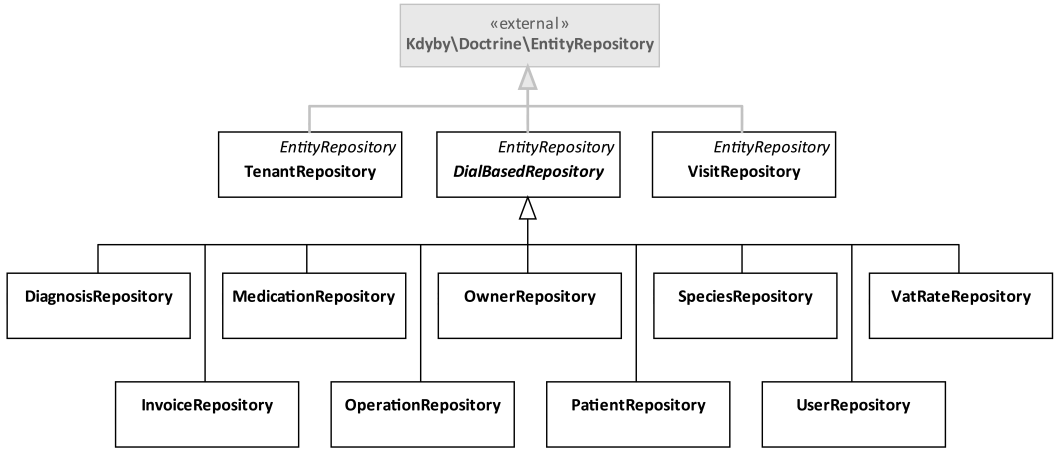
Obrázek A.2: Diagram tříd a závislostí - namespace `\App\Model\Facades`.
Elementy mimo namespace jsou vysázeny zašedle.



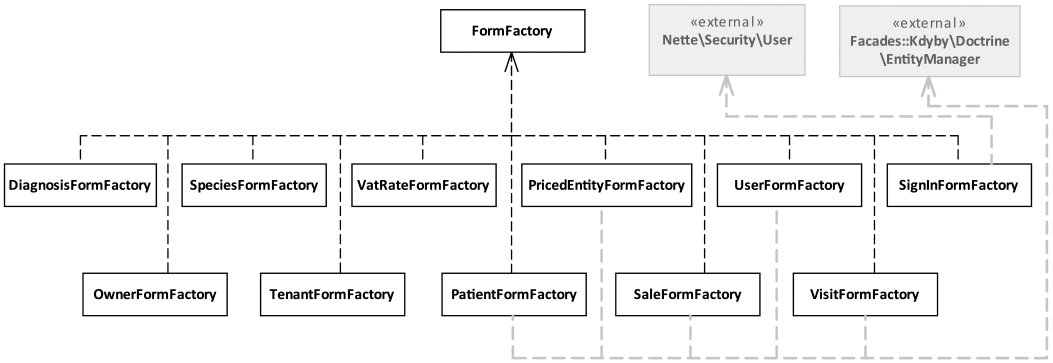
Obrázek A.3: Diagram tříd a závislostí - namespace `\App\Services`, včetně vnořených namespace.
Elementy mimo namespace jsou vysázeny zašedle.



Obrázek A.4: Diagram tříd a závislostí - namespace `\App\Model\Entities`. Elementy mimo namespace jsou vysázeny zašedle.



Obrázek A.5: Diagram tříd a závislostí - namespace `\App\Model\Repositories`. Elementy mimo namespace jsou vysázeny zašedle.



Obrázek A.6: Diagram tříd a závislostí - namespace `\App\Forms`.
Elementy mimo namespace jsou vysázeny zašedle. Závislosti vkládané pomocí DI kontejneru jsou zobrazeny přerušovanou čarou, pouze do první úrovně.

Příloha B – Obsah přiloženého CD

- `vetsystem` — zdrojové kódy aplikace
- `api` — programátorská dokumentace vygenerovaná nástrojem *phpDocumentor* (otevřete soubor `index.html`)
- `metrics` — metriky PHP kódu aplikace vygenerované nástrojem *phpMetrics* (otevřete soubor `index.html`)
- `DOKTOR - Bakalářská práce.pdf` — elektronická verze textu této práce
- `PŘÍLOHA - Kompletní schéma relační databáze.pdf`
- `PŘÍLOHA - Dokumentace případů užití.pdf`
- `PŘÍLOHA - Uživatelská dokumentace.avi`