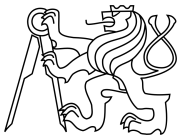
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Hematologický registr léčby (HeRgoT)
Student:	Petr Pikaus
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Cílem práce je vytvořit desktopovou aplikaci pro evidenci a vyhodnocení léčby hemofilických pacientů.

1. Na základě diskuse se zadavatelem proveďte specifikaci funkčních a nefunkčních požadavků na aplikaci.
2. Dle sebraných požadavků proveďte návrh systému, diskutujte možnosti a zvolte implementační platformu.
3. Návrh implementujte.
4. Implementujte sadu vhodných unit testů.
5. Aplikaci zdokumentujte.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 17. listopadu 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Hematologický registr léčby (HeRgoT)

Petr Pikaus

Vedoucí práce: Jiří Hunka

12. května 2016

Poděkování

Rád by jsem poděkoval vedoucímu práce, Ing. Jiřímu Hunkovi za nasměrování, na co se v práci zaměřit. Zadavateli projektu, MUDr. Peteru Salajovi za spolupráci na realizaci a PhDr. Vladimíru Dolejšovi za korektoru hemofilické části této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2016 .....

Odkaz na tuto práci

Pikaus, Petr. *Hematologický registr léčby (HeRgoT)*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce se zaobírá analýzou, návrhem, implementací a testováním registru pacientů s deficitem některého koagulačního faktoru. Rozebírá použité technologie a jejích uplatnění v projektu. Výstupem praktické části je aplikace pro systém Windows umožňující sledování a vyhodnocení léčby pacienta.

Klíčová slova hemofilie, koagulační faktor, registr pacientům, optimalizace léčby, digitalizace dat

Abstract

This bachelor thesis deals with analysis, design, implementation and testing of a registry of patients suffering from coagulation factor deficit. The thesis disccuses technologies that were used in the project. Output of the practical part is a Windows application that allows you to monitor and evaluate the patient treatment.

Keywords hemophilia, coagulation factor, registry of patients, treatment optimalization, data digitalization

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní moderní době se stále najdou odvětví, kde se technologie nepoužívají k jejich plnému potenciálu. Jedním z typických představitelů je lékařství, kde elektronizace probíhá velice pomalu a i teď jsou normou papírové formuláře. Toto konkrétně v lékařství, představuje zásadní problém pro vyhodnocování získaných dat ohledně léčby pacientů. Další využívání výsledků léčby má kritický vliv na zlepšování kvality a snižování nákladů léčby.

Hemofilie je závažné genetické onemocnění, které má za následek krvácení do kloubů, svalů a podkoží. Nejenom, že jsou pacienti omezeni ve svých aktivitách, ale zároveň trpí zhoršenou kvalitou života. Jedním z nejhorších následků hemofilie je omezení rozsahu pohybu kloubů, které může vést až k úplné invaliditě pacienta. Nalezením správné léčby můžeme snížit náklady, zatímco u pacienta snížíme následky a obtíže plynoucí z nemoci.

Tato práce se zaměří na analýzu problémové domény a následné vytvoření aplikace pro sběr a vyhodnocení dat pacientů trpících hemofilií. To bude probíhat ve spolupráci s vedoucím pražského Centra pro trombózu a hemostázu Ústavu hematologie a krevní transfuze, MUDr. Peterem Salajem.

Kapitola 1

Hemofilie

Pro pochopení významu aplikace je potřeba vyjasnit, co vlastně hemofilie je a jak ovlivňuje pacienty, kteří jí trpí. Mimo to bude rovněž vysvětleno, jak probíhá její léčba, jaká jsou její rizika a u koho se vyskytuje.

1.1 Faktory srážlivosti

Faktor srážlivosti, nebo-li koagulační faktor, je bílkovina obsažená v krvi, která přispívá k jejímu srážení. Těchto bílkovin je se v krvi nachází 13 [4, str.1690] a tvoří takzvanou koagulační kaskádu. Na počátku této kaskády dochází k reakci Fibrinogenu (faktor I) a trombinu (faktor II). Neaktivní trombin je běžně přítomný v krvi, po jeho aktivaci začíná proces srážení. Zbytek procesu sestává z následné reakce dalších faktorů, kde každý faktor aktivuje ten následující, a při existenci všech faktorů v potřebném množství dochází ke správnému srážení krve.

Deficit faktoru se udává v procentech běžného normálu, takže čím nižší procento, tím závažnější porucha.

1.2 Základní popis diagnózy

Hemofilie je onemocnění způsobené deficitem koagulačního faktoru VIII (hemofilie A) a IX (hemofilie B). Za následek má poruchu srážlivosti krve, která vede ke krvácení do svalů, kloubů a podkoží. Krvácení do kloubů představu veliké nebezpečí pro pacienta, protože poškozuje samotný kloub a snižuje jeho celkovou hybnost. Příklad toho, jak vypadá krvácení do kloubu, můžete vidět na obrázku 1.1.

Toto onemocnění je genetické a je způsobená defektem na chromozomu[1] $\mathcal{X}$. Muži mají pohlavní chromozomy$\mathcal{X}$ a $\mathcal{Y}$, zatímco ženy mají dva chromozomy

[1]Chromozómy jsou obsaženy v jádře buněk a jsou nositeli genetické informace ve formě DNA [5].

Obrázek 1.1: Krvácení do kloubu, takzvaná hemartróza [1]

$\mathcal{X}$. Když dochází k početí potomka, každý z rodičů předává jeden ze svých pohlavních chromozomů. Tento převzatý chromozom je náhodný a díky tomu je poměr mužů a žen celosvětově velice blízko 1, což potvrzují i známá data [6].

1.3 Výskyt nemoci

„V České republice je v současnosti registrováno přibližně 870 hemofiliků, 750 osob s hemofilií A (nedostatek faktoru VIII) a 120 osob s hemofilií B (nedostatek faktoru IX).“ [7]

Hemofilie se většinou vyskytuje u mužů, to je díky dvěma chromozomům $\mathcal{X}$ v ženském těle, kde přítomnost jednoho zdravého chromozomu je dostatečná pro správnou funkci. Díky tomu, může žena být přenašečkou hemofilie, aniž by trpěla klinickými příznaky. Žena může trpět projevy pouze tehdy, pokud má vadné oba dva chromozomy $\mathcal{X}$. To se objevuje pouze ve „velice raritních případech“ [2, str.256].

1.4 Léčba

Závažnost hemofilie je určena podle hladiny faktoru v krvi. Čím je jeho hladina nižší, tím musí být terapie intenzivnější, protože pacientovi stačí mnohem menší podnět k tomu, aby začal krvácet.

V lehčích případech stačí aplikovat chybějící faktor pouze po závažných úrazech či před velkými chirurgickými zákroky. V těch nejzávažnějších je po-

tom pacient odkázán na celoživotní udržování určité hladiny faktoru pomocí pravidelných aplikací faktoru, takzvané profylaxe. V tabulce 1.1 můžete vidět přibližné rozdělení závažnosti nemoci podle hladiny faktoru v krvi.

Absence koagulačních faktorů FVIII a FXIV			
	Těžká forma	Středně těžká forma	Lehká forma
FVIII:C, resp. FIX:C	< 1% (0,01 m.j./ml)	1-5% (0,01-0,05 m.j./ml)	> 5% (0,05 m.j./ml)
Krvácení	Časté spontánní + velké po minimálních úrazech a chirurgických výkonech	Ojedinělé spontánní + po úrazech a při chirurgických výkonech	Raritně spontánní + při větších úrazech a chirurgických výkonech

Tabulka 1.1: Vztah hladin FVIII:C, FIX:C k intenzitě krvácení [2, str.257]

1.4.1 Krvácivé epizody

Jedním z typů léčby je akutní léčba takzvaných krvácivých epizod. Ve chvíli krvácení si musí pacient k jeho zástavě aplikovat chybějící faktor. Dávka léku, v němž je faktor obsažen, je především odvislá od váhy pacienta a závažnosti krvácení.

Důvodu pro krvácení je mnoho, od spontánního krvácení, přes velkou fyzickou námahu až po úraz. Krvácení nemusí být zastaveno jednou aplikací. V těchto případech pacient pokračuje v aplikovaní léku v určitých časových intervalech až do jeho zastavení.

1.4.2 Profylaxe

Druhý typ léčby hemofilie je profylaktická léčba. Jejím základem je podávání preparátu pro udržení určité hladiny faktoru v krvi k předcházení krvácení. Důvodů k profylaktické léčbě je několik.

Obecná je podávána u pacientu, který má hladinu faktoru nižší než 2%. To je hranice, kdy pacient přestává spontánně krvácet. Tím se z pacienta z těžkou formou stává pacient se středně těžkou formou nemoci.

Operace před každým chirurgickým zákrokem se pacientovi podává velké množství faktoru. Podané množství se řídí podle potřeby dosáhnout hranice zdravého člověka. Několik dní po operaci se potom udržuje hranice 50%. Mezi chirurgické zákroky se řadí například i zákroky na stomatochirurgii, jako je trhání zubů apod.

Úraz když má pacient úraz, zvedá se hladina faktoru po celou dobu léčby úrazu pro předejití případných krvácení. Vyšší hladina se udržuje do té doby než je úraz vyléčen.

Závažné krvácení nastává-li u pacienta závažné krvácení, je i po jeho akutním zastavení zvýšená šance k dalšímu krvácení. Tomu se předejde zvýšením hladiny faktoru, což umožní vymizení již zastaveného krvácení bez dalších komplikací.

Rehabilitace například z důvodu poškození kloubů hemofilii je pacient nucen k rehabilitacím. V době rehabilitace se zvýší hladina faktoru, protože samotná rehabilitace je velkou zátěží pro pacienta a je to období zvýšené šance na krvácení.

1.4.3 Rizika

Léčba hemofilie s sebou dříve přinášela velká rizika. Jednalo se hlavně o nakažení hepatitidou a virem HIV. Vzhledem k tomu, že dříve se léky vyráběly exkluzivně z krevní plazmy a úroveň screeningu krve proti různým virovým onemocněním byla na velice nízké úrovni, bylo nakažení hemofilika po aplikaci koncentrátu velice pravděpodobné. K roku 1982 bylo přibližně 50% diagnostikovaných pacientů infikovaných virem HIV [8, str.1300].

V dnešní době to už neplatí. V první řadě se pro léky vyráběné z krevní plazmy naprosto změnil proces ověřování přijímané krve od dárce na různá virová onemocnění. Mimo to prochází krev procesem virové deaktivace [8, str.1300]. Dalším z důvodů, je příchod takzvaných rekombinantů na trh. To jsou léky, které nejsou vyráběny z lidské plazmy, ale místo toho jsou použity geny daných faktorů. Ty jsou extrahovány pomocí genetického inženýrství a následně se vkládají do nelidských buněk, které produkují žádaný faktor. Nejčastěji se používají buňky křeččích ledvin [9].

1.4.4 Náklady

Hemofilie je velice nákladné onemocnění na léčbu. V České republice vydala VZP v roce 2013 za nejdražšího pacienta 46 007 863,- [10]. Celková léčba poruch srážlivosti krve za rok 2014 stála VZP 470 milionů korun [11]. To jsou náklady jen za VZP, údaje od jiných pojišťoven nejsou dostupné.

Co se týče průměrné ceny léčby na pacienta, řadí se Česká republika mezi levnější země EU. V tabulce 1.2 můžete vidět náklady na pacienta léčeného profylaxí, pouze po nastání krvácivé epizody a průměrné náklady za všechny pacienty.

Průměrná cena na pacienta za rok			
	Epizody	Profylaxe	Průměr
Česká republika	24 tis. €	64 tis. €	34 tis. €
EU	30 tis. €	150 tis. €	-

Tabulka 1.2: Cena léčby hemofilických pacientů v ČR a EU [3]

Kapitola 2

Analýza domény

Na trhu dodnes neexistuje aplikace, která by pro danou problematiku byla vhodná. Hlavním důvodem jsou například lišící se podmínky na data kladené pojišťovnami po celém světě.

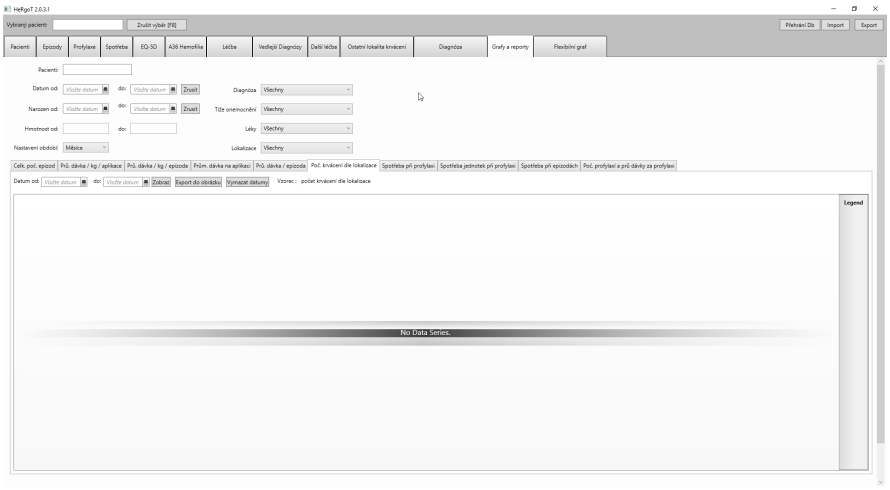
2.1 První verze aplikace

V roce 2012 vznikla první verze aplikace HeRgoT. Ta může být považována za prototyp. Aplikace umožňovala základní práci s pacienty a jejich daty, mezi které se řadí následující úkony

- správa
  - pacientů
  - epizod
  - profylaxí
- vykreslování grafů

Při převzetí projektu od tvůrce bylo nalezeno velké množství kritických nedostatků. Kromě chybějících atributů léčby se jednalo především o

- špatnou správu paměti
- pomalé zpracování dat
- nulové zabezpečení dat
- velký počet chyb ve vyhodnocování
- nekonzistentní databáze
- uživatelsky nepříjemné UX



Obrázek 2.1: Grafový modul první verze aplikace HeRgoT

Po zmapování stavu bylo rozhodnuto, že pokračovat v rozvoji aktuální aplikace nemělo smysl. Časová a finanční náročnost na opravení chyb, přidání chybějících vlastností a provedení optimalizace by byla mnohonásobně vyšší, než vybudovat aplikaci znovu.

2.2 Požadavky

Sběr požadavků probíhal tak, že byla sepsána dokumentace pro první verzi aplikace. Tato se následně prošla s MUDr. Peterem Salajem a byla upravena podle vznesených nároků. Ačkoli se věnovalo sběru požadavků velké množství času, s postupným vývojem se věci měnily. To bylo zapříčiněno hlavně následkem praktického použití aplikace, které ukázalo, že některé úkony potřebné ke správě dat byly velice náročné a mimo to poškozovaly kvalitu dat v databázi.

2.2.1 Funkční požadavky

1. Obecné
    a) Ověření identity uživatele
    b) Existence centralizovaného výběru pacienta
    c) Aplikace bude nabízet uložení změn při přechodu z detailu položky
    d) Aplikace bude ověřovat validitu zadaných dat podle pravidel popsaných v sekci 2.2.4
    e) Aplikace umožní export/import
    f) Aplikace nabídne přepnutí lokalizace podle dodaných lokalizačních materiálů zadavatelem

g) Aplikace bude uchovávat a umožní odeslání hlášení stavu vývojáři
h) Aplikace bude mít mód prezentace, který skryje stavovou lištu a záhlaví
i) Ve všech chvílích práce s aplikací bude možné vidět její verzi
j) Aplikace bude neustále zobrazovat vybraného pacienta a počet jeho epizod a profylaxí

2. Pacienti
   a) Aplikace umožní evidovat pacienty
   b) Seznam pacientů je možné řadit
   c) Na základě vybrané nemoci se případně zobrazí pole pro typ nebo závažnosti
   d) Pole pro vedlejší diagnózu bude nabízet již zadané diagnózy
3. Epizody
   a) Aplikace umožní evidovat epizody pacientů
   b) Seznam epizod je možné řadit
   c) Uživatel může spravovat vlastní lokality krvácení
   d) Časy zhodnocení aplikací se automaticky změní podle času poslední aplikace
   e) Pole pro lék proti bolesti bude nabízet již zadané léky
   f) Pole pro důvod krvácení bude nabízet již zadané důvody
4. Profylaxe
   a) Aplikace umožní evidovat profylaxe pacientů
5. Sklad
   a) Aplikace umožňuje sledovat skladové zásoby léků vybraných pacientů podle přípravku
6. Grafy
   a) Aplikace umožní vykreslovat grafy
   b) Na ose $\mathcal{X}$. bude časové body podle období (den, měsíc, rok)
   c) Graf bude mít 2 osy $\mathcal{Y}$.
   d) Uživatel může definovat typ grafu pro každou osu $\mathcal{Y}$. zvlášť
   e) Uživatel může definovat filtr grafu pro každou osu $\mathcal{Y}$. zvlášť
   f) Pro typy grafů, které nemají absolutní hodnotu, na grafu zobrazíme průměr
   g) Další data budou dostupná mimo graf

2.2.2 Neunkční požadavky

7. Aplikace bude vytvořena pro .NET 4.6
8. Aplikace bude mít funkci automatické aktualizace
9. Veškerá data budou uložena na zařízení uživatele
10. Ověření zařízení pro import/export dat
11. Externí zařízení pro autentizaci uživatelů
12. Podobné řešení UI jako u první verze

2.2.3 Definice požadavků

Pro správný návrh aplikace, je potřeba se ujistit, že všem požadavkům na aplikaci bylo porozuměno. Někdy to nemusí být jasné z jedné věty, a proto je v následující sekci podrobně rozebráno a vysvětleno vše, co zadavatel požadoval

1a Uživatel se pro práci s aplikací musí přihlásit pod svým účtem

1b Uživatel může v jakoukoli chvíli vybrat uživatele, se kterým bude pracovat. Pokud uživatel pouze vyhledává a žádného pacienta neoznačí, jediný vliv zaznamená seznam pacientů, který bude příslušně vyfiltrován. Pokud bude pacient vybrán, mimo předchozího bude vyfiltrován seznam epizod a profylaxí podle vybraného uživatele

1c Kdykoliv uživatel opouští sekci, kde prováděl jakékoli změny na datech, bude dotázán, zda si přeje změny uložit nebo zahodit

1d Všechny data mají určeny sadu pravidel, jež potvrzují jejich validitu. Aplikace neumožní uložení dat, která by nesplňovala všechna tato pravidla popsaná v sekci 2.2.4

1e Z aplikace bude možné převádět data mezi vlastní pamětí a externím zařízením

1f Aplikace bude lokalizovatelná. Lokalizační soubory budou dodány zadavatelem před kompilací aktuální verze. Pokud žádné dodány nebudou, automaticky se použijí soubory z nižší verze

1g Všechna chybová hlášení za běhu aplikace jsou uložena na disk uživatele. Na základě vědomé akce budou odeslána vývojáři

1h Na hlavní liště se bude nacházet tlačítko pro přechod do režimu celé obrazovky. Ten skryje hlavní a stavovou lištu. Ukončit tento režim bude možné buď klávesou „Escape“, nebo najetím myši na horní okraj obrazovky a kliknutím na tlačítko "ukončení režimu celé obrazovky"

1i Uživateli bude zobrazena verze aplikace bez ohledu na to, kde se právě nachází

1j Uživateli bude zobrazen počet vybraných pacientů a jejich epizod a profylaxí. Počet pacientů odpovídá počtu výsledků vyhledávání. Počet epizod a profylaxí je buď následně roven počtu všech záznamů v databázi, pokud není žádný uživatel vybrán, jinak je to počet záznamů u vybraného pacienta

2a Bude možné provádět CRUD operace na pacientech

2b Seznam pacientů bude možné řadit podle několika atributů. Pro atribut závažnosti nemoci se použije abecední řazení, nikoliv řazení podle hodnoty

2c U Hemofilie evidujeme takzvanou závažnost nemoci. U Von Willebrandovy choroby potom podtyp. Pole pro výběr těchto parametrů se zobrazí pouze tehdy, pokud jsme označili, že jí daný pacient trpí

2d Pole pro vedlejší diagnózu bude přijímat textový vstup od uživatele a na základě doposud vloženého textu vyhledá a zobrazí nabídku z již existujících diagnóz

3a Bude možné provádět CRUD operace na epizodách

3b Seznam epizod bude možné řadit podle několika atributů. Pro atribut intenzity krvácení se použije abecední řazení, nikoliv řazení podle hodnoty

3c Mimo pevných seznamů lokalit krvácení typů klouby, svaly a podkoží si bude moci uživatel přidávat vlastní lokality krvácení

3d Když je změněna hodnota atributu začátku času krvácení, upraví se časy zhodnocení, aby odpovídaly rozložení

3e Pole pro léky proti bolesti bude přijímat textový vstup od uživatele a na základě doposud vloženého textu vyhledá a zobrazí nabídku z již existujících léků

3f Pole pro důvod krácení bude přijímat textový vstup od uživatele a na základě doposud vloženého textu vyhledá a zobrazí nabídku z již existujících důvodů

4a Bude možné provádět CRUD operace na profylaxích

5 Uživatel si bude moct nechat vypsat stavy zásob a spotřeby léku pacientů, kteří ho zajímají

6a Součástí aplikace bude modul pro vyhodnocení dostupných údajů na grafu

6b Uživatel si bude moct volit periodicitu počítaných statistik z roční, měsíční, denní. Podle toho se vytvoří osa $\mathcal{X}$, kde každý bod bude reprezentovat dané období. Pokud v daném období neexistují žádná data, bude použita hodnota „0“

6c Na grafu bude hlavní (levá) a vedlejší (pravá) osa $\mathcal{Y}$.

6d Datový filtr se nastavuje pro určitou osu, takže můžeme na každé ose vykreslovat grafy nad jinou množinou dat

6e Typ grafu se nastavuje pro určitou osu, takže můžeme na každé ose vykreslovat jinou statistiku. Seznam typů grafů naleznete v tabulce 2.1

6f Určité typy grafu, jako je třeba počet pacientů, mají za výsledek absolutní hodnotu. Pokud se jedná o takovýto typ, zobrazí se na grafu absolutní hodnota. Pokud se jedna o statistický typ, bude na grafu zobrazen průměr.

6g Pro obě dvě osy $\mathcal{Y}$ si bude možné zobrazit tabulkový seznam všech hodnot, kde pro statistické typy grafu bude možné kromě průměru nalézt také medián a směrodatnou odchylku

7 Aplikace bude vyžadovat k běhu nainstalovaný .NET 4.6

8 Po spuštění se aplikace připojí na server a zkontroluje, jestli je k dispozici aktualizace. Pokud ano, stáhne instalační balík, nainstaluje novou verzi a znovu se spustí

9 Jediná komunikace po internetu bude probíhat v případě odesílání chybového hlášení nebo aktualizaci aplikace. Žádná data uživatele nebudou posílána přes síť. Data budou zašifrována standardní šifrou podle výběru, která znemožní při získání datového souboru data volně přečíst

10 Komunikace se zařízeními pro import a export bude povolena pouze pro zařízení, jejichž sériové číslo bylo dodáno zadavatelem a bylo přidáno do seznamu povolených zařízení. Navíc na sobě musí mít soubor potvrzující jejich identitu, ten je pro všechna zařízení stejný

11 Identita uživatelů bude ověřena vůči externímu zařízení

12 Návrh UI bude mít stejné klíčové prvky jako u první verze

2.2.4 Validace

Uložená data musí být konzistentní a kvalitní. Kvalitou dat rozumíme následující. Zprvu se jedná o existenci všech dat potřebných k vyhodnocení celku.

Název	Popis
Počet pacientů	Počet pacientů, kteří si aplikovali preparát v daném období z jakéhokoli důvodu
Počet epizod	Počet epizod v daném období
Počet profylaxí	Počet profylaxí, v jejichž rámci byl podán preparát v daném období
Celková spotřeba	Množství léku spotřebovaného v daném období z jakéhokoli důvodu
Dávka na kilogram	Když si pacient aplikoval lék z jakéhokoli důvodu, jak velká byla dávka vzhledem k jeho váze
Dávka na kilogram v rámci epizod	Když si pacient aplikoval lék kvůli krvácivé epizodě, jak velká byla dávka vzhledem k jeho váze
Dávka na kilogram v rámci profylaxí	Když si pacient aplikoval lék kvůli profylaxi, jak velká byla dávka vzhledem k jeho váze
Čas krvácení	Rozdíl mezi koncem a začátkem krvácení

Tabulka 2.1: Seznam typů grafů

Mimo to se snažíme o to, aby data nebyla duplicitní, například dva názvy, kde jeden je s diakritikou „zubní zákrok“ a druhý bez „zubni zakrok“

První sada pravidel kladená na data je definovaná už v databázovém modelu, který je obsažen v příloze. Kromě samotného modelu byla definována pravidla následující

- Patient

  Weight musí být větší než „0“

  DiagnosisSeverity neprázdný v případě diagnózy, která má závažnost

  DiagnosisSubType neprázdný v případě diagnózy, která má podtyp

- IllnessRecord

  TreatmentEnd možno vyplnit pouze tehdy, pokud je léčba nemoci ukončena

- InhibitorRecord

  Date musí být menší, než je čas uložení do databáze

- PrescriptionRecord

Date musí být menší, než je čas uložení do databáze

Amount musí být větší než „0“

- WeightRecord

  Date musí být menší, než je čas uložení do databáze

  Weight musí být větší než „0“

- Episode

  Applications musí obsahovat alespoň jeden záznam, pokud není **HasNoApplication** pravda

  StartTime musí být menší, než je **EndTime**

  EndTime musí být menší, než je čas uložení do databáze

- EpisodeApplication

  Date musí být menší než, je čas uložení do databáze

  Amount musí být větší než „0“

- ProphylaxisApplication

  Date musí být menší než, je čas uložení do databáze

  Amount musí být větší než „0“

2.3 Případy užití

Na obrázku 2.2 můžete vidět případy užití. V aplikaci je pouze jeden aktér, kterým je uživatel aplikace.

2.3.1 Přihlásit

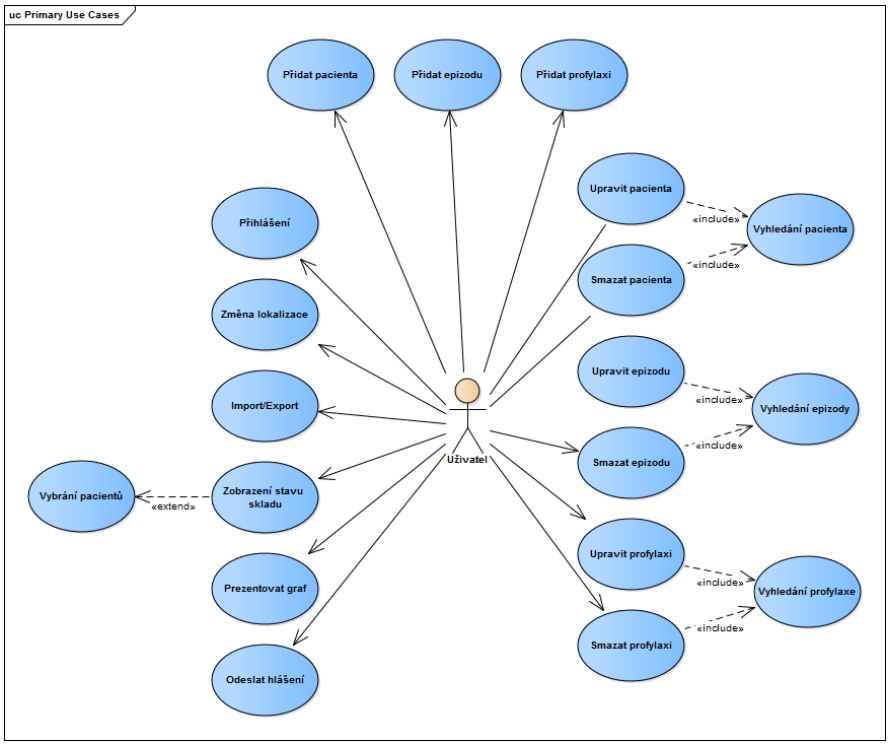
Přihlásí pacienta pro práci s aplikací

2.3.1.1 Hlavní scénář

1. Uživatel připojí externí zařízení
2. Uživatel vypíše svoje uživatelské jméno a heslo od externího zařízení
3. Systém přihlásí uživatele a přesměruje ho na záložku grafy

2.3.1.2 Chyby

1. Uživatel zadal neplatné údaje. Systém ho vyzve k opravě



Obrázek 2.2: Případy užití

2.3.2 Import/Export

Provede import nebo export dat mezi zařízení a externím USB diskem

2.3.2.1 Hlavní scénář

1. Uživatel připojí USB disk
2. Uživatel otevře nastavení aplikace
3. Uživatel klikne na import nebo export
4. Systém provede vybranou akci

2.3.2.2 Chyby

1. Není připojen autentizovaný USB disk. Systém uživatele upozorní.

2.3.3 Změnit lokalizaci

Změní jazyk aplikace

2.3.3.1 Hlavní scénář

1. Uživatel otevře nastavení aplikace
2. Z dostupné nabídky si vybere požadovaný jazyk
3. Systém načte texty pro vybraný jazyk

2.3.4 Zobrazit stav skladu

Zobrazí statistiky o lécích

2.3.4.1 Hlavní scénář

1. Uživatel přejde na záložku "sklad"
2. **Uživatel vybere skupinu pacientů** Pokud žádný pacient není vybrán, systém vybere všechny
3. Uživatel klikne na aktualizaci stavu
4. Systém zobrazí seznam léků, které byly vydání a informace o stavu

2.3.5 Odeslat hlášení

Odešle dostupná chybová hlášení vývojáři

2.3.5.1 Hlavní scénář

1. Uživatel otevře nastavení aplikace
2. Uživatel klikne na odeslání hlášení
3. Systém odešle hlášení

2.3.5.2 Chyby

1. Není možné vytvořit ZIP soubor pro odeslání. Systém uživatele informuje.
2. Není možné odeslat email. Systém uživatele informuje.

2.3.6 Prezentovat data

Zobrazí data na grafu a následně v detailu se systémem na celé obrazovce

2.3.6.1 Hlavní scénář

1. Uživatel přejde na záložku grafy
2. Uživatel přejde do režimu prezentace
3. Uživatel vybere periodicitu
4. Uživatel vybere typ grafu na hlavní ose $\mathcal{Y}$
5. **Uživatel nastaví filtry na hlavní ose $\mathcal{Y}$** Pokud filtry nejsou nastavené, pracuje se nad celou množinou dat.
6. Uživatel vybere typ grafu na vedlejší ose $\mathcal{Y}$
7. **Uživatel nastaví filtry na vedlejší ose $\mathcal{Y}$** Pokud filtry nejsou nastavené, pracuje se nad celou množinou dat.
8. Uživatel klikne na "vypočítat"
9. Systém vypočítá žádaná data a zobrazí je na grafu
10. Uživatel si zobrazí tabulkový detail dat

2.3.7 Přidat pacienta

Přidá nového pacienta do systému

2.3.7.1 Hlavní scénář

1. Uživatel přejde na záložku „Pacienti“
2. Uživatel klikne na tlačítko "přidat"
3. Systém uživatele přesměruje na stránku detailu pacienta
4. Uživatel vyplní data
5. Uživatel klikne na "uložit"
6. Systém uživatele informuje o uložení dat

2.3.7.2 Chyby

1. Některá vyplněná data nejsou validní. Systém uživatele informuje.

2.3.8 Upravit pacienta

Upraví osobní data pacienta

2.3.8.1 Hlavní scénář

1. Uživatel přejde na záložku „Pacienti“
2. Uživatel vybere pacienta ze seznamu
3. Uživatel klikne na "upravit"
4. Systém uživatele přesměruje na stránku detailu pacienta
5. Uživatel upraví data
6. Uživatel klikne na "uložit"
7. Systém uživatele informuje o uložení změn

2.3.8.2 Chyby

1. Nebyl vybrán pacient. Systém uživatele informuje.
2. Některá vyplněná data nejsou validní. Systém uživatele informuje.

2.3.9 Smazat pacienta

Smaže pacienta včetně všech jeho epizod a profylaxí

2.3.9.1 Hlavní scénář

1. Uživatel přejde na záložku „Pacienti“
2. Uživatel vybere pacienta ze seznamu
3. Uživatel klikne na "smazat"
4. Systém vyzve uživatele k potvrzení akce a vymaže data z databáze

2.3.9.2 Chyby

1. Nebyl vybrán pacient. Systém uživatele informuje.

2.3.10 Přidat epizodu

Přidá novou epizodu do systému

2.3.10.1 Hlavní scénář

1. Uživatel přejde na záložku „Epizody“
2. Uživatel klikne na tlačítko "přidat"
3. Systém uživatele přesměruje na stránku detailu epizody
4. Uživatel vyplní data
5. Uživatel z palety vybere pacienta, kterému epizodu přiřadit
6. Uživatel klikne na "uložit"
7. Systém uživatele informuje o uložení dat

2.3.10.2 Chyby

1. Některá vyplněná data nejsou validní. Systém uživatele informuje.
2. Nebyl vybrán pacient. Systém uživatele informuje.

2.3.11 Upravit epizodu

Upraví informace o krvácivé epizodě

2.3.11.1 Hlavní scénář

1. Uživatel přejde na záložku „Epizody“
2. Uživatel vybere epizodu ze seznamu
3. Uživatel klikne na "upravit"
4. Systém uživatele přesměruje na stránku detailu epizody
5. Uživatel upraví data
6. Uživatel klikne na "uložit"
7. Systém uživatele informuje o uložení změn

2.3.11.2 Chyby

1. Některá vyplněná data nejsou validní. Systém uživatele informuje.

2.3.12 Smazat epizodu

Smaže epizodu

2.3.12.1 Hlavní scénář

1. Uživatel přejde na záložku „Epizody“
2. Uživatel vybere epizodu ze seznamu
3. Uživatel klikne na "smazat"
4. Systém vyzve uživatele k potvrzení akce a vymaže data z databáze

2.3.12.2 Chyby

1. Nebyla vybrána epizoda. Systém uživatele informuje.

2.3.13 Přidat profylaxi

Přidá novou profylaxi do systému

2.3.13.1 Hlavní scénář

1. Uživatel přejde na záložku „Profylaxe“
2. Uživatel klikne na tlačítko "přidat"
3. Systém uživatele přesměruje na stránku detailu profylaxe
4. Uživatel vyplní data
5. Uživatel z palety vybere pacienta komu profylaxi přiřadit
6. Uživatel klikne na "uložit"
7. Systém uživatele informuje o uložení dat

2.3.13.2 Chyby

1. Některá vyplněná data nejsou validní. Systém uživatele informuje.
2. Nebyl vybrán pacient. Systém uživatele informuje.

2.3.14 Upravit profylaxi

Upraví informace o profylaxi

2.3.14.1 Hlavní scénář

1. Uživatel přejde na záložku „Profylaxe“
2. Uživatel vybere profylaxi ze seznamu
3. Uživatel klikne na "upravit"
4. Systém uživatele přesměruje na stránku detailu profylaxe
5. Uživatel upraví data
6. Uživatel klikne na "uložit"
7. Systém uživatele informuje o uložení změn

2.3.14.2 Chyby

1. Některá vyplněná data nejsou validní. Systém uživatele informuje.

2.3.15 Smazat profylaxi

Smaže profylaxi

2.3.15.1 Hlavní scénář

1. Uživatel přejde na záložku „Profylaxe“
2. Uživatel vybere profylaxi ze seznamu
3. Uživatel klikne na "smazat"
4. Systém vyzve uživatele k potvrzení akce a vymaže data z databáze

2.3.15.2 Chyby

1. Nebyla vybrána profylaxe. Systém uživatele informuje.

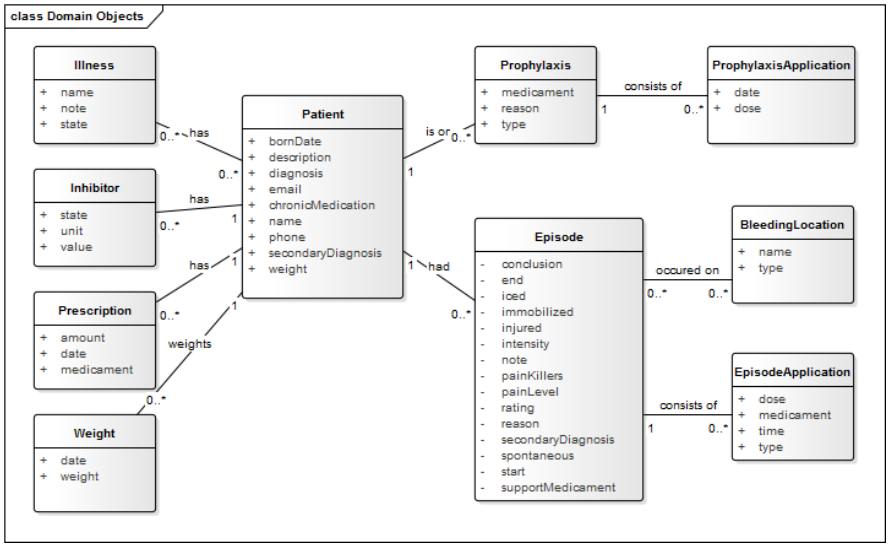
Pokrytí funkčních požadavků případy užití můžete vidět v tabulce 2.2

2.4 Doménový model

Po provedení kompletní analýzy byl sestaven doménový model. Ten můžete vidět na obrázku 2.3.

	Přihlásit	Přidat pacienta	Upravit pacienta	Smazat pacienta	Přidat epizodu	Upravit epizodu	Smazat epizodu	Přidat profylaxi	Upravit profylaxi	Smazat profylaxi	Změnit lokalizaci	Odeslat hlášení	Prezentovat data	Import/Export dat	Zobrazit stav skladu
1a	✓														
1b					✓			✓							
1c		✓	✓	✓	✓	✓	✓	✓	✓	✓					
1d		✓	✓		✓	✓		✓	✓						
1e														✓	
1f											✓				
1g												✓			
1h													✓		
1i	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1j		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2a		✓	✓	✓											
2b			✓	✓											✓
2c		✓	✓												
2d		✓	✓												
3a					✓	✓	✓								
3b						✓	✓								
3c					✓	✓									
3d					✓	✓									
3e					✓	✓									
3f					✓	✓									
4a								✓	✓	✓					
5a														✓	
6a													✓		
6b													✓		
6c													✓		
6d													✓		
6e													✓		
6f													✓		

Tabulka 2.2: Namapování funkčních požadavků a případů užití



Obrázek 2.3: Doménový model prostředí HeRgoT

Kapitola 3

Návrh systému

Po sebrání funkčních a nefunkčních požadavků, zmapování problémové domény a vymodelování případu užití přichází na řadu návrh řešení nefunkčních požadavků a datových struktur v aplikaci.

3.1 Autentizace

Hlavní požadavky na aplikaci, které mají vliv na autentizaci uživatele, jsou běh aplikace bez nutnosti připojení k internetu a existence externího zařízení, které bude nosičem identity uživatele.

3.1.1 RSA

RSA je asymetrická šifra, která je postavena na existence privátního a veřejného klíče. Data která byla jedním klíčem zašifrována, mohou být tím druhým dešifrována. Na základě toho víme, že pokud jsme schopni veřejným klíčem data dešifrovat, je původ dat ověřen.

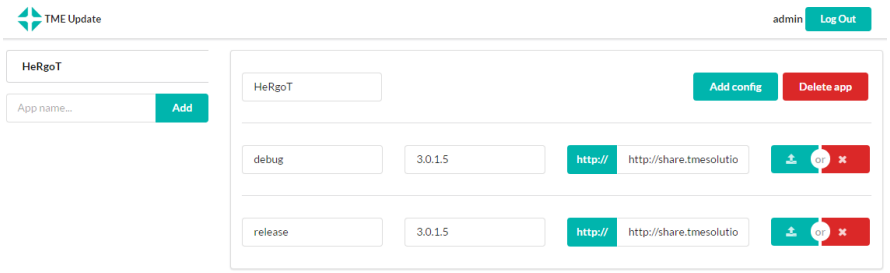
3.1.2 eToken

eToken je autentizační USB modul, který slouží jako nosič pro digitální certifikáty. Tyto certifikáty staví na šifře RSA. Token je možné využít k zašifrování dat privátním klíčem.

3.1.3 Použití certifikátů k autentizaci

Každému uživateli aplikace bude vydán token, na kterém je pro něj vygenerovaný certifikát. Následná autentizace probíhá tak, že aplikace bude uchovávat seznam uživatelů a jejich veřejných klíčů v následujícím formátu

username uživatelské jméno přidělené uživateli



Obrázek 3.1: Webový nástroj pro správu aktualizací

pubkey věřejný klíč uživatele ve formátu XML, z něhož může být vytvořena instance **RSACryptoServiceProvider** pro dešifrování

Při pokusu o přihlášení systém najde uživatele s příslušným uživatelským jménem, následně zašifruje blok dat použitím veřejného klíče, který má uložený. Zašifrovaná data systém odešle na dešifrování, a jestli se vrácená data rovnají těm před začátkem procesu, je uživatel úspěšně přihlášen. Pokud jakýkoli z těchto kroků selže, nebylo možné uživatele ověřit, a stav systému se nezmění.

3.2 Aktualizace

Z důvodu budoucích změn a oprav bude potřeba systém aktualizovat. Manuální aktualizace jsou velice nepraktické a časově náročné. Z toho důvodu se bude systém umět sám aktualizovat.

3.2.1 Správa aktualizací

Pro správu aktualizací byl vytvořen jednoduchý webový systém, jehož jsem autorem. Webový system je napsán ve frameworku **Meteor**. Systém umožňuje evidovat aplikace, které mohou mít více aktualizačních profilů pro potřeby testování, ostrého nasazení a podobně. Každý tento profil obsahuje verzi a instalační soubor k ní patřící. Ukázku systému můžete vidět na obrázku 3.1.

3.2.2 API

Pro aplikaci bude dostupné REST API, které umožní kontrolu dostupných aplikací. Kontrola aktualizací bude dostupná na adrese

`http://update.hergot.cz/app/{_app}/profile/{_profile}?version={_version}`

_app název aplikace

_profile aktualizační profil

__version aktuální verze aplikace

V případě úspěšného nalezení aplikace a profilu bude návratový kód **200 OK** a tělo bude obsahovat JSON s následujícími atributy

version dostupná verze na serveru

url adresa, ze které je možné stáhnout instlační soubor

update pokud byl zaslán nepovinný atribut **version** a liší se od verze dostupné na serveru, bude hodnota **true** jinak **false**

3.2.3 Průběh aktualizace

Při spuštění aplikace odešle dotaz na server. Pokud je na serveru dostupná nová verze, systém ji stáhne a spustí na pozadí. Následně se vypne, aby aktualizace mohla nahradit soubory. Vzhledem k velikosti instalačního balíku okolo 10 MB bylo rozhodnuto, že se aktualizace nebude modularizovat a k dispozici bude celý instalační balík. Jak můžete vidět, aktualizace jsou povinné a provádí se před přihlášením do systému

3.3 Zálohy

Aplikace bude mít seznam seriálových čísel autorizovaných USB disků. Mimo to musí na sobě disk mít speciální soubor, jehož obsah je systémem kontrolován.

3.3.1 Export

Databázový soubor bude překopírován z klientské stanici na připojený USB disk. Jméno zálohy bude obsahovat aktuální datum a čas.

3.3.2 Import

Systém najde nejnovější soubor zálohy na připojeném USB disku. Tento soubor přesuna na klientské zařízení a přenačte data v aplikaci.

3.4 Bezpečnost dat

Aby útočník nemohl obejít přihlašovaní, je potřeba zabezpečit data. Vzhledem k tomu, že všechna data jsou uložena u uživatele, je nutné datový soubor zašifrovat. To ovlivnilo i výběr databáze. Více v sekci 4.3

3.5 Třídní rozdělení

Mimo samotné třídy reprezentující data, o kterých si můžete přečíst víc v sekci 3.6, jsou důležité následující.

AutoCompleteBox dědí od třídy **RadAutoCompleteBox** a umožňuje uživateli dvojklikem zobrazit kompletní seznam nabídky

BindableTextBloxk dědí od třídy **TextBlock** a umožňuje pomocí funkce DataBinding generovat seznam textu na základě navázaného seznamu objektů typu **Inline**, které jsou základním stavebním prvkem bloku

FilterableComboBox dědí od třídy **ComboBox** a upravuje systém filtrování nabízených položek. Implementace podporuje nastavení atributu *DisplayMemberPath*, podle kterého se určí zobrazení i filtrovaní dat

MultiSelectDataGrid je rozšířením klasické **DataGrid** a umožňuje vybrání více záznamů. Podporuje funkci DataBinding

SortableTrackedBindingList<T>[12] je úpravou **TrackedBindingList<T>**, která umožní řazení záznamů, která jsou cílem funkce DataBinding

TitledFlipView je odvozeno od **FlipView** a přináší jednoduchou možnost nastavení vlastního nadpisu pro obsah tohoto kontejneru. Implementace využívá takzvané „Attached Properties“[13]. Příklad jejich použití můžete vidět na ukázce 1

LinqExtension je statická třída, která rozšiřuje dostupné funkce na seznamech implementujících *IEnumerable<T>* bezpečné vyhledání maxima a minima. Tyto funkce jsou použity při hledání minimálních dat během vypočítávání statistik

Comparer je statická třída, která má metody pro porovnání dvou prvků a vracení minima, respektive maxima

Filter je třída, která uchovává nastavení filtrů pro jednotlivé osy

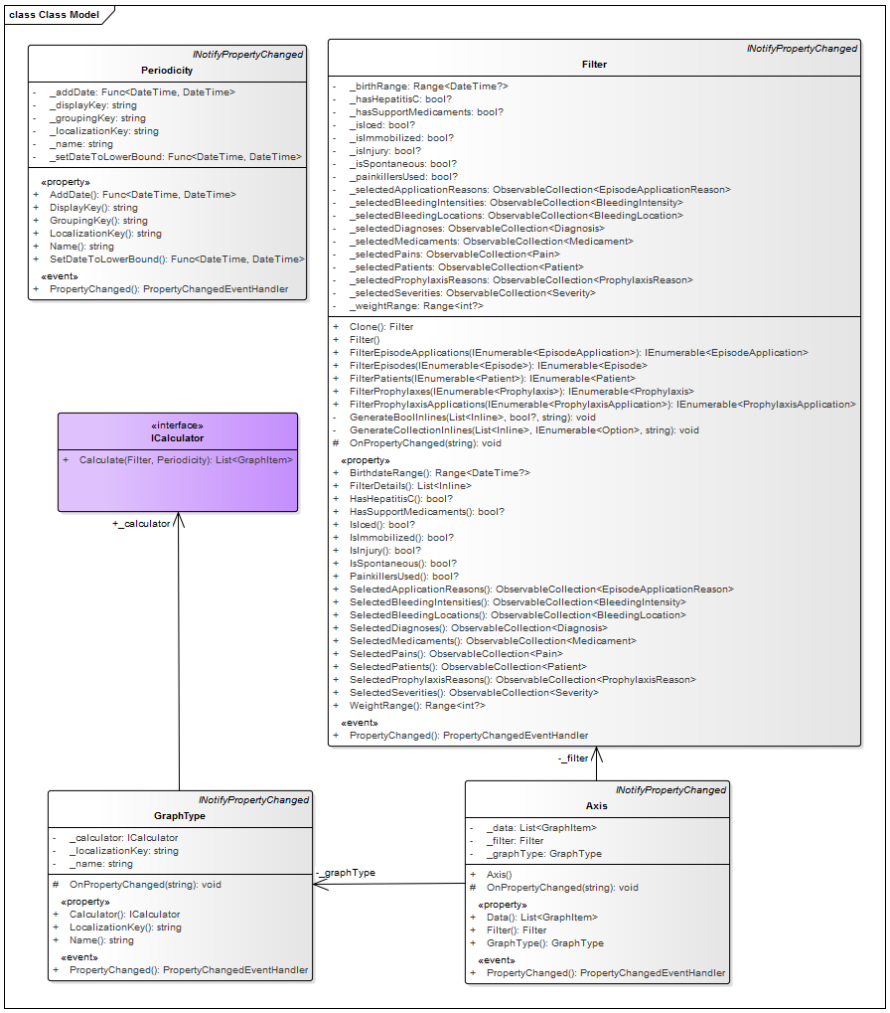
Periodicity je třída, která nastavuje, v jakých časových intervalech budou data zpracovávána

GraphType je třída určující, o který typ grafu se jedná

Axis reprezentuje osu grafu a k ní vážící se typ grafu a filtr

ICalculator je rozhraní, které umožňuje nastavení správného typu grafu

Ukázku statistické části třídního modelu můžete vidět na obrázku 3.2. Kompletní třídní model je součástí přílohy této práce.



Obrázek 3.2: Třídní diagram statistiky

3.6 Databázový model

Vzhledem k požadavku na lokalizovatelnost aplikace, bylo rozhodnuto o ukládání takzvaných lokalizačních klíčů do databáze. Tyto klíče jsou využity v lokalizačních souborech, kde je k nim přiřazena hodnota pro daný jazyk. Při načtení dat či změně lokalizace jsou hodnoty načteny. Každá tabulka v databázi reprezentuje právě jednu třídu v aplikaci. Všechny třídy, jejichž obrazem jsou tabulky obsahující zmíněný lokalizační klíč, mají také atribut pro uchování následně načtené hodnoty z lokalizačních souborů. Data uložená v databázi mají následující strukturu

BleedingIntensity slouží pro uchování nabídky intenzity krvácení v rámci krvácivých epizod

```
<controls:TitledFlipView>
        <telerik:RadCartesianChart
                controls:TitledFlipView.Title="Graf"/>
        <DataGrid
                controls:TitledFlipView.Title="Hlavní osa"/>
        <DataGrid
                controls:TitledFlipView.Title="Vedlejší osa"/>
<controls:TitledFlipView/>
```

Zdrojový kód 1: Zjednodušená ukázka použití „Attached Properties“

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

BleedingLocation slouží pro uchování nabídky místa krvácení v rámci krvácivých epizod

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

CustomName hodnota vlastního textu, jedná-li se o položku spravovanou uživatelem

Typ typ dané lokality. Jedná se o kloub, podkoží, sval a nebo vlastní položku

Diagnosis slouží pro uchování nabídky diagnózy pacienta

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

HasSeverity určuje, jestli se daná diagnóza dělí podle závažnosti

HasSubType určuje, jestli má daná diagnóza další podtypy

DiagnosisSeverity slouží pro uchování nabídky závažností diagnóz

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

DiagnosisSubType slouží pro uchování nabídky podtypů diagnóz

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

Episode reprezentuje konkrétní krvácivou epizodu

ID primární klíč záznamu

StartTime čas začátku krácení

EndTime čas konce krácení

IsIced jestli bylo místo krvácení ledováno

IsImmobilized jestli krvácení způsobilo nehybnost

IsSpontaneous jestli bylo krvácení spontánní, tedy bez jakéhokoli podnětu od zvýšené aktivity po úraz

IsInjury jestli krvácení začalo následkem úrazu

Note poznámka lékaře či pacienta k epizodě

Painlevel úroveň bolesti v místě krvácení

PainKillers jaké léky na zmírnění bolesti byly použity

PainKillersUsed jestli byly použity léky na zmírnění bolesti

Reason důvod začátku krvácení, jedná se o volný text

HasNoApplication jestli nebyl podán žádný preparát k zástavě krvácení

SupportMedicament podpůrné léky pro zmírnění následků krvácení

IntensityID cizí klíč odkazující na položku intenzity krvácení

ConclusionID cizí klíč odkazující na zhodnocení účinnosti léčby

PatientID cizí klíč odkazující na pacienta

EpisodeApplication reprezentuje podání preparátu v rámci epizody

ID primární klíč záznamu

Amount množství podaného léku

Time čas podání léku

MedicamentID cizí klíč odkazující na lék, který byl podán

ReasonID důvod k podání léku. Například zajišťovací dávka, která je podána i po tom, co je krvácení zastaveno

EpisodeID cizí klíč odkazující na epizodu, pod kterou aplikace spadá

EpisodeApplicationReason slouží pro uchování nabídky důvodu podání léku

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

IsInitial jestli daný důvod může být jako první na časové ose aplikací léku v rámci epizody

IsUnique jestli daný důvod může být použit pouze jednou v rámci konkretní epizody

EpisodeApplicationReason vazební tabulka pro redukci vztahu M:N mezi epizodou a místy krvácení

EpisodeID cizí klíč záznamu epizody

BleedingLocationID cizí klíč odkazující na lokalitu krvácení

EpisodeNoApplication reprezentuje nepodání léku

ID primární klíč záznamu

Note vlastní poznámka lékaře či pacienta ohledně okolností, které vedly k nepodání léku

ReasonID cizí klíč odkazující na důvod pro nepodání léku

EpisodeNoApplicationReason slouží pro uchování nabídky důvodu k nepodání žádného léku

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

EpisodeRating reprezentuje jedno zhodnocení krvácivé epizody, těch je v rámci epizody více

ID primární klíč záznamu

PainLevel úroveň bolesti

IsNotBleeding jestli krvácení právě neprobíhá

IsNotRated jestli hodnocení nebylo vyplněno

Delay doporučený odstup hodnocení od začátku krvácení

Time opravdový čas zhodnocení

EpisodeID cizí klíč odkazující na hodnocenou epizodu

EpisodeRatingConclusion slouží pro uchování nabídky zhodnocení léčby krvácivé epizody

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

Illness slouží pro uchování nabídky nemocí

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

HasClarification jestli má daná nemoc upřesnění diagnózy

IllnessRecord realizuje vazbu mezi nemocí a pacientem. Mimo to určuje stav léčby

- **ID** primární klíč záznamu
- **Note** poznámka k nemoci
- **TreatmentEnd** čas konce léčby
- **StateID** cizí klíč odkazující na stav léčby. Ten zároveň určuje, jestli záznam může evidovat čas konce léčby
- **IllnessID** cizí klíč odkazující na nemoc
- **PatientID** cizí klíč odkazující na pacienta

IllnessState slouží pro uchování nabídky stavů léčby nemoci

- **ID** primární klíč záznamu
- **LocalizationKey** lokalizační klíč dané položky
- **AllowsTreatmentEnd** jestli daný stav označuje ukončení léčby (bez ohledu na úspěch)

InhibitorRecord reprezentuje stav hladiny inhibitoru u pacienta

- **ID** primární klíč záznamu
- **Date** datum a čas provedeného měření
- **Value** vlastní hladina inhibitoru
- **UnitID** cizí klíč odkazující na jednotku měřené hladiny
- **StateID** cizí klíč odkazující na stav inhibitoru
- **PatientID** cizí klíč odkazující na pacienta

InhibitorState slouží pro uchování nabídky stavů inhibitoru

- **ID** primární klíč záznamu
- **LocalizationKey** lokalizační klíč dané položky
- **IsInital** jestli daný stav může být jako první na časové vývoje inhibitoru u pacienta

InhibitorUnit slouží pro uchování nabídky jednotek měření inhibitoru

- **ID** primární klíč záznamu
- **LocalizationKey** lokalizační klíč dané položky

Medicament slouží pro uchování nabídky léků k podání

- **ID** primární klíč záznamu
- **LocalizationKey** lokalizační klíč dané položky

Patient reprezentuje konkrétního pacienta v systému

- **ID** primární klíč záznamu
- **FirstName** křestní jméno
- **LastName** příjmení
- **Email** emailová adresa
- **Phone** telefonní číslo
- **Description** upřesnění diagnózy pacienta
- **ChronicMedication** chronická medikace pacienta spojená s jinými nemocemi
- **BorDate** datum narození
- **Weight** váha pacienta v době zadání do systému
- **SecondaryDiagnosis** sekundární diagnóza nemoci
- **DiagnosisID** cizí klíč odkazující na diagnózu
- **DiangosisSubTypeID** cizí klíč odkazující na podtyp diagnózy
- **DiagnosisSeverityID** cizí klíč odkazující na závažnost diagnózy

PrescriptionRecord reprezentuje změny stavu skladu, které mohou být od vydání léku po odpis

- **ID** primární klíč záznamu
- **Date** datum a čas provedení změny
- **Amount** množství léku
- **MedicamentID** cizí klíč odkazující na lék
- **TypeID** cizí klíč odkazující na typ změny stavu, tedy o jakou akci se jedná. Ta samotná akce má přiřazeno, zda je vliv na stav pozitivní nebo negativní
- **PatientID** cizí klíč odkazující na pacienta

PrescriptionType slouží pro uchování nabídky typů změn skladu

- **ID** primární klíč záznamu
- **LocalizationKey** lokalizační klíč dané položky
- **Influence** určuje následný vliv na výpočet statistik
- **HasReason** jestli změna stavu skladu s tímto typem má pole pro důvod

Prophylaxis reprezentuje konkrétní instanci profylaxe pacienta

- **ID** primární klíč záznamu
- **Ended** jestli byla profylaxe ukončena

MedicamentID cizí klíč odkazující na lék

TypeID cizí klíč odkazující na typ profylaxe

ReasonID cizí klíč odkazující na důvod profylaxe

PatientID cizí klíč odkazující na pacienta

ProphylaxisApplication reprezentuje profylaktické podání v rámci profylaxe

ID primární klíč záznamu

Date datum a čas podání léku

Amount množství podaného léku

MedicamentID cizí klíč odkazující na lék

ReasonID cizí klíč odkazující na důvod podání léku

ProphylaxisID cizí klíč odkazující na pacienta

ProphylaxisReason slouží pro uchování nabídky důvodů profylaxe

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

ProphylaxisType slouží pro uchování nabídky typů profylaxí

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

SupportMedicament slouží pro uchování nabídky podpůrných léků

ID primární klíč záznamu

LocalizationKey lokalizační klíč dané položky

WeightRecord reprezentuje profylaktické podání v rámci profylaxe

ID primární klíč záznamu

Date datum a čas měření

Weight váha pacienta

PatientID cizí klíč odkazující na pacienta

sqlite_sequence je systémová tabulka, která uchovává stav sekvencí primárních klíčů ostatních tabulek

name název tabulky, ke které se záznam váže

seq hodnota posledního záznamu v dané tabulce

Databázový model je součástí přílohy této práce.

Kapitola 4

Technologie

Tato kapitola shrne, jaké technologie byly použity při vývoji aplikace a z jakého důvodu byly vybrány. Hlavním cílem je splnění všech požadavků. Mimo to je kladen důraz na rychlé dokončení projektu. Vzhledem k tomu, že se jedná o firemní projekt, který má rozpočet, bylo možné brát v potaz komerční řešení.

4.1 WPF

WPF je moderní UI framework pro tvorbu aplikací běžících na platformě .NET. Mezi jeho hlavní přednosti patří vykreslování ve vektorové grafice, což umožňuje škálování aplikace bez následků na kvalitu obsahu. To ocení především lidé využívající funkci DPI škálování představeného se systémem Windows Vista. Dalším faktorem pro výběr bylo oddělení uživatelského rozhraní a samotného kódu, který pohání aplikaci. Uživatelské rozhraní je realizováno v jazyce XAML, samotný kód je následně v C#. Microsoft představil spolu s WPF řadu API, která se starají o umožnění práce s vizuálními objekty právě.

4.1.1 Návrhový vzor

MVVM, neboli Model-View-ViewModel je návrhový vzor představený přímo společností Microsoft. Byl postaven speciálně pro potřeby WPF. Sestává se ze tří komponent. Existuje rozšíření tohoto vzoru, takzvané MVCVM, které přináší zapojení složky „Controller“. U větších projektů, se málo kdy setkáme s čistým MVVM. Jeho nedostatkem je problematické zpracování služeb, dostupných během celého běhu aplikace. Toto bylo vyřešeno v aplikaci tak, že všechny služby jsou dostupné přes rozhraní statických tříd. Jednou nevýhodou tohoto přístupu je to, že by bylo velice náročné uživateli nabídnou možnost výběru. Například databázového úložiště, protože vše leží na konkrétní implementaci a není možné využít společného rozhraní různých tříd.

Model je datová struktura, které slouží jako úložiště dat. Mezi zástupce modelu, patří všechny databázové struktury

View je grafická šablona, jejíž jedinou činností je vykreslování dat

ViewModel plní funkci správce **View**. Nabízí všechny potřebné objekty, pro správnou funkci View a reaguje na jeho podněty

Pro představu, můžete vidět jednoduchý přihlašovací formulář a jeho implementaci v čistém MVVM na ukázce 2. Více o rozhraní **ICommand** v kapitole 5.4

4.2 MahApps

MahApps je UI framework pro WPF. Nabízí sadu vzhledů pro již existující ovládací prvky a zároveň přináší prvky zcela nové. Tento framework klade velký důraz na dodržování principů MVVM. Jedná se o open-source projekt. Hlavním záměrem pro vznik tohoto frameworku bylo rozšíření WPF o Metro styl [14].

4.3 SQLcipher

SQLcipher je rozšíření SQLite, které poskytuje transparentní 256-bitové AES šifrování databázových souborů [15]. Firma Zenetic distribuuje ADO.NET providery, které slouží jako rozhraní pro práci s databází. Ty naprosto korespondují s verzí SQLite co se týče rozhraní a liší se pouze v implementaci. Díky tomu je možné použít SQLCipher tam, kde jsou očekávány knihovny SQLite. Neocenitelnou výhodou je to, že nejsme nijak omezeni v použití ORM nástroje.

4.4 Telerik

Je společnost zabývající se tvorbou UI komponent a vývojových nástrojů [16]. V aplikaci byly využity následující

RadCartesianChart pro vykreslování grafů. Teto grafový nástroj splňoval požadavky na škálování a selekci dat. Dále podporuje dvě osy $\mathcal{Y}$. Celá grafová komponenta je navržená s důrazem na DataBinding

RadDateTimePicker pro zadání data a času. Součástí WPF ani MahApps bouhužel není jedna komponenta, která by reprezentoval čas i datum. Vzhledem k tomu, že komponenty od Teleriku dodržují konvence WPF, bylo možné docílit jednoduchého zařazení této komponenty do systému. Jediné co se provedlo byla úpráva grafické XAML šablony, kde došlo k nahrazení barev. Struktura i funkce komponenty zůstaly beze změny

```
class Command : ICommand {
    public Predicate<object> CanExecuteDelegate { get; set; }
    public Action<object> ExecuteDelegate { get; set; }

    public bool CanExecute(object parameter) {
        if( CanExecuteDelegate != null )
            return CanExecuteDelegate(parameter);
        return true;
    }

    public event EventHandler CanExecuteChanged {
        add { CommandManager.RequerySuggested += value; }
        remove { CommandManager.RequerySuggested -= value; }
    }

    public void Execute(object parameter) {
        if( ExecuteDelegate != null )
            ExecuteDelegate(parameter);
    }
}
class LoginVM{
    public string Username {get;set}
    public string Password {get;set;}
    public ICommand SingIn {get;} = new Command{
        Execute = x => {
            //pokus o přihlášení
        };

}

<StackPanel>
    <StackPanel.DataContext>
        <LoginVM/>
    </StackPanel.DataContext>

    <TextBox Text="{Binding Username}"/>
    <TextBox Text="{Binding Password}"/>
    <Button Command="{Binding SignIn}"/>
</StackPanel>
```

Zdrojový kód 2: Ukázka MVVM na přihlašovacím formuláři

Kapitola 5

Implementace

V této kapitole jsou shrnuty hlavní části implementace aplikace. Začneme představením fundamentálních principů OOP v jazyce C#. Dále si rozebereme zajímavé přístupy, které WPF spolu s MVVM přináší.

5.1 Property aneb atribut

Jak víme, v modelu OOP má každá třída své atributy. C# rozšiřuje základní třídní proměnné, které známe z mnoha programovacích jazyků, a přináší takzvané „Properties“. Mnoho principů WPF je postaveno právě na jejich existenci. Hlavní rozdíly mezi proměnnými a „Properties“ je následující

- Property
  - umožňuje vlastní implementaci
  - možnost povolit či zamezit přístup zvlášť pro přiřazení a čtení
  - podpora funkce DataBinding
- Proměnná
  - základní struktura pro uchování dat
  - využívá pouze operátoru přiřazení pro daný datový typ

U „Properties“ můžeme použít vlastnost automatické deklarace a ta se postará o vytvoření proměnné, která bude uchovávat hodnotu. Případně můžeme implementovat přiřazení a čtení sami, což většinou vyžaduje vytvoření proměnné, která bude uchovávat danou hodnotu. To není potřeba v případě, jedná-li se o „Property“, která pracuje nad cizí množinou dat. Na ukázce 3 můžete vidět použití všech výše zmíněných přístupů.

```
1  class User {
2      private string _username;
3      public string Username{
4          get{ return _username; }
5          set{
6              if(value.Length >= 5)
7                  _username = value;
8          }
9      }
10     public string Index {get; private set;}
11     public string FirstName {get;set;}
12     public string LastName {get;set;}
13     public string FullName {
14         get{
15             return FirstName + " " + LastName;
16         }
17     }
18
19 }
```

Zdrojový kód 3: Použití properties a proměnných

5.2 DataBinding a INotifyPropertyChanged

Jednou z úžasných schopností WPF je vytvoření spojení mezi komponentou a datovou strukturou, takzvaný DataBinding. Všechny objekty, které chtějí informovat o změně, musí implementovat rozhraní

INotifyPropertyChange. Toto rozhraní definuje událost s názvem *PropertyChanged*.

Kdykoli použijeme dostupné rozhraní pro DataBinding, které většinou definujeme v XAML, vytvoří se vazební objekt. Ten ověří, jestli zdroj notifikací implementuje **INotifyPropertyChanged**. Pokud ano, přihlásí se k odběru události *PropertyChanged*. Kdykoli je tato událost zavolána, provede se kontrola, o jaký atribut se jedná, a případně notifikuje odběratele o změně. Pro DataBinding je vyžadován objekt, který dědí od třídy **DependencyObject**. Nebudu zabíhat dále do detailu, jen řeknu, že všechny UI komponenty od něj dědí.

Na ukázce 4 můžete vidět, jakým způsobem dochází k oznámení o změně. Událost má přiřazena prázdný delegát, aby nedocházelo k výjimkám za běhu aplikace při pokusu o notifikaci. Bohužel není možné použít vlastnost notifikací spolu s automatickými *Properties*, ačkoli je to vysoce požadovaná vlastnost.

```
1  class Patient : INotifyPropertyChanged {
2
3      public event PropertyChangedEventHandler PropertyChanged = delegate { };
4
5      private string _username;
6      public string Username{
7          get{ return _username; }
8          set{
9              if(_username = value)
10                 return;
11             _username = value;
12             PropertyChanged(this, new PropertyChangedEventArgs("Username"));
13         }
14     }
15 }
```

Zdrojový kód 4: Provázání datové struktury a uživatelského rozhraní

Obrázek 5.1: Příklad informování uživatele o chybě v datech

5.3 Validace dat s IDataErrorInfo

Validace dat je esenciálním prvkem všech aplikací. Slouží pro kontrolu a dává uživateli zpětnou vazbu o validitě zadaných dat. K dosažení těchto požadavků se ve WPF používá rozhraní **IDataErrorInfo**. Toto rozhraní přidává třídě indexer, na než je možné se dotazovat pomocí názvu property, kterou chcete validovat. Druhým přídavkem je property *Error*, která slouží k validaci objektu jako celku.

DataBinding je navržen s tímto rozhraním na mysli. Při navazování je možné specifikovat, že požadujeme validovat navázaný atribut. Pokud dotázaný indexer vrátí neprázdný řetězec, bude komponenta notifikována o existenci chyby, pro kterou vykreslí vizuální stav a předá uživateli informaci o chybě. Ukázku vizuálního stavu můžete vidět na obrázku 5.1.

Na ukázce 5 můžete vidět definici třídy, která implementuje **INotifyPropertyChanged** a **IDataErrorInfo**. Pro zkrácení ukázky je vynechána implementace properties *StartTime* a *EndTime*. Tu můžete vidět v sekci 5.2. Dále tam můžete vidět napojení na uživatelské rozhraní jak je popsáno výše.

5.4 ICommand - jiný přístup k událostem

Obecně v aplikacích potřebujeme registrovat události, které se odehrávají na uživatelském rozhraní, a následně vykonat kód, který se k tomu váže. WPF i zde upravuje návrh a zavádí příkazy. Příkazem je jakýkoliv objekt implementující **ICommand**. Toto rozhraní přidává následující

CanExecute jestli je možné příkaz právě spustit. Používá se pro podmínění spuštění, popřípadě synchronizaci

CanExecuteChanged událost oznamující změnu proveditelnosti

Execute akce, která by měla být vykonána. Jedná se o funkce s návratovým typem *void* a přebírající vstupní parametr typu *object*

Hlavní výhodou příkazů je jejich navázání na uživatelské rozhraní. Komponenty, se kterými je komunikováno, mohou mít navázané příkazy pomocí funkce DataBinding. Typickým příkladem je stisk tlačítka. Příkaz je samozřejmě možné vykonat z kódu. Příklad implementace příkazu můžete vidět na ukázce 2.

```
<StackPanel>
    <t:TimePicker SelectedValue=
        "{Binding StartTime, ValidatesOnDataErrors=True,NotifyOnValidationError=True}"/>
    <t:TimePicker SelectedValue=
        "{Binding EndTime, ValidatesOnDataErrors=True,NotifyOnValidationError=True}"/>
</StackPanel>

class Episode : IDataErrorInfo, INotifyPropertyChanged {

    public string Error {
        get {
                PropertyInfo[] properties = GetType().GetProperties();
                foreach(PropertyInfo property in properties) {
                    if(!string.IsNullOrEmpty(this[property.Name]))
                        return this[property.Name];
                }
                if(EndTime < StartTime)
                    return "Začátek krvácení musí být před koncem";
                return string.Empty;
        }
    }

    public string this[string columnName] {
        get {
            switch(columnName) {
                case "StartTime":
                    if( StartTime > DateTime.Now )
                        return "Čas začátku krvácení musí být v minulosti";
                    return string.Empty;
                case "EndTime":
                    if( EndTime > DateTime.Now )
                        return "Čas konce krvácení musí být v minulosti";
                    return string.Empty;
                default:
                    throw new NotImplementedException();
            }
        }
    }
}
```

Zdrojový kód 5: Validace dat s použitím IDataErrorInfo

Kapitola 6

Testy

V této sekci se zaměříme na testy provedené v rámci aplikace. Pro aplikaci byla napsána sada jednotkových testů pokrývajících statistický modul.

6.1 Jednotkové testy

Jednotkové testy jsou nedílnou součástí vývoje aplikace. Umožňují nám opakovaně vykonávat testy, při každé úpravě systému. To má za následek ušetření nákladů na testování. Jednotkové testy však nejsou všemohoucí. Nenahrazují ruční testování aplikace, které rozšiřuje ověření systému. Hlavním účelem tvorby jednotkových testů je příprava sady pravidel, které kód musí splňovat k tomu, aby byl předán dále.

V aplikaci byl použit testovací framework NUnit. Jednalo se o převedení velice populárního frameworku pro jednotkové testování v Javě, JUnit, ale s poslední verzí byl NUnit od základu přepsán [17]. Základní obsah testovacích tříd je následující

SetUp slouží k nastavení počátečního stavu pro sadu testů, typicky inicializace databáze a podobně

TearDown slouží k uklizení používaných zdrojů po vykonání sady testů, typicky odpojení od databáze a podobně

Test jedná se o jednu testovací metodu. Testovací třída obvykle obsahuje více testovacích metod

Na ukázce 6 můžete vidět implementaci jednoduché testovací třídy

6.2 Manuální testování

Vzhledem k mezerám jednotkového testování je potřeba testovat aplikaci i manuálně. Tohoto testování bylo v rámci projektu HeRgoT vytvořením tes-

```
class Statistics {

    public ObservableCollection<EpisodeApplicationReasons> EpisodeApplicationReasons { get; set; }

    [SetUp]
    public void Initialize() {
        DatabaseService.Initialize();
        EpisodeApplicationReasons = DatabaseService.EpisodeApplicationReasons();
    }
    [TearDown]
    public Dispose(){
        DatabaseService.Dispose();
    }
    [Test]
    public void NewBleedingExists() {
        Assert.AreEqual(true, EpisodeApplicationReasons.Contains(
                                  Data.EpisodeApplicationReasons.NewBleeding
                              )
                        );
    }
}
```

Zdrojový kód 6: Použití NUnit pro jednotkové testy

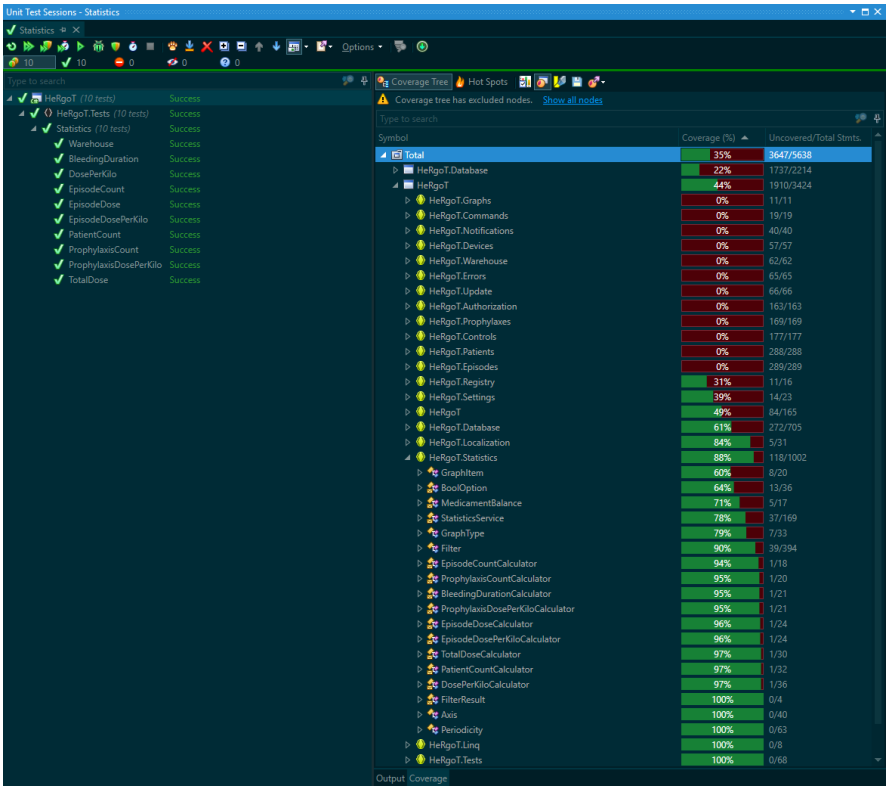
tovacích případů. „Testovací případ, často se využívá i anglický výraz „test case“, popisuje konkrétní akce prováděné s určitou softwarovou komponentou a jejich očekávané výsledky“ [18].

Testovací případy jsou předány manuálním testerům. Ty následně procházejí příklady krok po kroku a validují stav. Naše testovací případy se skládají především z vyplňování filtrů a následné kontroly vykreslených hodnot.

6.3 Prokrytí kódu testy

Pokrytí kódu obsahuje vše týkající se výpočetní logiky. Kód, který nebyl pokryt testy, se váže k uživatelskému rozhraní a k jeho otestování bylo využito manuálního testování.

Na obrázku 6.1 můžete vidět mapu pokrytí kódu testy. Ta je vyhodnocena jako počet řádku kódu, které byly dosaženy během vykonávání testů. Tento způsob vyhodnocení nám přináší například přehled o tom, které stavy objektu a jaké aplikační procesy jsme ještě neotestovali.



Obrázek 6.1: Mapa pokrytí kódu testy

Kapitola 7

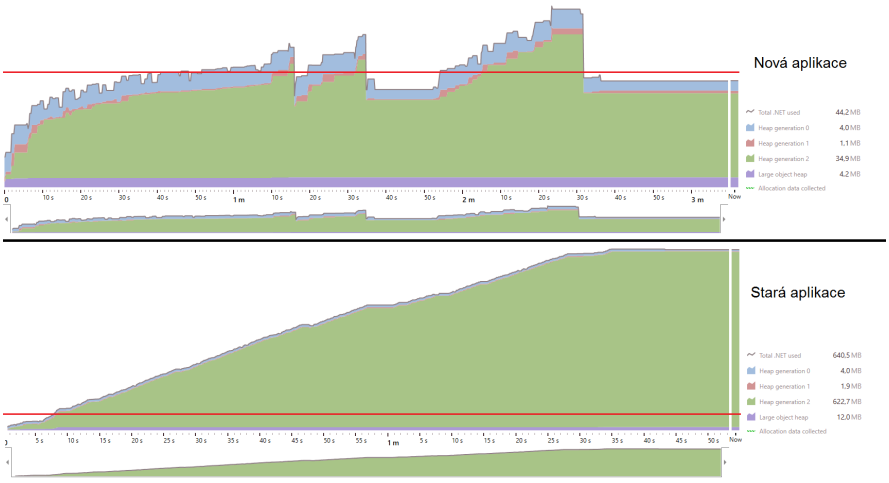
Srovnání verzí

Na začátku tohoto projektu jsme si zmínili důvody ke tvorbě nové aplikace. Teď se podíváme na to, jaké se našlo řešení a jak byly případné chyby odstraněny.

7.1 Špatná správa paměti

Během používání aplikace stoupala využitá paměť. I po době v řádu minut bylo možné dosáhnout 1 000 MB. Toto není nijak omezeno. Na obrázku 7.1 můžete vidět porovnání staré a nové aplikace z pohledu paměťové náročnosti, kde červená čára značí 50 MB paměti.

Jak můžeme vidět, ve staré aplikaci neprobíhá uvolnění paměti. Na platformě .NET se o toto stará takzvaný „Garbage Collector“, zkráceně GC. Největším důvodem bylo použití nevhodné komponenty pro navigaci aplikací.



Obrázek 7.1: Porovnání správy paměti

Typ grafu	Nová aplikace	Stará aplikace
Počet pacientů	790 ms	140 622ms
Počet epizod	11 ms	44 438ms
Počet profylaxí	39 ms	8 003ms
Celková spotřeba	39 ms	49 939 ms
Dávka na epizodu	33 ms	61 962ms
Dávka na kilogram	105 ms	87 41 ms
Dávka na kilogram v rámci epizod	22 ms	104 455 ms
Dávka na kilogram v rámci profylaxí	39 ms	17 146 ms
Čas krvácení	22 ms	- ms

Tabulka 7.1: Srovnání výpočetní doby grafů

Byla použita instance třídy **Frame**. Jedná se komponentu s navigační historií. Kdykoliv uživatel naviguje mezi záložkami, přidá se nový vizuální objekt do fronty. Tato fronta udržuje referenci na daný objekt po celou dobu svého života, což je v tomto případě běh aplikace. Díky tomu nemůže GC uvolnit alokované prostředky.

7.2 Pomalé zpracování dat

Pomalý běh činil aplikaci nepoužitelnou. Pro graf typu *Čas krvácení* dochází k zacyklení a výpočet není nikdy ukončen. Přehled rychlosti výpočtu můžete vidět v tabulce 7.1. Grafy byly počítány s 160 pacienty, 3200 epizodami a 350 profylaxemi.

7.3 Nulové zabezpečení dat

V původní verzi aplikace neexistovala žádná autentizace a databázový soubor bylo možné číst pomocí volně dostupných aplikací, ať už speciálních databázových nástrojů, nebo třeba poznámkového bloku. O tom, jak byly tyto problémy vyřešeny, si můžete přečíst v kapitolách 3.1 a 4.3.

7.4 Velký počet chyb ve vyhodnocování

Vyhodnocování dat je podloženo jednotkovými testy a testovacími případy. Více v sekci 6.

7.5 Nekonzistentní databáze

Při vytvoření nové aplikace docházelo ke změně struktury a technologie databáze. Toto mělo za následek migraci dat, během které došlo k manuální úpravě

dat.

Závěr

V rámci této bakalářské práce jsem si osvojil základy analýzy, návrhu a tvorby aplikace.

Jedním z hlavních aspektů byla komunikace s klientem. Zde jsem se naučil, jak od klienta získat popis domény, a na základě tohoto zaznamenat analýzu. Hlavními složkami analýzy byly funkční a nefunkční požadavky, případy užití aplikace a doménový model. Pro potřeby analýzy jsem si osvojil notaci UML. Vývoj tohoto projektu byl veden spíše agilním způsobem. Zákazník reagoval na aplikaci a následně požadoval změny a další funkcionalitu. V případě většiny projektů by toto mohl být problém, ale za tímto projektem stáli sponzoři, kteří s tímto přístupem neměli problém.

Po pochopení cíle uživatele bylo potřeba provést návrh systému. Dvěma hlavními body návrhu byla realizace nefunkčních požadavků a samotná architektura systému. Pod architekturu se řadí implementační součásti jako třídní diagram a databázový model. Nedílnou součástí bylo vybrání vhodných technologií. Vzhledem k tomu, že se jedná o komerční projekt, bylo možné zakoupit placené nástroje pro ušetření času a zvýšení kvality softwaru.

Na konec přišla samotná implementace. Tato část byla výrazně jednodušší, ačkoli časově náročnější. Hlavním důvodem bylo to, že běžně používám zvolené technologie a mám zažité principy pro tvorbu aplikací s jejich využitím. Největší novinkou pro mě byly certifikáty použité pro autentizaci uživatelů. Zde jsem se seznámil s jejich strukturou a rozhraním na platformě .NET pro práci s nimi.

Projekt sahá daleko za hranice bakalářské práce a jeho vývoj bude nadále pokračovat. Mezi hlavní kroky, které budou následovat, se řadí rozšíření pokrytí jednotkovými testy. Dále by aplikace měla implementovat uživatelské role a podle toho zobrazovat obsah a umožňovat práci s daty. Grafový modul by měl uživateli umožnit uložit a následně využít nastavení filtrů. Aplikace je aktuálně nasazena, přičemž nahradila starou verzi.

Literatura

[1] Hemophilia A, hemarthrosis. [Cited 2016-03-10]. Dostupné z: http://www.uaz.edu.mx/histo/pathology/ed/ch_20a/c20a_sc1.jpg

[2] Cetkovský, P.: *Intenzivní péče v hematologii*. Praha: Galén, první vydání, c2004, ISBN 80-7262-255-2.

[3] NÁKLADY NA LÉČBU HEMOFILIE V ČR. 2016, [Cited 2016-03-12]. Dostupné z: http://www.prolekare.cz/hemofilie-novinky/naklady-na-lecbu-hemofilie-v-cr-5878

[4] Wintrobe, M. M.; Lee, G.: *Wintrobe's clinical hematology*, ročník 2. Baltimore: Williams Wilkins, 10 vydání, c1999-, ISBN 0683-18242-0.

[5] Hemofilie. [Cited 2016-03-10]. Dostupné z: http://www.baxter.cz/pro_odborniky_ve_zdravotnictvi/hemofilie/

[6] SEX RATIO. [Cited 2016-03-07]. Dostupné z: https://www.cia.gov/library/publications/the-world-factbook/fields/2018.html

[7] Hemofilie v české republice. [Cited 2016-03-11]. Dostupné z: http://www.hemofilie.cz/co-je-hemofilie

[8] Hoffman, R.: *Hematology: basic principles and practice*. New York: Churchill Livingstone, 1991, ISBN 0-443-08643-5.

[9] Factor replacement therapy. [Cited 2016-03-04]. Dostupné z: http://www.hemophilia.ca/en/bleeding-disorders/hemophilia-a-and-b/the-treatment-of-hemophilia/factor-replacement-therapy/

[10] Za nejdražšího klienta dala loni VZP 46 milionů, v TOP 20 byl roční chlapeček i 71letý důchodce. 2014, [Cited 2016-03-10]. Dostupné z: https://www.vzp.cz/o-nas/aktuality/za-nejdrazsiho-klienta-dala-loni-vzp-46-milionu-v-top-20-byl-rocni-chlapecek-i-71lety-duchodce

[11] Hemofilie patří k nejdražším nemocem, VZP stojí ročně 470 milionů korun. 2015, [Cited 2016-03-10]. Dostupné z: https://www.vzp.cz/o-nas/aktuality/hemofilie-patri-k-nejdrazsim-nemocem-vzp-stoji-rocne-470-milionu-korun

[12] Custom Data Binding, Part 2. 2005, [Cited 2016-03-20]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms993236.aspx

[13] Attached Properties Overview. [Cited 2016-03-23]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms749011(v=vs.100).aspx

[14] let your desktop apps shine. [Cited 2016-03-24]. Dostupné z: http://mahapps.com/

[15] SQLCipher. [Cited 2016-03-17]. Dostupné z: https://www.zetetic.net/sqlcipher/

[16] About Telerik. [Cited 2016-03-26]. Dostupné z: http://www.telerik.com/company

[17] NUnit. [Cited 2016-04-11]. Dostupné z: http://www.nunit.org/

[18] Test Script – testovací scénář. [Cited 2016-04-11]. Dostupné z: http://testovanisoftwaru.cz/dokumentace-v-testovani/test-case/

Seznam tabulek

Seznam obrázků

Seznam zdrojových kódů

Příloha **A**

Seznam použitých zkratek

VZP Všeobecná zdravotní pojišťovna

CRUD manipulace s daty. Vytvořit, číst, upravit, smazat

UI uživatelské rozhraní

ORM mapování tříd na struktury databáze

Příloha **B**

Obsah přiloženého CD

- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `models` ..... modely analýzy a návrhu
  - `class.png` ..... třídní diagram
  - `database.png` ..... databázový model