FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Aplikace na podporu diagnostiky a léčby symptomů dolních cest močových (UROsoft)
Student:	Bc. Petr Pikaus
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Vytvořte vhodné řešení pro pacientskou aplikaci určenou ke sběru dat od osob trpících dysfunkcí močových cest.
Na základě vyhodnocení sesbíraných dat v případě potřeby doporučte pacientovi návštěvu lékaře. Vyhodnocení musí probíhat v souladu s postupy uznávanými Českou urologickou společností.
Umožněte pacientovi zobrazení historických dat a zároveň mu v podobě grafů zobrazte trend stavu jeho zdravotního stavu.
Proveďte implementaci navrženého řešení. Použijte standardní postupy pro návrh a testování UI aplikace, včetně zapojení samotných pacientů.
Respektujte platnou legislativu upravující zpracování osobních údajů.
Aplikaci nasaďte do testovacího provozu.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 15. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Aplikace na podporu diagnostiky a léčby symptomů dolních cest močových (UROsoft)

Bc. Petr Pikaus

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

9. května 2019

Poděkování

V první řadě bych rád poděkoval as. MUDr. Kamilu Švabíkovi, Ph.D. za osvětlení problematiky, definici požadavků na mobilní aplikaci a kontrolu odborné části této práce. Dále společnosti TME solutions, s. r. o, za umožnění realizace tohoto projektu a jejím zaměstnancům, především Bc. Patriku Cachnínovi, za konzultace při návrhu podoby aplikace. Obrovské díky patří mé rodině za neutuchající podporu.

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, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 9. května 2019 .....

Odkaz na tuto práci

Pikaus, Petr. *Aplikace na podporu diagnostiky a léčby symptomů dolních cest močových (UROsoft)*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Dysfunkce dolních cest močových trápí velkou část populace. Tato diplomová práce se zaobírá analýzou, návrhem, implementací a testováním aplikace pro podporu diagnostiky a léčby těchto obtíží. Jejím hlavním cílem je zlepšit aktuální stav sběru dat od pacientů pomocí elektronizace tohoto procesu. Rozebírá přínosy vybraného řešení. Návrh a implementace dbají na budoucí rozvoj. Výstupem praktické části je pacientská mobilní aplikace pro platformy Android a iOS, která slouží k diagnostice symptomů dolních cest močových, propojuje pacienty a lékaře a v neposlední řadě zpřesněním dat zlepšuje kvalitu poskytnuté péče.

Klíčová slova dolní cesty močové, hyperaktivní močový měchýř, pacientská aplikace, mikční deník, dotazníky kvality života

Abstract

Lower urinary tract symptoms widely affect the general population. This Master's thesis deals with analysis, design, implementation and testing of an application for enhanced diagnosis and treatment of these problems. The main objective is to improve the current process of collecting data by computerization. It discusses the benefits of the selected solution. Architecture and implementation heavily focus on the ease of future development. The output of the practical part is a mobile application for patients which runs on Android and iOS. The core features are: diagnosis of the lower urinary tract symptoms, provision patients with a convenient way of accessing a doctor and last but not least improved accuracy of the collected data by reminders and interaction.

Keywords lower urinary tract, overactive bladder, patient application, voiding diary, quality of life questionnaires

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Močový trakt je nepostradatelnou součástí lidského těla, jejímž úkolem je vylučování odpadních látek z těla ven. Při zaměření na dolní cesty močové zjistíme, že je s nimi spjato velké množství nemocí a potíží, které trápí širokou populaci. Často se jedná o problémy zhoršující kvalitu života pacientů bez přímého ohrožení jejich života. Pro potřeby diagnostiky a léčby těchto problémů se mimo jiné využívají informace od pacientů popisující jejich stav a funkčnost dolních cest močových. Nejběžnější formou získávání těchto informací jsou dotazníky kvality života a mikční deníky. Lékaři typicky rozdávají papírové formuláře, které pacienti vyplňují. Bohužel zde často dochází k předání nepřesných informací, což má za následek zhoršení kvality poskytnuté péče.

Tato práce uvede čtenáře do problematiky močového traktu, vysvětlí, co ho vlastně tvoří, jaká jsou nejčastější onemocnění spojená s dolními cestami močovými a jaké diagnostické postupy lékaři v praxi používají. Následně předloží analýzu problémové domény se zaměřením na sběr a vyhodnocení dat od pacientů. Dalším krokem bude výběr vhodné platformy, která by maximalizovala použitelnost pro hlavní segment pacientů včetně zvolení použitých technologií. Poté dojde k vytvoření návrhů uživatelského rozhraní včetně jejich vyhodnocení. V neposlední řadě bude diskutována stránka architektury a samotné implementace. Na závěr dojde ke zhodnocení kvality aplikace za pomoci testování její funkčnosti a použitelnosti. Hlavním odborníkem konzultovaným pro potřeby aplikace bude as. MUDr. Kamil Švabík, Ph.D.

Kapitola 1

Močový trakt

Jedná se o systém dutých orgánů nacházejících se v lidském těle, jehož cílem je mimo jiné regulovat množství vody a vyloučit z těla ven odpadní látky. Tento systém dělíme na horní a dolní cesty močové [1]. Rozložení v lidském těle můžete vidět na obrázku 1.1.

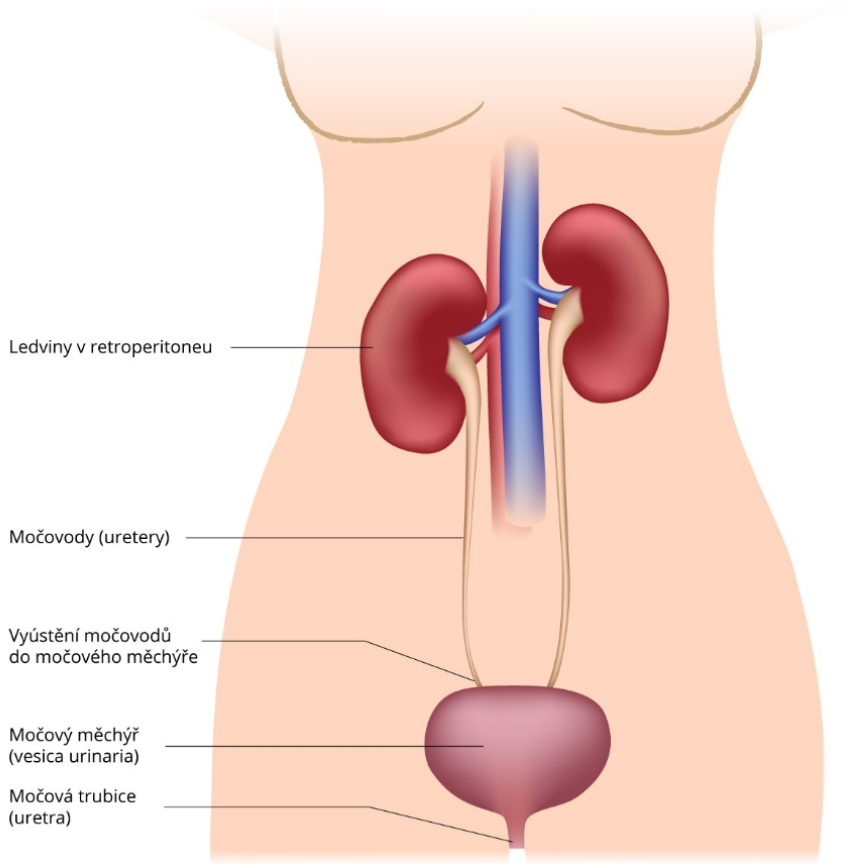
Horní cesty močové se skládají z ledvin a močovodu. Hlavním úkolem ledvin je tvorba moči, díky které dochází k vyloučení přebytečných a škodlivých látek z těla ven. Dále se ledviny starají o znovuvstřebání látek, které jsou pro tělo užitečné. Z ledvin se moč dostává do močovodu, kterým je přepravena do močového měchýře [7].

Dolní cesty močové tvoří močový měchýř a močová trubice. Močový měchýř má dvě funkce, jímací a vylučovací [8]. Jímací fáze způsobuje příjem a retenci moči v močovém měchýři. Postupným plněním a expanzí močového měchýře dochází k tlaku na receptory uložené ve svalovině okolo něj a ty následně odesílají signály do mozku. Po zpracování přijatých signálů se dostavuje pocit nucení na močení [9].

1.1 Symptomy dolních cest močových

Když se bavíme o symptomech dolních cest močových, označovaných v praxi také jako LUTS (lower urinary tract symptoms), mluvíme o projevech onemocnění, které mají u pacienta za následek návštěvu lékaře [10]. Obvykle se jedná o potíže s významným vlivem na zhoršení kvality života v podobě zvýšené nezaměstnanosti, snížené pracovní produktivity, špatného spánku, zhoršené kvality sexuálního života a zvýšené míry výskytu deprese [11]. Tyto symptomy se dělí na jímací, obstrukční a postmikční.

Jímací symptomy ovlivňují schopnost močového měchýře vykonávat jeho jímací funkci, tedy přijímat a zadržovat moč. Tyto symptomy jako celek nazýváme hyperaktivní měchýř (OAB) [12] a patří mezi ně především následující:



Obrázek 1.1: Podoba močového traktu v lidském těle [1]

- **urgence** – náhlý a závažný pocit nucení na močení, který může vyvrcholit inkontinencí [12];
- **frekvence** – počet mikcí za 24 hodin. Za normální považujeme 7 mikcí [12];
- **nykturie** – počet mikcí, které nastaly v průběhu noci. Pacient tedy spal, vzbudil se, vykonal potřebu a šel znovu spát. Spánkem zde máme na mysli pouze hlavní spánek dne. Ačkoli by se měly započítávat jen mikce, kde pocit na močení způsobil vzbuzení pacienta [12], v praxi není dost často možné odlišit důvod probuzení. Přípustná je jedna nykturie za noc [12];
- **inkontinence** – dělí se na **stresovou** a **urgentní**. O stresové inkontinenci mluvíme, pokud došlo k úniku moči bez varování, zatímco urgentní inkontinence popisuje únik moči, před nímž pacient pociťoval nucení na močení. Jedním z mnoha spouštěčů stresové inkontinence je kýchnutí,

kdy má tlak vyvinutý břišní stěnou za následek iritaci močového měchýře a následné uvolnění moči.

Druhou kategorii tvoří **obstrukční symptomy**, které obecně narušují schopnost močového měchýře plnit jeho vylučovací funkci. Řadí se mezi ně problém se zahájením močení, slabý nebo přerušovaný proud moči a výkyvy síly proudu moči před ukončením močení [10, 13].

Poslední, tedy **postmikční symptomy** jsou navázány na aktivní ukončení mikce a spadá pod ně pocit nedostatečného vymočení a nedobrovolný únik moči po ukončeném močení [14].

1.2 Diagnostika LUTS

Mezi hlavní nástroje, které lékaři používají pro diagnostiku poruch dolních cest močových, se řadí mikční deník a dotazníky kvality života [15, s. 85]. Po jejich vyhodnocení získává lékař dodatečné informace, které následně používá k případné léčbě. Dalšími diagnostickými pomůckami jsou fyzikální vyšetření, zobrazovací metody, endoskopické vyšetření a chemické, mikroskopické a kultivační vyšetření moči [13].

1.2.1 Mikční deník

Mikční deník, někdy také zvaný mikční karta, je záznamový arch, na který pacient zapisuje příjmy tekutin a výdaje moči v průběhu celého dne. V názoru na minimální délku mikčního deníku se různé zdroje rozcházejí, avšak s ohledem na možné denní odchylky se vždy pohybují mezi 2 a 3 dny [15, 16]. Kvůli tomu, že první ranní mikce každého dne je započítávána do noční funkce močového traktu předešlého dne, je zapotřebí zaznamenat ještě první ranní mikci po posledním dni mikčního deníku. Na obrázku 1.2 vidíme podobu jednodenního mikčního deníku.

1.2.2 Dotazníky kvality života

Tyto dotazníky si kladou za cíl zjistit kvalitu života pacienta a upozornit na možné symptomy dolních cest močových. Výsledkem jejich vyhodnocení nejsou instrukce k postupu v terapii, ale pouze upozorňují na zvýšenou až závažnou nutnost návštěvy lékaře. Lékař následně provede diagnózu za pomoci odpovědí poskytnutých v rámci daného dotazníku a výstupů z dalších metod diagnostiky zmíněných dříve.

1.2.2.1 IPSS

Benigní hyperplazie prostaty je jednou z hlavních příčin LUTS u mužů. Z toho důvodu se IPSS (International Prostate Symptop Score) řadí mezi primární

Den

Příjem tekutin	
Čas	Množství tekutin (v mililitrech)
Celkem:	

Příjem tekutin	
Čas	Množství tekutin (v mililitrech)
Celkem:	

Denní úniky	
Mokré spodní prádlo	
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne
Ano	Ne

Noc

Čas uložení se ke spánku:	..... hod.
Čas vstávání po probuzení:	..... hod.
Celková doba spánku:	..... hod.
Délka prvního nepřerušeného spánku	V minutách .....

	Čas	Mililitry
Objem noční moči	.....	.....
	.....	.....
	.....	.....
	.....	.....
	.....	.....
Objem první raní moči	..... ml	

Počet močení za noc	x
V noci jsem se šel vymočit	1× 2× 3× ..×
z toho .....x nutkání na močení	
z toho .....x jsem nemohl/a spát	
z toho .....x jiné důvody (uveďte):	

Obrázek 1.2: Jednodenní mikční deník [2]

dotazníky v tomto segmentu. Dotazník se skládá z 8 otázek a součet bodů za jednotlivé otázky udává, jak urgentně by měl člověk vyhledat lékaře pro další vyšetření. Skóre má 3 úrovně závažnosti: ≤ 7 bodů, ≤ 19 bodů a ≥ 20 bodů, přičemž návštěva lékaře je doporučena už od 8 bodů. Alternativou je použití VPSS (New Visual Prostate Symptom Score), jehož výsledky silně korelují s výsledky IPSS, zatímco úroveň inteligence potřebná pro vyplnění tohoto dotazníku je výrazně nižší [17]. Na obrázku 1.3 můžete vidět znění dotazníku IPSS.

1.2.2.2 OAB-V8

Tento dotazník slouží pro rychlou a snadnou diagnostiku OAB. Deník tvoří 8 otázek, a pokud je pacient muž, přičítá si 2 extra body. V případě, že celkové

	Vůbec ne	Asi v 1/5 případů	V méně než 1/2 případů	Asi v 1/2 případů	Ve více než 1/2 případů	Téměř vždy
Neúplné vyprázdnění: Jak často jste během posledního měsíce měl po vymočení pocit nevyprázdněného močového měchýře?	0	1	2	3	4	5
Frekvence: Jak často jste během posledního měsíce musel znovu močit dříve než za 2 hodiny po předchozím vymočení?	0	1	2	3	4	5
Přerušované močení: Jak často jste během posledního měsíce pozoroval, že se močení několikrát přerušilo a znovu začalo?	0	1	2	3	4	5
Naléhavost: Jak často jste během posledního měsíce jen s potížemi močení oddálil?	0	1	2	3	4	5
Oslabení proudu moče: Jak často jste během posledního měsíce měl slabý proud moči?	0	1	2	3	4	5
Tlačení na močení: Jak často jste během posledního měsíce musel tlačit, abyste začal močit?	0	1	2	3	4	5
Noční močení: Jak často jste během posledního měsíce musel v noci kvůli močení vstávat? (Průměrně za noc)	Nikdy	1x	2x	3x	4x	5x a více

Obrázek 1.3: Mezinárodní skóre prostatických symptomů [3]

skóre překročí 7 bodů, je pacientovi doporučena návštěva lékaře pro další vyšetření. Podobu dotazníku si můžete prohlédnout na obrázku 1.4.

1.3 Léčba

Léčba symptomů dolních cest močových se dělí do 3 kategorií: změna denního režimu, medikamentózní léčba a chirurgický zákrok.

Hlavní výhodou **změny denního režimu** je charakter této léčby. Jedná se o neinvazivní postup, jehož efektivita je přímo závislá na pacientovi a času, který jí věnuje. Obecně se pro rychlý nástup kýženého klinického efektu doporučuje doplnit tyto behaviorální opatření o farmakoterapii [16]. Typické zásahy do režimu pacienta jsou tyto:

Nakolik Vás obtěžovalo...	vůbec ne	trochu	docela	poněkud	více hodně	velmi hodně
1. časté močení během dne?	0	1	2	3	4	5
2. nepříjemné nucení na močení?	0	1	2	3	4	5
3. náhlé nucení na močení s malými nebo žádnými varovnými projevy?	0	1	2	3	4	5
4. nepředvídaný únik malého množství moči?	0	1	2	3	4	5
5. noční močení?	0	1	2	3	4	5
6. noční probuzení z důvodu potřeby se vymočit?	0	1	2	3	4	5
7. nekontrolovatelné nucení na močení?	0	1	2	3	4	5
8. pomočení při silném nucení na močení?	0	1	2	3	4	5
9. Jste muž?	ANO			NE		

Obrázek 1.4: Dotazník hodnotící hyperaktivitu močového měchýře [4]

- snížení příjmu tekutin v časech, kdy je nadměrné močení nejvíce obtěžující,
- snížení příjmu kofeinu nebo alkoholu kvůli jejich diuretickým účinkům,
- zvýšení maximální funkční kapacity močového měchýře pomocí cvičení,
- dvojité močení, kdy člověk sedí v pozici ideální pro močení, vyčká po dobu 20–30 vteřin a následně se pokusí vymočit znovu [18, 19].

Farmakoterapie se zaměřuje především na správnou funkci močového měchýře a prostaty. Základ farmakologické léčby tvoří antimuskarinika, jejichž úkolem je blokovat cholinergní muskarinové receptory. Díky nim dochází ke snížení počtu a amplitudy urgencí. Bohužel použití antimuskarinik může vést k nežádoucím účinkům v podobě ovlivnění jiných orgánů lidského těla. Typicky se jedná o projevy v podobě sucha v ústech, rozmazaného vidění a obstipace. U mužů je tato léčba doplněna o inhibitory 5-AR, které zmenšují prostatu a tím se snižuje riziko retence moči. Pokud je hmotnost prostaty více než 30–40 g, jsou kromě inhibitorů 5-AR běžně nasazovány i alfa-blokátory [12].

Chirurgické zákroky jsou v porovnání s předešlými metodami výjimečné. Pokud není dosaženo uspokojivého výsledku jinou cestou, přichází na řadu semiinvazivní či operativní metody léčby, jako transuretrální incize prostaty, transuretrální resekce prostaty, transvezikální prostatektomie, transuretrální elektrovaporizace prostaty, Stollerova aferentní neurostimulace, sakrální neuromodulace, augmentační cystoplastika a další [15, 16, 12].

Kapitola 2

Analýza

Analytická část práce na projektu obsahovala nejdříve zmapování dostupnosti konkurenčních aplikací. Poté, co bylo vyhodnoceno, že žádná existující aplikace přímo neohrožuje nasazení a provoz tohoto projektu, se přešlo na doménovou analýzu. Ta zahrnovala především zjištění funkčních a nefunkčních požadavků, výběr vhodné platformy a technologií, sepsání případů užití a tvorbu doménového modelu. Každou z těchto částí si teď rozebereme v detailu.

2.1 Zmapování konkurence

Prvním krokem v analýze bylo zjištění dostupných aplikací, jejichž doména se znatelně protíná s tou naší. Největší část naší cílové skupiny tvoří lidé nad 55 let, z čehož pramení malé očekávání znalosti anglického jazyka, a je tedy nutné se zaměřovat na aplikace s českou lokalizací. Přední aplikací, která se zaobírá podobnou problematikou, je SeniControl. Jedná se o aplikaci od společnosti Seni, která se věnuje výrobě inkontinenčních pomůcek. Tato aplikace je dostupná pro iOS a Android a právě obdržela českou lokalizaci. Z pohledu na Google Play, resp. Apple Store, ale zjistíme, že aplikace má mnoho problémů s použitelností. Po stažení a vyzkoušení aplikace bych jako hlavní problémy uvedl následující: nereaguje na pokyny, nevyzývá pacienta k zaznamenání dat, je orientovaná na pomůcky vyráběné společností Seni a v neposlední řadě aplikace používá elementy uživatelského rozhraní, které považuji za méně intuitivní pro starší lidi. Oproti tomu UROsoft cílí nejen na sběr a zpracování pacientských dat, ale především na propojení pacienta s lékařem a s tím spojené zlepšení léčby. Z toho vyvozuji, že aktuálně neexistuje žádný konkurenční produkt, který by ovlivnil spuštění projektu UROsoft.

2.2 Výběr platformy

Při výběru cílové platformy byla základem konzultace s MUDr. Švabíkem a následná verifikace u dalších praktických urologů, gynekologů a urogynekologů. Výstupem bylo, že primární platformou pro zadávání dat by měla být mobilní aplikace. Hlavními důvody byly:

portabilita – pacient má mobilní zařízení vždy u sebe, a nebude tak docházet k pozdnímu zadání, které aktuálně způsobuje znatelné nepřesnosti v zaznamenaných datech, jelikož pacienti často vyplňují deník na konci dne nebo i těsně před návštěvou lékaře;

notifikace – mobilní zařízení nám umožní pacienta v pravidelných intervalech upomínat o potenciálně chybějících záznamech, což přinese jejich další zpřesnění.

Sekundární platformu by měla především kvůli snadnějšímu cílení online kampaní tvořit webová aplikace. Ta bude vhodná i pro případy, kdy pacient nemá mobilní aplikaci, a v neposlední řadě pro zjednodušení vzdáleného zadávání dat rodinnými příslušníky za jejich rodiče, prarodiče aj.

Tato práce se bude zaobírat návrhem REST API pro komunikaci s mobilní aplikací a návrhem a implementací mobilní aplikace. Vývoj API a webové aplikace bude proveden společností LinkSoft Technologies, a. s.

2.3 Zjištění požadavků na aplikaci

Sběr požadavků probíhal za přítomnosti MUDr. Švabíka. Úspěšně jsme skloubili jeho znalosti a praktické zkušenosti na poli léčby LUTS s mými poznatky ohledně vývoje mobilních aplikací.

2.3.1 Funkční požadavky

1. Obecné požadavky
   a) Aplikace musí obsahovat informace shrnující její funkcionality.
   b) Aplikace musí obsahovat podmínky používání, které respektují platnou legislativu.
   c) Aplikace nesmí o pacientovi sbírat jiné informace, než je v danou dobu nezbytně nutné.
   d) Domácí stránkou pro nepřihlášeného uživatele je přihlašovací obrazovka.
   e) Domácí stránkou pro přihlášeného uživatele s rozepsaným deníkem je obrazovka rozpracovaného deníku.

f) Domácí stránkou v ostatních případech je stránka, kde uživatel vidí žádosti od lékaře a má možnost si vyplnit hlavní dotazník pro své pohlaví nebo přejít na stránku se všemi dotazníky.

2. Přihlášení

a) Pro práci s aplikací je nutné být přihlášený.

b) Formát e-mailu je při přihlášení validován, ale použití neregistrovaných domén nikoliv.

c) Ze stránky přihlášení může uživatel přejít na registraci a obnovení hesla.

d) Uživatel není aplikací svévolně odhlášen.

e) Když uživatel přechází na stránku pro obnovu zapomenutého hesla, je hodnota pole e-mailu ze stránky přihlášení přenesena.

f) Pokud není zadaný e-mail vázán k žádnému účtu, je tato informace uživateli sdělena.

g) Pokud je zadaný e-mail validní, ale zadané heslo není správné, je tato skutečnost uživateli sdělena.

h) Uživatel se může odhlásit.

3. Obnova zapomenutého hesla

a) Uživatel si může zapomenuté heslo obnovit po zadání své e-mailové adresy, na kterou je účet vedený.

b) Formát e-mailu je validován, ale použití neregistrovaných domén nikoliv.

c) Pokud není zadaný e-mail vázán k žádnému účtu, je tato informace uživateli sdělena.

4. Registrace

a) Registrace je volně dostupná.

b) Registrace vyžaduje následující údaje: pohlaví pacienta, e-mail, heslo a souhlas s podmínkami užití aplikace.

c) Minimální délka hesla je 6 znaků.

d) Maximální délka hesla je 107 znaků.

e) Formát e-mailu je validován, ale použití neregistrovaných domén nikoliv.

f) Pokud zadaný e-mail již existuje, je tato informace uživateli sdělena.

g) Po registraci je uživatel přesměrován na hlavní stránku pro přihlášené uživatele.

5. Změna hesla
    a) Uživatel má možnost si kdykoliv heslo změnit.
    b) Změna hesla vyžaduje staré a nové heslo, přičemž na nové heslo jsou kladeny stejné nároky jako při registraci.
    c) Pokud staré heslo není správné, je o tom uživatel informován.
6. Nastavení profilu
    a) Pacient si může upravit svůj profil.
    b) Pacient si může smazat svůj rozšířený profil, pokud rozšíření existuje a pacient nemá sdílení s lékařem ve stavu čekání na přijetí nebo přijato.
    c) Pacient si může upravit svoji obvyklou dobu vstávání a usínání. Tato změna není reflektována v právě rozpracovaném deníku, pokud nějaký existuje.
7. Sdílení s lékařem
    a) Aplikace umožňuje pacientovi sdílet svoje data s lékařem.
    b) Pacient si může vyhledat lékaře.
    c) Pacient může odeslat lékaři žádost o sdílení. Pokud tak ještě neučinil, je vyzván k doplnění rozšířeného profilu v podobě jména, příjmení a data narození. Tyto informace jsou vyžadovány z důvodu identifikace pacienta lékařem.
    d) Při vyhledávání jsou lékaři zobrazeni dle rostoucí vzdálenosti jejich nejbližšího pracoviště vůči pacientově poloze.
    e) Pacient si může seřadit lékaře dle vzdálenosti od libovolné obce v ČR.
    f) Pacient si může filtrovat vyhledané lékaře dle jejich jména.
    g) Pacient si může zobrazit seznam lékařů, se kterými sdílí data, resp. jimž odeslal žádost o sdílení. U každého takového záznamu bude mít uživatel dostupnou informaci o stavu sdílení.
    h) Pacient si může zobrazit detail lékaře a lékařova pracoviště budou seřazena dle vzdálenosti od aktuální polohy pacienta.
8. Sdílení záznamů
    a) Pacient bude moci sdílet svoje vybrané záznamy na libovolný e-mail.
    b) Pacient může sdílet maximálně 10 záznamů v rámci jednoho e-mailu.

c) Pacientovi jsou na výběr zobrazeny chronologicky řazené záznamy, ze kterých si volí maximálně 10.

d) Formát zadaného e-mailu je validován, ale použití neregistrovaných domén nikoliv.

9. Notifikace

a) V aplikaci budou dva druhy notifikací: jedny spojené s lékařem a druhé systémové.

b) Aplikace bude registrovat dva kanály pro distribuci notifikací. První jsou lokální notifikace na úrovni operačního systému a druhé jsou notifikace stále viditelné v aplikaci.

c) Notifikace má k sobě vázanou akci, která by se měla vykonat při kliknutí na ni.

d) Seznam notifikací, jejich distribučních kanálů a akcí si můžete prohlédnout v tabulce 2.1.

10. Dotazníky

a) Pacient si může zobrazit dostupný seznam dotazníků k vyplnění. U každého typu vidí, kdy ho naposledy vyplnil.

b) Pacient si může vyplnit dostupný deník.

c) Dostupné deníky jsou uvedeny v tabulce 2.2.

11. Mikční deník

a) Pacient bude mít možnost založit si mikční deník s volitelným začátkem v budoucnosti a s volitelnou délkou 1–7 dnů. Výchozí začátek deníku je následující den a délka jsou 2 dny.

b) Před založením deníku se aplikace zeptá na obvyklý čas usínání a vstávání pacienta, pokud tuto informaci ještě nemá.

c) Pacient může začít vyplňovat deník 3 hodiny před obvyklým časem vstávání v den, na který si nastavil začátek deníku. Pokud je tedy začátek deníku 21. 1. 2019 a obvyklý čas vstávání 08:00, je možné začít deník vyplňovat od 21. 1. 2019 05:00.

d) Po založení deníku se pacientovi zobrazí stránka shrnující průběh deníku. Pokud pacient již může deník vyplňovat, je mu na konci shrnutí zobrazeno tlačítko „Pokračovat“, v opačném případě je zobrazeno tlačítko „Zrušit deník“.

i. Po odsouhlasení shrnutí deníku je pacient přesměrován na detail rozpracovaného deníku. Pacient musí odsouhlasit shrnutí přesně jednou za každý deník.

Název notifikace	Akce	Kanály lokální (L) aplikační (A)
Informace o ukončení deníku z důvodu nezaznamenané mikce	Přesměrování na stránku pro vytvoření deníku	A, L
Informace o ukončení deníku z důvodu nezaznamenaného příjmu	Přesměrování na stránku pro vytvoření deníku	A, L
Upomenutí zaznamenání první ranní mikce	Přesměrování na stránku přidání mikce	L
Upomenutí zaznamenání mikce	Přesměrování na stránku přidání mikce	L
Upomenutí zaznamenání příjmu	Přesměrování na stránku přidání příjmu	L
Lékař přijal žádost o sdílení	Přesměrování na stránku detailu lékaře	A, L
Lékař odmítl žádost o sdílení	Přesměrování na seznam lékařů	A, L
Lékař zrušil sdílení	Přesměrování na seznam lékařů	A, L
Lékař požaduje vyplnění dotazníku	Přesměrování na první stránku daného dotazníku	A, L
Lékař požaduje vyplnění deníku	Přesměrování na stránku vytvoření deníku	A, L

Tabulka 2.1: Notifikace pro pacienta

Název dotazníku	Pro pohlaví
Mezinárodní skóre prostatických symptomů (IPSS)	Muži
Hyperaktivní močový měchýř (OAB-V8)	Všechna

Tabulka 2.2: Dostupné dotazníky

ii. Po zrušení deníku je pacient přesměrován na domovskou stránku.

e) Pacient si může zobrazit přehledovou stránku rozpracovaného deníku, která mu umožní:

i. přidat záznam o mikci,

ii. přidat záznam o příjmu,

iii. přejít na stránku se záznamy v aktuálním deníku,

iv. předčasně ukončit deník.

f) Pokud pacient nevyplní první ranní mikci do 8 hodin od obvyklého času vstávání, je rozpracovaný deník ukončen.

g) Pokud pacient nevyplní příjem tekutin do 6 hodin od první ranní mikce, resp. od posledního příjmu daného dne, pokud nějaký existuje, dojde k ukončení deníku.

h) Pokud je prodleva mezi zadanými příjmy větší než 6 hodin, dojde k ukončení deníku.

i) Pokud pro počet mikcí M_c, čas první ranní mikce $M_f t$ a aktuální čas T neplatí $M_f t + 4 * M_c > T$, dojde k upozornění na potenciálně chybějící záznamy.

j) Pokud je prodleva mezi zadanými příjmy větší než 3 hodiny, dojde k upozornění na potenciálně chybějící záznamy.

12. Záznam mikce

a) Při zaznamenávání mikcí nás zajímá čas, množství a okolnosti, zda šlo o inkontinenci bez pocitu nucení k močení a zda se jednalo o nykturii a urgence. Aplikace bude používat rozšířenou a efektivní škálu pro měření urgence PPUS (Patient Perception of Urgency Scale) [16, 20, 21], která je zobrazena v tabulce 2.3.

b) Záznam mikce nesmí být v budoucnosti.

c) Záznam mikce nesmí předcházet začátku deníku.

13. Záznam příjmu

a) U záznamů o příjmech tekutin vyžadujeme čas, množství a typ nápoje. Důvodem pro uvedení typu nápoje jsou především diuretické a antidiuretické vlivy přijímaných tekutin.

b) Ve spolupráci s MUDr. Švabíkem byly definovány následující typy nápojů: voda, káva a čaj, energetické nápoje a kola, citrusové nápoje, alkohol, jiné.

c) Záznam příjmu musí být v minulosti.

d) Záznam příjmu nesmí předcházet začátku deníku.

Hodnota	Název	Popis
0	Žádné nucení	Necítil/a jsem potřebu vyprázdnit močový měchýř, ale vymočil/a jsem se z jiných důvodů
1	Mírné nucení	Mohl/a jsem močení oddálit tak dlouho, jak bylo nutné bez obav z pomočení
2	Středně silné	Mohl/a jsem močení na krátkou chvíli oddálit bez obav z pomočení
3	Silné nucení	Močení jsem nemohl/a oddálit, ale musel/a jsem spěchat na toaletu, abych se nepomočil/a
4	Urgentní únik	Pomočil/a jsem se před příchodem na toaletu

Tabulka 2.3: Škála urgence PPUS

14. Detail deníku

    a) Pacient vidí všechny záznamy, které jsou v rámci deníku uvedeny.

    b) Zobrazené záznamy jsou řazeny chronologicky a jsou roztříděny do skupin podle dne, do kterého spadají.

    c) Pacient si může prohlédnout detail záznamu.

    d) Pokud se pacient nachází v detailu právě rozpracovaného deníku, může záznamy mazat a upravovat.

    e) Při úpravě záznamů se nesmí stát, že by pacient přesunul jedinou mikci daného dne do dne jiného.

15. Archiv deníků

    a) Pacient si může zobrazit archiv deníků, které v minulosti dokončil.

    b) Pacient může přejít na detail libovolného deníku.

16. Grafy

    a) V grafech jsou reflektovány pouze ukončené deníky.

    b) Grafy jsou počítány vždy jako průměr za deník.

    c) Pacient si bude moci zobrazit grafy běžně používané pro léčbu OAB a noční polyurie [16, 22]. Konkrétně jsou grafy popsány v tabulce 2.4.

    d) Všechny grafy, které mají více než jednu metriku, se ve chvíli, kdy se zobrazují hodnoty za alespoň 3 deníky, přepnou ze sloupcového zobrazení do spojitého.

Název	Popis
Frekvence	Počet mikcí v průběhu celého dne (den i noc) od první mikce v daném dni do poslední mikce, která nastala v průběhu spánku (včetně).
Nykturie	Počet mikcí, které proběhly v průběhu spánku, tj. pacient se probudí, jde se vymočit a poté znovu usne.
Urgence	Zobrazení míry urgencí v průběhu celého dne (den i noc) od první ranní mikce v daném dni do poslední mikce, která nastala v průběhu spánku (včetně). Urgence je v grafu zobrazena jako počet urgencí v dané hladině (0–4). Jednotlivé sloupce by v grafu měly být odstínovány dle závažnosti. Součet sloupců se rovná počtu frekvencí minus počet inkontinencí bez varování.
Inkontinence	Počet úniků bez varování od první ranní mikce po poslední mikci během noci (včetně) a počet urgencí označených jako 4. V grafu jsou tedy dva sloupce.
Objem příjmu za 24 hodin	Celkový příjem tekutin, který nastal od první ranní mikce do první ranní mikce následujícího dne vyjma této mikce.
Objem moči za 24 hodin	Celkový součet objemu mikcí, které nastaly od první ranní mikce včetně do první ranní mikce následujícího dne vyjma.
Objem noční mikce	Celkový součet objemu všech nykturií a navíc první ranní mikce následujícího dne.

Tabulka 2.4: Podporované grafy

2.3.2 Poznámky

12a Zadání informace o inkontinenci a urgenci v běžném deníku probíhá typicky separátně. Tady dojde ke změně a pacient nejdříve odpovídá na otázku, zdali došlo k inkontinenci. Pokud odpoví ano, zobrazí se mu otázka, zdali před únikem cítil urgenci. Toto opatření je zde z toho důvodu, že pacientovi je poměrně těžké vysvětlit pojem inkontinence bez varování. Tímto se zadávání stává mnohem přímočařejší a méně náchylné k chybám.

16d Ačkoli toto není matematicky přesné, s Bc. Cachnínem a Mgr. Břicháčkem jsme se usnesli, že výpovědní hodnota takového zobrazení bude pacientům výrazně bližší. Sloupcový graf se při více záznamech stává zmatečný a nelze si z něj lehce odnést dostupnou informaci.

2.4 Nefunkční požadavky

1. Minimální podporovaná verze operačního systému Android je 5.0.
2. Minimální podporovaná verze operačního systému iOS je 8.0.
3. Osobní data pacienta jsou v zařízení uložena zašifrovaná.
4. Texty v aplikaci musí být lehce upravitelné a lokalizovatelné.
5. Aplikace bude fungovat i bez připojení k internetu. Synchronizace bude probíhat na pozadí v předem vymezených intervalech. Pouze akce, které vyžadují bezprostřední odpověď serveru, budou vyžadovat připojení. Jedná se o přihlášení, registraci, obnovu hesla, vyhledání lékaře a sdílení záznamů.
6. Aplikace bude schopná se v jakémkoli okamžiku úspěšně vypořádat s výpadky připojení. Například po vytvoření zdroje na serveru a před přijetím odpovědi od serveru.
7. Komunikace se serverem bude probíhat přes TLS 1.2.

2.5 Tvorba případů užití

V této části se podíváme na případy užití. Na závěr se podíváme na pokrytí funkčních požadavků a na to, jak byly řešeny nefunkční požadavky. Pokud není explicitně specifikováno jinak, budeme pro snazší orientaci pro všechny případy užití počítat s následujícími předpoklady:

- Chyby spojené s konkrétním datovým polem jsou za předpokladu, že danou chybu lze vyhodnotit v zařízení, zobrazeny pod daným polem. Pokud je potřeba komunikace se serverem, jsou oznámení prezentována v podobě dialogu.

- Uživatel se vždy nachází na domovské stránce pro daný stav aplikace.

2.5.1 Přihlášení

Pacient bude přihlášen do aplikace.

2.5.1.1 Předpoklady

Pacient je již registrován, aplikace je spuštěna a nachází se v nepřihlášeném stavu.

2.5.1.2 Hlavní scénář

1. Pacient vyplní svůj e-mail.

2. Pacient vyplní svoje heslo.

3. Pacient kliká na „Přihlásit se“.

4. Pacientovi se načítá hlavní stránka.

2.5.1.3 Chyby

1. Pacient není připojený k internetu.

2. Pacient zadal špatné heslo.

3. Pacient zadal e-mail ve špatném formátu.

4. Pacient zadal e-mail, který není v databázi.

2.5.2 Registrace

Pacient si vytvoří účet a bude přihlášen do aplikace.

2.5.2.1 Předpoklady

Aplikace je spuštěna a nachází se v nepřihlášeném stavu.

2.5.2.2 Hlavní scénář

1. Pacient kliká na „Chci se zaregistrovat“ a je mu zobrazen registrační formulář.
2. Pacient vybere svoje pohlaví.
3. Pacient vyplní svůj e-mail.
4. Pacient vyplní požadované heslo.
5. Pacient odsouhlasí podmínky užití.
6. Pacient kliká na „Registrovat se“.
7. Pacient je přihlášen a načítá se mu hlavní stránka pro přihlášené uživatele.

2.5.2.3 Chyby

1. Pacient není připojený k internetu.
2. Pacient nezadal alespoň jedno z následujícího: pohlaví, e-mail, heslo.
3. Pacient zadal e-mail ve špatném formátu.
4. Pacient zadal zabraný e-mail.
5. Pacient zadal heslo, které nesplňuje kritéria.

2.5.3 Obnovení hesla

Pacientovi bude odeslán e-mail s postupem pro obnovení hesla.

2.5.3.1 Předpoklady

Pacient je již registrován, aplikace je spuštěna a nachází se v nepřihlášeném stavu.

2.5.3.2 Hlavní scénář

1. Pacient kliká na „Zapomněl jsem své heslo“ a je přesunut na stránku pro obnovu hesla.
2. Pacient vyplní svůj e-mail.
3. Pacient kliká na „Zaslat nové heslo“.
4. Pacientovi se zobrazuje dialog s informací o zaslaném e-mailu.
5. Pacient potvrzuje dialog a je přesunut na stránku přihlášení.

2.5.3.3 Alternativní scénář

1. Pacient se pokusí přihlásit.
2. Systém zobrazí chybu s nesprávným heslem.
3. Pacient kliká na „Zapomněl jsem své heslo“ a je přesunut na stránku pro obnovu hesla.
4. Pacientův e-mail je automaticky vyplněn dle zadání na stránce přihlášení.
5. Pacient kliká na „Zaslat nové heslo“.
6. Pacientovi se zobrazuje dialog s informací o zaslaném e-mailu.
7. Pacient potvrzuje dialog a je přesunut na stránku přihlášení.

2.5.3.4 Chyby

1. Pacient není připojený k internetu.
2. Pacient zadal e-mail ve špatném formátu.
3. Pacient zadal e-mail, který není v databázi.

2.5.4 Zobrazení informací o aplikaci

Pacientovi bude zobrazena stránka s informacemi o aplikaci.

2.5.4.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.4.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Nastavení“.
3. Pacientovi je zobrazena stránka s nastavením.
4. Pacient kliká na „Informace“.
5. Pacientovi je zobrazena stránka s informacemi o aplikaci.

2.5.5 Zobrazení podmínek užití aplikace

Pacientovi bude zobrazena stránka s podmínkami užití aplikace.

2.5.5.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.5.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Nastavení“.
3. Pacientovi je zobrazena stránka s nastavením.
4. Pacient kliká na „Podmínky užívání“.
5. Pacientovi je zobrazena stránka s podmínkami užití aplikace.

2.5.6 Změna hesla

Pacient si změní heslo ke svému účtu.

2.5.6.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.6.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Nastavení“.
3. Pacientovi je zobrazena stránka s nastavením.
4. Pacient kliká na „Změna hesla“.
5. Pacientovi je zobrazena stránka pro změnu hesla.
6. Pacient zadává svoje aktuální heslo.
7. Pacient zadává svoje nové heslo.
8. Pacient kliká na „Změnit heslo“.
9. Pacientovi je zobrazen dialog o úspěšné změně.
10. Pacient potvrzuje zobrazený dialog.
11. Pacient je přesměrován na hlavní stránku.

2.5.6.3 Chyby

1. Pacient není připojený k internetu.
2. Pacientovo nové heslo nesplňuje kritéria.
3. Pacientovo aktuální heslo neodpovídá.

2.5.7 Změna spacího režimu

Pacient si vyplní nebo změní spací režim.

2.5.7.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.7.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Nastavení“.
3. Pacientovi je zobrazena stránka s nastavením.
4. Pacient kliká na „Úprava profilu“.
5. Pacientovi je zobrazena stránka se správou profilu.
6. Pacient kliká na čas usínání.
7. Pacientovi se objevuje dialog pro výběr času.
8. Pacient vybírá svůj obvyklý čas usínání a potvrzuje výběr.
9. Pacient kliká na čas vstávání.
10. Pacientovi se objevuje dialog pro výběr času.
11. Pacient vybírá svůj obvyklý čas vstávání a potvrzuje výběr.
12. Pacient kliká na „Uložit spánkový režim“.
13. Pacientovi je zobrazen dialog o úspěšné změně.

2.5.7.3 Chyby

1. Pacient nevyplnil čas usínání.
2. Pacient nevyplnil čas vstávání.

2.5.8 Úprava profilu

Pacient si upraví profil.

2.5.8.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.8.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Nastavení“.
3. Pacientovi je zobrazena stránka s nastavením.
4. Pacient kliká na „Úprava profilu“.
5. Pacientovi je zobrazena stránka se správou profilu.
6. Pacient si upravuje ty z následujících položek, které chce měnit:
   - pohlaví,
   - jméno, příjmení a datum narození (pokud má rozšířenou registraci).
7. Pacient kliká na „Uložit profil“.
8. Pacientovi je zobrazen dialog s potvrzením o změně.

2.5.8.3 Chyby

1. Pacient není připojený k internetu.

2.5.9 Smazání rozšířené registrace

Pacient si smaže rozšířený profil.

2.5.9.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má vyplněný rozšířený profil.

2.5.9.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Nastavení“.
3. Pacientovi je zobrazena stránka s nastavením.

4. Pacient kliká na „Úprava profilu“.
5. Pacientovi je zobrazena stránka se správou profilu.
6. Pacient kliká na „Smazat údaje“.
7. Pacientovi je zobrazen dialog s potvrzením o smazání.
8. Pacient potvrzuje zobrazený dialog.
9. Pacientovi je zobrazen dialog o úspěšném smazání.
10. Z uživatelského rozhraní se smaže rozšířená registrace.

2.5.9.3 Chyby

1. Pacient není připojený k internetu.
2. Pacient má sdílení s lékařem, které je ve stavu, kdy čeká na přijetí nebo je přijato.

2.5.10 Odhlášení z aplikace

Pacient bude odhlášen a zobrazí se mu přihlašovací stránka.

2.5.10.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.10.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Nastavení“.
3. Pacientovi je zobrazena stránka s nastavením.
4. Pacient kliká na „Odhlásit“.
5. Pacientovi je zobrazen dialog s potvrzením o odhlášení.
6. Pacient potvrzuje zobrazený dialog.
7. Pacientovi je zobrazena přihlašovací stránka.

2.5.11 Odeslání žádosti o sdílení

Lékaři bude odeslána žádost o sdílení údajů.

2.5.11.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.11.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Můj lékař“.
3. Pacientovi je zobrazena přehledová stránka lékařů, se kterými sdílí údaje.
4. Pacient kliká na „Vyhledat lékaře“.
5. Pacientovi je zobrazena stránka se seznamem dostupných lékařů.
6. Pokud pacient sdílí údaje o své poloze, dojde k seřazení lékařů podle vzdálenosti pacienta od jejich nejbližšího pracoviště a tato vzdálenost je u každého lékaře zobrazena.
7. Pacient zadává jméno svého lékaře.
8. Seznam obsahuje pouze lékaře, kteří odpovídají zadanému textu.
9. Pacient kliká na požadovaného lékaře.
10. Pacientovi je zobrazena stránka s detailem lékaře.
11. Pacient kliká na „Sdílet údaje s lékařem“.
12. Pacientovi je zobrazen dialog o úspěšném odeslání žádosti.

2.5.11.3 Alternativní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Můj lékař“.
3. Pacientovi je zobrazena přehledová stránka lékařů, se kterými sdílí údaje.
4. Pacient kliká na „Vyhledat lékaře“.
5. Pacientovi je zobrazena stránka se seznamem dostupných lékařů.
6. Pokud pacient sdílí údaje o své poloze, dojde k seřazení lékařů podle vzdálenosti pacienta od jejich nejbližšího pracoviště a tato vzdálenost je u každého lékaře zobrazena.
7. Pacient si vybírá město, kde by rád navštívil lékaře.
8. Seznam lékařů je seřazen dle vzdálenosti od vybraného města a tato vzdálenost je u každého lékaře zobrazena.

9. Pacient kliká na požadovaného lékaře.
10. Pacientovi je zobrazena stránka s detailem lékaře.
11. Pacient kliká na „Sdílet údaje s lékařem“.
12. Pacientovi je zobrazen dialog o úspěšném odeslání žádosti.

2.5.11.4 Chyby

1. Pacient není připojený k internetu.

2.5.12 Zrušení sdílení

Žádost (potažmo sdílení) bude zrušena.

2.5.12.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.12.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Můj lékař“.
3. Pacientovi je zobrazena přehledová stránka lékařů, se kterými sdílí údaje.
4. Pacient kliká na požadovaného lékaře.
5. Pacientovi je zobrazena stránka s detailem lékaře.
6. Pacient kliká na „Sdílení je aktivní“, resp. „Čeká na schválení“.
7. Pacientovi je zobrazen dialog s potvrzením o zrušení.
8. Pacient potvrzuje zobrazený dialog.
9. Pacientovi je zobrazen dialog o úspěšném zrušení sdílení.

2.5.12.3 Chyby

1. Pacient není připojený k internetu.

2.5.13 Zobrazení detailu lékaře, se kterým pacient sdílí data

Pacient si zobrazí všechny známé údaje o lékaři, se kterým sdílí data.

2.5.13.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.13.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Můj lékař“.
3. Pacientovi je zobrazena přehledová stránka lékařů, se kterými sdílí údaje.
4. Pacient kliká na požadovaného lékaře.
5. Pacientovi je zobrazena stránka s detailem lékaře. Pokud pacient sdílí údaje o své poloze, jsou zařízení řazena dle aktuální vzdálenosti od pacienta.

2.5.14 Sdílení dat na e-mail

Pacient odešle zvolené deníky, resp. dotazníky exportované do PDF na libovolný e-mail.

2.5.14.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.14.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na ikonu pro sdílení.
3. Pacientovi je zobrazena stránka pro sdílení.
4. Pacient zadá cílový e-mail.
5. Pacient si vybere dotazníky a deníky, které by rád odeslal, z chronologicky řazeného seznamu.
6. Systém umožní vybrat maximálně 10 položek.
7. Pacient kliká na „Odeslat vybrané záznamy“. Toto je možné, pouze pokud byl vybrán alespoň jeden záznam.
8. Pacientovi je zobrazen dialog o úspěšném odeslání.

2.5.14.3 Chyby

1. Pacient není připojený k internetu.
2. Pacient nezadal cílový e-mail.
3. Pacient zadal e-mail ve špatném formátu.

2.5.15 Vyvolání akce přes aplikační notifikaci

Pacientovi přijde notifikace a uživatel vyvolá akci s ní spojenou. Popsaný případ je aplikovatelný pro všechny notifikace popsané v tabulce 2.1 doplněním příslušného typu notifikace N_t a očekávané akce N_a.

2.5.15.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má notifikaci typu N_t.

2.5.15.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Upozornění“.
3. Pacientovi je zobrazena stránka se všemi upozorněními, která byla distribuována v rámci aplikačního kanálu.
4. Pacient kliká na notifikaci N_t.
5. Systém provádí akci N_a.
6. Pokud byla notifikace nepřečtená, stává se přečtenou.

2.5.16 Vyvolání akce přes lokální notifikaci

Pacientovi přijde notifikace a uživatel vyvolá akci s ní spojenou. Popsaný případ je aplikovatelný pro všechny notifikace popsané v tabulce 2.1 doplněním příslušného názvu N_t notifikace a očekávané akce N_a.

2.5.16.1 Předpoklady

Pacient je přihlášen a má notifikaci typu N_t. Nezáleží na tom, zda je aplikace spuštěná, či nikoliv.

2.5.16.2 Hlavní scénář

1. Pacient přistupuje k lokálním notifikacím.
2. Pacient nachází notifikaci typu N_t a kliká na ni.
3. Systém provádí akci N_a.

2.5.17 Vyplnění dotazníku

Pacient vyplní dotazník typu Q_t. Podporované dotazníky jsou uvedeny v tabulce 2.2.

2.5.17.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen.

2.5.17.2 Hlavní scénář

1. Pacient otevírá postranní menu.
2. Pacient kliká na „Dotazníky“.
3. Pacientovi se zobrazuje stránka s dotazníky. Vidí zde všechny dostupné dotazníky pro své pohlaví a u každého z nich datum posledního vyplnění.
4. Pacient kliká na požadovaný dotazník typu Q_t.
5. Pacientovi je zobrazena stránka pro vyplnění dotazníku. Po zodpovězení každé otázky se mu odemkne tlačítko pro přechod k následující otázce.
6. Pacient odpovídá na všechny otázky.
7. Pacient kliká na „Dokončit“.
8. Pacientovi je zobrazen dialog o úspěšném dokončení.
9. Pacient potvrzuje dialog.
   - Pokud má dotazník vyhodnocení:
     a) Pacientovi je zobrazena stránka s vyhodnocením.
     b) Pacient kliká na „Pokračovat“.
10. Stránka se zavírá a pacient je přesměrován na seznam dotazníků.

2.5.17.3 Alternativní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Upozornění“.
3. Pacient kliká na žádost o dotazník typu Q_t.
4. Odehrávají se kroky 5–9 z hlavního scénáře.
5. Stránka se zavírá a pacient je přesměrován na seznam notifikací.

2.5.18 Založení mikčního deníku

Pacient si založí mikční deník.

2.5.18.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient nemá probíhající mikční deník.

2.5.18.2 Hlavní scénář

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Můj mikční deník“.
3. Pacientovi se zobrazuje stránka pro založení mikčního deníku.
4. Pacient si nastavuje délku trvání mikčního deníku.
5. Pacient si vybírá datum začátku deníku.
6. Pacient kliká na „Založit deník“.
7. **Nepovinné**: Pokud pacient nemá obvyklý čas vstávání a usínání:
   a) Zobrazí se modální stránka pro vyplnění obvyklého času vstávání a usínání.
   b) Pacient kliká na čas usínání.
   c) Pacientovi se zobrazuje dialog pro výběr času.
   d) Pacient vybírá svůj obvyklý čas usínání a potvrzuje výběr.
   e) Pacient kliká na čas vstávání.
   f) Pacientovi se zobrazuje dialog pro výběr času.
   g) Pacient vybírá svůj obvyklý čas vstávání a potvrzuje výběr.
   h) Pacient kliká na „Uložit spánkový režim“.
8. Pacient je přesměrován na přehledovou stránku mikčního deníku.

2.5.18.3 Alternativní scénář

1. Pacient si kliká na „Založit deník“ na domovské stránce.
2. Odehrávají se kroky 3–8 z hlavního scénáře.

2.5.18.4 Alternativní scénář 2

1. Pacient si otevírá postranní menu.
2. Pacient kliká na „Upozornění“.
3. Pacient kliká na žádost o deník.
4. Pacientovi se zobrazuje stránka pro založení mikčního deníku s délkou trvání dle volby lékaře.
5. Odehrávají se kroky 4–8 z hlavního scénáře.

2.5.19 Chyby

1. Datum začátku není v budoucnosti.

2.5.20 Potvrzení mikčního deníku

Pacient potvrdí shrnutí mikčního deníku a bude moci začít vyplňovat údaje.

2.5.20.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má rozpracovaný mikční deník, který už může vyplňovat, ale ještě neodsouhlasil shrnutí.

2.5.20.2 Hlavní scénář

1. Pacient kliká na „Pokračovat“.
2. Pacient je přesměrován na detail probíhajícího deníku.

2.5.21 Zrušení nepotvrzeného mikčního deníku

Pacient zruší mikční deník před jeho začátkem.

2.5.21.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má probíhající mikční deník, který ještě nemůže vyplňovat.

2.5.21.2 Hlavní scénář

1. Pacient kliká na „Zrušit deník“.
2. Pacient je přesměrován na domovskou stránku.

2.5.22 Přidání mikce

Pacient přidá mikci do právě rozpracovaného deníku.

2.5.22.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má rozpracovaný mikční deník, u kterého už odsouhlasil shrnutí.

2.5.22.2 Hlavní scénář

1. Pacient kliká na „Přidat močení“.
2. Pacient je přesměrován na stránku nového močení.
3. Pacient má předvolený čas na aktuální hodnotu.
4. Pacient má předvoleno, zda došlo k mikci během spánku, podle aktuálního času a uloženého spánkového režimu.
5. Pacient může upravit čas mikce a to, zda k mikci došlo během spánku.
6. Pacient vybírá, zda došlo k inkontinenci:
   - **Ano** – vybírá, zda před samotnou inkontinencí cítil nucení k močení.
   - **Ne**
     a) Pacient kliká na urgenci.
     b) Pacientovi je zobrazena modální stránka pro výběr urgence z nabídky popsané v tabulce 2.3.
     c) Pacient vybírá požadovanou urgenci.
     d) Pacient kliká na „OK“ a dialog se zavírá.
7. Pacient vyplňuje množství moči.
8. Pacient kliká na „Uložit“.
9. Pacient je přesměrován na detail rozpracovaného deníku.

2.5.22.3 Alternativní scénář

1. Pacient přistupuje k lokálním notifikacím.
2. Pacient nachází notifikaci typu „Nevyplňená ranní mikce“ nebo „Dlouho nevyplněná mikce“ a kliká na ni.
3. Odehrávají se kroky 2–9 z hlavního scénáře.

2.5.23 Chyby

1. Pacient nevyplnil některou z položek popsaných ve scénáři.
2. Čas mikce je v budoucnosti.
3. Čas mikce předchází začátku deníku.

2.5.24 Úprava mikce

Pacient upraví již přidanou mikci.

2.5.24.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má rozpracovaný mikční deník, u kterého už odsouhlasil shrnutí.

2.5.24.2 Hlavní scénář

1. Pacient kliká na „Upravit záznamy“.
2. Pacient je přesměrován na stránku se seznamem chronologicky řazených záznamů, které jsou kategorizovány dle mikčního dne.
3. Pacient kliká na vybranou mikci.
4. Pacient kliká na „Upravit záznam“.
5. Pacient se dostává na stránku mikce s předvolenými hodnotami, které při uložení zadal.
6. Pacient upravuje požadované hodnoty.
7. Pacient kliká na „Uložit“.
8. Pacient je přesměrován na stránku se seznamem záznamů.

2.5.25 Chyby

1. Chyby popsané v případu užití 2.5.22.
2. Mikce bude přesunuta do jiného dne, ale jedná se o poslední mikci aktuálního dne.
3. Mikce naruší blok souvislých denních mikcí nebo nykturií.

2.5.26 Přidání příjmu

Pacient si přidá záznam o příjmu tekutin v rámci právě rozpracovaného deníku.

2.5.26.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má rozpracovaný mikční deník, u kterého už odsouhlasil shrnutí.

2.5.26.2 Hlavní scénář

1. Pacient kliká na „Přidat příjem“.
2. Pacient je přesměrován na stránku nového příjmu.
3. Pacient má předvolený čas na aktuální hodnotu.
4. Pacient vybírá typ nápoje:
   a) Pacient kliká na typ nápoje.
   b) Pacientovi je zobrazena modální stránka pro výběr typu nápoje z nabídky popsané ve funkčním požadavku 13b.
   c) Pacient vybírá požadovaný typ nápoje.
   d) Pacient kliká na „OK“ a dialog se zavírá.
5. Pacient vyplňuje množství vypitých tekutin.
6. Pacient kliká na „Uložit“.
7. Pacient je přesměrován na detail rozpracovaného deníku.

2.5.26.3 Alternativní scénář

1. Pacient přistupuje k lokálním notifikacím.
2. Pacient nachází a kliká na notifikaci typu „Chybějící záznamy o příjmech“.
3. Odehrávají se kroky 2–7 z hlavního scénáře.

2.5.27 Chyby

1. Pacient nevyplnil některou z položek popsaných ve scénáři.
2. Čas příjmu je v budoucnosti.
3. Čas příjmu předchází začátku deníku.

2.5.28 Úprava příjmu

Pacient upraví již přidaný příjem tekutin.

2.5.28.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má rozpracovaný mikční deník, u kterého už odsouhlasil shrnutí.

2.5.28.2 Hlavní scénář

1. Pacient kliká na „Upravit záznamy“.
2. Pacient je přesměrován na stránku se seznamem chronologicky řazených záznamů, které jsou kategorizovány dle mikčního dne.
3. Pacient kliká na vybraný příjem tekutin.
4. Pacient kliká na „Upravit záznam“.
5. Pacient se dostává na stránku příjmu tekutin s předvolenými hodnotami, které při uložení zadal.
6. Pacient upravuje požadované hodnoty.
7. Pacient kliká na „Uložit“.
8. Pacient je přesměrován na stránku se seznamem záznamů.

2.5.29 Chyby

1. Chyby popsané v případu užití 2.5.26.

2.5.30 Zobrazení detailu záznamu již dokončeného deníku

Pacient si zobrazí záznam, který patří k již dokončenému deníku.

2.5.30.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má alespoň jeden dokončený deník.

2.5.30.2 Hlavní scénář

1. Pacient otevírá postranní menu.
2. Pacient kliká na „Grafy a záznamy“.
3. Pacientovi je zobrazena stránka s grafy.
4. Pacient kliká na ikonu archivu v navigační liště.
5. Pacient je přesměrován na stránku se seznamem chronologicky řazených dokončených deníků.

6. Pacient kliká na vybraný deník.
7. Pacient je přesměrován na stránku se seznamem záznamů.
8. Pacient kliká na vybraný záznam.
9. Pacient kliká na „Zobrazit záznam“.
10. Pacient se dostává na stránku detailu záznamu s předvolenými hodnotami, které při uložení zadal.

2.5.31 Zobrazení grafu

Pacient si zobrazí graf typu G_t, z nabídky popsané v tabulce 2.4.

2.5.31.1 Předpoklady

Aplikace je spuštěna a pacient je přihlášen. Pacient má alespoň jeden dokončený deník.

2.5.31.2 Hlavní scénář

1. Pacient otevírá postranní menu.
2. Pacient kliká na „Grafy a záznamy“.
3. Pacientovi je zobrazena stránka s grafy, kde vidí průměrné hodnoty za všechny dokončené deníky.
4. Pacient kliká na „Následující graf“, dokud mu není zobrazen graf typu G_t.

2.5.32 Zobrazení výchozí stránky

Pacientovi je zobrazena výchozí stránka pro aktuální stav aplikace.

2.5.32.1 Předpoklady

Aplikace je vypnuta.

2.5.32.2 Hlavní scénář

1. Pacient spouští aplikaci, do které není přihlášen.
2. Pacientovi je zobrazena přihlašovací stránka.

2.5.32.3 Alternativní scénář

1. Pacient spouští aplikaci, do které je přihlášen, a má rozpracovaný deník.
2. Pacientovi je zobrazena přehledová stránka rozpracovaného deníku.

2.5.32.4 Alternativní scénář 2

1. Pacient spouští aplikaci, do které je přihlášen, a nemá rozpracovaný deník.
2. Pacientovi je zobrazena stránka, na které si může založit deník, vyplnit výchozí dotazník, zobrazit všechny dostupné dotazníky a vidí případné žádosti od lékaře.

2.5.33 Automatické ukončení deníku

Pacientovi je sdělena informace o automatickém ukončení deníku z důvodu neaktivity.

2.5.33.1 Předpoklady

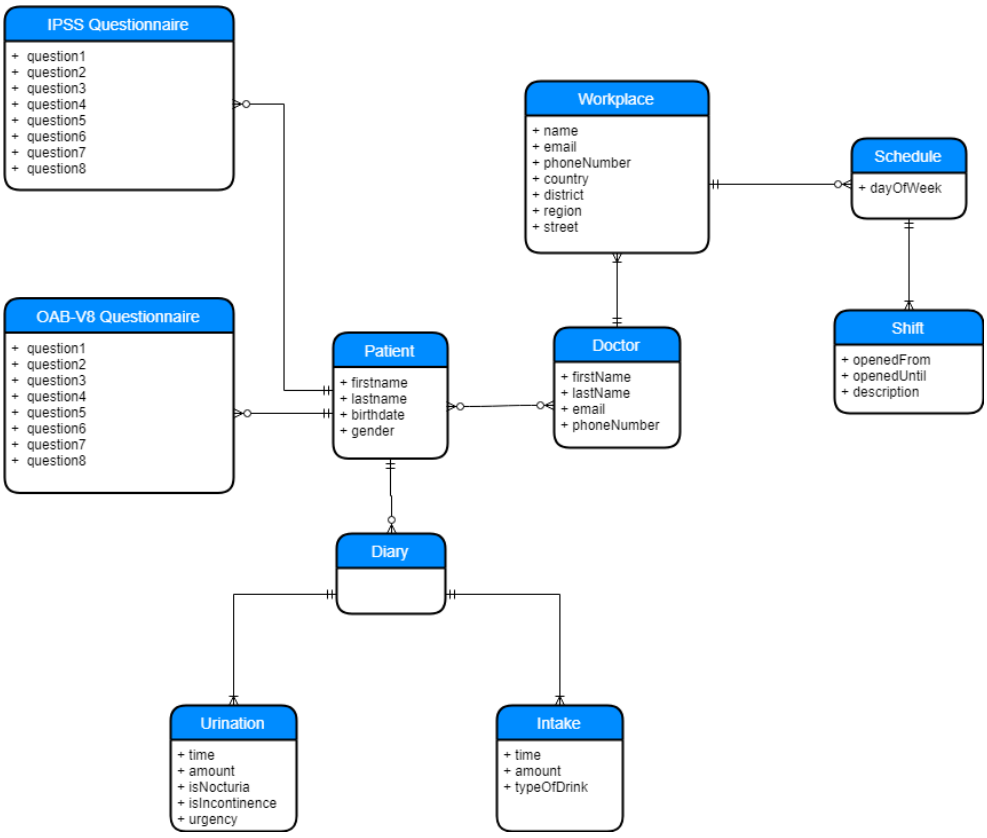
Aplikace je zapnutá a uživatel je přihlášen.

2.5.33.2 Hlavní scénář

1. Pacient porušil některé z pravidel 11f–11g.
2. Pacientovi je ukončen deník.
3. Deník je uložen bez právě probíhajícího dne.
4. Pacient je přesměrován na domovskou stránku.
5. Pacientovi je zobrazen dialog o ukončení deníku.
6. Pacientovi jsou odeslány lokální a aplikační notifikace o ukončení deníku spolu s důvodem.

2.6 Vybudování doménového modelu

Na základě provedené analýzy vznikl doménový model, který můžete vidět na obrázku 2.1.



Obrázek 2.1: Doménový model

Kapitola 3

Technologie

Po dokončení pečlivé analýzy přišel na řadu výběr technologií, které budou v projektu použity. Tento krok musí předcházet návrhu systému a jeho architektury, protože různé platformy a programovací jazyky používají jiná paradigmata.

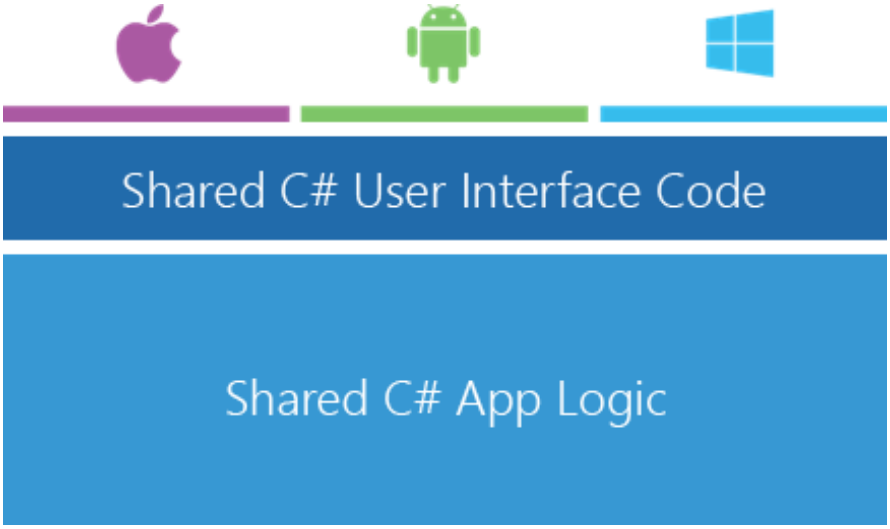
3.1 Xamarin.Forms – iOS a Android

Při vývoji mobilních aplikací vždy docházelo k tvorbě různých verzí v příslušném jazyce pro danou platformu – Java nebo Kotlin pro Android, Objective-C nebo Swift pro iOS a C# pro Windows Phone. V posledních letech se tento přístup do značné míry mění a vývojáři využívají nástroje pro multiplatformní vývoj. Provázanost se systémem, na kterém aplikace běží, se pohybuje na škále od zobrazení lokálního HTML obsahu v prohlížeči po kompilaci a využití nativních komponent zařízení. Aktuálně se mezi nejpopulárnější řadí Xamarin, React Native, Flutter a Cordova.

Xamarin.Forms je open-source C# framework pro tvorbu multiplatformních mobilních aplikací. Projekt byl založen v roce 2011 jako startup a později odkoupen a v roce 2016 uvolněn pro veřejnost společností Microsoft. Nejběžnější způsob pro tvorbu jednotlivých komponent je rozložení na uživatelské rozhraní v XAML a případnou implementaci v C#. Ačkoli je teď Xamarin vlastněn Microsoftem, dialekt použitého XAML se stále mírně liší od těch pro UWP a WPF. Tento framework poskytuje úroveň abstrakce nad nativními komponentami, která je potom za běhu mapovaná právě na příslušné nativní elementy.

3.1.1 Architektura

Xamarin.Forms nabízí perfektní kombinaci sjednocení kódu a možnosti přizpůsobení na jednotlivých platformách. Projekt se skládá z knihovy .NET Standard a dílčích projektů, jejichž hlavním úkolem je zavedení aplikace na



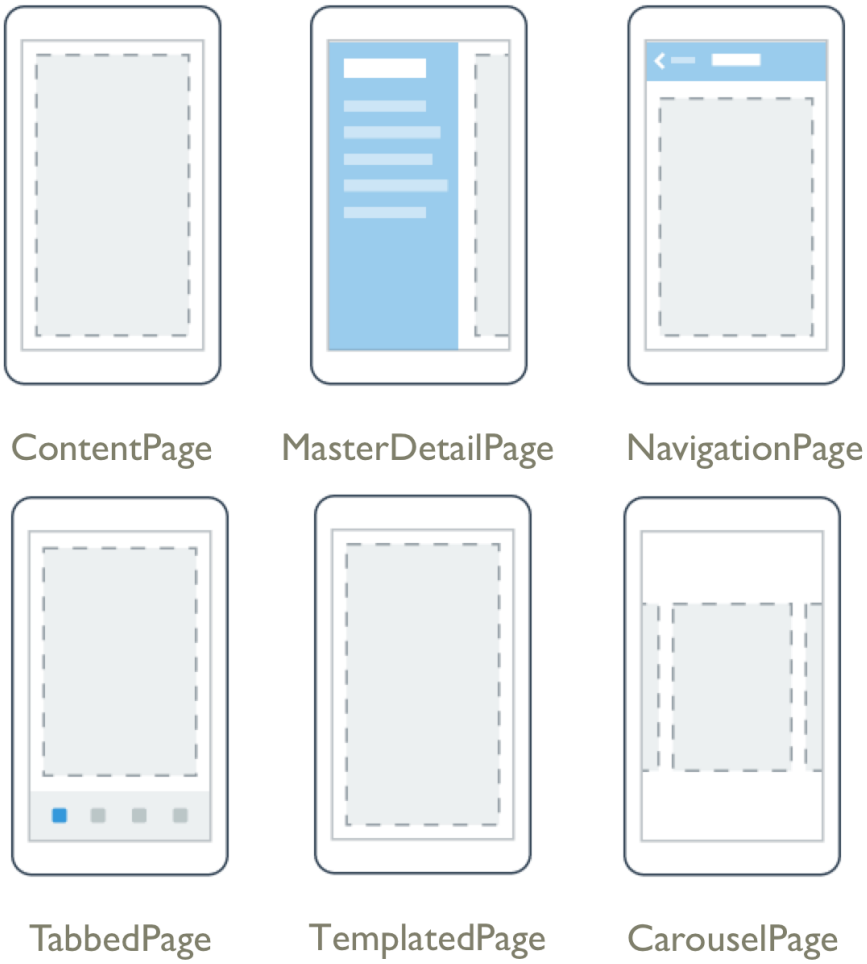
Obrázek 3.1: Architektura aplikace Xamarin.Forms [5]

jednotlivých platformách a případná konfigurace funkcí, jako jsou push notifikace, běh na pozadí aj. Pokud je potřeba přístup k nativním funkcím, které framework ještě nevystavuje, stačí, aby se ve sdíleném projektu definovala podoba rozhraní a následně byla v každém z platformních projektů dodána specifická implementace. Vizualizaci architektury můžete vidět na obrázku 3.1.

Základem interakce s uživatelem jsou stránky. Na iOS je každá stránka mapována na `UIViewController`. Naopak na Androidu sice stránka vypadá jako `Activity`, ale ve skutečnosti tomu tak není a existuje jen jedna sdílená aktivita po celý čas běhu aplikace. Při změně struktury uživatelského rozhraní, ať už způsobené přechodem na jinou stránku, či náhradou nějaké části té stávající, dochází k instanciaci a vykreslení nových komponent a aktualizaci těch stávajících. Vzhledem k tomu, že Xamarin.Forms tvoří vlastně obal a správce nativních elementů, je zřejmé, že část výkonu zařízení padne na jeho režii.

3.1.2 Vykreslování komponent

Komponenty k vykreslení na obrazovce tvoří stromovou strukturu, v jejímž kořeni se vždy nachází stránka. Xamarin.Forms definuje několik typů stránek a jejich přehled můžete vidět na obrázku 3.2. Všechny komponenty, které jsou na stránce zobrazeny, dědí od `VisualElement`, což je základní třída pro všechny komponenty, která vývojáři zpřístupňuje hlavní potřebné atributy a metody pro vykreslení elementu na obrazovce. Každý `VisualElement` má k sobě registrovaný separátní `Renderer` v každém z platformních projektů. Jedná se o most mezi komponentou použitou ve sdíleném kódu a nativním elementem. Stará se o vytvoření a správu elementu. Pokud například změ-



Obrázek 3.2: Stránky poskytované v rámci Xamarin.Forms [6]

níme barvu textu, je to právě daný `Renderer`, jehož úkolem je dostat informaci o změně k nativnímu elementu. Když systém potřebuje vykreslit novou komponentu, vyhledá pro ni registrovaný `Renderer`, vytvoří jeho instanci a požádá ho o inicializaci. Pokud není pro požadovaný typ registrován žádný `Renderer`, systém projde dědičnou hierarchii, než najde typ, pro který je nějaký `Renderer` registrován [23]. Všechny komponenty poskytnuté samotným frameworkem mají svoje výchozí implementace, avšak ty mohou být zaměněny za jiné a tím se docílí změny chování napříč aplikací.

3.1.3 Rozšíření Xamarin.Essentials

Do nedávné doby nebyla interakce s hardwarem a operačním systémem silně integrovaná do frameworku a vývojář byl nucen instalovat si volitelné komu-

nitní balíčky, které vystavovaly jednotné rozhraní. Na konci roku 2018 vyšel oficiální baliček s názvem Xamarin.Essentials, který si klade za cíl poskytnutí přístupu k běžně používaným funkcím telefonu, jako jsou poloha, rotace, připojení k internetu aj.[1] Tento balíček je automaticky instalován při vytvoření nového projektu s Xamarin.Forms.

3.1.4 UI komponenty od společnosti Syncfusion

Syncfusion[2] je společnost zaobírající se vývojem vlastních komponent pro široké spektrum platforem a jednou z nich je právě Xamarin.Forms. Komponenty jsou výrazně přizpůsobitelné a skvěle doplňují ty poskytnuté samotným frameworkem. Knihovny budou využity jak pro záměnu výchozích komponent, tak pro přidání zcela nových jako prostředí pro vykreslování grafů. Pokud dochází k nahrazení nějaké existující komponenty Xamarin.Forms, je tomu tak z důvodu lepší podpory přizpůsobení.[3]

3.2 Úprava binárních výstupů pomocí Fody

V každém jazyce je velké množství kódu, který systém potřebuje pro správné fungování, ale vývojáře to pouze zpomaluje a zároveň to činí kód méně přehledným a udržitelným. Právě tento problém řeší v případě platformy .NET nástroj Fody[4]. Jedná se o rozšiřitelný systém manipulace vygenerovaného IL[5] kódu. Minimalizuje potřebu znát rozhraní nástrojů MSBuild a Visual Studio. V případě potřeby automatizace nějakého úkonu je následně třeba vytvořit si vlastní rozšíření, které provede příslušné manipulace.

3.2.1 PropertyChanged.Fody

Řadí se mezi nejpopulárnější rozšíření nástroje Fody. Stará se o přidání oznámení o změně hodnoty property, aby na ně mohly UI komponenty reagovat a aktualizovat, co je prezentováno uživateli. Nástroj zvládá kontrolu rovnosti, tranzitivní závislosti, a pokud během kompilace nepotřebujeme, aby objekt implementoval `INotifyPropertyChanged`, stačí třídu dekorovat atributem `AddINotifyPropertyChangedInterfaceAttribute`. Jak taková transformace vypadá, můžete vidět na ukázkách 1 a 2.

[1] Kompletní seznam k dispozici na https://docs.microsoft.com/cs-cz/xamarin/essentials.

[2] Dostupné na https://www.syncfusion.com.

[3] Příkladem je tlačítko, jehož výchozí implementace podporuje pouze text a ikonu. Implementace od společnosti Syncfusion umožňuje vykreslit uvnitř libovolný obsah.

[4] Dostupný na https://github.com/Fody/Fody.

[5] Intermediate Language – výstup kompilace, který je za běhu přeložen na strojový kód.

```
internal class MyReactiveObject : INotifyPropertyChanged {
    public int Id { get; set; }
    public int Category { get; set; }
    public string Name => Category + "_" + Id;
    public event PropertyChangedEventHandler PropertyChanged;
}
```

Zdrojový kód 1: Podoba reaktivního objektu před manipulací

```
internal class MyReactiveObject : INotifyPropertyChanged {
    private int _id;
    public int Id {
        get => _id;
        set => {
            if(this._id == value)
                return;
            this._id = value;
            OnPropertyChanged();
            OnPropertyChanged("Name");
        }
    }
    private int _category;
    public int Category {
        get => _category;
        set => {
            if(this._category == value)
                return;
            this._category = value;
            OnPropertyChanged();
            OnPropertyChanged("Name");
        }
    }

    public string Name => Category + "_" + Id;
    public void OnPropertyChanged( [CallerMemberName]
                                    string propertyName = null) {
        PropertyChanged?.Invoke(new PropertyChangedEventArgs(propertyName));
    }
    public event PropertyChangedEventHandler PropertyChanged;
}
```

Zdrojový kód 2: Podoba reaktivního objektu po manipulací

3.3 Perzistence dat v zařízení

Během použití aplikace je třeba alespoň podmnožinu dat ukládat lokálně. Pro tento problém máme možnost využít uložení do uživatelských preferencí nebo data uložit do lokální databáze. Většinou dochází ke kombinaci obojího, protože preference se nehodí pro uložení většího množství dat. Často ale naopak umožňují pohodlnější manipulaci například s informacemi o přihlášeném uživateli, požadovaném jazyce aplikace aj.

3.3.1 Realm

Realm[6] je náhradou za klasické relační databáze. Při jeho návrhu a implementaci bylo hlavním záměrem poskytnout databázový nástroj, který by řešil úskalí vývoje mobilních aplikací. Jednou z výhod tohoto nástroje je jednoduché rozhraní tvořené několika třídami, které respektují zvyklosti daného programovacího jazyka. Bohužel podčást projektu, konkrétně implementace pro .NET, je relativně nová a stále zaostává za ostatními, co se týče podporovaných funkcí. V projektu se pracuje se standardními objekty, které Realm efektivně ukládá do souboru. Samozřejmostí je podpora šifrování dat, pro kterou stačí konfiguraci dodat klíč dlouhý přesně 64 bajtů [24]. Realm je dostupný pro všechny významné jazyky[7] na poli vývoje mobilních aplikací.

Všechny objekty a dotazy nad databází jsou reaktivní. Pokud tedy získáme referenci na objekt, který je v databázi následně upraven, tato změna je promítnuta do reference, kterou jsme už měli. Obdobně pokud iterujeme přes query, její výsledek vždy reflektuje stav v databázi. Zároveň všechny objekty implementují `INotifyPropertyChanged` pro možnost přímého zobrazení uživateli, které se bude automaticky aktualizovat.

Mezi největší nevýhody tohoto nástroje se řadí nízká podpora operátorů LINQ [25], nemožnost uložení časové zóny u typu `DateTimeOffset`[8], chybějící podpora pro `TimeSpan` a `enum`, nemožnost nastavit závislosti pro mazání dat a v neposlední řadě chybějící podpora automatických migrací. Jak se některé ze zmíněných nedostatků dají řešit, můžete vidět na ukázce 3.

[6] Dostupný na https://realm.io.
[7] Aktuálně jsou podporovány Java, Kotlin, Objective-C, Swift, Javascript, .NET.
[8] Při uložení dochází k převodu na UTC čas.

```
1  public enum MyEnum {Value,AnotherValue}
2  public class MyClass {
3      public int MyEnumValueData { get; set; }
4      public MyEnum MyEnumValue {
5          get => (MyEnum)MyEnumValueData;
6          set => MyEnumValueData = (int)value;
7      }
8
9      public long TimeData { get; set; }
10     public TimeSpan Time {
11         get => TimeSpan.FromTicks(TimeData);
12         set => TimeData = value.TotalTicks;
13     }
14
15     private double DateOffset { get; set; }
16     private DateDateOffset DateData { get; set; }
17
18     public DateDateOffset Date {
19         get => new DateDateOffset(DateData.DateTime.AddHours(DateOffset),
20                                   DateSpan.FromHours(DateOffset));
21         set {
22             DateData = value.UtcDateDate;
23             DateOffset = value.Offset.TotalHours;
24         }
25     }
26 }
```

Zdrojový kód 3: Obcházení limitací Realm

Kapitola 4

Návrh aplikace

Jakmile byly vybrány technologie, následoval návrh systému. Nejdříve proběhl návrh uživatelského rozhraní na úrovni tvorby drátěného modelu a jeho interní testování. Poté došlo k tvorbě prototypu a ten byl testován uživateli. Po reflektování připomínek testerů byl vytvořen grafický návrh, který poté sloužil jako podklad pro tvorbu aplikace. Posledním krokem návrhu byla definice API rozhraní, které server vystavuje.

4.1 Návrh uživatelského rozhraní

Prvním krokem návrhu podoby uživatelského rozhraní byla tvorba drátěného modelu. K tomuto účelu byl využit nástroj Balsamiq[9]. Jedná se o interaktivní prototypovací nástroj, který umožňuje následný export do PDF. Do této fáze nebyli zapojeni žádní pacienti a hlavním bodem byly informace od MUDr. Švabíka. Zdrojový soubor je k nalezení v příloze.

Po dokončení drátěného modelu přišlo na řadu interní testování. Některé vybrané nedostatky byly v modelu odstraněny a Kristián Štech ho převedl na klikatelný prototyp. Ten byl testován s vybranými uživateli, během čehož došlo ke zjištění významného množství závad. Zde jsou ty hlavní, seřazené podle závažnosti a počtu výskytů:

1. Pojem „únik bez varování“ byl uživatelům (mimo lékaře) naprosto cizí. Tlačítko s informacemi buď nebylo použito, nebo znění informace vůbec nepomohlo.
2. Velké množství údajů pro registraci, neochota některé z nich vyplnit a zmatečnost registračního formuláře.
3. Přístup ke všem funkcím mimo mikční deník. Menu v navigační liště bylo naprosto ignorováno. V několika ojedinělých případech tam uživatelé hledali nastavení a informace.

[9] Dostupný na https://balsamiq.com.

4. Znění otázky, zda močení proběhlo během spánku.
5. Požadavek na dotazníky po dokončení registrace. Uživatelé nevěděli, co mají dělat.
6. Dlouhé dotazníky a problém se soustředěním.
7. Odhady moči se projevily jako naprosto zcestné.
8. Uživatelé nabývali dojmu, že jejich lékař nutně musí být v aplikaci dostupný.
9. Obnovení hesla a potřeba zadat e-mail, který byl použit při registraci.
10. Nastavení výchozí stránky pro registraci.
11. Při vytvoření deníku v budoucnosti nebyly jasně komunikovány požadavky po celou dobu, která předcházela zahájení deníku.
12. Zobrazení vyhodnocení získaných dat.

Jak lze vidět, nestačilo v tento okamžik provést drobné změny, ale bylo nutné spíše celý návrh přepracovat. S ohledem na množství zpětné vazby byla zvolena cesta přeskočení další iterace návrhu a grafickému studiu se rovnou zadala příprava materiálů. Podoba uživatelského rozhraní byla konzultována s Bc. Cachnínem. Vypracované materiály od studia Finley, a. s., obsahovaly seznam použitých barev, fontů a ikon a ilustrativní vizualizace jednotlivých stránek.

4.2 Datový model

Vzhledem k tomu, že nebyla použita klasická relační databáze, potažmo nějaký ORM nástroj pracující nad ní, databázový model nemáme. Všechny níže popsané třídy dědí od `RealmObject`. To nám umožňuje pracovat s daty v databázi a promítat změny konzumentům. Přestože typicky nepracujeme přímo s property definovanými na `RealmObject`, základní množinu tvoří:

IsManaged – zjištění, zda je tento objekt spravovaný databází;

IsValid – kontrola, zda je tento objekt validní. Pokud objekt odstraníme z databáze, stává se nevalidním a veškerá manipulace s ním skončí výjimkou;

Realm – reference na správce tohoto objektu.

Pro zjednodušení popisu vynechám primární klíče, vazby mezi třídami a pomocné property. Dále vypustím atributy s jasným významem nebo případné komprimace zápisu. Kompletní diagram datového modelu je k nalezení v příloze.

AppUser uchovává data přihlášeného uživatele a obsahuje e-mail a pohlaví.

ExtendedProfile – rozšířená registrace pacienta obsahující jeho identifikační údaje: jméno, příjmení a datum narození.

SessionInfo reprezentuje autentizační informace pro aktuální spojení se serverem. Když dojde k vypršení platnosti přístupového tokenu, dojde k jeho automatickému obnovení pomocí obnovovacího tokenu, který je uložen v zabezpečeném úložišti telefonu.

- **AccessToken** – přístupový token, jenž autentizuje uživatele při komunikaci se serverem.
- **ExpiresAt** – čas, kdy skončí platnost tokenu.

Notification – notifikace, která je uživateli zobrazena v rámci aplikace. Texty notifikací nejsou uloženy v databázi, ale jsou načteny za běhu. Použitím tohoto návrhu je možné neomezeně měnit jazyk aplikace a znění notifikací.

- **Data** – serializovaná data pro práci s notifikací. Ne všechny notifikace musí nutně mít nějaká data.
- **DoctorId** – hodně notifikací je spojeno s lékaři, a proto došlo k umístění identifikátoru lékaře do obecné části. Cílem je vyhnout se deserializaci dat a snížit tak výpočetní náročnost.
- **LocalNotificationId** – pokud byla notifikace publikována do obou kanálů, jedná se o identifikátor lokální notifikace. Důvodem je možnost smazat lokální notifikaci, pokud si ji uživatel zobrazí v aplikaci.
- **Type** – typ notifikace. Je použit pro identifikaci správné obsluhy dané notifikace.

Title – titul, který lékař získal. Nese informaci, zda se nachází před, nebo za jménem, a jeho pořadí v kontextu ostatních titulů.

Doctor – lékař, se kterým pacient sdílí údaje, resp. o sdílení zažádal.

- **ConnectionStatus** – stav spojení pacienta s lékařem.

Workplace – pracoviště, na kterém lékař působí.

- **Name** – název pracoviště. Tento údaj je prezentován pacientům. Příkladem může být „FN Vinohrady“.

DaySchedule – pracovní rozvrh v rámci jednoho dne.

Shift – jedna část pracovní doby.

Address – přesná adresa pracoviště obsahující GPS souřadnice pro určení vzdálenosti od pacienta.

SyncStatus reprezentuje informaci o stavu synchronizace dat. Systém podle toho rozhoduje, zda by měl nahrávat změny na server.

RequestStatus – stav žádosti od lékaře. Slouží k tomu, aby měl uživatel notifikaci v mobilním zařízení označenou za přečtenou, pokud si ji přečte ve webové aplikaci.

QuestionnaireType je použit pro vyhledání odpovídající obsluhy.

QuestionnaireRequest – požadavek od lékaře na vyplnění dotazníku. Zahrnuje typ dotazníku.

DiaryRequest – požadavek od lékaře na vyplnění deníku. Zahrnuje doporučenou délku trvání deníku. Pacient může doporučení ignorovat a délku změnit.

Diary – mikční deník. Spadají sem jak ty dokončené, tak případně právě rozpracované.

- **StartData** – datum začátku.
- **EndDate** – datum plánovaného konce.
- **Length** – plánovaný počet dní.
- **ServerId** – identifikátor záznamu na serveru.
- **ActualEndDate** – opravdový konec deníku. V případě, že byl deník předčasně ukončen, bude se lišit od EndDate.
- **CurrentDay** – pořadové číslo právě probíhajícího dne deníku.
- **SleepTimeOffset** – jedná se o čas, který je přičítán k datu dne pro získání přesné doby usínání.
- **WakeUpTimeOffset** – jedná se o čas, který je přičítán k datu dne pro získání přesné doby vstávání.

DiaryStatus – stav deníku. Určuje zdali deník právě probíhá, resp. způsob jeho dokončení.

DiaryDay – jeden den deníku, který obsahuje k němu vázané záznamy. Mimo jiné kvůli optimalizaci výpočetní náročnosti sleduje a aktualizuje informace o první a poslední mikci.

- **DayNumber** – pořadové číslo dne. Pro poslední den odpovídá hodnotě CurrentDay u deníku.
- **SleepTime** – předpokládaný čas usínání pacienta v tento mikční den.

WakeUpTime – předpokládaný čas vstávání pacienta v tento mikční den.

Intake – záznam příjmu tekutin.

Urination – záznam mikce. Při zadání mikce dochází k transformaci dat pro uložení ve stejné struktuře, ve které jsou data následně odeslána na server.

4.3 Definice REST API

Mobilní aplikace komunikuje se serverem pomocí REST API. Při vývoji byla využita platforma Apiary[10], která umožňuje definovat podobu rozhraní a zároveň se chovat jako prostředník pro snazší ladění. Kompletní definice rozhraní ve formátu pro Apiary je v příloze.

4.3.1 Synchronizace

Pro synchronizaci dat mezi serverem a klientem jsou využity hlavičky `If-Modified-Since` a `Last-Modified`. Při vůbec prvním dotazu na server klient neodesílá žádnou synchronizační hlavičku. Při obdržení odpovědi ze serveru si uloží hodnotu `Last-Modified` a tu při příštím volání odesílá pod `If-Modified-Since`.

4.3.2 Lokalizace

Server podporuje hlavičku `Accept-Language`, dle které lokalizuje data a zprávy pro klienta.

4.3.3 Autentizace

Pro ověření původu požadavků se používá hlavička `Api-Key`, která je odesílána spolu se všemi požadavky na server. Hodnota se v čase nemění a je pro všechny klienty stejná. Důvodem její existence je jednoduchá filtrace případných podvodných nebo útočných požadavků na server.

Pro autorizaci uživatelů se jako autorizační token používá JWT (Json Web Token). Jedná se o čitelný a verifikovatelný formát, do nějž můžeme zahrnout libovolné dodatečné informace. Při registraci nebo přihlášení je uživateli vrácena následující autentizační struktura:

access_token – přístupový token, který je zasílán serveru s každým požadavkem;

expires_at – čas vypršení platnosti tokenu;

[10] Dostupná na https://apiary.io.

refresh_token – token pro získání nového přístupového tokenu. Slouží pro udržení spojení se serverem a zvýšení zabezpečení.

Po vypršení platnosti autentizačního tokenu dochází k jeho obnovení. Klient serveru zasílá obnovovací token a zpět dostává identickou autentizační strukturu jako při přihlášení, resp. registraci. Jak je zřejmé, obnovovací token se mění. Je tomu tak kvůli zvýšení zabezpečení. Pro případ, že by došlo k jeho zcizení útočníkem, je jeho platnost omezená.

Kapitola 5

Architektura a implementace

V tuto chvíli následovala příprava architektura systému a samotná implementace. Během definice architektury nedocházelo k typické tvorbě diagramu třídní struktury, ale postupovalo se návrhem podoby rozhraní přímo v aplikaci Visual Studio. S vylepšeními, které se objevují ve vývojových prostředích považují toto za krok správných směrem. Nejenom, že je to pro většinu vývojářů mnohem příjemnější, ale zároveň se nám dostává ekvivalentního výstupu, zatímco šetříme čas a náklady.

5.1 MVVM architektura

Nejpoužívanější architekturou v Xamarin.Forms je Model–View–ViewModel. Rozdělení částí je následující.

Model je datový model, který reprezentuje podobu dat v perzistentním úložisti.

View je grafické rozhraní, které je prezentováno uživateli a ten s ním může interagovat. View může být celá stránka nebo jen její podčást. Jednotlivé komponenty vystavují `BindableProperty` property, které mohou být následně navázány na hodnoty z view modelu.

ViewModel je srdce této architektury. Každé view má svůj view model. Ten slouží jako adaptér mezi částmi View a Model. Jeho nejčastějším úkolem je transformovat data a implementovat požadované akce, které má uživatel přístupné.

Všechny view modely v projektu dědí od třídy `BaseViewModel`, která implementuje `IBaseViewModel` a `INotifyPropertyChanged`. Tato rozhraní nám zajišťují podporu validace dat a publikaci změn dat ke konzumentům.

5.1.1 Zpracování akcí

Běžně se pro práci s akcemi vyvolanými uživatelem aplikace používají příkazy vystavené view modelem. V tomto projektu došlo k implementaci rozhraní `ISmartCommand`, které implementuje `ICommand`, takže je plně kompatibilní se systémem a zároveň nám poskytuje následující možnosti:

- předat odkaz na synchronní či asynchronní funkci s typovaným vstupním parametrem, která bude zavolána v případě požadavku na vykonání příkazu. Implementace zohledňuje to, že není možné vykonat příkaz, který již probíhá;
- zařazení příkazu do skupiny. V rámci jedné skupiny může v danou chvíli probíhat pouze jeden příkaz. Toto je zapotřebí, aby se předešlo nechtěným stavům, kdy uživatel vyvolá dvě různé akce najednou.

Do budoucna by bylo vhodné rozšířit implementaci tak, aby byla schopná dynamicky reagovat na změny dat view modelu a příslušně upravovat, zda může být daný příkaz vykonán. Pro jednoduchost bylo stejného efektu dosaženo navázáním podmínek na property `IsEnabled` jednotlivých komponent uživatelského rozhraní.

5.2 Validace

Většina vstupů od uživatele vyžaduje validaci. V rámci tohoto projektu byl navržen lehce rozšiřitelný model validace, který je mimo jiné aplikovatelný na vstupní a výstupní parametry metod, návratové hodnoty, objekty aj.

IValidationRule je základní stavební prvek celého validačního systému. Pravidlo reprezentuje požadavek na data, který musí být splněn. Validační metoda přijímá kromě dat samotných také kontext, ve kterém se data nachází, a dodatečné parametry pro validaci. Kontextem může být cokoliv od objektu, na kterém je property definována, po HTTP požadavek. Při implementaci vlastního pravidla stačí oddědit od třídy `ValidationRule` a poskytnout vlastní implementaci pro generickou variantu rozhraní `IValidationRule`. Na ukázce 4 můžete vidět implementaci validace délky řetězce. V rámci systému byla implementována pravidla pro validaci hodnoty booleanu, délky řetězců, toho, zda je datum v minulosti, či budoucnosti, a mnoho jiných.

IValidationRuleProvider slouží ke získání instance validačního pravidla na základě typu.

IValidationDescriptor definuje potřebné informace pro validaci. Aktuálně ho implementuje `ValidationDescriptorAttribute` pro podporu validace objektů pomocí atributů.

```
1  public interface IStringLengthParams {
2      int MinLength { get; }
3      int MaxLength { get; }
4      NullStringHandling NullHandling { get; }
5  }
6  public class StringLengthRule : ValidationRule<string,
7                                                  IStringLengthParams,
8                                                  object> {
9      public override bool Validate( string value,
10                                    IStringLengthParams @params,
11                                    object context ) {
12         if( @params.NullHandling.ValidityKnown(ref value, out var valid) )
13             return valid;
14         return value.Length >= @params.MinLength &&
15                value.Length < @params.MaxLength;
16     }
17 }
```

Zdrojový kód 4: Implementace validace délky řetězců

ErrorCode je identifikátor chyby, pokud není pravidlo splněno.

Params jsou parametry, které jsou předány pravidlu v době vyhodnocení.

Rule jde o typ pravidla pro vyhodnocení validity.

Severity značí, zda se jedná o chybu, či pouhé varování.

Targets určuje, zda by mělo dojít k aplikaci pravidla na data samotná, nebo jednotlivé elementy v případě, že se jedná o kolekci. Zde je možné zvolit obojí.

Conditions zahrnuje podmínky, jaké jsou kladeny na cíl, aby došlo k vyhodnocení pravidla. Aktuálně je podporována možnost `NoError`, tedy předchozí validace nedopadla chybou.

IValidationOption slouží pro řízení validace objektů. Aktuálně podporuje nastavení rekurzivního průchodu objekty, což může být nastaveno na procházet – procházet, pokud nedošlo k chybě – neprocházet.

IErrorMessageProvider slouží ke generování chyb na základě poskytnutého kódu.

IErrorMessageCoercer – poté co obdržíme chybovou zprávu, může být potřeba ji ještě upravit. Dochází zde k manipulaci s textem. Při manipulaci jsou k dispozici stejná data, jako má validační pravidlo.

IErrorMessageCoercerProvider slouží pro registraci a získání `IErrorMessageCoercer`. Registrace je závislá na kódu chyby a případném jazyce. Pokud jazyk není při registraci specifikován, je daný `IErrorMessageCoercer` poskytnut při všech požadavcích.

IValidator slouží k validaci dat. Vystavuje několik přetížení metody `Validate`. Obecně platí, že mu volající předá data, která chce validovat, a případný externí seznam `IValidationDescriptor`. Validátor provede příslušné vyhodnocení. Poskytnutá implementace zvládá procházet objektový graf, pracovat s kolekcemi aj. Návratovou hodnotou je vždy seznam chyb, které nastaly.

IValidatableObject je základ objektů, které podporují dynamickou validaci a jsou si toho vědomy.[11]

IValidatableProperty je wrapper okolo vlastního typu dat, která chceme validovat. Umí provádět automatickou validaci při změně hodnoty a implementuje `INotifyPropertyChanged`. Typicky se vyskytuje právě na view modelu.

IValidationDependency mapuje validační závislosti mezi různými property. `IValidatableObject` sleduje změny v hodnotách, na kterých existují závislosti, a provádí případnou revalidaci.

IDependencyScanner vyhledává validační závislosti v rámci objektu.

IPropertyNameProvider je použit pro získání názvu property při generování chyby.

IPropertyInitializer prochází a inicializuje všechny instance `IValidatableProperty` na objektu implementujícím `IValidatableObject`.

Na ukázce 5 můžete vidět, jak do sebe všechno zapadá a zároveň výrazně automatizuje práci vývojáře.

5.3 Lokalizace

S ohledem na požadavek lokalizace bylo potřeba mít veškeré texty ve vícejazyčné podobě. Práci s texty zjednodušilo rozšíření ResXManager[12]. Jedná se o nástroj pro správu textů, které jsou připojeny jako embedované zdroje. Vývojář má přehled o podporovaných jazycích, má možnost exportu a importu pro předání ke korektuře aj.

[11] Pozor, neplést si s možností validac! Data nemusí implementovat toto rozhraní, aby mohla být validována.

[12] Dostupné na https://marketplace.visualstudio.com/items?itemName=TomEnglert.ResXManager.

```
1  public class SignInViewModel : BaseViewModel {
2      public SignInViewModel( IValidationConfiguration conf ) : base(conf) {
3          SignIn = new SmartCommand<object>(ExecuteSignInAsync);
4      }
5      [DataNotNull(1)]
6      [Email(2, Order = 1, Conditions = ExecutionConditions.NoError)]
7      public IValidatableProperty<string> Email { get; set; }
8
9      [StringLength(3, MinLength = 6, MaxLength = 107)]
10     public IValidatableProperty<string> Password { get; set; }
11
12     public ISmartCommand SignIn { get; }
13
14     public void ExecuteSignInAsync(object arg){
15         // nechceme automaticky validovat před prvnim vyvolanim akce
16         AutomaticallyValidate = true;
17         if(HasErrors)
18             return;
19         //kód pro přihlášení
20     }
21 }
```

Zdrojový kód 5: View model s validací

ILocalizationProvider slouží jako zdroj lokalizovaných podkladů. Nemá žádné povědomí o jazyce aplikace apod. Disponuje pouze jednou metodou, která na základě klíče a požadovaného jazyka vrátí lokalizovaný objekt. Jedná se o generickou třídu, která může pracovat s čímkoliv od řetězců po obrázky. V projektu byla dodána implementace, která vyhledává data v embedovaných zdrojích.

ILocalizationService je hlavní třída pro lokalizaci dat v aplikaci. Zapouzdřuje `ILocalizationProvider` a přidává nastavení jazyka.

IThreadCultureBinder slouží k manipulaci jazyka vláken při změně jazyka `ILocalizationService`. `ILocalizationService`.

IStringLocalizar zapouzdřuje `ILocalizationService` pro přímočařejší využití.

TranslateExtension představuje rozšíření pro lokalizaci v XAML. Vrací hodnotu lokalizovanou implementací `ILocalizationService` dle aktuálního jazyka aplikace. Ukázka 6 demonstruje lokalizaci v XAML.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:localization="clr-namespace:Lib;assembly=SrcLib"
             x:Class="App.SimplePage">
    <Button Text="{localization:Translate Actions:ShowNotification"
            Command="{Binding ShowNotification}"/>
</ContentPage>
```

Zdrojový kód 6: Lokalizace v XAML

5.4 Navigace v aplikaci

Typickým problémem, který musíme řešit u jakéhokoliv typu aplikace, je navigace. Právě z toho důvodu vznikl `INavigatorBase` a okolní rozhraní. Hlavní prioritou byla jednoduchost konzumace tohoto rozhraní. Uživatel pouze odesílá požadavky na navigaci a informaci, jaké komponenty se týkají, zbytek zařídí systém sám.

IHost je obal okolo prostředku, který bude reagovat na navigační požadavek. V aplikaci jsou aktuálně dvě implementace: jedna obaluje rozhraní od frameworku `INavigation` a druhá pracuje nad `MasterDetailPage`.

IBindingBehavior provádí samotnou registraci hostitelů pro požadované komponenty. Různé implementace rozhodují o tom, jaké je požadované chování při pokusu o registraci hostitele pro komponentu, pro kterou již registrace existuje. Aktuálně jsou podporované režimy zachovej první, použij novou a vyhoď výjimku.

IHostBinder uchovává všechny existující registrace hostitelů.

INavigatorBase definuje metody pro obousměrnou navigaci. Vzhledem k existenci systémů, kde bude typicky pouze jedna komponenta[13], je možné vynecháním identifikátoru komponenty využít takzvané zjednodušené navigace. `IPageNagivator` dědí toto rozhraní a jeho implementace přidává parametry navigace potřebné pro mobilní zařízení. Příklad registrace hostitele a navigace můžete vidět na ukázce 7.

- Dopředná navigace vyžaduje název komponenty a stránku, která by měla být prezentována. Mezi volitelné spadá způsob zobrazení a to, zda se jedná o nový kořen. Aktuálně podporované způsoby zobrazení jsou buď přes celou stránku, nebo v rámci aplikace,
- Zpáteční navigace podporuje počet stránek k odstranění a případně informaci, zda by mělo být odstraněno vše až ke kořeni.

[13] Mobilní aplikace je skvělým příkladem.

```
1  public partial class MainPage {
2      private readonly IPageNavigator _navigator;
3      private readonly MasterDetailHost _host;
4      private readonly IHomePageFactory _factory;
5
6      public MainPage( IPageNavigator navigator,
7                       SidebarPage sidebarPage,
8                       IHomePageFactory homePageFactory ) {
9          _navigator = navigator;
10         _host = new MasterDetailHost(this, true);
11         _factory = homePageFactory;
12
13         InitializeComponent();
14
15         Master = sidebarPage;
16         Detail = new NavigationPage(homePageFactory.Create());
17     }
18
19     protected override void OnAppearing() {
20         base.OnAppearing();
21         _navigator.HostBinder.Bind(_navigator.DefaultComponent, _host, null);
22     }
23
24     public async void ReloadHomePage(){
25         var @params = new PageNavigationParams(true);
26         await _navigator.NavigateToAsync(homePageFactory.Create(), @params);
27     }
28 }
```

Zdrojový kód 7: Registrace navigace v aplikaci

5.5 Perzistence dat

Každou entitu uloženou v databázi spravuje repozitář. Sdílená definice `IRealmRepository` definuje základní CRUD operace, které kopírují ty dostupné na třídě `Realm`. Mimo to přidává různé doplňky pro zkrácení zápisu. Především se jedná o podporu filtrovacích výrazů a filtrování již materializované kolekce[14]. Všechny repozitáře používají pro získání instance databáze `IRealmProvider`. Na ukázce 8 můžete vidět částečnou definici `RealmRepository`.

Část dat byla uložena mimo Realm. Konkrétně to byla data přihlášeného

[14] Realm nepodporuje značné množství operací, a tak je nejdříve potřeba zhmotnit kolekci v paměti a až poté filtrovat. Při zhmotnění do paměti Realm načte pouze obal, a ne data samotná.

```
public class RealmRepository<TEntity> : IRealmRepository<TEntity>
                                        where TEntity : RealmObject {
    public RealmRepository( [NotNull] IRealmProvider provider ) {
        Provider = provider;
    }
    protected IRealmProvider Provider { get; }

    public TEntity Find( Expression<Func<TEntity, bool>> selector ) {
        return Provider.Instance.All<TEntity>().SingleOrDefault(selector);
    }
    public IEnumerable<TEntity> MaterializedAll( Func<TEntity, bool> filter ) {
        return Provider.Instance.All<TEntity>().ToList().Where(filter);
    }
    public virtual void RemoveRange( IEnumerable<TEntity> items ) {
        var realm = Provider.Instance;
        realm.Write(() => {
            foreach( var item in items )
                realm.Remove(item);
        });
    }
}
```

Zdrojový kód 8: Zkrácená definice repozitáře

uživatele a autentizační údaje pro spojení se serverem. Z povahy věci byl obnovovací token uložen v zabezpečeném úložišti pomocí `SecureStorage`[15]. Implementace je na jednotlivých platformách následující.

Android – data jsou šifrována blokovou šifrou AES. Klíč je na novějších verzích Androidu uložen v KeyStore. Starší verze KeyStore podporovaly pouze RSA klíče, a tak dochází ke generování AES klíče za běhu, jeho zašifrování dodaným RSA klíčem a k následnému uložení ve sdílených preferencích. Data jsou ve všech verzích Androidu uložena ve sdílených preferencích.

iOS – data se vždy nacházejí v úložisti KeyChain [26].

5.6 Vlastní komponenty

V rámci projektu bylo vytvořeno několik vlastních komponent. Typicky dědí od `View`, `TemplatedView` nebo `ContentView`. Některé komponenty nemají

[15] Jedna z komponent balíčku Xamarin.Essentials.

žádnou další implementaci a slouží pouze ke zjednodušení globálního stylování. U komponent uživatelského rozhraní je všechen kód součástí třídy a není zde použit view model. Většina vlastností se nastavuje pomocí deklarovaných properties.

DataView umožňuje vykreslit jakýkoliv objekt za pomoci `DataTemplate` nebo `DataTemplateSelector`.

DatePicker je komponenta pro výběr data. Je možné omezit výběr dostupných let. Zobrazuje aktuálně vybranou hodnotu a při kliknutí otevírá dialog pro úpravu výběru.

TimePicker představuje komponentu pro výběr času. Zobrazuje aktuálně vybranou hodnotu a při kliknutí otevírá dialog pro úpravu výběru.

GenderPicker zobrazuje všechna dostupná pohlaví, která mají ikonu a název.

IconText se skládá ze dvou instancí třídy `Label`. Jedna používá jako font ten s ikonami. Umožňuje nastavit rozložení, vzájemnou vzdálenost, směr uspořádání aj.

HtmlWebView vykresluje obsah lokálních souborů do `WebView`. Je použit pro informace o aplikaci a podmínky užití.

QuestionnairePresenter zobrazuje ovládací strukturu vyplňovaného dotazníku – aktuální stav, navigační tlačítka a především aktuální otázku.

QuestionView je abstraktní třída pro jednotlivé reprezentace různých typů otázek.

SingleChoiceQuestionView rezprezentuje otázku s jednou možností odpovědi. Aktuálně implementované dotazníky využívají toto zobrazení pro všechny otázky.

5.7 Notifikace

Práce s notifikacemi probíhá pomocí sjednoceného rozhraní `INotificationService`. Zde je možné posílat požadavky na zobrazení notifikací do libovolného kanálu, odebírat aplikační notifikace a sledovat počet nepřečtených notifikací, ať už všech, či jen těch spojených s lékařem. Vzhledem k tomu, že dochází k manipulaci s uživatelským rozhraním, je zapotřebí akce delegovat na hlavní vlákno.

INotificationRequest představuje požadavek na zobrazení notifikace do libovolného z kanálů:

DoctorId – ID lékaře v případě, že je notifikace spjatá s lékařem.

Type – typ notifikace pro správné načtení a obsluhu.

Channels – do jakých kanálů má být notifikace publikována.

DisplayAt – kdy by měla být notifikace zobrazena. Pokud není vyplněno, dojde k okamžitému zobrazení.

CreatedAt – kdy byla notifikace vytvořena. Když zobrazujeme například žádost od lékaře, je potřeba zobrazit datum žádosti, a nikoliv synchronizace.

INotificationLoader je rozhraní definující dvě metody. Jedna slouží k načtení notifikace pro lokální zobrazení a ta druhá pro zobrazení v rámci operačního systému.

INotificationHandler představuje rozhraní pro interakci s notifikacemi. Umožňuje notifikacím reagovat na aktivace (kliknutí), když byly uživatelem zobrazeny.

INotificationService je služba umožnující správu notifikací. Stará se o zobrazení a interakci s notifikacemi. Když uživatel s nějakou interaguje, je o tom `INotificationService` zpraven, vyhledá příslušnou obsluhu pro daný typ notifikace a deleguje reakci na interakci s ní.

AppNotifications je reaktivní kolekce notifikací ke zobrazení v aplikaci.

UnreadNotificationsCount je počet nepřečtených notifikací.

UnreadDoctorNotificationsCount představuje počet nepřečtených notifikací spojených s lékařem.

5.8 Dialogy

Nedílnou součástí mobilní aplikace je práce s dialogy. Obecně se jedná o komponenty pro sdělení informace uživateli, potažmo získání informace od uživatele, které jsou vykresleny přes aktuální obsah aplikace. Dialogy jsou použity především pro:

- zobrazení informace o průběhu akce,
- zobrazení výsledku akce,
- získání dodatečných informací o mikci,
- výběr konfigurace, kterou má aplikace použít.

Xamarin.Forms nevystavuje jednotné rozhraní pro práci s dialogy, a tak byla použita knihovna ACR User Dialogs[16]. Nabízí veškeré potřebné modely:

[16] Dostupné na https://github.com/aritchie/userdialogs.

zobrazení informace uživateli, získání hesla od uživatele pro změnu konfigurace[17] a hlášení o průběhu akce, jako je přihlášení do aplikace.

5.9 Synchronizace

Veškerá synchronizace pacientových dat probíhá periodicky na pozadí. Při vytváření záznamů na zařízení je jim přiřazeno GUID a po odeslání na server je jim serverem přiřazen další identifikátor pro jeho potřeby. V Androidu je k provedení synchronizace použit `JobScheduler` a v případě iOS technika background fetch. Obojí nechává systému volnou ruku, aby úlohy spouštěl ve vhodných chvílích pro optimalizaci využití zdrojů systému a dosáhl s tím spojené úspory energie.

ISynchronizer je základní stavební blok celého synchronizačního systému. Poskytuje metodu `Synchronize`[18], která vrací počty přidaných, upravených a smazaných objektů. Každý synchronizátor má prioritu, kterou může jeho volající zohledňovat.

ISynchronizationGroup představuje synchronizační skupinu, která jednotlivé sychronizátory spojuje do logických svazků. Sama implementuje rozhraní `ISynchronizer`, a je tedy možné je libovolně vnořovat. Aktuálně je v aplikaci definována pouze jedna skupina a její implementaci můžete vidět na ukázce 9.

SyncStatus – status lokálního záznamu ve spojení se synchronizací. Stavy jsou následující:

- **New** – záznam byl vytvořen na zařízení a ještě nebyl odeslán na server.
- **Modified** – záznam byl upraven na zařízení a jeho starší podoba existuje na serveru.
- **Synchronized** – záznam je aktuální.

IsolationLevel – při synchronizaci běžně chceme, aby chyba v jedné části neovlivnila běh celého systému. Úrovně izolace vyjadřují, jaké výjimky by měly být odchyceny a zpracovány bez dalších projevů. Aktuálně podporované jsou tyto úrovně:

- **None** – výjimky jsou propagovány dále.
- **SynchronizationGroup** – výjimka v rámci jednoho synchronizátoru ukončí procesování fronty a dojde k navrácení mezivýsledků.
- **Synchronizer** – všechny synchronizátory jsou plně izolovány a celý proces pokračuje.

```
public async Task<ISynchronizationResult> Synchronize() {
    var results = new List<ISynchronizationResult>();
    try {
        var synchronizers = _synchronizers.ToList();
        foreach( var synchronizer in synchronizers ) {
            try {
                results.Add(await synchronizer.Synchronize());
            } catch {
                if( IsolationLevel < IsolationLevel.Synchronizer )
                    throw;
            }
        }
        return results.MergeResults();
    } catch {
        if( IsolationLevel < IsolationLevel.SynchronizationGroup )
            throw;
        return results.MergeResults();
    }
}
```

Zdrojový kód 9: Implementace výchozí synchronizační skupiny

Aktualizovány jsou deníky, dotazníky, lékaři, požadavky od lékařů a profil uživatele.

5.10 Komunikace s API

Za běhu aplikace je nutné komunikovat se serverem skrze jeho API. Ke zjednodušení práce byl vytvořen `IHttpApiClient`, který umožňuje:

- vytvářet požadavky k odeslání,
- autentizovat požadavky těsně před jejich odesláním,
- obnovovat platnost autentizačního systému,
- upravovat požadavky před jejich odesláním,
- deserializovat odpovědi serveru do požadované podoby. Skládá se z deklarace požadovaného datového typu těla odpovědi a hlaviček.

[17] Změna konfigurace je prvek pro vývojáře a testery. Z tohoto důvodu je potřeba zadat heslo, aby nedošlo k mylné úpravě uživatelem.

[18] Mělo by dojít k přejmenování metody na `SynchronizeAsync`

IAuthenticator autentizuje jednotlivé požadavky na server. Dokáže rozhodnout, zda požadavek vyžaduje autentizace a jestli je zapotřebí obnovit platnost před autorizací následujícího požadavku.

IRequestPreprocessor přijímá požadavek, který následně libovolně transformuje. Aktuálně implementované preprocesory se starají o ověření stavu připojení k internetu a vložení hlavičky požadovaného jazyka. API klíč je vkládán jako výchozí hlavička pro všechny požadavky.

IBodySerializer se stará o serializaci dat odesílaných na server. Aktuální implementace používá Newtonsoft.JSON[19]. V budoucnosti dojde k nahrazení statické deserializace těla odpovědi abstrakcí `IBodyDeserializer`.

IHeaderParser provádí deserializaci hlaviček odpovědi a vrací požadovaný datový typ. Při instanciaci klienta dochází k registraci potřebných parserů.

IHttpApiClientProvider slouží pro získání instance API klienta, se kterým by měl konzument pracovat. Umožňuje recyklaci pro znovunačtení aktuální konfigurace.

IHttpApiClient je použit pro veškerou komunikaci s API. Ukázka 10 demonstruje implementaci odeslání požadavků.

V aplikaci jsou definováni různí klienti, kteří obstarávají tematickou část API (autentizace, deníky, dotazníky, ...). Všechny implementace využívají `IHttpApiClient` k odesílání požadavků skrze závislost na `IHttpApiClientProvider`.

5.11 Vkládání závislostí

S návrhem architektury systému je neoddělitelně spojena tvorba závislostí mezi jednotlivými komponentami. Jedním z aspektů udávajících, jak lehké je spravovat, upravovat a rozšiřovat kód, je to, jak volně jsou jednotlivé komponenty svázané. Volné svázání nám udává, jaké množství informací má jedna komponenta o jiné. Čím více závislostí nějaká třída má, tím těžší je její znovupoužití, pochopení a udržování. Obecně platí, že nechceme tvořit závislosti mezi komponentami, a jedním ze způsobů, jak snížit provázanost, je abstrakce požadavků a vkládání závislostí.

Ilustrativní příklad, který si teď rozebereme, je logování. Komponenta ví, že bude potřebovat zapisovat informace o průběhu vykonávání svých funkcí, ale už vůbec by nemělo být v její kompetenci rozhodovat o tom, kam budou informace zapsány. To je úkolem konzumenta této komponenty. Zápis informací má jasně danou strukturu a to, zdali jsou data zapisována do konzole, souboru,

[19] Dostupný na https://www.newtonsoft.com/json.

```
1  public async Task<IResponse<THeaders, TBody>> SendAsync<THeaders, TBody>
2                                                  ( HttpRequestMessage request )
3      where TBody : class, IResponseBody
4      where THeaders : class, IResponseHeaders {
5
6      var requiresAuth = Authenticator.RequiresAuthentication(request);
7      if( requiresAuth && Authenticator.RequiresRefresh ) {
8          await Authenticator.RefreshAsync(this);
9      }
10     if( requiresAuth ) {
11         Authenticator.Authenticate(request);
12     }
13     var response = await Client.SendAsync(request);
14     var headers = ParseHeaders<THeaders>(response.Headers,
15                                          response.Content.Headers);
16
17     var content = await response.Content.ReadAsStringAsync();
18     var body = JsonConvert.DeserializeObject<TBody>(content);
19
20     return new Response<THeaders, TBody>
21                (response.StatusCode, headers, body, null);
22 }
23
24 public HttpRequestMessage CreateRequest( HttpMethod method,
25                                          string requestUri,
26                                          object body ) {
27
28     var request = new HttpRequestMessage(method, requestUri);
29
30     foreach( var preprocessor in Preprocessors )
31         preprocessor.Process(request);
32
33     Serializer.AddBody(request, body);
34     return request;
35 }
```

Zdrojový kód 10: Odeslání požadavku na server

databáze, či jen zahozena, je irelevantní. V tomto případě tedy můžeme definovat rozhraní, které nám umožní zapsat informace, a následná implementace této abstrakce může být lehce zaměněna bez ovlivnění ostatních částí systému. Různé možnosti implementace logování můžete vidět na ukázce 11.

Jedním z důležitých rozhodnutí je volba úrovně abstrakce. Všichni budou souhlasit, že mít závislost na řetězci je v pořádku a že ho není třeba abstra-

```
1  public interface ILogger {
2      public void Log( string message );
3  }
4  public class FileLogger : ILogger {
5      private readonly string _fileName;
6      public FileLogger(string fileName) {
7          _fileName = fileName;
8      }
9      public void Log( string message ){
10         File.AppendAllText(_fileName, message);
11     }
12 }
13 public class NullLogger : ILogger {
14     public void Log( string message ){}
15 }
16 public class MyComponent(){
17     private readonly ILogger _logger;
18     public MyComponent( ILogger logger ){
19         _logger = _logger;
20         _logger.Log("Created an instance of " + nameof(MyComponent));
21     }
22 }
```

Zdrojový kód 11: Abstrakce logování

hovat. Ačkoli abstrakce přináší velké množství výhod, nesmíme zapomínat, že s sebou přináší náklady v podobě režie. Proto dělíme závislosti do dvou kategorií [27]:

stabilní – klasicky typy, na které můžeme tvořit přímé závislosti. Jedním z příkladů je Base Class Library[20]. Výběrová kritéria by byla následující:

- nové verze nepřinesou nekompatibilní změny;
- typy obsahují deterministické algoritmy. Jedním z důvodů tohoto požadavku je, že testování komponenty, která závisí například na třídě `Random`, je problematické. Lepším přístupem by bylo provedení abstrakce a následně v rámci testování poskytnutí deterministické implementace;
- neexistuje předpoklad nutnosti nahrazení, obalení nebo dekorování třídy v budoucnosti.

nestabilní – závislosti, které porušují alespoň jedno z následujících:

[20] Knihovna s definicí základních datových typů. Je součástí frameworku .NET.

- komponenta ještě neexistuje nebo není plně implementována;
- komponenta není dostupná na všech strojích;
- komponenta má nedeterministické chování;
- komponenta nutně vyžaduje konfigurací běhového prostředí.

5.11.1 Kontejner

Kontejner pro vkládání závislostí nám umožňuje registraci a následné získání jednotlivých komponent. Přestože se dílčí schopnosti liší mezi jednotlivými kontejnery, základní funkce mívají stejné. Řadí se mezi ně především níže uvedené:

- možnost registrace komponent pro nějakou abstrakci nebo sama sebe;
- možnost registrace externí instance pro libovolný typ;
- možnost získání komponenty pro nějaký typ;
- pokud je třeba vytvořit instanci objektu, dojde ke zjištění a zajištění závislostí vybraného konstruktoru[21] a následné instanciaci pomocí něj;
- správa životního cyklu objektu. Sem spadá instanciace, inicializace a případný úklid. Nejpopulárnější životní cykly jsou:

  Transient – při každém dotazu je vrácena nová instance,

  Singleton – po celou dobu života kontejneru je vracena stejná instance,

  WebRequest – v rámci jednoho webového požadavku vrátí na dotaz vždy stejnou komponentu,

  Scoped umožňuje vytvářet vlastní sekce a v rámci nich vrací stejnou komponentu.

Ukázka 12 demonstruje registraci závislostí a jejich následné získání za použití DryIoc[22]. Jedná se o kontejner použitý v tomto projektu.

[21] Výběr konstruktoru je čistě na kontejneru. Častým přístupem je výběr takového, který obsahuje největší počet uspokojitelných závislostí.

[22] Dostupné na https://github.com/dadhi/DryIoc.

```
1  public interface INumberGenerator {
2      public int Next();
3  }
4  public class StaticNumberGenerator {
5      public int Number { get; set; }
6      public StaticNumberGenerator( int number ){
7          Number = number;
8      }
9      public int Next() => Number;
10 }
11 // registrační část
12 var container = new DryIoc.Container();
13 var requiredNumber = 5;
14 container.Register<INumberGenerator,StaticNumberGenerator>(
15     // výchozí hodnota pro závislost
16     made: Parameters.Of.Name("number", defaultValue: requiredNumber)
17 );
18 // funkční část
19 var generator = container.Resolve<INumberGenerator>();
20 Assert.Equals(requiredNumber, generator.Number);
21
22 var number = generator.Next();
23 Assert.Equals(requiredNumber, number);
```

Zdrojový kód 12: Konfigurace kontejneru pro vkládání závislostí

Kapitola 6

Testování aplikace

Po dokončení implementace přišlo na řadu testování aplikace jak z pohledu správné funkčnosti, tak použitelnosti. Aktuálně nejsou v projektu nasazeny jednotkové testy z důvodu potřeby rychlého vydání beta verze. Návrh systému dodržuje všechny principy pro snadnou testovatelnost kódu a k jejich zapracování dojde v budoucnosti.

6.1 Validace funkčnosti

Jedním ze způsobů testování je vytvoření, provedení a následná validace testovacích případů. Každý z nich obsahuje:

- počáteční stav systému;
- data, která mají být během vykonávání zadána. Například se jedná o hodnotu e-mailu a hesla pro verifikaci přihlášení;
- seznam kroků, které jsou následně v definovaném pořadí prováděny. Každý krok obsahuje popis akce k provedení a očekávaný stav systému po dokončení tohoto kroku.

S ohledem na vysokou rozpracovanost případů užití mobilní aplikace vycházela tato část testování právě z nich. Cílem bylo provést hlavní a všechny případné alternativní scénáře s variací vstupních dat a stavů systému pro dosažení

- úspěšného dokončení všech scénářů,
- vygenerování všech definovaných chyb

pro každý z případů užití. Během testování došlo ke generování neočekávaných stavů, což bylo způsobeno buď chybou v implementaci, nebo nedokonalým zadáním. Podle důvodu selhání došlo k řešení buď opravou v kódu, nebo dospecifikací požadovaného chování.

Jednou z podstatných částí této fáze bylo testování aplikace po sestavení pro vydání. Zde došlo k odhalení velkého množství případů, kdy ProGuard[23] odstranil implementaci tříd, které byly v systému využity, a tím způsobil pády aplikace.

6.2 Heuristická analýza

Patří mezi základní metody testování použitelnosti aplikace. Je založena na tom, že testování provádí odborníci, a nikoliv typičtí uživatelé. Neexistuje přesně daný postup, ale v praxi často používanou šablonou je publikace Jakoba Nielsena o 10 principech použitelnosti aplikace. První verze byla publikována v roce 1990 ve spolupráci s Rolfem Molichem [28, 29] a k následné úpravě došlo v roce 1994 na základě analýzy 249 problémů použitelnosti [30].

6.2.1 Viditelnost stavu systému

Stav systému by měl být uživateli vždy prezentován jasně a v rámci přiměřeného času. V aplikaci je řešeno toto:

- Akce vyžadující odpověď serveru zobrazují uživateli dialog o průběhu dané akce. Všechny ostatní akce by měly proběhnout do 300 ms.
- Aplikace je konstruována tak, aby akce pokud možno nezabírala více než jednu stránku. Například, pokud je potřeba se dostat na detail záznamu v rámci dokončeného deníku, každá stránka jasně komunikuje, kde se uživatel právě nachází.
- Vyplnění dotazníku tvoří výjimku z předešlého pravidla a je zde dbáno na sdělení uživateli, kolik otázek už zodpověděl a kolik mu jich ještě chybí k vyplnění. Toto je řešeno pomocí textové informace v ovládacím panelu a vizuálního zobrazení v horní části stránky.

6.2.2 Shoda s reálným světem

Všechny termíny použité v aplikaci by se měly shodovat s chápáním uživatele. Právě z tohoto důvodu došlo například k odstranění pojmu „inkontinence bez varování“ či nahrazení slova „mikce“ slovem „močení“.

6.2.3 Kontrola uživatele a jeho svoboda

Uživatel by měl mít možnost jednoduše zrušit akci a vrátit se do předchozího stavu. Všechny obrazovky v aplikaci disponují jednoduchou možností zrušení akce, která uživatele vrátí na předchozí obrazovku beze změny stavu aplikace. Aktuálně není možné modifikovat deníky a dotazníky po jejich odeslání. Pokud

[23] Nástroj pro optimalizaci bajt kódu Javy.

tedy uživatel chybně vyplní poslední údaj a dojde k finalizaci záznamu, už není možné ho opravit.

6.2.4 Konzistence a standardy

Návrh systému by měl zohledňovat standardy používané pro danou platformu. Spadá sem cokoli od výrazů po komponenty. Jednou z věcí, které byly změněny pro splnění tohoto bodu, bylo odstranění výběru množství tekutin pomocí posuvníku. Uživatelé tento element vůbec neznali, a tak byl změněn na rotující výběr, který se běžně používá v mobilních zařízeních.

6.2.5 Prevence chyb

Je lepší chybám předcházet než na ně později upozorňovat. Aplikace jasně komunikuje a případně omezuje formu zadávaných dat. Během testování nebyla odhalena žádná část aplikace, která by způsobovala zvýšený výskyt chyb. Pokud k nějaké došlo, jednalo se o různé překlepy, které dle mého není možno řešit lépe.

6.2.6 Flexibilita a efektivita použití

Systém by měl zkušenému uživateli umožňovat přizpůsobovat si často používané akce. Zde jsem neidentifikoval žádný prostor pro možnou implementaci. Je nežádoucí, aby aplikace podporovala výchozí hodnoty apod.

6.2.7 Rozpoznání namísto vzpomínání

Uživatel by neměl být zatěžován rozpomínáním, co je od něj právě vyžadováno, a měl by mít všechny informace potřebné k dokončení úkonu viditelné na právě prezentované stránce. Toto je dodrženo napříč aplikací. Ukázkou může být výběr urgence: uživatel je přesměrován na novou stránku, kde je mu přesně sděleno, co od něj systém požaduje.

6.2.8 Estetický a minimalistický design

Dialogy by neměly obsahovat nepotřebné informace. V aplikaci obsahují typ hlášky (info, chyba, ...) a její znění. Typ považuji za důležitý, protože umožňuje rychlou identifikaci a případné přeskočení dialogu.

6.2.9 Řešení nastalých chyb

Chybové hlášky by měly být formulovány jazykem, a nikoliv chybovými kódy. Aplikace uživateli zobrazuje velice precizní informace o chybách a o tom, co je potřeba udělat pro jejích odstranění. Příkladem je „Heslo musí být dlouhé alespoň X znaků.“, kde uživatel jasně vidí požadovanou délku hesla a nemusí hádat.

6.2.10 Nápověda a dokumentace

Ačkoli by měl být systém použitelný bez další dokumentace, někdy jí může být zapotřebí. V aplikaci jsou strukturované informace o jednotlivých funkcích. Zde jsem identifikoval jednu závadu a tou je umístění informací. Uživatel musí jít nejdříve do nastavení. Jako možná řešení bych viděl přesunutí informací do postranního menu nebo přidání funkce vyhledávání.

6.3 Testování s uživateli

Testování s uživateli opět vycházelo z případů užití. Oproti testovacím případům zde bylo cílem sledovat, jak jednoduše se uživateli podaří následovat scénář, který mu není předložen. Obecně se od tohoto kroku ve vývoji aplikace očekává identifikace míst, která jsou pro uživatele zmatečná a mají za následek špatnou použitelnost aplikace. Často jsou tyto problémy způsobeny předpoklady vývojářů o znalostech uživatelů, kteří jimi ale ve skutečnosti nedisponují.

Každému uživateli byla předána aplikace v nepřihlášeném stavu spolu s krátkým popisem, k čemu slouží. Následně dostával uživatel k plnění jednotlivé úkoly. Mezi hlavní problémy, na které uživatelé naráželi, se řadí následující:

1. Docházelo k pokusu o přihlášení před registrací.
2. Navigace na domovskou stránku nebyla zřejmá.
3. Matoucí žádost o potvrzování, zda šlo o první ranní mikci.
4. Špatná čitelnost textů.
5. Zaseknutí se na zadání objemu tekutin.

6.3.1 Návrh odstranění problému

Výstupy testování byly zpracovány a na jejich základě došlo k navržení možných řešení. Některá z nich byla rovnou implementována.

1 Nastala situace, že se uživatel pokusil o přihlášení před samotnou registrací. Po oznámení o neexistenci účtu došlo vždy ke správnému přechodu na stránku s registrací. Možné zjednodušení by bylo při přechodu na registraci ze stránky přihlášení předvyplnit e-mail, který byl případně na stránce přihlášení zadán, stejně jako je tomu v případě obnovy hesla.

2 Aktuálně probíhá navigace na domovskou stránku pomocí loga UROsoft v postranním menu. To by mohlo jít vyřešit přidáním ikony domečku před logo, resp. přidáním nové položky menu „Hlavní stránka“.

3 Při zadávání mikce během času bdění následujícího dne dochází ke zobrazení dialogu s výběrem, zda šlo o první ranní močení, nebo pacient ještě nešel spát. Tato informace je potřebná pro správné sledování mikčních dnů v případech, že pacient nešel spát či nevstával dle obvyklého režimu. Návrh řešení je přidat toleranční dobu, během které se aplikace nebude pacienta ptát a rovnou mikci označí za první ranní. Pokud je tedy obvyklá doba vstávání v 8:00 a tolerance 2 hodiny a k zadání mikce dojde po 6:00, systém bude mikci automaticky považovat za první ranní. Doporučená tolerance bude diskutována s MUDr. Švabíkem.

4 Oproti návrhu došlo k celoplošnému zvětšení použitého písma pro zlepšení čitelnosti. Tato změna měla uspokojivé výsledky.

5 Posuvník byl změněn na dialog s rolovací nabídkou výběru. Bylo dosaženo uspokojivých výsledků, ale je třeba doplnit do dialogu jednotky, pro některé uživatele to bez nich nebylo jasné.

6.4 Vyhodnocení testování

Aplikace je pacienty plně použitelná, ale existují aspekty, které by šlo zlepšit. Některé problémy byly již odstraněny a ostatní budou řešeny v nadcházejícím čase. Další testování zahrne širší spektrum uživatelů, především se zaměřením na pacienty.

Závěr

V rámci této diplomové práce se mi podařilo proniknout do prostředí léčby potíží dolních cest močových. Na základě konzultací s MUDr. Švabíkem jsem vybral vhodnou platformu pro elektronizaci sběru dat. Jako hlavní jsem zvolil mobilní aplikaci, která umožní zvýšit kvalitu získaných dat a pohodlí samotných pacientů. Následně jsem provedl analýzu domény jako celku, zmapoval funkční a nefunkční požadavky, na jejich základě jsem připravil scénáře užití a posléze jsem tuto část zakončil tvorbou doménového modelu. Následoval výběr vhodných technologií, kde jsem se zaměřil na možnost jednoduše podporovat obě dvě platformy, iOS a Android.

Analytickou část následoval návrh systému. V prvním kroku jsem připravil drátěný model a provedl první kolo testování. Na základě výskytu velkého množství problémů spojených s použitelností jsem podobu aplikace kompletně přepracoval. Po dokončení návrhu uživatelského rozhraní jsem vytvořil datový model. Posledním krokem této části byla definice REST API pro napojení klientské mobilní aplikace na webovou službu.

Návrh architektury aplikace jsem prováděl přímo ve vývojovém prostředí. Dbal jsem na dodržování uznávaných principů pro vývoj aplikací, jako jsou SOLID, DRY, GRASP a Deméteřin zákon. Díky tomu je výsledný produkt méně náchylný k chybám, snadno udržovatelný a lehce rozšiřitelný. Poté jsem provedl samotnou implementaci mobilní aplikace. Zdrojové kódy jsem kvůli snadnému využití v dalších projektech rozložil do několika balíčků. Vývoj jsem zakončil testováním funkčnosti a použitelnosti aplikace. Verifikace funkčnosti jsem prováděl za pomoci testovacích scénářů, které silně vycházely z případů užití. Naopak při testování s uživateli jsem jim nechával volnou ruku se projevit a poukázat na nedostatky.

Projekt touto diplomovou prací nekončí a práce na něm bude probíhat dál. Především se jedná o rozšíření testování, zapracování připomínek a implementaci dalších funkcionalit. V nejbližší budoucnosti je pravděpodobné, že aplikace bude rozšířena o mapu toalet a instruktážní modul, který by pacientovi pomáhal se cviky pro zlepšení stavu jeho močových cest.

Literatura

[1] Holmannová, D.: Vylučovací soustava. [cit. 2019-04-27]. Dostupné z: https://www.symptomy.cz/anatomie/vylucovaci-soustava

[2] Pitná a mikční karta. [cit. 2019-04-28]. Dostupné z: https://www.musimcasto.cz/download/dotaznik1.pdf

[3] Mezinárodní skóre prostatických symptomů IPSS. [cit. 2019-04-28]. Dostupné z: http://www.cus.cz//wp-content/uploads/2013/02/Dotaznik-Mezinarodni-skore-prostatickych-symptomu-IPSS.pdf

[4] Zachoval, R.; Krhut, J.; Zámečník, L.; aj.: Dotazníky hodnotící kvalitu života u pacientů s inkontinencí moči a hyperaktivním měchýřem. *Urologie pro praxi*, ročník 7, č. 6, 2006: s. 286–296, ISSN 1803-5299.

[5] Montemagno, J.: Meet Xamarin.Forms: 3 Native UIs, 1 Shared Code Base. [cit. 2019-05-05]. Dostupné z: https://devblogs.microsoft.com/xamarin/meet-xamarin-forms-3-native-uis-1-shared-code-base

[6] Britch, D.: Xamarin.Forms Pages. [cit. 2019-05-05]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/user-interface/controls/pages

[7] Holmannová, D.: Horní cesty močové. [cit. 2019-04-27]. Dostupné z: https://www.symptomy.cz/anatomie/horni-cesty-mocove

[8] Kawaciuk, I.: *Urologie*. Praha: Galén, první vydání, c2009, ISBN 9788072626267.

[9] Holmannová, D.: Dolní cesty močové. [cit. 2019-04-27]. Dostupné z: https://www.symptomy.cz/anatomie/dolni-cesty-mocove

[10] Vávrová, L.: Diagnostika a léčba benigní hyperplazie prostaty. *Postgraduální medicína: odborný časopis pro lékaře*, ročník 18, č. 6, 2016: s. 654–660, ISSN 1212-4184.

[11] Coyne, K. S.; Sexton, C. C.; Irwin, D. E.; aj.: The impact of overactive bladder, incontinence and other lower urinary tract symptoms on quality of life, work productivity, sexuality and emotional well-being in men and women: results from the EPIC study. *BJU International*, ročník 101, č. 11, 2008: s. 1388–1395.

[12] Sobotka, R.: Dysfunkce dolních cest močových – možnosti diagnostiky a léčby. *Postgraduální medicína: odborný časopis pro lékaře*, ročník 18, č. 6, 2016: s. 583–591, ISSN 1212-4184.

[13] Zachoval, R.; Zálenský, M.; Heráček, J.; aj.: Neurogenní dysfunkce dolních močových cest. *Urologie pro praxi*, ročník 5, č. 2, 2004: s. 73–77, ISSN 1803-5299.

[14] Klečka, J.; Hora, M.; Eret, V.; aj.: Vybrané názvosloví urodynamiky a mikčních symptomů dolních cest močových. *Urologie pro praxi*, ročník 13, č. 2, 2012: s. 75–78, ISSN 1803-5299.

[15] Hanuš, T.; Macek, P.: *Urologie pro mediky*. V Praze: Univerzita Karlova, nakladatelství Karolinum, vydání první vydání, 2015, ISBN 978-80-246-3032-8.

[16] Vrtal, R.; Vidlář, A.; Študent, V.: Diagnostika a léčba hyperaktivního měchýře. *Urológia pre prax*, ročník 8, č. 5-6, 2007: s. 189–192, ISSN 1337-107X.

[17] Roy, A.; Singh, A.; Sidhu, D.; aj.: New visual prostate symptom score versus international prostate symptom score in men with lower urinary tract symptoms: A prospective comparision in Indian rural population. *Nigerian Journal of Surgery*, ročník 22, č. 2, 2016: s. 111–117, doi: 10.4103/1117-6806.189002.

[18] Oelke, M.; Bachmann, A.; Descazeaud, A.; aj.: EAU Guidelines on the Treatment and Follow-up of Non-neurogenic Male Lower Urinary Tract Symptoms Including Benign Prostatic Obstruction. *European Urology*, ročník 64, č. 1, 2013: s. 118 – 140, ISSN 0302-2838, doi:https://doi.org/10.1016/j.eururo.2013.03.004. Dostupné z: `http://www.sciencedirect.com/science/article/pii/S0302283813002285`

[19] Nall, R.: A guide to double voiding and bladder-emptying techniques. 2017, [cit. 2019-04-28]. Dostupné z: `https://www.medicalnewstoday.com/articles/316706.php`

[20] Mathias, S. D.; Crosby, R. D.; Nazir, J.; aj.: Validation of the Patient Perception of Intensity of Urgency Scale in Patients with Lower Urinary Tract Symptoms Associated with Benign Prostatic Hyperplasia. *Value in Health*, ročník 17, č. 8, 2014/12/01: s. 823–829, ISSN 1098-3015,

doi:10.1016/j.jval.2014.09.002. Dostupné z: https://doi.org/10.1016/j.jval.2014.09.002

[21] Cartwright, R.; Srikrishna, S.; Cardozo, L.; aj.: Validity and reliability of the patient's perception of intensity of urgency scale in overactive bladder. *BJU International*, ročník 107, č. 10, 2011: s. 1612–1617, doi:10.1111/j.1464-410X.2010.09684.x, https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1464-410X.2010.09684.x. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1464-410X.2010.09684.x

[22] Zachoval, R.; Krhut, J.; Fügner, D.; aj.: Doporučené postupy diagnostiky a léčby nykturie. *Urológia pre prax*, ročník 8, č. 5-6, 2007: s. 204–211, ISSN 1337-107X.

[23] Britch, D.: Renderer Base Classes and Native Controls. [cit. 2019-05-05]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/app-fundamentals/custom-renderer/renderers

[24] Getting Started. [cit. 2019-05-06]. Dostupné z: https://realm.io/docs/dotnet/latest/

[25] LINQ support in Realm .NET. [cit. 2019-05-06]. Dostupné z: https://realm.io/docs/dotnet/latest/api/linqsupport.html

[26] Montemagno, J.; Dunn, C.; Alt, J.; aj.: Xamarin.Essentials: Secure Storage. [cit. 2019-05-07]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/essentials/secure-storage

[27] van Deursen, S.; Seemann, M.: *Dependency Injection Principles, Practices, and Patterns*. Shelter Island: Manning Publications, první vydání, 2019, ISBN 9781617294730.

[28] Molich, R.; Nielsen, J.: Improving a Human-computer Dialogue. *Commun. ACM*, ročník 33, č. 3, Březen 1990: s. 338–348, ISSN 0001-0782, doi:10.1145/77481.77486. Dostupné z: http://doi.acm.org/10.1145/77481.77486

[29] Nielsen, J.; Molich, R.: Heuristic Evaluation of User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, New York, NY, USA: ACM, 1990, ISBN 0-201-50932-6, s. 249–256, doi:10.1145/97243.97281. Dostupné z: http://doi.acm.org/10.1145/97243.97281

[30] Nielsen, J.: Enhancing the Explanatory Power of Usability Heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, New York, NY, USA: ACM, 1994, ISBN 0-

89791-650-6, s. 152–158, doi:10.1145/191666.191729. Dostupné z: http://doi.acm.org/10.1145/191666.191729

[31] Sochorová, N.: Inkontinence moči a jednorázové absorpční pomůcky. *Urológia pre prax*, ročník 9, č. 1, 2008: s. 34–36, ISSN 1337-107X.

Příloha A

Seznam použitých zkratek

UI User interface

XAML Extensible application markup language

IL Intermmediate Language

ORM Object-relational mapping

LINQ Object-relational mapping

CRUD Create Read Update Delete

BCL Base Class Library

IPSS International Prostate Symptom Score

OAB Overactive bladder

SOLID Akronym pěti návrhových principů: Single responsibility principle, Open–closed principle, Liskov substitution principle, Interface segregation principle, Dependency inversion principle.

DRY Don't repeat yourself

GRASP General Responsibility Assignment Software Patterns

Příloha **B**

Obsah přiložené SD karty

- `UROsoft.apk` ..... Aplikace pro Android
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `apiary.bp` ..... zdrojová forma definice API v Apiary.io
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `analysis`
  - `data_model.png` ..... kompletní datový model
  - `balsamiq.bmpr` ..... dratěný model v Balsamiq
- `screenshots` ..... Snímky jednotlivých obrazovek aplikace