FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Mobilní aplikace pro pomoc léčby dětské obezity u společnosti MEDASOL
Student:	Lukáš Pekař
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem bakalářské práce je vytvořit mobilní aplikaci pro Android. Aplikace bude sloužit pro pomoc s léčbou a prevencí dětské obezity u společnosti MEDASOL. Jádrem aplikace bude sdílení fotografií jednotlivých jídel pacientů s obezitology, kteří jim budou poskytovat zpětnou vazbu.

1. Seznamte se s diplomovou prací „Vytvoření business modelu projektu dětská obezita“ od Bc. Andrei Holoubkové, která obsahuje rozsáhlou analýzu pro vytvoření mobilní aplikace.

2. Udělejte návrh aplikace a vyberte vhodné technologie.
3. Implementujte mobilní aplikaci pro android, která bude umožňovat snadnou zpětnou vazbu pro pacienty od lékařů, na základě vyfocených jídel.
4. Proveďte řádné otestování aplikace.
5. Dané řešení zhodnoťte a navrhněte možná budoucí vylepšení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 22. ledna 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Mobilní aplikace pro pomoc léčby dětské obezity u společnosti MEDASOL

Lukáš Pekař

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petra Pavlíčková, Ph.D

30. července 2020

Poděkování

Rád bych poděkoval Ing. Petře Pavlíčkové, Ph.D. za hodnotné rady, věcné připomínky a pomoc při vypracování této bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při 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 citovaného zákona.

V Praze dne 30. července 2020 . . . . . .

Odkaz na tuto práci

Pekař, Lukáš. *Mobilní aplikace pro pomoc léčby dětské obezity u společnosti MEDASOL*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato bakalářská práce se zabývá vývojem mobilní aplikace MedaFit na operační systém Android, která pomáhá léčit dětskou obezitu u společnosti Medasol. Na základě dřívější diplomové práce zabývající se business modelem pro tento projekt, je provedena specifikace, návrh a implementace dané aplikace s použitím metodik pro softwarové inženýrství. Jádrem aplikace je sdílení fotografií jídel od léčeného dítěte směrem k lékaři a herní prvky (gamifikace). Výsledkem práce je funkční mobilní aplikace na Android, jež může společnost Medasol užívat pro léčbu dětské obezity.

Klíčová slova mobilní aplikace, dětská obezita, léčba, gamifikace, REST API, klient-server, Android, .NET Core

Abstract

This bachelor thesis is focused on mobile application development of MedaFit application on operating system Android that helps cure childhood obesity with help of Medasol company. Specification, design and implementation of the application is done using methods of software engineering and is based on previous diploma thesis about business model for this application. The core of application is sharing meal photos between child and doctor and also gamification. The output of this work is functional mobile application on Android that can be used by Medasol for curing childhood obesity.

Keywords mobile appliaction, childhood obesity, treatment, gamification, REST API, client-server, Android, .NET Core

Obsah

Seznam obrázků

Úvod

V současné době lze pozorovat stále se objevující nové nápady na zakomponování moderních technologií do všech možných odvětví lidské činnosti. Není tomu jinak ani ve světě zdravotnictví, konkrétně v oblasti dětské obezity.

Spolu s narůstajícím počtem obézních dětí vzniká velká motivace pro nová řešení tohoto problému s využitím moderních technologií. Doktoři ze společnosti Medasol proto požádali ČVUT FIT o spolupráci na vývoji aplikace, jež má hravou formou pomoci obézním dětem k hubnutí.

Pro tyto účely vznikla diplomová práce *Vytvoření business modelu projektu dětská obezita*, kterou vypracovala Bc. Andrea Holoubková [1]. Diplomová práce se věnuje analýze dané problematiky a poskytuje návrh prototypu mobilní aplikace. Základem aplikace je sdílení přijímané potravy dítěte s obezitology. Nyní když je k dispozici business model pro celý projekt dětské obezity, už nic nebrání tomu, aby se daná aplikace vyvinula a projekt realizoval.

Tato práce se věnuje právě návrhu a implementaci mobilní aplikace pro pomoc s dětskou obezitou u společnosti Medasol. Důležitou roli hraje snadné ovládání aplikace a herní prvky, jež mají podnítit u dětí zájem o hubnutí. Obezitologové na druhé straně chtějí přehledný obraz toho, jaký mají děti příjem potravy. Na základě zpětné vazby jsou pak děti hodnoceny virtuálními mincemi podle toho, jak si podle obezitologů vedou. Rodiče budou následně moci za virtuální mince své dítě odměňovat.

Do budoucna se plánuje v aplikaci využít technologie strojového učení pro analýzu fotografií jednotlivých jídel, tak že aplikace bude schopná hned rozeznat, jak „zdravé“ jídlo je bez nutnosti zásahu obezitologa. Má práce se však tímto tématem nezaobírá.

Mou motivací pro vytvoření této práce je skutečnost, že mobilní aplikace může pomoci mnoha dětem potýkajícím se s obezitou. Zároveň souzním se způsobem, jakým chtějí obezitologové ze společnosti Medasol tento problém řešit a věřím že cesta, kterou se vydali, je správná.

Kapitola 1

Cíl práce

Cílem práce je vyvinout mobilní aplikaci pro Android, jež bude pomáhat léčit dětskou obezitu u společnosti Medasol. Základním kamenem pro vývoj aplikace je diplomová práce *Vytvoření business modelu projektu dětská obezita* od Bc. Andrei Holoubkové [1]. Jako jádro aplikace slouží sdílení fotografií jednotlivých jídel obézního dítěte s obezitologem. Obezitolog následně předá dítěti a jeho rodičům zpětnou vazbu a ohodnotí jej virtuálními mincemi.

Vzhledem k tomu, že aplikace má potenciál mít dlouhou životnost, je žádoucí, aby byla snadno rozšiřitelná a udržovatelná. Z toho důvodu je druhým cílem práce, aby sloužila jako podklad pro programátory navazující na mou práci.

Posledním cílem práce je otestování aplikace a její příprava k nasazení. Závěr práce by měl obsahovat zhodnocení vyvinuté aplikace a možná budoucí vylepšení.

Kapitola 2

Rešeršní část

2.1 Softwarový proces

Cílem softwarového inženýrství je efektivně vytvářet software. Pro dosažení tohoto cíle se určuje Softwarový proces. Softwarový proces zahrnuje postupy a metodiky, jak software vytvářet [2].

V praxi se osvědčilo rozdělení tvorby software na jednotlivé fáze, dovolím si zde citovat přednášku z kurzu *Softwarové Inženýrství II.* „*Pro úspěšné vytvoření software je třeba projít následujícími kroky:*

- *specifikaci -— co bude systém dělat, jak se změní,*
- *architekturu a design —- z jakých kostek a jak se bude systém skládat,*
- *implementaci -— vlastní výroba systému,*
- *validaci —- ověření, že systém dělá co má.*

Žádný z těchto kroků nelze vynechat.“ [3]

2.2 Mobilní aplikace

Mobilní aplikace jsou software určený pro mobilní zařízení, které vykonává nějakou úlohu pro uživatele. V dnešní době se tyto zařízení často označují jako chytré telefony.

Z pohledu vývoje se mobilní aplikace v mnohém podobají jakémukoliv jinému software, jako např. integrace s hardware zařízení nebo požadavcích na výkon. Mobilní aplikace se však typicky vyznačují následujícími vlastnostmi:

- spoluprací s ostaními aplikacemi,

- zpracováváním vstupů z různých senzorů (gesta na dotykové obrazovce, zvuk z mikrofonu, GPS, Fotoaparát a mnoho dalších),
- schopností fungovat na různých zařízeních,
- uživatelským rozhraním aplikace, které se přizpůsobuje dané platformě,
- komplexním testováním z důvodu různých zařízení,
- braním v potaz omezený zdroj energie z baterie a tudíž neprováděním energeticky náročných operací.

Z pohledu uživatele jsou mobilní aplikace jednoduché na používání, levné a typicky není potřeba příliš drahé zařízení pro jejich běh. Dalo by se tedy říci, že mobilní aplikace jsou z uživatelského hlediska velice dostupné.

Použití mobilních aplikací je široké a zahrnuje např. hry, kancelářské programy, sociální sítě, aplikace pro audiovizuální zpracování, GPS navigace a mnoho dalších [4][5].

2.3 Model případů užití

Model případů užití slouží v etapě analýzy tvorby softwaru k detailní specifikaci funkčních požadavků. Oproti samotným funkčním požadavkům mají případy užití zpravidla vyšší granularitu a tedy poskytují větší míru detailu.

Typickou první položkou modelu případů užití je seznam aktérů, kde aktér reprezentuje roli uživatele vzhledem k systému, externí systém nebo čas.

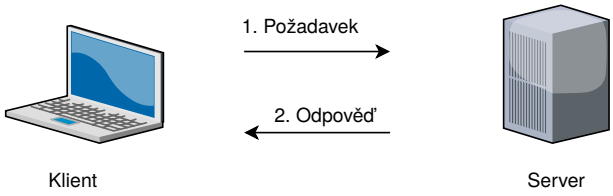
Dále model obsahuje diagram případů užití, jež zachytává přiřazení jednotlivých případů užití k aktérům. Na první pohled je zřejmé, které případy užití využívá jaký aktér. Zároveň také zaznamenává generalizaci aktérů.

Nejdůležitější částí celého modelu jsou samotné případy užití. Jeden případ užití reprezentuje sekvenci kroků, jaké uživatel vykonává při plnění nějaké úlohy v daném systému. Popis případu užití typicky obsahuje jeden hlavní scénář a případně vedlejší scénáře. V případě, že je vedlejších scénářů více, se doporučuje zaznamenat scénáře pomocí diagramu aktivit. Naopak v situacích, kde je naprosto zřejmé, jak má vypadat plnění případu užití postačí slovní popis [6].

Případem užití může být např. výběr hotovosti z bankomatu, kde by byl aktérem vlastník kreditní karty (zákazník banky daného bankomatu). Hlavní scénář by v jednotlivých krocích popisoval úspěšný výběr. Vedlejší scénáře by naopak detailně specifikovaly neúspěšné výběry.

2.4 Klient-server

Klient-server označuje vztah mezi dvěma programy, kde klient zasílá požadavky na server a server provádí úlohy pro plnění daných požadavků. Na straně ser-



Obrázek 2.1: Klient-server

veru tak vzniká centralizovaný systém. Protikladem jsou distribuované systémy, které pro svůj běh žádný server nepotřebují [7].

Typicky se klient-server používá pro dynamické webové stránky a mobilní či desktopové aplikace využívající externí systémy.

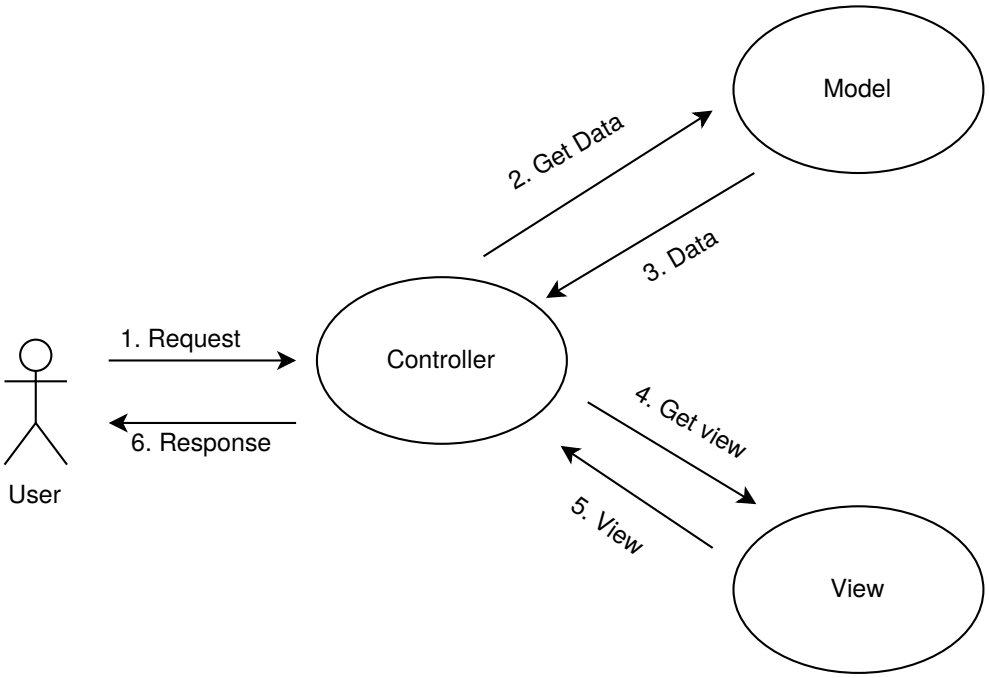
2.5 REST API

REST API představuje aplikační rozhraní systému pro jiné části systému, nebo pro externí aplikace. Zkratka REST stojí za Representational State Transfer a určuje pravidla, jak bude dané API vypadat. Definuje, že se ke komunikaci použije sada bezestavových operací nebo také, že každou operaci lze najít pomocí unikátního identifikátoru. Nejčastěji se REST API realizuje pomocí protokolu HTTP, kde jsou operace identifikovány pomocí URL [8].

2.6 MVC

MVC neboli Model-View-Controller je architektonický vzor, jež rozděluje aplikaci do tří částí: modelů, kontrolerů a pohledů a má za cíl oddělit prezentační vrstvu od zbytku aplikace [9].

Typické využití MVC lze vidět u dynamických webových stránek, kde se aplikuje následovně. Požadavek na webový server zachytí kontroler, který v závislosti na požadavku získá z modelu potřebná data. Následně kontroler vybere v závislosti na požadavku správný pohled, ten naplní daty a pošle uživateli jako odpověď na daný požadavek. Celý proces lze vidět na obrázku 2.2.



Obrázek 2.2: Použití MVC u webové aplikace

Kapitola 3

Analýza

3.1 Léčba dětské obezity, Medasol

Dětská obezita je v současné době velkým problémem ve vyspělých zemích. Samotná nemoc má mnoho nežádnoucích následků na fyzické i psychické zdraví dětí. Vzhledem k počtu dětí, které jí trpí, je považována za jednu z nejvážnějších nemocí 21. století [10].

Společnost Medasol, kterou tvoří evropský tým lékařů se sídlem v Praze [11], se mimo jiné snaží v oblasti léčby dětské obezity udělat pokrok. Chtějí ho dosáhnout pomocí hravé mobilní aplikace MedaFit, kde na jedné straně obézní děti fotí svá jídla a na straně druhé stojí lékaři z Medasolu, kteří dané fotografie vyhodnocují a děti odměňují. Třetím aktérem v aplikaci jsou rodiče dětí, kteří můžou dohlížet na to, jak se dítě stravuje a zároveň jej mohou odměňovat za získané virtuální mince z aplikace.

Místo násilného nucení dětí do přísné diety a cvičení má aplikace vzbudit v dětech hravost a soutěživost. Pomocí virtuálních mincí, jež dítě dostane v průběhu léčby s pomocí aplikace, bude mít dítě větší motivaci k správné životosprávě.

Rozsáhlou analýzu toho, jak má aplikace vypadat, udělala ve své diplomové práci *Vytvoření business modelu projektu dětská obezita* Bc. Andrea Holoubková [1]. Na jejích základech tvořím mobilní aplikaci a snažím se tak uvést její snahu v reálný projekt, jež se bude k léčbě dětské obezity u společnosti Medasol využívat.

3.2 Definice pojmů

V této části popíši pojmy, jež budu dále práci využívat.

- **Pacient** — Dítě, jež využívá aplikaci pacienta, ve které zaznamenává jídla.

- **Doktor** — Obezitolog ze společnosti Medasol, který dává pacientům zpětnou vazbu. Využívá doktorskou aplikaci.
- **Administrátor** — Uživatel doktorské aplikace se speciálními právy pro správu doktorů.
- **Doktorská aplikace** — Administrační webová aplikace určená k administraci pacientů, doktorů a k editaci a potvrzování zpětných vazeb.
- **Aplikace pacienta** — Mobilní aplikace, jež využívají pacienti k zaznamenávání jídel a prohlížení zpětných vazeb.
- **Nutriční hodnota** — Nutriční hodnotou je myšlena čtveřice čísel označující počet gramů bílkovin, počet gramů sacharidů, počet gramů tuků a počet kcal.
- **Nutriční tabulka** — Nutriční tabulka se skládá z jednotlivých řádků, kde každý řádek obsahuje textový popis jídla či části jídla a nutriční hodnotu k dané části jídla.
- **Vyhodnocení** — Vyhodnocení se skládá z počtu mincí udělených ke konkrétní zpětné vazbě a textového ohodnocení ke konkrétní zpětné vazbě.
- **Zpětná vazba** — Zpětná vazba se vždy váže ke konkrétnímu dni a pacientovi a obsahuje nutriční tabulku, komentáře k jednotlivým jídlům a vyhodnocení.

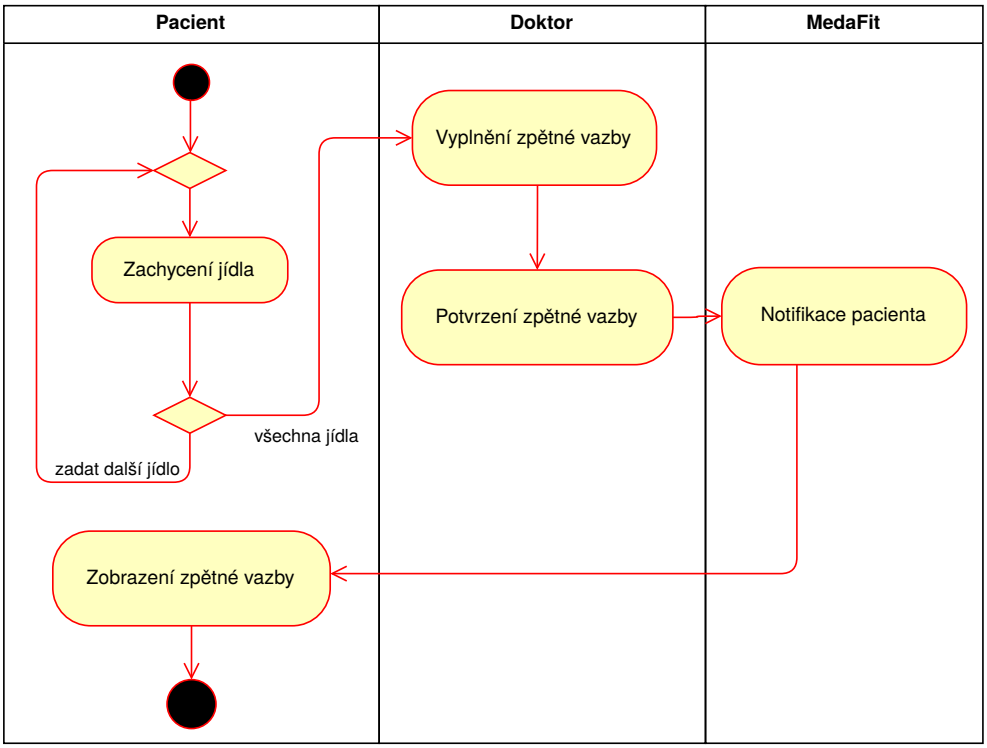
3.3 Typický cyklus použití aplikace

Předpokládejme, že jsou vytvořené uživatelské účty jak pro pacienta, tak pro doktora. Typický cyklus celé aplikace poté probíhá následovně.

Pacient přes den vyfotí každé své jídlo, které se zaznamená a uloží. Jakmile bude mít pacient zachycena všechna svá jídla pro daný den, tak doktor na druhé straně vyplní zpětnou vazbu a jejím potvrzením notifikuje pacienta o nové zpětné vazbě. Pacient si následovně zobrazí zpětnou vazbu a dozví se, jak si daný den vedl, co se životosprávy týče. Daný cyklus je zobrazen jako diagram aktivit v obrázku 3.1.

3.4 Požadavky

V následujícím textu jsou popsány požadavky na aplikaci MedaFit. Je zvoleno rozdělení na požadavky funkční a nefunkční. Vzhledem k tomu, že se z uživatelského hlediska jedná o dvě aplikace (doktorská aplikace a aplikace pacienta), je zvoleno označení požadavků následovně:



Obrázek 3.1: Typický cyklus použití aplikace

- **FP-P-*x*** — označuje funkční požadavek na aplikaci pacienta, kde x je číslo požadavku.
- **FP-D-*x*** — označuje funkční požadavek na doktorskou aplikaci, kde x je číslo požadavku.
- **NP-P-*x*** — označuje nefunčkní požadavek na aplikaci pacienta, kde x je číslo požadavku.
- **NP-D-*x*** — označuje nefunčkní požadavek na doktorskou aplikaci, kde x je číslo požadavku.

3.4.1 Funkční Požadavky

- **FP-P-1 Přihlášení pacienta** — Pacient se bude moci přihlásit do aplikace pacienta pomocí dvojice email a heslo. Bez úspěšného přihlášení nebude možné v aplikaci pacienta jakkoliv pokračovat.
- **FP-P-2 Zobrazení domovské stránky** — Pacient bude moci vidět svou domovskou stránku, která bude zachycovat počet jeho mincí, množství kcal za současný den a všechna jím zachycená jídla pro současný den.

- **FP-P-3 Možnost vidět předešlé dny** — Pacient bude moci po překliknutí vidět jím zachycená jídla na domovské stránce i pro jiný než současný den. V Této situaci však nebude možné přidávat další jídla.
- **FP-P-4 Zachycení nového jídla** — Pacient bude moci pro současný den zachytit nové jídlo a zařadit jej do jedné z kategorií: snídaně, svačina, oběd, večeře.
- **FP-P-5 Zobrazení zpětné vazby** — Pokud bude k danému dni vytvořena potvrzená zpětná vazba od jednoho z doktorů, pacient si ji bude moci zobrazit.
- **FP-P-6 Odměna** — Pacient bude moci být odměněn za získané mince. Pro odměnění bude potřeba zadat počet mincí, textový popis odměny a rodičovské heslo.
- **FP-P-7 Notifikace** — Jakmile doktor vyplní zpětnou vazbu pro uživatele a potvrdí ji, tak na mobilní telefon pacienta přijde notifikace s upozorněním, že na daný den byla přidána zpětná vazba.
- **FP-D-1 Přihlášení doktora** — Doktor se bude moci přihlásit do doktorské aplikace. Bez přihlášení nebude moci provádět žádné jiné akce.
- **FP-D-2 Registrace pacienta** — Doktor bude moci zaregistrovat nového pacienta a vyplnit jeho údaje: jméno, příjmení, email, heslo, přezdívka, věk, výška, váha, doporučené nutriční hodnoty a rodičovské heslo.
- **FP-D-3 Editace pacienta** — Doktor bude moci upravit pacientovi existující údaje. Kromě emailu může upravit všechny hodnoty, jež byly zadány při registraci.
- **FP-D-4 Odstranění pacienta** — Doktor bude moci odstranit pacienta, tak aby se již nemohl přihlásit do aplikace pacienta.
- **FP-D-5 Registrace doktora** — Administrátor bude moci registrovat nové doktory. U doktorů se budou evidovat: jméno, příjmení, email, heslo a titul.
- **FP-D-6 Editace doktora** — Administrátor bude moci upravovat existující údaje doktorů. Kromě emailu bude možné editovat všechny údaje zadané při registraci.
- **FP-D-7 Odstranění doktora** — Administrátor bude moci odstranit doktora, tak aby se již nedostal do doktorské aplikace.
- **FP-D-8 Zobrazení zpětných vazeb** — Doktor si bude moci u konkrétního pacienta zobrazit jeho zpětné vazby. Zpětné vazby se budou automaticky vytvářet nevyplněné u dnů, u kterých je zaznamenáno alespoň

jedno jídlo ze strany pacienta. Dny, na které ještě nebyla zpětná vazba potvrzena, budou speciálně označeny.

- **FP-D-9 Editace zpětné vazby** — Doktor bude moci vyplnit zpětnou vazbu pro konkrétní den a pacienta.
- **FP-D-10 Potvrzení zpětné vazby** — Doktor bude moci potvrdit vyplněnou zpětnou vazbu a tím notifikovat pacienta.

3.4.2 Nefunkční požadavky

- **NP-P-1 Operační systém** — Aplikace pacienta bude fungovat na operačním systému Android od verze Android 5.0 Lollipop.
- **NP-P-2 Snadná ovladatelnost** — Aplikace pacienta bude mít jednoduché a přehledné GUI, jež dokáže ovládat dítě.
- **NP-D-1 Kompatibilita** — Doktorská aplikace bude správně fungovat z webového prohlížeče Google Chrome alespoň od verze 70.

3.5 Model případů užití

Model případu užití zaznamenává a zpřesňuje definované funkční požadavky z předchozí kapitoly.

3.5.1 Aktéři

Uživatele aplikace MedaFit lze rozdělit do čtyř skupin.

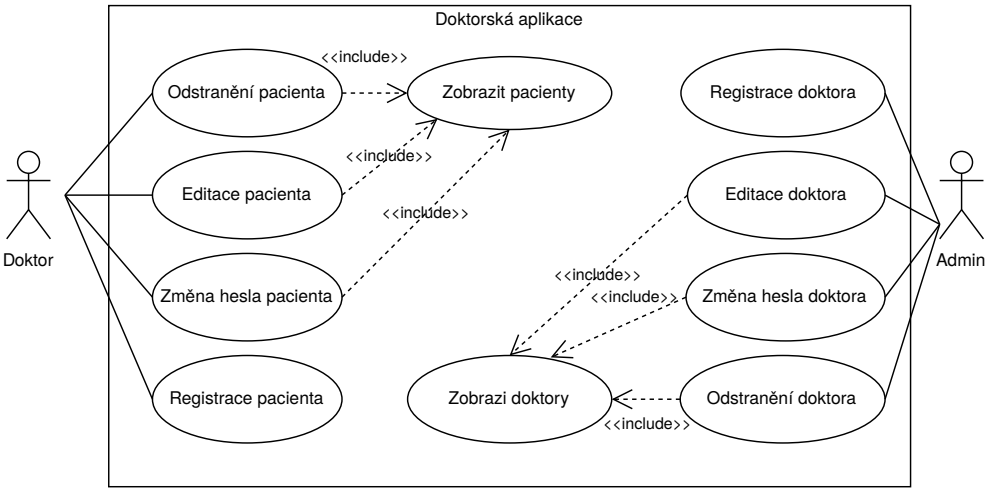
- **Doktor** — obezitolog ze společnosti Medasol
- **Administrátor** — uživatel se speciálními právy pro správu doktorů
- **Pacient** — léčené dítě
- **Rodič** — rodič léčeného dítěte

3.5.2 Správa doktorů a pacientů

UC1: Registrace pacienta

Případ užití umožňuje doktorovi zaregistrovat nového pacienta do systému. Při registraci vyplní základní údaje pacienta: jméno, příjmení, věk, výšku, váhu, email, heslo, rodičovské heslo a doporučené denní nutriční hodnoty.

Aktér: Doktor



Obrázek 3.2: Diagram případů užití doktorské aplikace

UC2: Editace pacienta

Případ užití umožňuje doktorovi editovat základní údaje pacienta, které jsou stejné jako u registrace pacienta s výjimkou emailu a hesla.

Aktér: Doktor

UC3: Změna hesla pacienta

Případ užití umožňuje doktorovi změnit heslo pacienta.

Aktér: Doktor

UC4: Odstranění pacienta

Případ užití umožňuje doktorovi odstranit pacienta ze systému tak, že se daný pacient nebude moci nadále přihlásit a aplikaci pacienta používat.

Aktér: Doktor

UC5: Zobrazení pacientů

Případ užití umožňuje doktorovi zobrazit seznam pacientů seřazených dle příjmení a jména.

Aktér: Doktor

UC6: Registrace doktora

Případ užití umožňuje administrátorovi zaregistrovat nového doktora do systému. Při registraci vyplní základní údaje doktora: jméno, příjmení, titul, email a heslo.

Aktér: Administrátor

UC7: Editace doktora

Případ užití umožňuje administrátorovi editovat základní údaje doktora, které jsou stejné jako u registrace doktora s výjimkou emailu a hesla.

Aktér: Administrátor

UC8: Změna hesla doktora

Případ užití umožňuje administrátorovi změnit heslo doktora.

Aktér: Administrátor

UC9: Odstranění doktora

Případ užití umožňuje administrátorovi odstranit doktora ze systému tak, že se daný doktor nebude moci nadále přihlásit a doktorskou aplikaci používat.

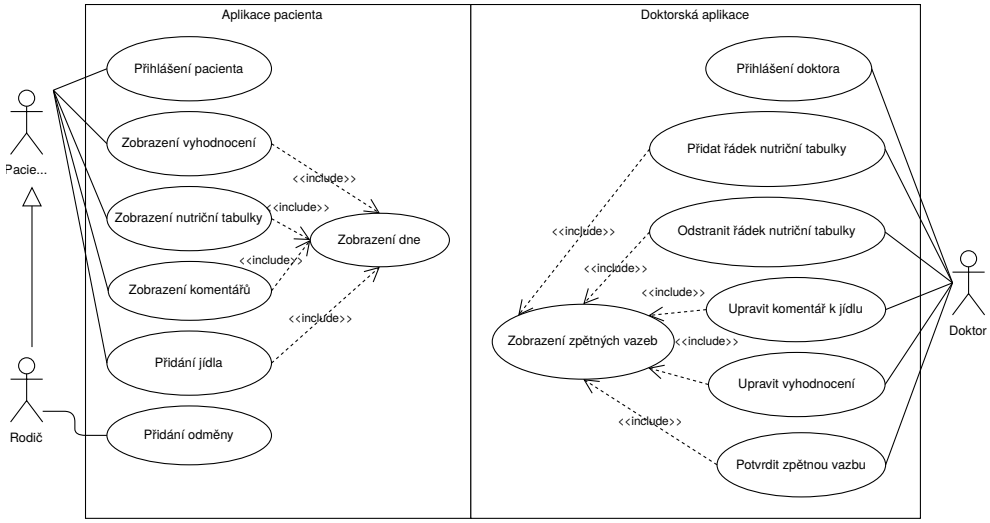
Aktér: Administrátor

UC10: Zobrazení doktorů

Případ užití umožňuje administrátorovi zobrazit seznam doktorů seřazených dle příjmení a jména.

Aktér: Administrátor

3.5.3 Aplikace pacienta a zpětné vazby



Obrázek 3.3: Diagram případů užití aplikace pacienta a zpětné vazby

UC11: Přihlášení pacienta

Případ užití umožňuje pacientovi přihlásit se do aplikace pomocí emailu a hesla.

Aktér: Pacient, Rodič

Hlavní scénář: Klasické přihlášení

1. Pacient zadá svůj email a heslo.
2. Systém zkontroluje validitu emailu a hesla a v případě úspěchu autentizuje pacienta.

Vedlejší scénář: První přihlášení

Scénář začíná ve 2. kroku hlavního scénáře, kdy systém úspěšně autentizoval pacienta poprvé.

1. Pacient je přesměrován na úvodní obrazovku, ve které se mu představí maskot aplikace MedaFit.

UC12: Zobrazení dne

Případ užití umožňuje pacientovi zobrazit všechna jídla, která v daný den v aplikaci zaznamenal. Zároveň mu také zobrazí počet kcal, jež doktor k tomu dni vyplnil v nutriční tabulce. Pokud ještě není nutriční tabulka vyplněná v potvrzené zpětné vazbě pro daný den, zobrazí se 0.

Aktér: Pacient, Rodič

UC13: Zobrazení vyhodnocení

Případ užití umožňuje pacientovi zobrazit vyhodnocení pro daný den, pokud dané vyhodnocení existuje v potvrzené zpětné vazbě pro daný den.

Aktér: Pacient, Rodič

UC14: Zobrazení nutriční tabulky

Případ užití umožňuje pacientovi zobrazit nutriční tabulku pro daný den, pokud daná nutriční tabulka existuje v potvrzené zpětné vazbě pro daný den.

Aktér: Pacient, Rodič

UC15: Zobrazení komentářů

Případ užití umožňuje pacientovi zobrazit komentáře k jídlům pro daný den, pokud dané komentáře existují v potvrzené zpětné vazbě pro daný den.

Aktér: Pacient, Rodič

UC16: Přidání jídla

Případ užití umožňuje pacientovi přidat pro daný den fotografii jídla.
Aktér: Pacient, Rodič
Hlavní scénář: Přidání jídla

1. Pacient si zobrazí současný den.
2. Pacient klikne na ikonu pro přidání jídla.
3. Systém zobrazí pacientovi všechny předurčené typy jídel.
4. Pacient vybere typ jídla.
5. Systém spustí fotoaparát, aby mohl pacient vyfotit dané jídlo.
6. Pacient vyfotí dané jídlo.
7. Systém fotografii i s vybraným typem uloží.

UC17: Přidání odměny

Případ užití umožňuje rodiči přidat pacientovi odměnu za získané mince. Zvolí počet mincí, zadá textovou podobu odměny a rodičovské heslo. V případě, že je rodičovské heslo platné a pacient má dostatečný počet mincí, uloží odměnu.
Aktér: Rodič

UC18: Přihlášení doktora

Případ užití umožňuje doktorovi přihlásit se pomocí svého emailu a hesla do doktorské aplikace.
Aktér: Doktor

UC19: Zobrazení zpětných vazeb

Případ užití umožňuje doktorovi zobrazit si zpětné vazby pro konkrétního pacienta.
Aktér: Doktor
Hlavní scénář: Zobrazení zpětných vazeb

1. Systém doktorovi zobrazí seznam pacientů.
2. Doktor si vybere pacienta, jehož zpětné vazby chce zobrazit.
3. Systém vytvoří nové nepotvrzené zpětné vazby u těch dnů, u kterých pacient přidal alespoň jedno jídlo a zároveň u nich neexistuje zpětná vazba.
4. Systém zobrazí doktorovi zpětné vazby rozdělené na potvrzené zpětné vazby a nepotvrzené zpětné vazby.

UC20: Přidat řádek nutriční tabulky

Případ užití umožňuje doktorovi přidat řádek do nutriční tabulky v konkrétní zpětné vazbě.

Aktér: Doktor

UC21: Odstranit řádek nutriční tabulky

Případ užití umožňuje doktorovi odstranit řádek z nutriční tabulky v konkrétní zpětné vazbě.

Aktér: Doktor

UC22: Upravit komentář k jídlu

Případ užití umožňuje doktorovi upravit text komentáře k danému jídlu v konkrétní zpětné vazbě.

Aktér: Doktor

UC23: Upravit vyhodnocení

Případ užití umožňuje doktorovi upravit textovou podobu vyhodnocení a počet mincí udělené za daný den v konkrétní zpětné vazbě.

Aktér: Doktor

UC24: Potvrdit zpětnou vazbu

Případ užití umožňuje doktorovi potvrdit zpětnou vazbu. Potvrzená zpětná vazba se ukáže konkrétnímu pacientovi v aplikaci pacienta. Na mobilní telefon pacienta přijde notifikace s upozorněním, že byla přidána zpětná vazba.

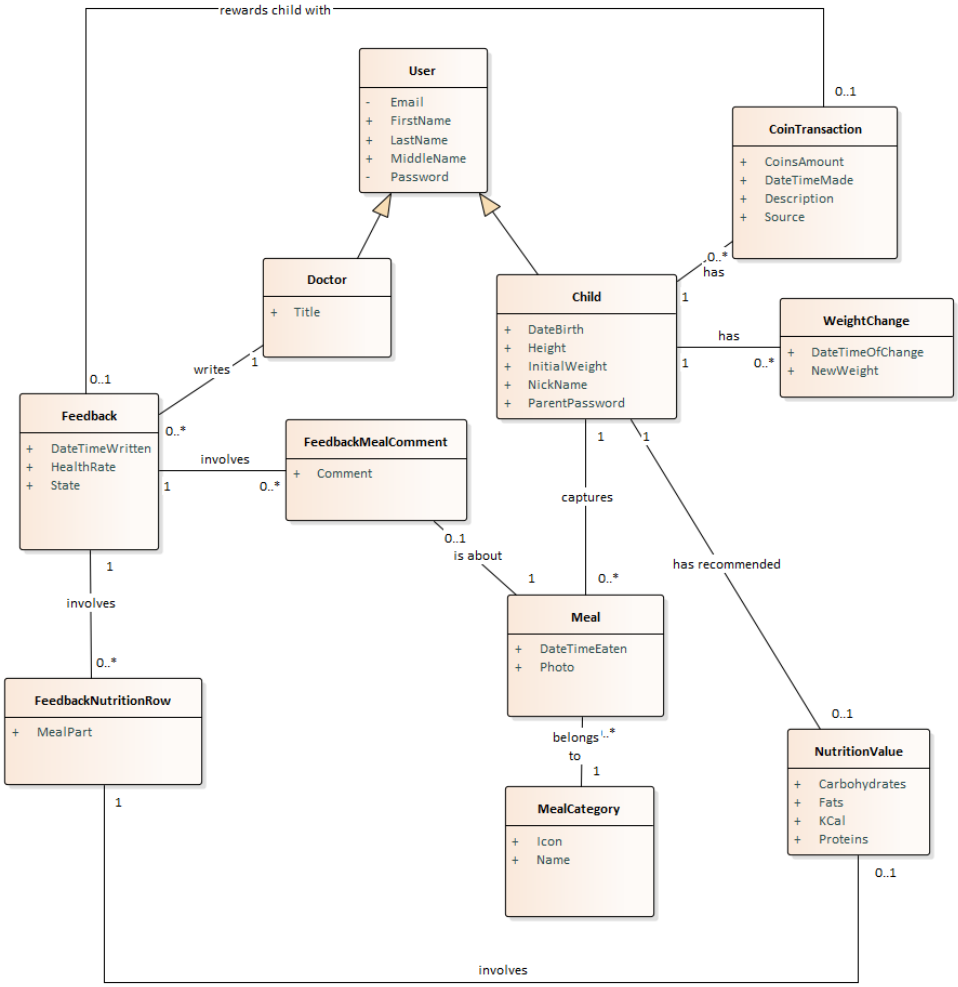
Aktér: Doktor

Hlavní scénář: Potvrzení zpětné vazby

1. Doktor si zobrazí konkrétní zpětnou vazbu.
2. Doktor klikne na tlačítko pro potvrzení zpětné vazby.
3. Systém doktorovi zobrazí potvrzovací okno pro ujištění, zda danou operaci chce skutečně vykonat.
4. Doktor potvrdí svůj záměr kliknutím na potvrzovací tlačítko.
5. Systém potvrdí zpětnou vazbu.
6. Systém odešle na mobilní telefon pacienta notifikaci o nové zpětné vazbě.

3.6 Analytický doménový model

Pro snazší zorientování v analyzovaném systému je vytvořen analytický doménový model. Cílem tohoto modelu je mimo jiné popis dat a vazeb mezi jednotlivými entitami. Dále také slouží jako základ pro návrh databáze a ulehčuje dělání návrhových rozhodnutí. Pro vytvoření modelu je použit diagram tříd z rodiny diagramů UML. Doménový model je zachycen v obrázku 3.4 [12].



Obrázek 3.4: Analytický doménový model

Kapitola 4

Návrh

4.1 Klient-server

Pro realizaci mobilní aplikace pacienta na Android i doktorské aplikace na webu je použita metoda klient-server. V případě aplikace pacienta server poskytuje mobilní aplikaci REST API a pro doktorskou aplikaci je využito MVC (Model-View-Controller).

Pro možnou budoucí rozšiřitelnost na různé mobilní zařízení (případně jiné klienty) se nechá co nejvíce logiky zpracovávat server, aby se pro každého nového klienta nemusela znovu implementovat potřebná logika. Jinak řečeno je zde snaha dodržet koncept DRY (Dont repeat yourself) na úrovni architektury.

4.2 Volba technologií

Tato část se zabývá volbou technologií, které jsou pro realizaci projektu MedaFit zvolené.

Pro vytvoření serverové části aplikace je použita techologie .NET Core s relační databází MSSQL. Důvodem výběru této technologie byl fakt, že technologie dokáže splnit všechny požadavky, jež jsou na serverovou aplikaci a zároveň s ní mám předešlé zkušenosti.

Konkrétně pro REST API a doktorskou aplikaci je použita šablona projektu ASP.NET Core MVC. Front-end doktorské aplikace je realizován pomocí technologie Razor Pages společně s Bootstrapem a JQuery.

Dále byly zvažovány tři možné způsoby, jak vytvořit klienta pro mobilní aplikaci. Vybíráno bylo z následujících třech možností:

- webovou Single Page Application (SPA) napsanou pomocí JavaScriptového frameworku Angular,
- Android aplikaci napsanou v Javě, nebo Kotlinu,
- Android aplikaci napsanou v .NET pomocí technologie Xamarin.

Webová SPA byla zavržena z důvodu obtížné podpory práce s fotoaparátem v mobilních telefonech. Zbyly tedy možnosti pro vývoj nativní Android aplikace buď v Javě, Kotlinu nebo v Xamarinu. Všechny tyto možnosti dokáží naplnit požadavky mobilní aplikace. Nakonec byla zvolena z důvodu předchozí zkušenosti technologie Xamarin.

4.2.1 .NET Core

Technologie .NET Core je open source platforma pro vývoj různorodých aplikací na mnoho platforem, o kterou se v první řadě stará společnost Microsoft. Pyšní se především svou schopností fungovat na různých operačních systémech zahrnujících Microsoft Windows, macOS nebo Linux. Typicky se pro vývoj využívá programovací jazyk C# [13].

4.2.2 ASP.NET Core

Technologie ASP.NET Core je framework běžící na platformě .NET Core, jež umožňuje vývoj webových a cloudových aplikací. Mezi hlavní přínosy frameworku patří MVC, Razor Pages, Model Binding a další [14].

4.2.3 Xamarin

Xamarin je open source platforma vyvýjená hlavně společností Microsoft určená pro vytváření mobilních aplikací pro operační systémy Android a iOS. Pro vývoj se používá prostředí .NET společně s programovacím jazykem C# [15].

4.2.4 Entity Framework Core

Pro práci s relační databází je použito objektově-relační mapování (ORM) realizováno pomocí technologie Entity Framework Core.

Tato technologie umožňuje namapovat objekty z objektově orientovaného programování do relací v relační databázi. K vytvoření databázového schématu je použita strategie Code First, kde jsou nejprve vytvořeny entitní třídy, jejichž jednotlivé atributy mají speciální anotace. Následně se pomocí databázových migrací aktualizuje databáze a vytvoří se požadované schéma. K dotazování se na data a manipulaci s nimi se používá technologie Language Integrated Query (LINQ) [16][17].

4.2.5 Razor Pages

Razor Pages je technologie umožňující snadnou tvorbu dynamických webových stránek ve frameworku ASP.NET Core. Obsahuje syntax pro vkládání dat přímo do HTML. V architektuře MVC se stará o generování pohledů [18].

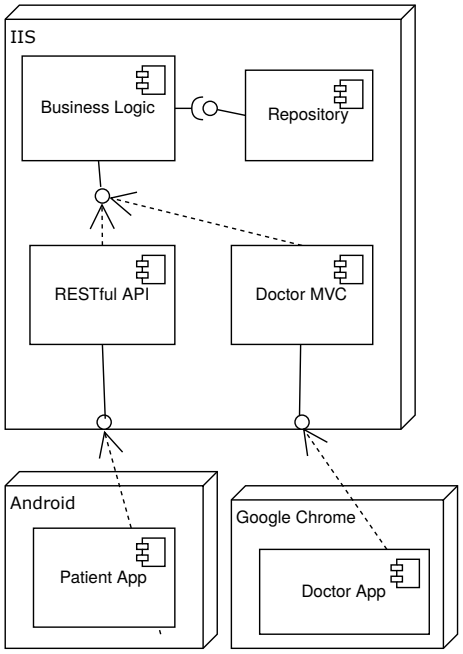
4.2.6 JQuery

JQuery je knihovna napsaná v programovacím jazyce JavaScript, která usnadňuje manipulaci s HTML stránkami. Knihovna obsahuje funkcionality pro práci s jednotlivými elementy na stránce, událostmi, asynchronními voláními a mnoho dalších [19].

4.2.7 Bootstrap

Bootstrap je HTML, CSS a JavaScriptový framework pro vytváření responzivních webových stránek. Pro stylování jednolivých elementů na webové stránce se primárně používá HTML atribut class [20].

4.3 Komponenty



Obrázek 4.1: Komponenty

Celý systém MedaFit je rozdělen do šesti komponent. Komponenty Repository, Business Logic, REST API a Doctor MVC budou nasazeny na straně serveru. Komponenta Patient App (odpovídá aplikaci pacienta) poběží na systému Android a komponenta Doctor App (odpovídá doktorské aplikaci) poběží ve webovém prohlížeči. Komponenty jsou zachyceny v diagramu v obrázku 4.1.

4.3.1 Repository

Komponenta Repository představuje datovou vrstvu aplikace. Pomocí ORM (Object Relational Mapping) má jako jediná z komponent přístup k relační databázi MSSQL.

Technologie komponenty je .NET Core společně s Entity Framework Core. Od statních komponent je oddělena definovaným rozhraním.

4.3.2 Business Logic

Komponenta Business Logic obsahuje veškerou business logiku aplikace. Je závislá na komponentě Repository, která ji umožňuje přístup k datům.

Pro definici rozhraní business logiky se používá návrhový vzor fasáda. Pro komponentu je použita technologie .NET Core.

4.3.3 REST API

Komponenta REST API představuje z pohledu serveru prezentační vrstvu, jež využívá aplikace pacienta pro komunikaci se systémem. Je závislá na komponentě Business Logic.

Rozhraní je definováno pomocí URL adres. Technologií této komponenty je ASP.NET Core MVC.

4.3.4 Doctor MVC

Komponenta Doctor MVC slouží k obsluze doktorské aplikace. Je závislá na komponentě Business Logic.

Rozhraní je definováno pomocí URL adres. Technologií této komponenty je ASP.NET Core MVC.

4.3.5 Patient App

Komponenta Patient App představuje aplikaci pacienta určenou pro mobilní telefony s operačním systémem Android. Je závislá na komponentě REST API.

Komponenta má grafické uživatelské rozhraní takové, aby jej dokázaly snadno ovládat děti. Technologií této komponenty je Xamarin Android.

4.3.6 Doctor App

Komponenta Doctor App představuje doktorskou aplikaci určenou pro webové prohlížeče. Je závislá na komponentě Doctor MVC.

Komponenta má grafické uživatelské rozhraní v podobě HTML webových stránek. Pro realizaci této komponenty jsou použity technologie Razor Pages, HTML, CSS, JavaScript, JQuery a Bootstrap.

4.4 Databáze

Pro ukládání dat jsem zvolil relační databázi MSSQL, ke které se přistupuje pomocí objektově-relačního mapování zprostředkovaného technologií Entity Framework Core.

Jednotlivé tabulky databáze jsou generovány z entitních tříd za pomocí strategie code first, jež funguje tak, že nejprve vytvoříme entitní třídy, kterým přidáme potřebné anotace a tabulky do databáze necháme vygenerovat. Úprava databázového schématu probíhá formou databázových migrací.

Pro cizí klíče, jež jsou často využívané k spojování jednotlivých tabulek, jsou vytvořené indexy.

4.5 Cíle návrhu

Výše uvedená architektura a návrh má za cíl udělat systém přehledným, snadno rozšiřitelným a testovatelným.

Ačkoliv bude z počátku vývoje dodržování architektury pracnější, ve fázi údržby by měla umožnit snadný rozvoj či opravy chyb. Vzhledem k tomu, že jsou k systému už teď vymyšlena rozšíření, je žádoucí stanovit pevné návrhové základy.

Všechny technologie, jež jsou k vývoji použité patří do rodiny .NET Core a v případě aplikace pacienta se jedná o Mono (open source software umožňující použití .NET technologií v operačním systému Android). Díky tomu lze psát téměř veškerý kód, který neslouží k strukturování prezentační vrstvy, jen za použití programovacího jazyka C#.

Kapitola 5

Implementace

5.1 Vývojové prostředí

Vzhledem k tomu, že drtivá většina napsaného kódu je v programovacím jazyce C# a technologii .NET Core, jako primární IDE je zvoleno Visual Studio 2019 s balíčky pro vývoj webů a aplikací v technologii Xamarin.

Pro práci s databází je použito Microsoft SQL Server Management Studio a pro její běh lokální SQL Server.

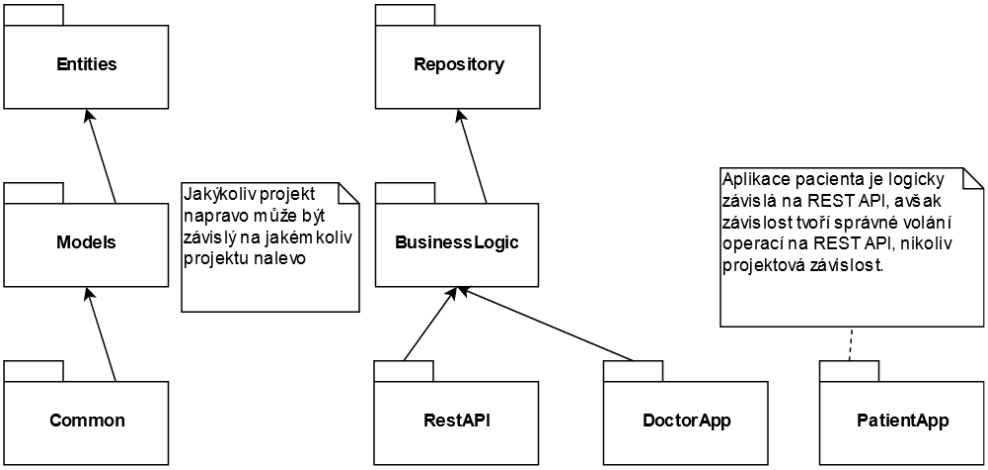
Lokální spuštění webových aplikací (REST API, Doctor MVC) je obstaráno pomocí IIS Express. Pro aplikaci pacienta byl použit Android emulátor.

5.2 Struktura projektů

Prvním úkolem při implementaci bylo založení projektů a určení jejich závislostí. Pro založení projektů se vycházelo z návrhu komponent z kapitoly 4.3. Ze samotných komponent bylo založeno pět projektů.

- **PatientApp** — projekt představující aplikaci pacienta, technologií je Xamarin.Android
- **DoctorApp** — projekt představující doktorskou aplikaci, technologií je ASP.NET Core MVC, závislý na projektu BusinessLogic
- **RestAPI** — projekt poskytující REST API pro projekt PatientApp, závislý na projektu BusinessLogic
- **BusinessLogic** — projekt představující business logiku, technologií je .NET Core, závislý na projektu Repository
- **Repository** — projekt představující datovou vrstvu, technologií je .NET Core

Dále bylo potřeba přidat projekty pro entitní třídy, modely a sdílenou logiku celého systému. Všechny ostatní projekty mohly být na těchto projektech závislé. Nastala zde situace, kdy na jednom projektu je závislá .NET Core aplikace a zároveň Mono aplikace. Tyto sdílené projekty jsou založeny v technologii .NET Standard, která tuto závislost umožňuje.



Obrázek 5.1: Projekty a jejich závislosti

Dohromady se tedy nakonec založilo osm projektů a byly jim určeny závislosti podle návrhu. Výsledné projekty lze vidět v obrázku 5.1.

5.3 Configuration Management

Pro configuration management je zvolena platforma Azure DevOps [21], která je nakonfigurovaná pro použití systému správy verzí git.

Vzhledem k tomu, že se o celý počáteční vývoj staral jeden vývojář, nebylo potřeba dodržovat žádná pravidla pro práci s gitem. S přechodem do stavu údržby se však tato situace změní a mým doporučením je používat jednu hlavní větev (master), ve které se budou objevovat pouze jednotlivé release. Další větev bude sloužit pro slučování změn (develop) a každá jedna chyba či změnové řízení bude vyvíjena ve vlastní soukromé větvi.

Portál Azure DevOps také slouží pro systém jako zdroj dokumentace, který obstarává wiki a tiketovací systém pro správu chyb či změnových řízení. Samotný portál toho umožňuje daleko více, avšak pro projekt MedaFit jsem si vystačil s vyjmenovanými funkcionalitami.

5.4 Dependency injection

Dependency injection je návrhový vzor, který umožňuje inversion of control [22] mezi třídou a jejími závislostmi. Závislostí rozumíme jakýkoliv objekt, jež

třída potřebuje pro svůj běh.

Definice třídy a závislosti

```
// Rozhrani zavislosti
4 references
public interface IZavislost
{
}

// Zavislost tridy MojeTrida
1 reference
public class Zavislost : IZavislost
{
}

// Trida, do niz se vlozi instance zavislosti
1 reference
public class MojeTrida
{
    private readonly IZavislost zavislost;

    0 references
    public MojeTrida(IZavislost zavislost)
    {
        this.zavislost = zavislost;
    }
}
```

Vložení závislosti

Konfigurace dependency injection

```
0 references
public void ConfigureServices(IServiceCollection services)
{
    // konfigurace dependency injection
    services.AddTransient<IZavislost, Zavislost>();
}
```

Obrázek 5.2: Ukázka použití dependency injection

Při vytváření třídy za použití návrhového vzoru dependency injection jsou v dané třídě definovány závislosti většinou schované za rozhraní. Dále se nakonfiguruje, jaké instance závislostí se mají za rozhraní přiřadit. Závislosti jsou určeny jako parametry konstruktoru v dané třídě a dependency injection vloží (tzv. „nainjektuje“) dané instance, když instancuje samotnou třídu. Použití tohoto návrhového vzoru lze vidět na obrázku 5.2 [23].

Technologie .NET Core, jež je využita téměř ve všech projektech, dependency injection podporuje a v aplikaci MedaFit je využita na mnoha místech.

5.5 Realizace jednotlivých vrstev

V následujících částech je popsáno, jak se postupovalo při implementaci jednotlivých vrstev aplikace a také je zde popis technik a návrhových vzorů, jež jsou v daných vrstvách použity.

5.5.1 Datová vrstva

Datová vrstva se nachází v projektu Repository a jejím účelem je přístup k datům či jejich zápis. Jako jediná z vrstev má přístup do relační databáze a to skrze technologii Entity Framework Core.

Pro každou část (modul) aplikace MedaFit jsou vytvořeny rozhraní repozitáře I*Nazev*Repository a jeho implementaci *Nazev*Repository, kde *Nazev* označuje danou část aplikace. Příkladem může být část aplikace manipulující s doktory (správa doktorů v doktorské aplikaci), kde je vytvořeno rozhraní IDoctorRepository a jeho implementaci DoctorRepository. Pro každou repository třídu je nakonfigurováno v datové vrstvě dependency injection.

Pro projekt Repository je použita následující struktura:

- **DbContexts** — definice databázových kontextů pro Entity Framework Core
- **Interfaces** — rozhraní pro jednotlivé repozitáře
- **Migrations** — databázové migrace
- **Repositories** — repozitáře
- **RepositoryDI.cs** — konfigurace dependency injection

5.5.2 Vrstva business logiky

Vrstva business logiky se nachází v projektu BusinessLogic a jejím účelem je schraňovat veškerou business logiku aplikace MedaFit. Pro její realizaci jsou použity návrhové vzory Facade a Command [24].

Každá operace, jež je v systému prováděna, má svou vlastní třídu *Nazev*Operation, kde *Nazev* označuje název operace. Všechny tyto třídy mají jednu veřejnou metodu Execute, která danou operaci provede. Jedná se o návrhový vzor Command s tím, že se třídám dá do názvu postfix Operation.

K ucelení několika operací dohromady do soudržného celku je využit návrhový vzor Facade, který má za úkol poskytnout jednotné rozhraní pro více operací. Ke každé části aplikace existuje rozhraní I*Nazev*Facade a jeho implementace *Nazev*Facade, jež si pomocí dependency injection získá všechny závislé operace a v jednotlivých metodách je provolá.

Pro projekt BusinessLogic je použita následující struktura:

- **Facades** — jednotlivé fasády
- **Interfaces** — rozhraní pro fasády
- **Operations** — jednotlivé operace
- **BusinessLogicDI.cs** — konfigurace dependency injection

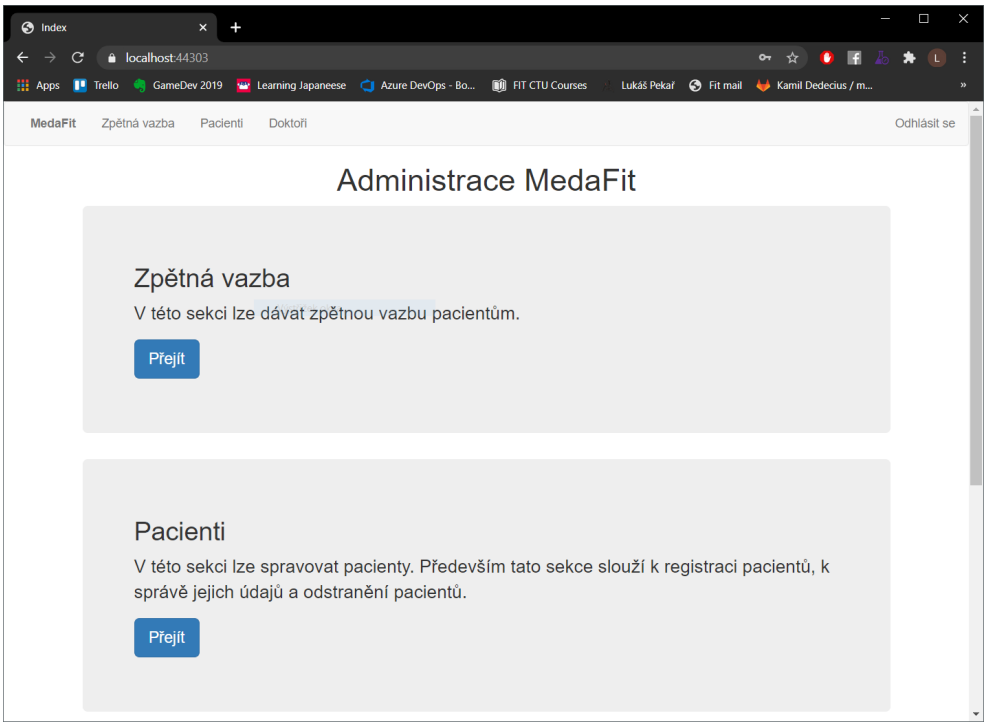
5.5.3 REST API

REST API bylo vytvořeno jako ASP.NET Core Web API, do nějž byly přidány kontrolery z šablony MVC. Tyto kontrolery a definovaný routing určují, jaké URL se budou volat pro které akce. Do každého kontroleru se pomocí dependency injection vloží instance potřebné fasády, které pro REST API zajistí všechny potřebné informace.

5.5.4 Prezentační vrstva

Prezentační vrstvy jsou z pohledu celého systému dvě. První je aplikace pacienta na operační systém Android a tou druhou doktorská aplikace fungující jako webová stránka. Zatímco doktorská aplikace je se serverovou částí spojena v jednom projektu pomocí technologie ASP.NET Core MVC, aplikace pacienta komunikuje s poskytovaným REST API.

5.5.4.1 Doktorská aplikace

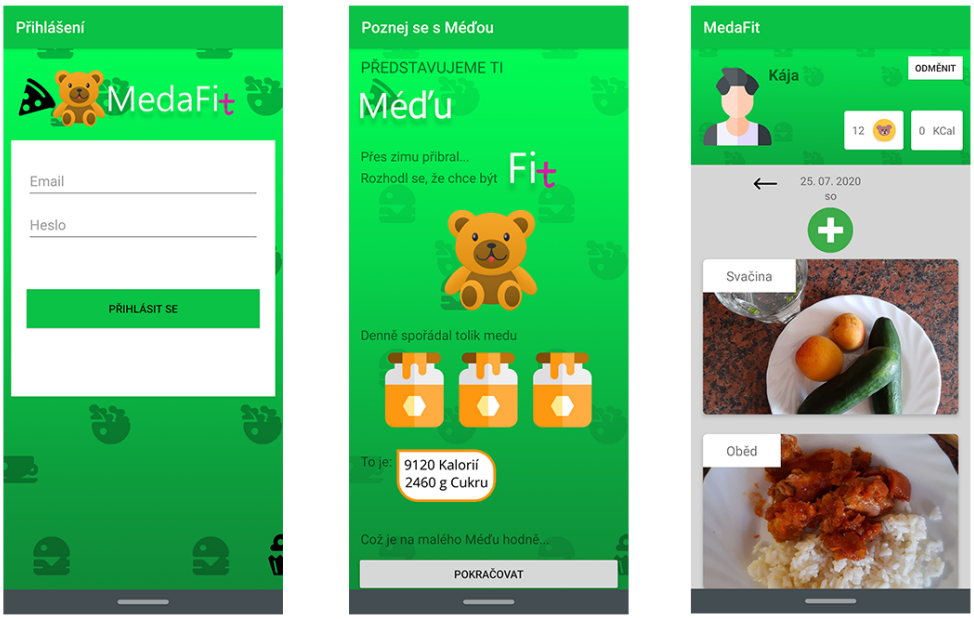


Obrázek 5.3: Ukázka doktorské aplikace

Pro realizaci prezentační vrstvy doktorské aplikace jsem použil technologii Razor Pages společně s HTML, CSS, JavaScriptu, JQuery a Bootstrap. Jednotlivé obrazovky jsou udělány jako pohledy v architektuře MVC.

Cílem při vytváření této vrstvy bylo udělat doktorskou aplikaci co možná nejjednodušší a nejpřívětivější pro doktory a případně administrátory. Ukázku doktorské aplikace lze vidět v obrázku 5.3.

5.5.4.2 Aplikace pacienta



Obrázek 5.4: Ukázka aplikace pacienta

Prezentační vrstva aplikace pacienta zahrnuje grafické uživatelské rozhraní určené pacientům. Dle nefunkčních požadavků viz kapitola 3.4.2, konkrétně požadavku NP-P-2, má být ovladatelná dítětem. Pro uskutečnění požadavku bylo uživatelské rozhraní navrženo co možná nejjednodušeji.

Při návrhu rozhraní se vycházelo z práce *Vytvoření business modelu projektu dětská obezita*, kterou vypracovala Bc. Andrea Holoubková [1]. Byla zachována barevná paleta, pozadí, logo a velká část rozvržení obrazovek. Z praktických důvodů se však některé obrazovky změnily.

Obrazovky na Android jsou vytvořeny pomocí XML souborů, napsaných ve Visual Studio Xamarin.Android Designer [25]. Ukázku obrazovek lze vidět na obrázku 5.4.

5.6 Autentizace & Autorizace

Tato kapitola se věnuje realizaci autentizace a autorizace. Jak přístup do aplikace pacienta, tak do doktorské aplikace musí být zabezpečený dle funkčních

požadavků FP-P-1 a FP-D-1 z kapitoly 3.4.2. Pro implementaci dané funkcionality je využita technologie Microsoft Identity Platform [26].

Pro uživatele, role, přiřazení rolí k uživatelům atd. byly do databáze vygenerovány tabulky pomocí Entity Framework Core. Pro definice entitních tříd stačilo stáhnout správný NuGet balík. Jiný NuGet balík zase umožnil využití tříd UserManager a SignInManager pomocí dependency injection, jež jsou využity k realizaci přihlašování, registrace a změny hesla jak u doktorů, tak u pacientů.

Pro doktory a administrátory jsou vytvořeny role. Doktorům je přiřazena role doktora, jakmile administrátor doktora zaregistruje. Administrátorskou roli nelze získat jinak, než manuálním přiřazením ze strany vývojáře.

5.6.1 Autentizace na aplikaci pacienta

Pro realizaci autentizace v aplikaci pacienta je využita technologie přístupových tokenů, jež se posílají s každým HTTP požadavkem.

Pokud se na aplikaci pacienta nenachází přístupový token, je přesměrován na přihlašovací obrazovku. Po zadání platného emailu a hesla se vygeneruje přístupový token s určenou expirací a odešle se na aplikaci pacienta, která si jej uloží pro další komunikaci s REST API.

```
3 references
public HttpResponseMessage Get(string uri)
{
    var result = httpClient.GetAsync(uri).Result;

    if (TokenExpired(result))
    {
        bool success = RefreshToken();

        if (success)
        {
            return Get(uri);
        }
        else
        {
            DeleteTokens();
            throw new RefreshTokenFailedException();
        }
    }

    return result;
}
```

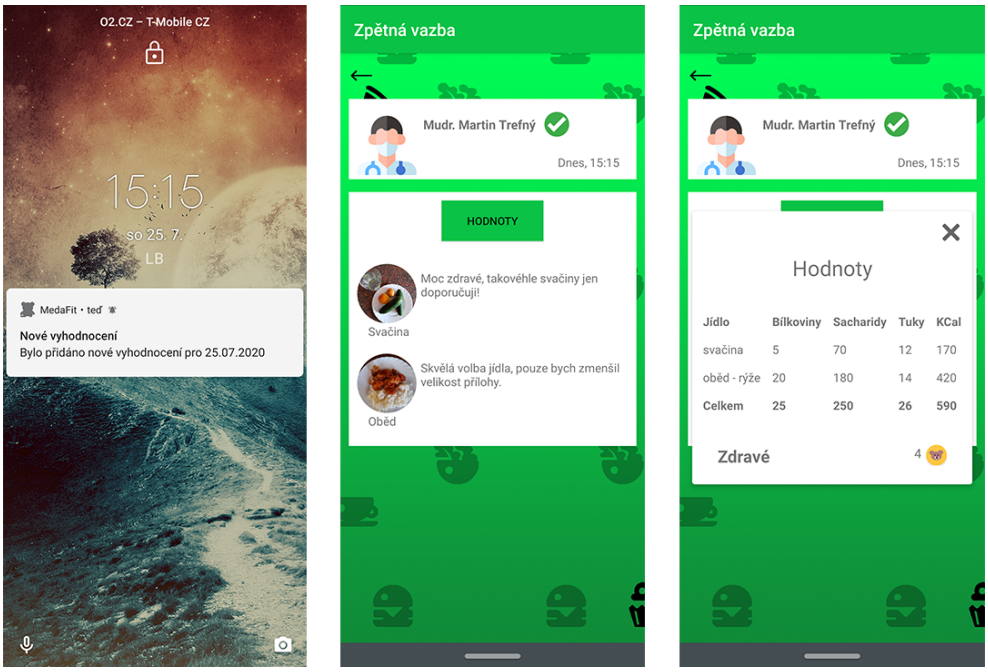
Obrázek 5.5: Ukázka metody Get ve třídě HttpClientWrapper

Vzhledem k tomu, že aplikaci budou pacienti využívat několikrát za den hlavně k focení jídel, je velmi žádoucí, aby se nemuseli pro každé využití apli-

kace, případně po každé expiraci přístupového tokenu, znovu přihlašovat. Toho se docílilo pomocí obnovovacího tokenu, jež je vygenerován ve stejnou chvíli jako přístupový token. V případě, že na REST API přijde HTTP požadavek s expirovaným přístupovým tokenem, REST API vrátí specifickou odpověď, jež označuje expirovaný přístupový token. Aplikace pacienta následně může získat nový přístupový token tím, že na REST API pošle svůj expirovaný přístupový token a také obnovovací token. Pokud jsou oba tokeny správné, REST API vygeneruje nový přístupový token a ten pošle na aplikaci pacienta. Ta poté může zopakovat akci, jež chtěla provést před expirací přístupového tokenu [27].

Pro přístupové a obnovovací tokeny je využita technologie JSON Web Tokens [28]. Implementace obnovování tokenů se vyřešila novou třídou HttpClientWrapper, jež tvoří obal (wrapper) pro třídu HttpClient (třída, jež se typicky využívá pro HTTP komunikaci) a která se využívá pro všechna volání na REST API. [29]. V třídě HttpClientWrapper byly znovu implementovány metody pro Get, Post, Put a Delete, tak aby v případě, že přístupový token expiroval, poslaly na REST API požadavek pro jeho obnovu a požadavek zopakovaly. Ukázka kódu z třídy HttpClientWrapper pro metodu Get lze vidět v obrázku 5.5.

5.7 Notifikace



Obrázek 5.6: Ukázka notifikace a zpětné vazby

Poté co doktor potvrdí zpětnou vazbu pro určitého pacienta, tak danému pacientovi přijde na jeho mobilní telefon notifikace. Těmto notifikacím se říká push notifikace a v aplikaci MedaFit jsem je realizoval pomocí technologie Firebase Cloud Messaging [30].

K rozeznání jednotlivých zařízení Firebase Cloud Messaging používá tzv. registrační tokeny. Registrační tokeny jsou typicky vytvořeny a uloženy při prvním spuštění aplikace. Mohou se však v průběhu používání aplikace kdykoliv obnovovat. Při implementaci notifikací v aplikaci pacienta bylo potřeba, aby se získaný registrační token uložil do mezipaměti v případě, že nebyla provedena řádná autentizace. Pokud autentizace provedena byla a daný registrační token ještě nebyl uložen na straně serveru, odešle se na REST API požadavek k uložení registračního tokenu.

Když přijde na pacientův mobilní telefon notifikace, tak se po kliknutí na ni otevře aplikace pacienta na zpětné vazbě, o níž notifikace informovala. Notifikaci a následně zobrazenou zpětnou vazbu ukazuje obrázek 5.6.

5.8 Prostředí

Vývoj aplikace podobné velikosti jako MedaFit obnáší vytvoření různých prostředí, které každé slouží k jinému účelu. Pro systém MedaFit byly používány tři prostředí: lokální, testovací a akceptační.

- **lokální prostředí** — odpovídá vývojovému prostředí viz kapitola 5.1
- **testovací prostředí** — prostředí, kde doktorská aplikace a REST API běží na IIS a také, kde jsou tyto aplikace přístupné z internetu. Toto prostředí slouží k testování aplikace v reálných podmínkách.
- **akceptační prostředí** — prostředí, kde doktorská aplikace a REST API běží na IIS a také, kde jsou tyto aplikace přístupné z internetu. Toto prostředí slouží k akceptačnímu testování ze strany zákazníka.

Všechna zmíněná prostředí včetně přístupů jsou popsána ve wiki na portálu Azure DevOps.

5.9 Testování

Po počátečním vývoji aplikace přišlo na řadu testování. Pro účely testování je vytvořeno testovací prostředí viz kapitola 5.8. Jako hlavní zdroj byl pro testování použit model případů užití, kde byl každý jeden případ užití vyzkoušen, zda skutečně funguje. Dané chyby byly zaznamenány na portál Azure DevOps a následně opraveny.

Dále je doporučeno provést akceptační testování prováděné samotnými klienty na akceptačním prostředí.

Kapitola 6

Zhodnocení a možná vylepšení

6.1 Zhodnocení

Výsledek práce, i přesto, že je na něm spousta možností, jak jej vylepšit, jde správným směrem k udržovatelnému a rozšiřitelnému systému, který společnost Medasol může začít využívat pro léčbu dětské obezity.

Z architektonického pohledu se jedná o systém skládající se ze tří vrstev: prezentační vrstva, business logika a datová vrstva. Účel vrstev byl po celou dobu dodržován a díky tomu se zajistila udržovatelnost, rozšiřitelnost a testovatelnost.

Mobilní aplikace pacienta je navržena tak, aby jej dokázaly ovládat děti. Intuitivní a jednoduchá ovladatelnost společně s přehledným uživatelským rozhraním zajišťuje snadné používání systému. Vzhledem k tomu, že mobilních zařízení s operačním systémem Android 5.0 nebo vyšší je nepřeberné množství, mobilní aplikaci nebylo možné otestovat za všech podmínek. Pro vývoj byly použity emulátory pro Android 5.0, Android 10.0 a reálná zařízení Motorola One Vision s operačním systémem Android 10.0 a Huawei P10 Lite s operačním systémem Android 8.0.

Administrační doktorská aplikace zajišťuje přehledný portál, ve kterém lze snadno spravovat pacienty, doktory a zpětné vazby. Cílem administrace bylo umožnit doktorům snadné zadávání zpětných vazeb směrem k pacientům, a proto byl největší důraz kladen právě na editor zpětné vazby obsahující nutriční tabulku, komentáře k jídlům, vyhodnocení a potvrzení zpětné vazby. Z hlediska funkcionalit tedy doktorská aplikace odpovídá zadaným požadavkům a je zcela použitelná. Potencionálním problémem této části systému mohou být použité technologie Bootstrap 3 a JQuery, které někteří již považují za zastaralé.

Největším nedostatkem této práce je absence akceptačního testování, které z časových důvodů nebylo provedeno. Testování proběhlo pouze na lokálním vývojovém prostředí a na testovacím prostředí, kde v obou případech prováděl testy vývojář.

6.2 Budoucí vylepšení

Výsledná aplikace MedaFit představuje hlavní jádro celého potencionálního systému a to sdílení jídel od pacientů směrem k doktorům. Zároveň také umožňuje doktorům dávat pacientům zpětnou vazbu a ohodnocení v podobě virtuálních mincí.

Jedním z hlavních kroků pro budoucí vývoj je realizace aplikace pacienta na operačním systému iOS, který běží na mobilních telefonech značky Apple.

Do budoucna se plánují rozšíření v podobě detailnější zpětné vazby k jednotlivým jídlům, jež zahrnují kontrolu velikosti porce jídla, správnost četnosti jídla či zda pacient jí ve správnou hodinu [1].

Dalším budoucím rozšířením je zvýšení gamifikace. Toho lze docílit pomocí odměňování pacienta po každém vyfocení jídla, což má motivovat pacienty k zodpovědnému zaznamenávání jejich denního příjmu. Dále lze navázat gamifikaci na váhu pacienta. Každá zaznamenaná pozitivní změna váhy přidá pacientovi virtuální mince. Zde je ovšem třeba brát na vědomí, že pacienti jsou děti, u kterých příbytek na váze nemusí být vždy špatně [1].

Mezi nejzajímavější budoucí vylepšení patří schopnost aplikace sama poznat, zda je jídlo zdravé či nikoliv pomocí napojení na strojové učení. Taková aktualizace by doktorům mohla značně usnadnit práci s identifikací zdravých a nezdravých jídel [1].

Posledním plánovaným vylepšením je realizace reportů pro jednotlivé pacienty, ve kterých by bylo na první pohled vidět, jak si pacient v čase vede co se týče jeho váhy, získaných mincí a případně jiných hodnot.

Závěr

Cílem práce bylo vyvinout mobilní aplikaci pro pomoc léčby dětské obezity u společnosti Medasol. Po vývoji přišlo na řadu otestování aplikace a její připravení k nasazení. Součástí práce bylo také provést dokumentaci pro budoucí vývojáře, jež budou na tomto projektu dále pracovat. V rešeršní části byly popsány metody, jak vývoj takového softwaru realizovat a zároveň jsou zde vysvětleny pojmy, se kterými pracuji v dalších částech práce.

V analytické části je sepsáno zvolené řešení, jež navazuje na diplomovou práci *Vytvoření business modelu projektu dětská obezita* od Bc. Andrei Holoubkové včetně funkčních a nefunkčních požadavků. Následně tato část obsahuje diagram případu užití a doménový model [1].

Práce pokračuje návrhem mobilní aplikace. V první řadě je popsána obecná architektura řešení, která slouží společně s požadavky jako základ pro výběr technologií, ve kterých je aplikace zhotovena. Návrh je také obohacen diagramem komponent, jež jsou každá samostatně popsány.

S hotovým návrhem přišel čas pro samotný vývoj. Zde hrála roli správná realizace systému s ohledem na návrh. Jsou zde popsány jednotlivé hlavní funkcionality systému (autorizace, notifikace, atd.) a také prostředí, ve kterých běží.

Další vývoj by do aplikace mohl vnést více prvků gamifikace a možnost využívat MedaFit i na mobilních telefonech s operačním systémem iOS. Je také v plánu automatické rozpoznávání zdravých či nezdravých jídel pomocí technologie strojového učení. V případě většího využití mobilní aplikace by bylo dobré provést výzkum, zda je takový způsob léčby obezity skutečně efektivní.

Výsledným produktem je funkční mobilní aplikace MedaFit, která odpovídá zadání a provedené analýze. Všechny cíle pro danou práci byly splněny. Jediným nedostatkem práce je absence akceptačního testování, jež nebylo z časových důvodů provedeno.

Bibliografie

1. HOLOUBKOVÁ, Andrea. *Vytvoření business modelu projektu dětská obezita*. 2020. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
2. MLEJNEK, Jiří. *Metodiky Vývoje [Přednáška]*. letní semestr 2019.
3. HLAVATÝ, Martin. *Softwarový proces [Přednáška]*. zimní semestr 2019.
4. WASSERMAN, Anthony I. Software Engineering Issues for Mobile Application Development. In: *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*. Santa Fe, New Mexico, USA: Association for Computing Machinery, 2010, s. 397–400. FoSER '10. ISBN 9781450304276. Dostupné z DOI: 10.1145/1882362.1882443.
5. ISLAM, Rashedul; ISLAM, Rofiqul; MAZUMDER, Tahindul. Mobile application and its global impact. *International Journal of Engineering & Technology (IJEST)*. 2010, roč. 10, č. 6, s. 72–78.
6. MLEJNEK, Jiří. *Analýza a sběr požadavků [Přednáška]*. letní semestr 2019.
7. *Client-Server Model* [online] [cit. 2020-07-19]. Dostupné z: https://techterms.com/definition/client-server_model.
8. LIEW, Zell. *Understanding And Using REST APIs* [online]. 2018 [cit. 2020-07-23]. Dostupné z: https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/.
9. SMITH, Steve. *Overview of ASP.NET Core MVC* [online]. 2020 [cit. 2020-07-23]. Dostupné z: https://docs.microsoft.com/en-gb/aspnet/core/mvc/overview?view=aspnetcore-3.1.
10. SAHOO, Krushnapriya; SAHOO, Bishnupriya; CHOUDHURY, Ashok Kumar; SOFI, Nighat Yasin; KUMAR, Raman; BHADORIA, Ajeet Singh. Childhood obesity: causes and consequences. *Journal of family medicine and primary care*. 2015, roč. 4, č. 2, s. 187.

11. *MEDASOL: medical access solutions* [online] [cit. 2020-05-02]. Dostupné z: http://medasol.eu/.

12. MLEJNEK, Jiří. *Analýza problémové domény [Přednáška]*. letní semestr 2019.

13. *.NET Core intro and overview* [online]. 2020 [cit. 2020-07-09]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/core/introduction.

14. ROTH, Daniel; ANDERSON, Rick; LUTTIN, Shaun. *Introduction to ASP.NET Core* [online]. 2020 [cit. 2020-07-09]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1.

15. *What is Xamarin?* [online] [cit. 2020-07-29]. Dostupné z: https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin.

16. *Language Integrated Query (LINQ)* [online]. 2017 [cit. 2020-07-29]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/.

17. *Overview of Entity Framework Core* [online]. 2016 [cit. 2020-07-29]. Dostupné z: https://docs.microsoft.com/en-us/ef/core/.

18. ANDERSON, Rick; NOWAK, Ryan. *Introduction to Razor Pages in ASP.NET Core* [online]. 2020 [cit. 2020-07-29]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-3.1&tabs=visual-studio.

19. *jQuery* [online] [cit. 2020-07-29]. Dostupné z: https://jquery.com/.

20. *Bootstrap: The world's most popular mobile-first and responsive front-end framework.* [online] [cit. 2020-07-29]. Dostupné z: https://getbootstrap.com/docs/3.3/.

21. *Azure DevOps* [online] [cit. 2020-07-24]. Dostupné z: https://dev.azure.com/.

22. *Architectural principles* [online]. 2018 [cit. 2020-07-24]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles#dependency-inversion.

23. SMITH, Steve; ADDIE, Scott; DAHLER, Brandon. *Dependency injection in ASP.NET Core* [online]. 2020 [cit. 2020-07-24]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.1.

24. GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0201633612.

25. *Using the Xamarin.Android Designer* [online]. 2018 [cit. 2020-07-24]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/android/user-interface/android-designer/designer-walkthrough?tabs=windows.

26. *Microsoft identity platform documentation* [online] [cit. 2020-07-24]. Dostupné z: https://docs.microsoft.com/en-us/azure/active-directory/develop/.

27. FIGUEIREDO, Rui. *Refresh Tokens in ASP.NET Core Web Api* [online] [cit. 2020-07-24]. Dostupné z: https://www.blinkingcaret.com/2018/05/30/refresh-tokens-in-asp-net-core-web-api/.

28. *JSON Web Tokens - jwt.io* [online] [cit. 2020-07-24]. Dostupné z: https://jwt.io/.

29. *HttpClient Class (System.Net.Http)* [online] [cit. 2020-07-24]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?view=netcore-3.1.

30. *Firebase Cloud Messaging* [online]. 2020 [cit. 2020-07-24]. Dostupné z: https://firebase.google.com/docs/cloud-messaging.

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

CSS Cascading Style Sheets

DRY Don't Repeat Yourself

GUI Graphical User Interface

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

IIS Internet Information Services

MVC Model-View-Controller

REST Representational State Transfer

SPA Single Page Application

URL Uniform Resource Locator

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `diag` ..... adresář s diagramy
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `Pekar_BP_MedaFit.pdf` ..... text práce ve formátu PDF