ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Müller** Jméno: **Jiří** Osobní číslo: **383792**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Konzultační systém pro diabetické pacienty

Název diplomové práce anglicky:

Recommendation system for diabetic patients

Pokyny pro vypracování:

Seznamte se s problematikou diabetických pacientů, zejména typu
1)Proveďte rešerši existujících podobných řešení, které zaznamenávají údaje z inzulínové pumpy a glukometru, případně kontinuálních senzorů, a dále data o pacientově denním režimu (strava, fyzická aktivita). Na základě rešerše kriticky zhodnoťte výhody a nevýhody jednotlivých řešení.
S využitím kritického hodnocení
ad 2) udělejte analýzu požadavků na funkcionality systému. Na základě analýzy navrhněte architekturu systému pro sledování pacientských dat lékařem a pro tvorbu doporučení pro pacienta. Systém má splňovat následující požadavky: efektivní a intuitivní ovládání, zobrazování požadovaných informací, zejména normoglykémie, hypo- a hyperglykémie, souhrn kalorického příjmu a výdeje za definované časové období (volitelné uživatelem - např. den, týden, měsíc); jednoduché vkládání dalších dat (např. upřesnění složení stravy, neměřené fyzické aktivity).
Navržený systém implementujte.
Implementovaný systém otestujte na reálných datech. Výsledky vyhodnoťte.

Seznam doporučené literatury:

[1] Škrha Jan et al. " Diabetologie " Galén, 2009
[2] Štechová Kateřina a kol. " Technologie v diabetologii " Maxdorf, 2016
[3] Journal of Diabetes Science and Technology - selected papers: http://journals.sagepub.com/home/dst

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Lenka Lhotská, CSc., katedra přírodovědných oborů FBMI

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **02.02.2018** Termín odevzdání diplomové práce: **25.05.2018**

Platnost zadání diplomové práce: **30.09.2019**

doc. Ing. Lenka Lhotská, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Konzultační systém pro diabetické pacienty

Bc. Jiří Müller

Vedoucí práce: doc. Ing. Lenka Lhotská, CSc.

24. května 2018

Poděkování

Rád bych poděkoval v první řadě paní prof. MUDr. Kateřině Štechové, Ph.D., bez které by tato práce nejspíš vůbec nevznikla. Její vhled do života diabetiků a práce jejich lékařů byl nedocenitelný. Dále bych rád poděkoval paní doc. Ing. Lence Lhotské, CSc. za příkladné vedení práce a ochotu kdykoliv pomoci.

Rád bych také zmínil grant č. 15-25710A AZV ČR "Identifikace individuální dynamiky glykemických exkurzí u pacientů s diabetem pro zlepšení rozhodovacích postupů ovlivňujících dávkování inzulínu", který zajistil hardware na měření galvanického odporu kůže.

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.

V Praze dne 24. května 2018 .....

České vysoké učení technické v Praze
Fakulta elektrotechnická

Odkaz na tuto práci

Müller, Jiří. *Konzultační systém pro diabetické pacienty.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, 2018.

Abstrakt

Tato práce si klade za cíl analyzovat, navrhnout, implementovat a na reálných datech otestovat konzultační systém pro pacienty s diabetem. Výstupem praktické části práce je prototyp mobilní aplikace, která umožňuje diabetikům nechat si na základě jídelníčku a dalších parametrů doporučit dávku inzulinu. A zároveň webová aplikace pro lékaře, kde je možno sledovat data od pacientů a upravovat parametry pro výpočet dávek.

Klíčová slova

diabetes, zdraví, mobilní aplikace, web, softwarové inženýrství, GSR

Abstract

The goal of this thesis is to analyze, implement and test recommendation system for diabetic patients. The result is prototype of mobile application for patients which can recommend the insulin dosage on the basis of diet and other parameters. And also web appliaction which allows doctors to follow patients' data and set parameters for dosage calculation.

Keywords

diabetes, health, mobile app, web, software engineering, GSR

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Podle ÚZIS[1] se jen v České republice v roce 2016 léčilo s diabetem 929 945 osob tj. asi 8,8% populace. Pacienti jsou odkázáni na léčbu inzulinem a právě jeho správné dávkování je kritické pro účinnou kompenzaci diabetu. Špatná kompenzace může být příčinou celé řady komplikací, jako jsou například kardiovaskulární onemocnění nebo retinopatie[2].

Tato práce si klade za cíl vyvinout aplikaci, která bude schopná pacientům doporučit optimální dávku inzulinu na základě konzumovaného jídla, aktuální glykémie, fyzické aktivity, stresu a mnoha dalších faktorů, které ovlivňují citlivost organismu na inzulin. Aplikace by zároveň měla umožňovat efektivní sdílení dat pacienta přímo s ošetřujícím lékařem, a tak zjednodušit léčbu.

Mojí hlavní motivací při tvorbě této práce je snaha alespoň trochu zlepšit kvalitu života velké části populace, která se musí potýkat s touto nelehkou chorobou.

[1] Ústav zdravotnických informací a statistiky ČR

[2] patologické změny sítnice a jejích cév

Kapitola 1

Analýza

1.1 Analýza problematiky

1.1.1 Diabetes mellitus

Diabetes mellitus (laicky cukrovka) je souhrnný název pro skupinu chronických onemocnění, projevujících se poruchou metabolismu sacharidů. Rozlišujeme dva základní druhy: diabetes I. a diabetes II. typu (viz dále). Jejich společným jmenovatelem je nedostatek (ať už relativní nebo absolutní) hormonu inzulinu. Ten umožňuje glukóze obsažené v krvi vstup do buněk, ve kterých je následně štěpena na jednodušší látky za vzniku energie. Normální hodnota glykemie u zdravého člověka se pohybuje v úzkém rozmezí 4-7 mmol/l [1]. V případě nedostatku inzulinu nedochází ke vstřebávání glukózy z krve, což vede k její zvýšené koncentraci - tzv. **hyperglykémii**. Opakem hyperglykémie je **hypoglykémie**. K té dochází naopak pokud je do krve uvolněno inzulinu příliš. [2] [3]. U zdravého člověka je všechen potřebný inzulin produkován v tzv. beta-buňkách Langerhansových ostrůvků ve slinivce břišní. Jeho uvolňování (spolu s dalšími hormony) do krve je velmi citlivě regulováno tak, aby množství glukózy v krvi zůstalo v normě.

1.1.1.1 Diabetes I. typu

Podstatou diabetu I. typu je zničení beta-buněk Langerhansových ostrůvků autoimunitním zánětem, tj. útok buněk imunitního systému právě proti beta-buňkám. Proč přesně k tomuto útoku dochází stále není známo. [2]

Když jsou beta-buňky zničeny, dojde k výpadku tvorby inzulinu a jedinou možností léčby je zahájení jeho podávání jiným způsobem - injekcí. [1]

1.1.1.2 Diabetes II. typu

Příčiny vzniku diabetu II. typu jsou odlišné. Hlavní úlohu hrají dva jevy: porucha citlivosti buněk na inzulin (tzv. inzulinorezistence) a porucha produkce

inzulinu v beta-buňkách Langerhansových ostrůvků (tzv. inzulinodeficience). Inzulinorezistence znamená, že k dosažení stejného efektu je potřeba inzulinu mnohem více. Zhoršení jedné poruchy navíc vede ke zhoršení druhé. [1] Hlavními riziky, která vzniku tohoto typu cukrovky předchází, jsou přejídání, nedostatek pohybu a obezita. [3]

1.1.2 Léčba inzulinem

> *„Léčba inzulinem je jedinou a nezastupitelnou léčbou nemocných s diabetem I. typu, která je léčbou život zachraňující. Pokud klesá endogenní sekrece inzulinu, musí být nahrazena inzulinem podávaným exogenně.“*[2]

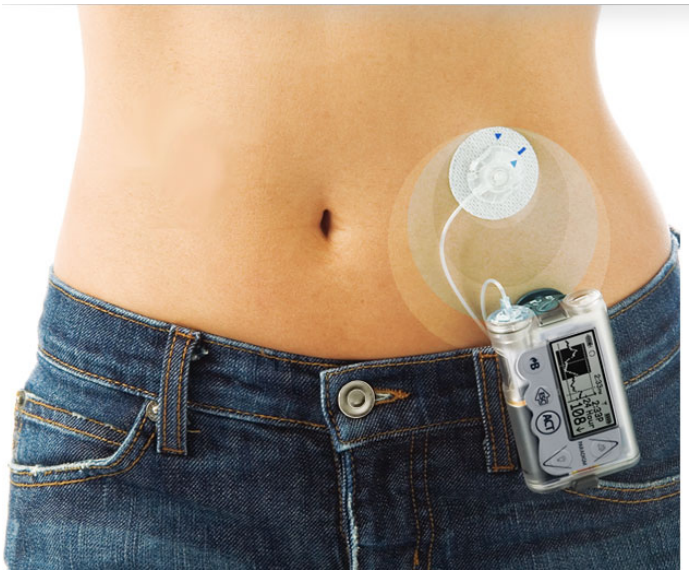
V zasadě existují dvě metody aplikace: injekční stříkačkou (inzulinové pero) a nebo inzulinovou pumpou. V případě použití injekcí si musí pacient několikrát denně – obvykle před jídlem – „ručně“ aplikovat dávku rychle působícího inzulinu. Mimo to si také musí pravidelně aplikovat pomalu působící inzulin na pokrytí bazální potřeby organismu[1].

Ruční aplikace injekcí se dá nahradit tzv. inzulínovou pumpou. Toto zařízení, velké asi jako mobilní telefon, obsahuje zásobník inzulinu a je napojeno na katetr, zakončený jehlou zavedenou do podkoží (obr. 1.1). To umožňuje dávkovat inzulin přímo do těla pacienta. Ten si může na pumpě nastavit jednak jednorázový bolus např. před jídlem, ale také stálé dávkování malého množství inzulinu na pokrytí bazální potřeby[1].

1.1.3 Průběh aplikace

Před každým jídlem by měl pacient provést následující kroky:

1. Změření aktuální glykémie glukometrem
2. Stanovení množství sacharidů v jídle, které se chystá konzumovat – tento bod bývá často „kamenem úrazu“. Ukazuje se, že pacienti často nesprávně odhadnou hmotnost porce a nebo množství sacharidů obsažené v konkrétní potravině. Toto je jeden z hlavních problémů, které se naše aplikace snaží řešit.
3. Stanovení množství inzulinu, který pokryje potřebu organismu pro dané jídlo – další problém, který by měla aplikace řešit.
4. Samotná aplikace inzulinu (ať injekčně nebo pomocí inzulinové pumpy) - probíhá obvykle asi 15 minut před jídlem (podle druhu inzulinu)[1]
5. Konzumace potraviny



Obrázek 1.1: Inzulinová pumpa [4]

1.2 Rešerše existující řešení

1.2.1 Aplikace pro monitoring jídelníčku

1.2.1.1 Calorie Counter by FatSecret

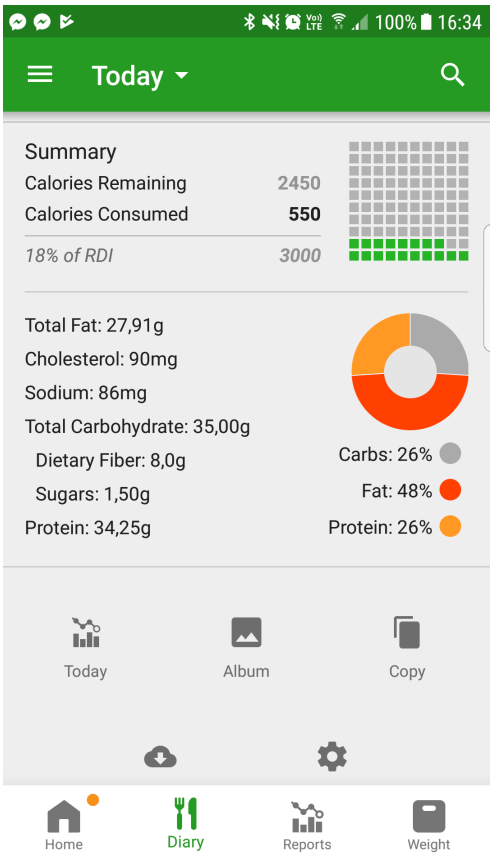
Mobilní aplikace navázaná na databázi potravin FatSecret[5], primárně zaměřená na úpravu tělesné hmotnosti. Po vyplnění základních údajů o uživateli (výška, váha, věk), umožňuje zaznamenávat jídla a fyzickou aktivitu (obr. 1.3). Ze zadaných údajů počítá kalorický příjem a zobrazuje nutriční hodnoty (obr. 1.2).

Výhody

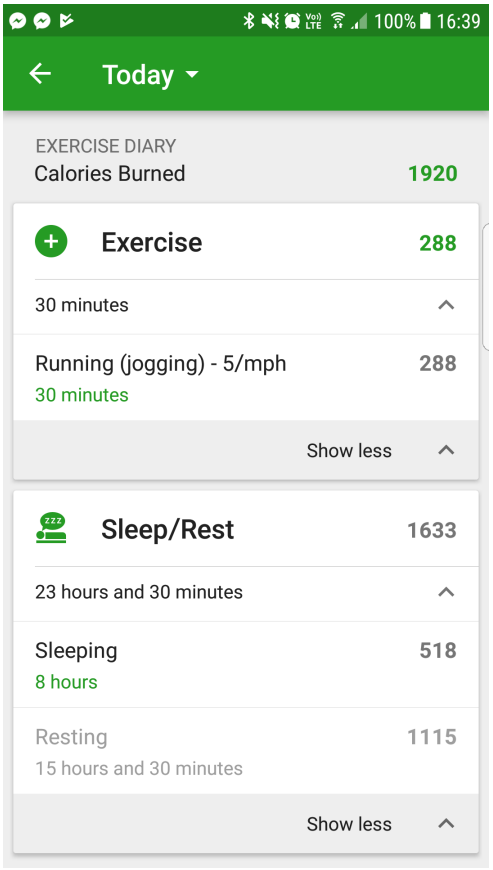
- Rozsáhlá databáze obsahuje i některé české potraviny
- Bez reklam a zdarma
- Možnost sdílet dietu, cvičení a váhu s lékařem nebo dietetikem

1.2.1.2 KalorickéTabulky.cz

České mobilní aplikace navázaná na stejnojmenný web. Funkcionalita je jinak velice podobná *Calorie Counter by FatSecret*. Mimo tělesné hmotnosti umožňuje zaznamenávat i procento tuku, obvod pasu, obvod boků a další. Zobrazovaný přehled a zadávání fyzické aktivity ukazují obr. 1.4 a 1.5.



Obrázek 1.2: FatSecret: Přehled



Obrázek 1.3: FatSecret: Fyzická aktivita

Výhody

- Databáze českých potravin
- Zdarma, ale obsahuje reklamy[3]

Nevýhody

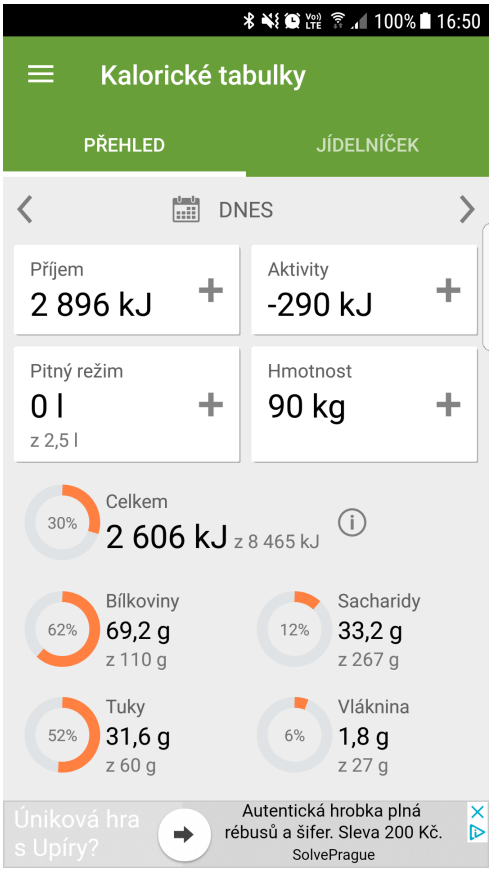
- Neumožňuje sdílet data s lékařem

1.2.2 Bolus kalkulátory

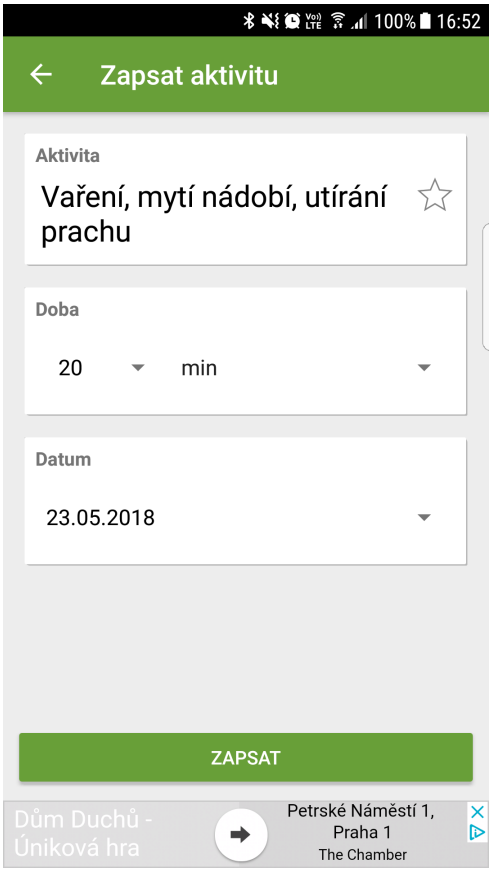
1.2.2.1 Diabetes:M

Diabetes:M[6][7] je jedna z pokročilých aplikací pro monitoring diabetu. V základu je zdarma, ale pro pokročilé funkce je nutné předplatné[4]. Výpočet

[3] reklamy je možné odstranit za jednorázový poplatek cca 95 Kč
[4] cca 150 Kč na měsíc nebo 1 500 Kč na rok



Obrázek 1.4: Kal.Tabulky.cz: Přehled



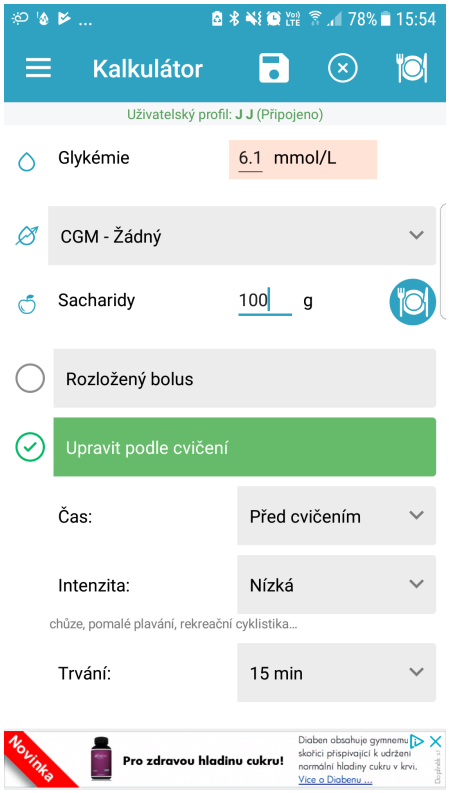
Obrázek 1.5: Kal.Tabulky.cz: Aktivita

bolusu a vyhledávání jídla v mobilní aplikaci zobrazují obrázky 1.6 a 1.7.

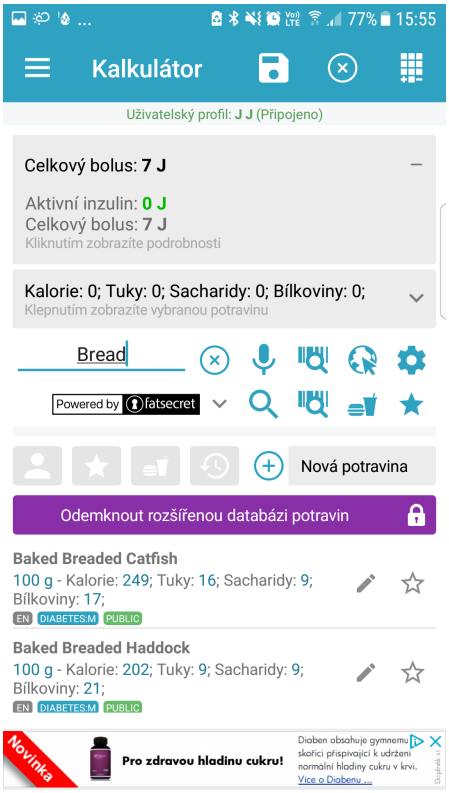
Výhody

- Při výpočtu je možno zadat velké množství informací
- Zahrnutí bílkovin a tuků do výpočtu
- Online databáze potravin[5]
- Možnost zadat konkrétní značku pomalého a rychlého inzulinu
- Možnost sdílet data s lékařem - prémiová funkce
- Připojení Bluetooth glukometrů - prémiová funkce

[5] FatSecret.com[5]



Obrázek 1.6: Diabetes:M: Výpočet bolusu



Obrázek 1.7: Diabetes:M: Vyhledávání jídla

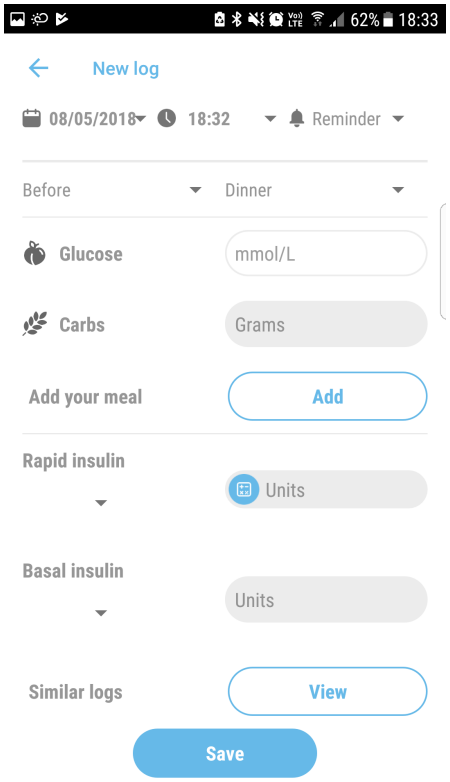
Nevýhody

- Komplikované a místy nepřehledné UI
- Úprava dávky podle fyzické aktivity - nutno zadat o kolik procent se má dávka snížit
- Úprava dávky podle nemoci - nutno zadat o kolik procent se má dávka zvýšit
- Reklamy ve verzi zdarma

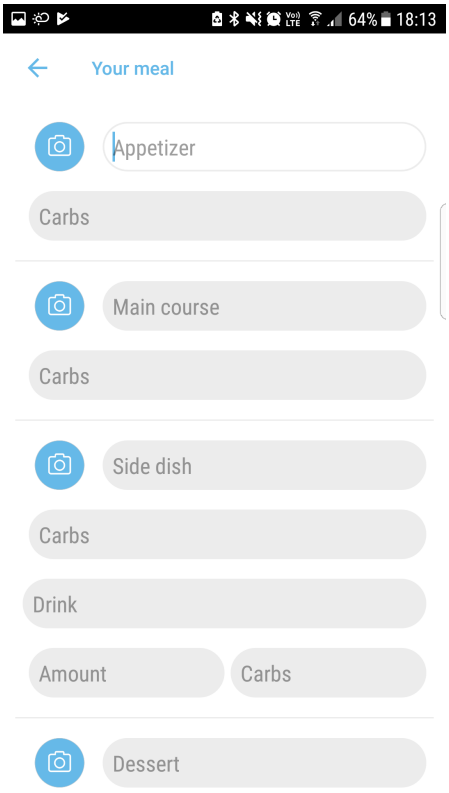
1.2.2.2 Social Diabetes

Social Diabetes[8][9] je další z pokročilých aplikací, která je v základu zdarma, ale pro pokročilé funkce je nutné mít předplatné[6]. Výpočet bolusu a zadávání jídla v mobilní aplikaci zobrazují obrázky 1.8 a 1.9.

[6] cca 410 Kč na rok



Obrázek 1.8: Social Diabetes: Výpočet bolusu



Obrázek 1.9: Social Diabetes: Zadávání jídla

Výhody

- Možnost zadat konkrétní značku pomalého a rychlého inzulinu
- Možnost sdílet data s lékařem - prémiová funkce
- Připojení Bluetooth glukometrů a kontinuálních senzorů

Nevýhody

- Aplikace sice obsahuje napojení na databáze potravin, takže lze vyhledat nutriční hodnoty pro konkrétní potravinu, ale nelze ji dále použít při výpočtu bolusu
- Bílkoviny a tuky nejsou nezahrnuty do výpočtu
- Nelze upravit dávku např. podle nemoci

1.3 Sběr požadavků

Sběr požadavků probíhal během osobních konzultací s prof. MUDr. Kateřinou Štechovou, Ph.D., která se zaměřuje na léčbu diabetu.

1.4 Případy užití

Případy užití popisují množinu akcí, které je možné vykonávat v prostředí systému. Jsou popisovány z pohledu zadavatele pomocí běžného jazyka a za pomoci pojmů z problémové domény. Byly identifikovány následující tři uživatelské role:

- Administrátor
- Lékař
- Pacient

1.4.1 Případy užití společné pro všechny role

Následující případy užití jsou společné pro všechny role. Jedná se především o správu přihlášení a nastavení účtu.

UC1: Přihlášení do aplikace

Popis: Uživatel se může do aplikace přihlásit. V případě privilegovaných uživatelů (Lékař, Administrátor) je přihlášení povinné.

UC2: Odhlášení z aplikace

Popis: Přihlášený uživatel se může, pokud se rozhodne ukončit práci s aplikací, odhlásit kliknutím na příslušné tlačítko.

1.4.2 Administrátor

Role **Administrátor** bude náležet uživatelům určeným ke správě celého systému. Tento uživatel bude především přiřazovat pacienty lékařům a případně řešit vzniklé problémy technického rázu.

UC3: Nastavení rolí uživatelů

Popis: Přidá popř. odebere roli uživateli. Tj. může např. uživateli přidat roli Lékař

UC4: Přiřazení pacienta lékaři

Popis: Administrátor přiřadí pacienta konkrétnímu lékaři, který tím získá přístup k datům a nastavení pacienta.

1.4.3 Lékař

Uživatel s rolí **Lékař** může mimo jiné analyzovat data pacientů, nahrávat údaje např. z kontinuálních senzorů a upravovat nastavení fyziologických parametrů pacientů.

UC5: Upload dat z kontinuálního senzoru

Popis: Lékař nahraje data vyexportované z kontinuálního senzoru. Data jsou nahrána ke konkrétnímu uživateli.

UC6: Procházení dat o glykemii

Popis: Lékař si může zobrazit graf glykemie konkrétního pacienta pro vybraný den.

UC7: Procházení dat o bolusech

Popis: Lékař si může zobrazit graf bolusů konkrétního pacienta pro vybraný den.

UC8: Procházení dat o konzumovaném jídle

Popis: Lékař si může zobrazit tabulku obsahující informace o zkonzumovaném jídle zadaném pacientem pro vybraný den. Hodnoty přijatých sacharidů jsou zobrazeny v grafu.

UC9: Nastavení parametrů pacienta

Popis: Lékař může pro konkrétního pacienta změnit hodnoty jednotlivých parametrů pro výpočet bolusu.

1.4.4 Pacient

Uživatel s rolí **Pacient** může zadávat konzumované potraviny a získávat doporučení velikosti dávky inzulinu.

UC10: Zadání bolusu

Popis: Pacient může zadat informace o aplikovaném bolusu.

UC11: Zadání konzumovaných potravin

Popis: Pacient může zadat informace o druhu, složení a množství konzumovaných potravin. Pacient může připojit i fotografii.

UC12: Vytvoření a úprava potraviny

Popis: Pacient může, pro potřeby budoucího zadávání, vytvořit novou vlastní potravinu a zadat její nutriční hodnoty.

UC13: Výpočet doporučené dávky inzulinu

Popis: Pacientovi bude po zadání konzumovaného jídla, fyzické aktivity, stresu a aktuální glykemie doporučena dávka inzulinu. Fyzické aktivita bude zjištěna na základě měření specializovaného HW.

1.5 Specifikace požadavků

1.5.1 Obecný přehled

1.5.1.1 Přehled hlavních funkcí systému

Systém je rozdělen na dvě hlavní komponenty: *Mobilní aplikace* určená pro pacienty a *Webová aplikace* určená pro lékaře. Tyto dvě součásti jsou propojeny pomocí API, skrze které si vyměňují data. Hlavní funkce mobilní aplikace:

- Doporučování velikosti dávky inzulinu
- Zadávání jídelníčku
- Sledování fyzické aktivity

Hlavní funkce webové aplikace:

- Upravování parametrů použitých pro výpočet dávky inzulinu
- Procházení hodnot glykémie pacienta
- Zobrazení jídelníčku pacienta
- Zobrazení dávek inzulinu

1.5.1.2 Uživatelé systému

Pacient

Největší část uživatelů systému budou tvořit pacienti, kteří ji budou používat pro doporučení dávek inzulinu a zaznamenávání konzumovaných potravin.

Lékař

O pacienty se budou starat lékaři. Ti budou mít na starosti správné nastavení aplikace tak, aby vyhovovala konkrétnímu pacientovi. Lékaři budou moci v systému procházet data pacientů o aplikovaných dávkách inzulinu, konzumovaných potravinách a také fyzické aktivitě.

Administrátor

Administrátoři budou mít za úkol hlavně přiřazování pacientů k lékařům a případně také přidávání nových lékařů do systému.

1.5.1.3 Operační prostředí

Mobilní aplikace

Mobilní aplikace bude fungovat na mobilních telefonech a tabletech s operačním systémem Android s následujícími parametry:

- verze Android OS: Lollipop 5.0 (API level 21) a vyšší
- Fotoaparát
- Bluetooth Low Energy
- Akcelerometr
- Konektivita k internetu - vyžadováno pouze pro synchronizaci dat a aktualizaci aplikace

Aplikace bude distribuována a aktualizována pomocí služby Google Play.

Webová aplikace

Podporované webové prohlížeče:

- Chrome ve verzi 50 a vyšší
- Firefox ve verzi 35 a vyšší

Backend server

Operační systém: Ubuntu Linux ve verzi 16.04.4 LTS 64-bit.

Databáze: PostgreSQL 10.3.

HW: 4GB RAM, HDD 1GB (pouze pro aplikaci), CPU Intel Core i7-4790K, Konektivita 300 Mbps

1.5.1.4 Dokumentace

Součástí vydání bude následující dokumentace:

- Instalační manuál Backend serveru, popisující jak provést nasazení webové aplikace
- Stručný manuál pro uživatele mobilní aplikace

1.5.2 Funkce systému

Následující sekce obsahuje kompletní specifikaci funkcí systému. Je rozdělena na dvě hlavní části: Mobilní aplikace a Backend server.

1.5.2.1 Mobilní aplikace

REQ1: Výpočet bolusu

Priorita: Vysoká

Popis: Uživateli je po zadání jídla, které bude konzumovat, aktuální glykemie a dalších parametrů (viz dále) doporučena velikost dávky inzulinu.

Popis interakce: Interakce začíná v okamžiku, kdy uživatel v hlavním menu vybere položku **Nový bolus**.

Na první obrazovce zadá uživatel potraviny, které bude konzumovat. Může tak učinit dvěma způsoby:

1. Pomocí vyhledávacího pole, které mu na základě názvu (nebo části) nabídne známé potraviny. Po vybrání z nabídky, je potravina přidána do seznamu.
2. Po kliknutí na ikonu dojde k aktivaci fotoaparátu a uživatel naskenuje čárový kód. Pokud je potravina s tímto čárovým kódem známá, je přidána do seznamu.

Pokud potravina, kterou chce uživatel vyhledat (ať už podle názvu nebo podle čárového kódu) neexistuje, je dotázán zda ji chce přidat ručně. Jestliže ano, je přesměrován na obrazovku přidání nové potraviny, specifikované v REQ5. Po úspěšném přidání je přesměrován zpět a nově přidaná potravina se objeví v seznamu potravin ke konzumaci.

U jídel přidaných do seznamu může uživatel zvolit množství. Zadá buď kolik gramů bude konzumovat nebo vybere ze seznamu nabízených velikostí (definovaných pro každou potravinu zvlášť).

Uživatel si dále může konzumované jídlo vyfotit. Pořízená fotografie je zobrazena a lze ji odstranit kliknutí na odpovídající ikonu.

Po přidání všech potravin uživatel přejde na další obrazovku kliknutím na ikonu šipky. Na této obrazovce zadá aktuální glykemii, plánovanou fyzickou aktivitu v dalších třech hodinách a další faktory (viz dále), které mohou mít na výpočet bolusu vliv.

Na další obrazovce je zobrazena vypočtená dávka. Samotný výpočet dávky je předmětem REQ2. Uživatel zároveň může zadat jinou velikost dávky (viz dále), pro kterou se rozhodl.

V případě, že je doporučená hodnota menší než nula (tj. není potřeba další dávka, ale naopak inzulinu je moc), zobrazí aplikace doporučenou dávku 0 a upozornění na hrozící hypoglykemii spolu s upozorněním, aby si pacient snížil bazální inzulin.

Interakce končí v okamžiku, kdy uživatel bolus uloží.

Funkční požadavky:

- Glykemie – číslo z intervalu <0,40> s přesností na jedno desetinné místo.
- Fyzická aktivita – Výběr z hodnot *Žádná, Nízká, Střední a Vysoká*. V případě vybrání jiné hodnoty než *Žádná*, je možné zvolit, že se jedná o vytrvalostní aktivitu.
- Další faktory, které lze vybrat:
  - Nemoc
  - Menstruace
  - Stres
  - Nefunkční pomůcka
  - Extrémní teplota
  - Léky
  - Jiné (možno specifikovat jaké)
- Jiná velikost dávky – celé číslo z intervalu <0, 100>

REQ2: Způsob výpočtu bolusu

Priorita: Vysoká

Popis: Tento požadavek popisuje přesný způsob výpočtu bolusu.

Popis interakce:

Funkční požadavky: Výpočet bolusu bude probíhat z dat zadaných v REQ1, parametrů z REQ3 a REQ4. A to podle následujícího vzorce:

$IU = \frac{S}{10} * CIR + \frac{BG_c - BG_t - PA}{ISF} - IOB$

kde:

- IU je výsledný doporučený počet jednotek inzulinu
- S je množství konzumovaných sacharidů v gramech
- CIR
- BG_c je aktuální glykemie

- BG_t je cílová glykémie
- PA je vliv fyzické aktivity [7]
- ISF
- IOB je Insulin On Board[8]

REQ3: Nastavitelné parametry pacienta

Priorita: Vysoká

Popis: Tento požadavek popisuje jaké parametry může lékař nastavit pacientovi.

Popis interakce:

Funkční požadavky:

- CIR – Carbohydrate To Insulin Ratio – kolik inzulinu je potřeba na metabolizování 10g sacharidů
  - Číslo s přesností na jedno desetinné místo
  - Možno specifikovat jinou hodnotu pro každou denní i noční hodinu
- ISF – Insulin Sensitivity Factor – kolik inzulinu je potřeba na snížení glykémie o 1 mmol/l
  - Číslo s přesností na jedno desetinné místo
  - Možno specifikovat jinou hodnotu pro každou denní i noční hodinu
- Pohlaví – muž nebo žena
- Výška (cm)
- Váha (kg)
- Obvod pasu (cm)
- Obvod boků (cm)
- Množství tělesného tuku (%)
- Množství svaloviny (%)
- BMR – Basal Metabollic Rate – množství inzulinu na pokrytí bazální potřeby organismu
- Doba aktivního inzulinu (h) – za jak dlouho dojde k metabolizaci inzulinu po aplikaci
- Cílová hodnota glykémie (mmol/l)
  - Hodnota pro den a noc

[7] o kolik mmol/l klesne glykemie při dané míře fyzické aktivity
[8] množství inzulinu v krvi např. z předchozího bolusu

- Vliv fyzické aktivity (mmol/l/h) – o kolik mmol/l klesne glykemie při dané míře fyzické aktivity za jednu hodinu
  - Hodnota pro nízkou, střední a vysokou zátěž

REQ4: Výpočet aktivního inzulinu

Priorita: Vysoká

Popis: Tento požadavek popisuje jak se počítá IOB[9] – Aktivní inzulin

Popis interakce:

Funkční požadavky: Pro výpočet se použije parametr Čas aktivního inzulinu (REQ3). Ten říká, jak dlouho po aplikaci je inzulin přítomen v krvi.

V okamžiku aplikace je aktivní inzulin roven velikost dávky. V čase se tato hodnota lineárně snižuje a dosáhne nuly[10] v čase aktivního inzulinu.

$IOB(t) = (1 - \frac{t-t_b}{t_{active}}) * IU$

kde:

- t je čas
- t_b je čas aplikace bolusu
- t_{active} je čas aktivního inzulinu
- IU jsou aplikovaná jednotky

REQ5: Přidání a editace jídla

Priorita: Vysoká

Popis: Pacient si může vytvořit novou potravinu v databázi a použít ji pro budoucí zadávání jídla. Existující potravinu je možné upravit.

Popis interakce: Novou potravinu může pacient přidat na obrazovce *Potraviny*, když klikne na tlačítko „+“. Potravinu lze také přidat při výpočtu bolusu, pokud není nenalezena podle jména nebo čárového kódu.

Kliknutím na konkrétní potravinu na obrazovce *Potraviny* ji lze editovat.

Funkční požadavky:
- Zadávané hodnoty

[9] Insulin on Board
[10] hodnota již dále neklesá

  - Název
  - Fotografie (nepovinná)
  - Popis (nepovinný)
  - Čárový kód – možno naskenovat fotoaparátem (nepovinný)
  - Glykemický index (pomalý, střední, rychlý)
  - Obsah sacharidů ve 100g
  - Původ jídla (domácí, koupené, z restaurace)

REQ6: Historie bolusů

Priorita: Vysoká

Popis: Uživatel si může zobrazit historie zaznamenaných bolusů

Popis interakce: Interakce začíná v okamžiku, kdy uživatel v hlavním menu vybere položku **Historie bolusů**.

Na obrazovce se uživateli zobrazí seznam všech zaznamenaných bolusů.

Funkční požadavky: Zobrazené položky:

- Název jídla
- Fotografie jídla - pokud byla pořízena
- Obsah sacharidů
- Glykemie
- Aplikovaný inzulin
- Doporučený inzulin - zobrazen pouze pokud se liší od aplikovaného
- Datum a čas

REQ7: Přihlášení

Priorita: Vysoká

Popis: Uživatel se může do aplikace přihlásit pomocí Google účtu.

Popis interakce: Interakce začíná v okamžiku, kdy uživatel v hlavním menu vybere položku **Přihlásit**, která je zobrazena pouze pokud již není přihlášený. Uživateli se zobrazí systémový dialog, ve kterém může vybrat z účtů, do kterých je přihlášen v systému Android nebo zadat jiný účet. Po vybrání účtu je zobrazeno upozornění o úspěšném popř. neúspěšném přihlášení.

Funkční požadavky:

REQ8: Odhlášení

Priorita: Vysoká

Popis: Uživatel se může odhlásit, pokud je přihlášený.

Popis interakce: Interakce začíná v okamžiku, kdy uživatel v hlavním menu vybere položku **Odhlásit**, která je zobrazena pouze pokud je přihlášený. Uživatel je následně odhlášen a je zobrazeno upozornění o odhlášení.

Funkční požadavky:

REQ9: Nahrávání dat o bolusech na Backend server

Priorita: Vysoká

Popis: Data o aplikovaných bolusech se pravidelně nahrávají na Backend server

Popis interakce: Nahrávání probíhá na pozadí, bez uživatelské interakce.

V pravidelných intervalech je spuštěn proces, který nahraje kompletní data o aplikovaných bolusech na Backend server skrze Bolus API specifikované v 2.2.3.

Funkční požadavky:

- Interval nahrávání – každou hodinu
- Podmínky nahrávání
  - Nahrávání povoleno v nastavení
  - Zřízení je připojeno k internetu
  - Uživatel je přihlášen

REQ10: Nahrávání dat ze senzorů na Backend server

Priorita: Vysoká

Popis: Data ze senzorů se pravidelně nahrávají na Backend server

Popis interakce: Nahrávání probíhá na pozadí, bez uživatelské interakce.

V pravidelných intervalech je spuštěn proces, který nahraje kompletní data ze senzorů na Backend server skrze Bolus API specifikované v 2.2.3.

Funkční požadavky:

- Interval nahrávání – každou hodinu
- Podmínky nahrávání
  - Nahrávání povoleno v nastavení
  - Zřízení je připojeno k internetu
  - Uživatel je přihlášen

REQ11: Aktualizace parametrů z Backend serveru

Priorita: Vysoká

Popis: Parametry nastavené lékařem na Backend serveru se promítnou do nastavení mobilní aplikace pacienta.

Popis interakce: Stahování probíhá na pozadí, bez uživatelské interakce.

V pravidelných intervalech je spuštěn proces, který stáhne nastavení parametrů pacienta z Backend serveru.

Funkční požadavky:

- Interval stahování – každou hodinu
- Podmínky stahování
  - Stahování parametrů je povoleno v nastavení
  - Zařízení je připojeno k internetu
  - Uživatel je přihlášen

1.5.2.2 Backend server

REQ12: Přihlášení

Priorita: Vysoká

Popis: Uživatel se musí do aplikace přihlásit.

Popis interakce: Nepřihlášený uživatel je při navštívení všech URL aplikace, přesměrován na přihlašovací stránku. Zde si může zvolit, jakou metodou se chce přihlásit. Po kliknutí na odpovídající tlačítko, je buď rovnou přihlášen, nebo přesměrován na stránku poskytovatele přihlašovací metody, kde může být dotázán, zda např. chce povolit aplikaci přístup k některým údajům.

V případě úspěšného přihlášení, je uživatel přesměrován zpět na stránku, na kterou původně přistoupil[11].

Pokud není přihlášení úspěšné, je zobrazena chybová hláška popisující důvody chyby.

V případě, že se k Backend serveru připojuje mobilní aplikace skrze REST API, musí se nejdříve autentizovat pomocí příslušného API.

Funkční požadavky:

- Podporované metody přihlášení
  - Google účet

[11] Obvykle se bude jednat o hlavní stránku aplikace

REQ13: Odhlášení

Priorita: Vysoká

Popis: Uživatel se může odhlásit, pokud je přihlášený.

Popis interakce: Interakce začíná v okamžiku, kdy uživatel v hlavním menu vybere položku **Odhlásit**. Uživatel je následně odhlášen a je přesměrován na přihlašovací stránku.

Funkční požadavky:

REQ14: Nastavení parametrů pacienta

Priorita: Vysoká

Popis: Lékař nastaví hodnoty jednotlivých parametrů. Ty budou následně použity pro výpočet bolusu.

Popis interakce: Lékař v hlavním menu vybere pacienta a zvolí **Nastavení parametrů**. Následně je mu zobrazen formulář, do kterého zadá požadované hodnoty. Každý parametr má výchozí hodnotu (společnou pro všechny pacienty). Tato hodnota může být zvolena zaškrtnutím **Výchozí hodnota**. Formulář se po kliknutí na **Uložit** uloží. Po uložení jsou hodnoty dostupné pro synchronizaci popsanou v REQ11.

Funkční požadavky:

- Datové typy parametrů
  - String
  - Integer
  - Float

REQ15: Nahrávání dat z kontinuálních senzorů glykemie

Priorita: Vysoká

Popis: Lékař může do aplikace nahrát data vyexportovaná ze senzoru glykémie.

Popis interakce: Lékař v hlavním menu vybere pacienta a zvolí **Upload dat**. Následně je mu zobrazen formulář, do kterého zadá soubor, který chce nahrát. Data se po kliknutí na **Uložit** nahrají na server. Tato data je následně možné procházet, jak je popsáno v REQ16.

V případně úspěšného nahrání je zobrazena hláška oznamující tuto skutečnost.

V případě neúspěchu je zobrazena chybová hláška s příčinou chyby.

Funkční požadavky:
- Podporované formáty:
  - Dexcom
  - Medtronic

REQ16: Procházení dat ze senzorů

Priorita: Vysoká

Popis: Lékař může v aplikaci procházet data zaznamenaná připojenými senzory, popř. ručně nahraná data (viz REQ15).

Popis interakce: Lékař v hlavním menu vybere pacienta a zvolí **Procházení dat**. Následně jsou mu zobrazeny grafy se zaznamenanými údaji za jeden den. Datum, pro které budou zobrazena data, je možné vybrat z kalendáře.

Funkční požadavky:
- Vzhled grafů:
  - Na ose X je čas měření
  - Na ose Y je zobrazena naměřená hodnota
- Zobrazené grafy:
  - Glykémie
  - Počet kroků
  - GSR
  - Srdeční tep
  - MM level

REQ17: Procházení dat o bolusech

Priorita: Vysoká

Popis: Lékař může v aplikaci procházet data o bolusech pacienta.

Popis interakce: Lékař v hlavním menu vybere pacienta a zvolí **Bolusy**. Následně je mu zobrazena tabulka s daty. Data pro zobrazení jsou získávána z mobilní aplikace, jak je popsáno v REQ9.

Funkční požadavky:
- Zobrazené hodnoty:
  - Datum a čas
  - Množství sacharidů
  - Aplikovaný inzulin
  - Doporučený inzulin
  - Fotografie jídla (pokud byla pořízena)

1.5.2.3 Požadavky na zabezpečení

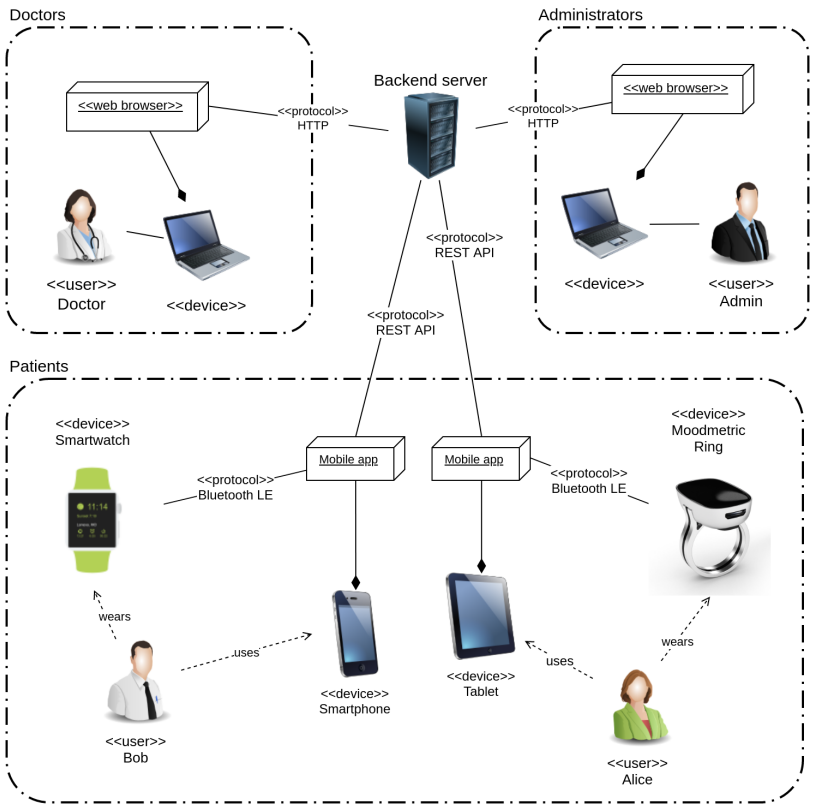
REQ18: Zabezpečená komunikace

Všechna komunikace mezi mobilní aplikací a Backend serverem musí probíhat šifrovaně, stejně jako komunikace mezi webovou aplikací a prohlížečem uživatele.

Kapitola 2

Návrh

Na základě specifikace požadavků z minulé kapitoly byli navrženy jednotlivé součásti výsledného řešení. Jako první byl vytvořen návrh celkové struktury systému. Jak ukazuje obr. 2.1 systém se bude skládat z několika hlavních součástí: Backend server, webová aplikace, mobilní aplikace a hardwarové periferie na měření fyzické aktivity a stresu.



Obrázek 2.1: Architektura systému

2.1 HW

Z případu užití UC13 vyplývá, že pro měření fyzické aktivity a ideálně i úrovně stresu budeme potřebovat specializovaný hardware, který budou mít pacienti stále na sobě a který bude mobilní aplikaci poskytovat real-time popř. na vyžádání data.

Pro stanovování fyzické aktivity jsou vhodné senzory srdečního tepu. Těch existují dva základní druhy: hrudní pásy (obr. 2.2) a optické senzory (obr. 2.3). Hrudní pásy vynikají přesností měření (plynoucí z jejich umístění na hrudníku a těsného kontaktu elektrod s pokožkou), ale jejich velkou nevýhodou je (ne)pohodlnost nošení. Elastický pás, který drží senzor na místě, stále mírně svírá hrudník, což může být nepříjemné a navíc občas dochází ke sklouzávání pásu dolů z ideální polohy pod prsními svaly.

Druhým typem jsou optické senzory tepu, nejčastěji umístěné na spodních stranách zařízení podobných hodinkám (sport-testery apod.). Pro svou správnou funkci potřebují těsně přiléhat na zápěstí, takže pásek musí být dostatečně utažený. Z toho plyne také určité nepohodlí. Ale ani při ideálním umístění nemají optické senzory vysokou přesnost[10].

Kvůli výše uvedeným důvodům použijeme senzor srdečního tepu pouze pro referenční měření.

Pro měření fyzické aktivity (zejména chůze a běhu) se také často používají krokoměry. Ty vynikají energetickou nenáročností, nízkou váhou a malými rozměry. Bohužel z principu jejich funkce – měření pochybu těla – nedokáží zaznamenat statické fyzické aktivity jako je např. jóga. I přes to ale mohou dobře posloužit pro měření každodenních aktivit jako jsou nakupování, uklízení nebo chůze po městě. Navíc, ač si to často neuvědomujeme, obvykle máme

Obrázek 2.2: Geonaute HR Belt[11]

Obrázek 2.3: Fitbit Surge[12]

krokoměr stále při sobě, protože je obsažen ve většině moderních mobilních telefonů. Integrovaný krokoměr použijeme pro stanovení každodenní fyzické aktivity.

Další možností jak stanovit fyzickou aktivitu a stres je GSR – Galvanic skin response (někdy také označované jako EDA – Electrodermal activity). Metoda, která stanovuje odpor pokožky při průchodu slabého elektrického proudu. Tato veličina se mění v závislosti na aktivitě potních žláz, které řídí sympatický nervový systém. Vyšší aktivita potních žláz indikuje vyšší míru vybuzení a tedy např. stresovou situaci.

Pro měření GSR existuje na trhu několik zařízení:

MAXREFDES73#

Referenční design MAXREFDES73# [13] společnosti Maxim Integrated , ve formě zařízení nositelného na zápěstí. Nutno podotknout, že není určeno pro produkční nasazení – nemá např. ani horní kryt (viz obr. 2.4) Hlavní parametry:

- GSR senzor
- Teploměr - měří teplotu kůže
- Konektivita Bluetooth Low Energy
- Výdrž na baterii – 15 h
- Cena: cca 200 USD

Obrázek 2.4: MAXREFDES73#

Moodmetric ring

Velice designově povedené zařízení Moodmetric ring [14] od společnosti Moodmetric má formu prstenu (obr. 2.5). Stejně jako předchozí zařízení poskytuje měření GSR. Hlavní parametry:

- GSR senzor
- MM level – Hodnota z intervalu 0 – 100 ukazující úroveň stresu jaké je aktuálně člověk vystaven[15]. Hodnota je srovnatelná napříč uživateli.
- Akcelerometr
- Konektivita – Bluetooth Low Energy
- Výdrž na baterii – 1 týden
- Cena: cca 189 EUR

Obrázek 2.5: The Moodmetric ring

Empatica Embrace

Empatica Embrace[16](obr. 2.6) je zařízení podobné MAXREFDES73#. Jeho hlavním problémem je, že není možné jej používat jinak než s oficiální aplikací výrobce a není tedy ani možné získat přistup k datům. Toto omezení nám bohužel brání, jinak slibné zařízení, použít.

- GSR senzor
- Teploměr - měří teplotu kůže
- Akcelerometr
- Gyroskop
- Konektivita Bluetooth Low Energy
- Výdrž na baterii – 30 h
- Cena: cca 250 USD

Obrázek 2.6: Empatica Embrace

Empatica E4

Empatica E4[17](obr. 2.7) je pokročilejší verzí Embrace cílenou primárně na výzkumníky. E4 poskytuje přístup k datům skrze vlastní API pro registrované vývojáře. Zařízení vypadá opravdu nadějně, ovšem faktorem, který pro nás limituje reálné nasazení je jeho cena blížící se 1 700 USD.

- GSR senzor
- Teploměr - měří teplotu kůže
- Akcelerometr

- Optický senzor tepové frekvence
- Tlačítko umožnující poznamenat čas nějaké události
- Konektivita Bluetooth Low Energy
- Výdrž na baterii – 24 h
- Cena: cca 1 690 USD

Obrázek 2.7: Empatica E4

2.1.1 Vybraný hardware

Všechna výše uvedená zařízení splňují naše požadavky na senzorovou výbavu, ale rozhodující faktorem hovořícím v neprospěch výrobků společnosti Empatica je v případě E4 vysoká cena a v případě Embrace nepřístupnost dat.

Pro další testování byly zakoupeny zařízení MAXREFDES73# a Moodmetric ring. Pro referenční měření byl vybrán hrudní senzor tepu Geonaute BluetoothSmart HRM Belt[11].

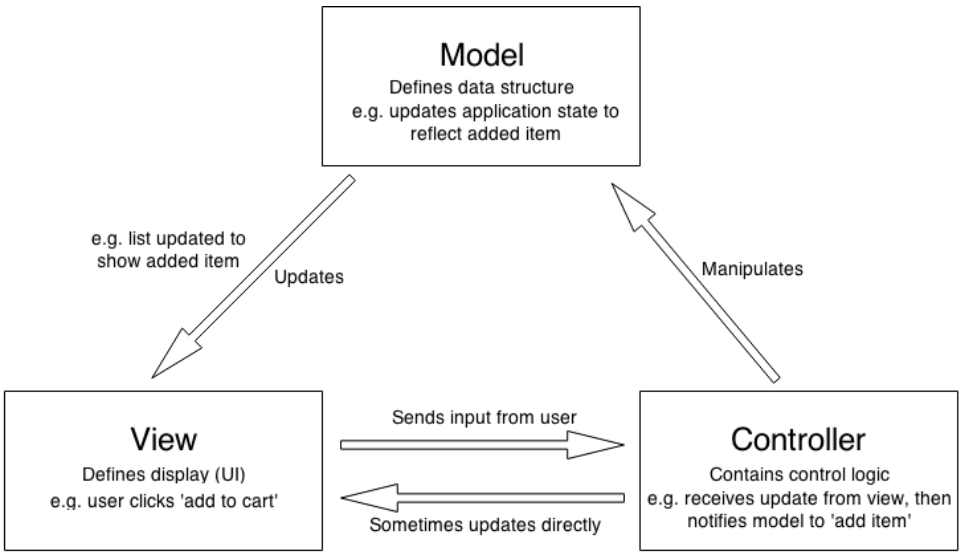
2.2 Backend server

2.2.1 Architektura

Pro aplikační backend byla zvolena klasická MVC[12] architektura[18]. Její fungování ukazuje obrázek 2.8. Uživatel vždy pracuje pouze s *View*, které obstarává veškeré zobrazování dat a také vstup od uživatele. V případě naší aplikace je *View* reprezentováno HTML stránkou s JavaScriptem.

Od *View* dostává data *Controller*, který je převede do formy, které rozumí *Model*, kterému data následně předá. Jako *Controller* v naší aplikaci slouží třídy se stejnojmenným sufixem. Definují na jakých URL je dostupné jaké *View*, jaká data se do něj posílají a naopak jaká data mohou být obdržena.

Model definuje jaká data mohou být v aplikaci obsažena a jak se s nimi pracuje. Určuje také stav aplikace.



Obrázek 2.8: MVC Architecture

2.2.2 Datový model

Datový model určuje s jakými daty a v jaké formě bude aplikace pracovat. Navržené entity, atributy a vztahy mezi entitami ukazuje diagram 2.9.

Středem datového modelu je entita *Pacient*, která má vazby na následující entity:

[12] Model-View-Controller

- *Bolus* – obsahuje všechna data vztahující se k jedné aplikaci inzulinu a s tím spojenou konzumaci jídla
- *Doctor* – určuje, jestli má lékař přístup k datům konkrétního pacienta
- *Measurement* – hodnota nějakého ukazatele v čase, např. srdeční tep, galvanický odpor, počet kroků, glykémie atd.
- *User* – vazba určující např. přihlašovací údaje do systému
- *Settings Value* – hodnota některého z parametrů pro výpočet bolusu

Dále je zajímavá entita *User* a s ní asociované entity *Role* a *Privilege*, které slouží k řízení práv uživatele.

2.2.3 REST APIs

Na základně požadavků byla navržena následující API, která budou sloužit pro komunikaci s mobilní aplikací případně s jinými klienty, kdyby to bylo v budoucnu potřeba.

Login API

Požadavky: REQ12

Endpoint: /login/googleToken/

HTTP Metoda: GET

Request: OAuth 2.0 access token v hlavičce X-ID-TOKEN

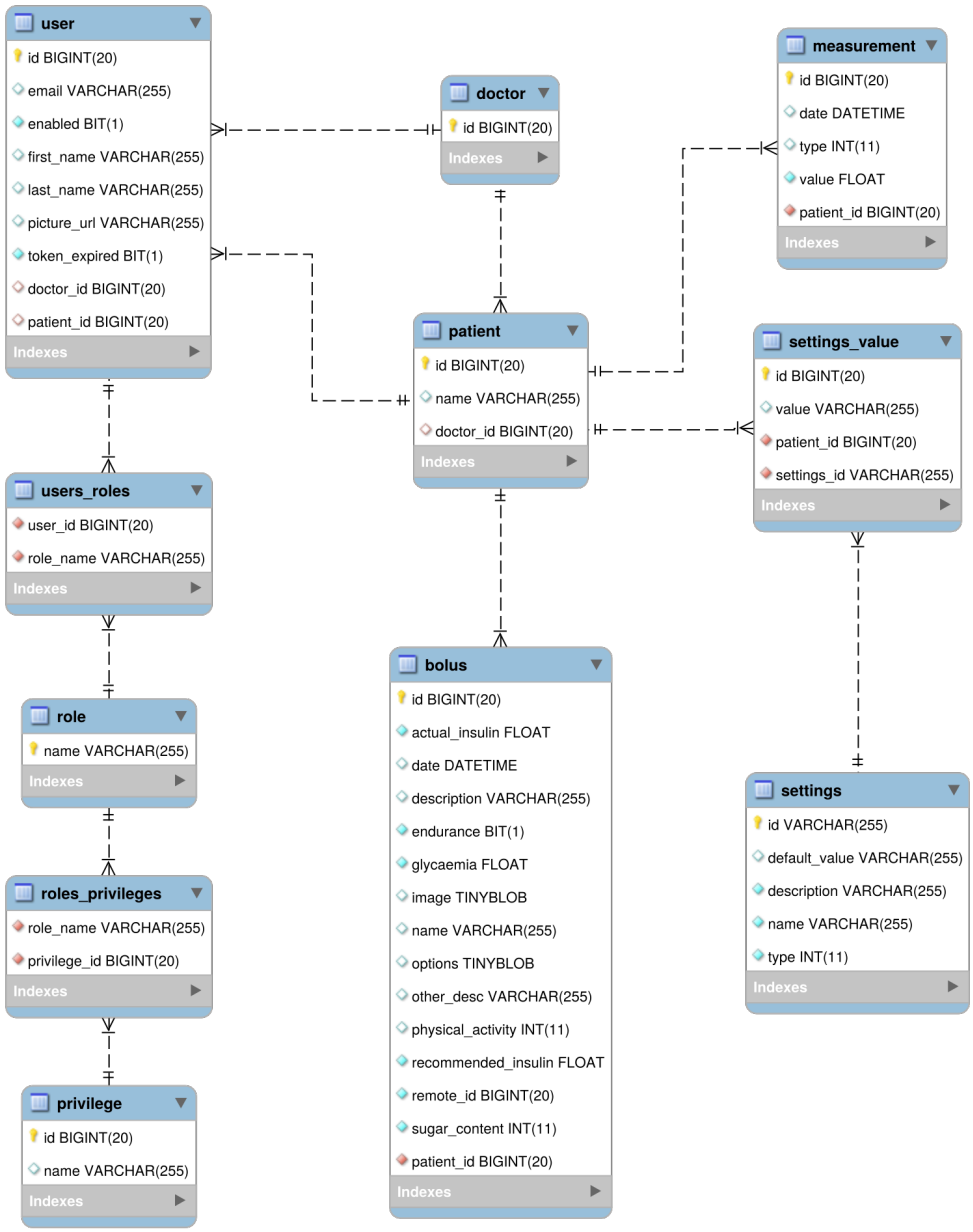
Ukázka requestu:
```
X-ID-TOKEN: eyJhbGciOiJSUzI1NiIsImtpZCI6IjNmM2VmOWM3ODAzY
```

Response: **204 No Content**

Cookie JSESSIONID, podle které bude ověřována identita při dalších requestech

Ukázka response:
```
Set-Cookie: JSESSIONID=E78C5C2F6EA8CDFBCB2587FBEAA02B3E;
Path=/; Secure; HttpOnly
```

Chybové stavy:
- 302 Found - Chybný token, přesměrování na přihlašovací stránku



Obrázek 2.9: Datový model - Backend

Bolus API

Požadavky: REQ9

Endpoint: /boluses/<email>/

HTTP Metoda: POST

Request: Pole objektů popisujících bolus

Ukázka requestu:


```
[
   {
      "actualInsulin":26.0,
      "date":"2018-05-02T17:59:24.066+0200",
      "endurance":false,
      "glycaemia":4.0,
      "id":2,
      "name":"Těstoviny",
      "options":[],
      "physicalActivity":"LOW",
      "recommendedInsulin":32.0,
      "remoteId":1,
      "sugarContent":20,
      "synced":"NOT_SYNCED"
   }
]
```


Response: **200 OK**

Pole objektů mapujících klientská ID na serverová ID (kvůli opakované synchronizaci)

Ukázka response:


```
[
   {
      "serverId":12,
      "clientId":"2"
   }
]
```


Chybové stavy:

- 400 Bad Request - Špatný formát requestu
- 401 Unauthorized - Neautorizovaný uživatel

Measurement API

Požadavky: REQ10

Endpoint: /measurements/<email>

HTTP Metoda: POST

Request: Pole objektů popisujících jednotlivá měření

Ukázka requestu:


```
[
   {
      "value":23432,
      "date":"2018-05-10T14:06:53.160+0000",
```

```
        "type":"GSR"
    },
    {
        "value":15,
        "date":"2018-05-10T14:06:53.842+0000",
        "type":"STEPS"
    },
    {
        "value":72,
        "date":"2018-05-10T14:06:54.185+0000",
        "type":"HR"
    }
]
```

Response: **200 OK**

Pole objektů tak jak byla data uložena

Ukázka response:

```
[
    {  "id":3454,
        "value":23432,
        "date":"2018-05-10T14:06:53.160+0000",
        "type":"GSR"
    },
    {
        "id":3455,
        "value":15,
        "date":"2018-05-10T14:06:53.842+0000",
        "type":"STEPS"
    },
    {
        "id":3456,
        "value":72,
        "date":"2018-05-10T14:06:54.185+0000",
        "type":"HR"
    }
]
```

Chybové stavy:

- 400 Bad Request - Špatný formát requestu
- 401 Unauthorized - Neautorizovaný uživatel

Settings API

Požadavky: REQ11

Endpoint: /settings/<email>/

HTTP Metoda: GET

Response: **204 No Content**

Cookie JSESSIONID, podle které bude ověřována identita při dalších requestech

Ukázka response:

```
[
   {
      "id":"CIR",
      "value":1.5
   },
   {
      "id":"ISF",
      "value":1
   },
   {
      "id":"targetGlycaemia",
      "value":5.6
   }
]
```

Chybové stavy:

- 401 Unauthorized - Neautorizovaný uživatel

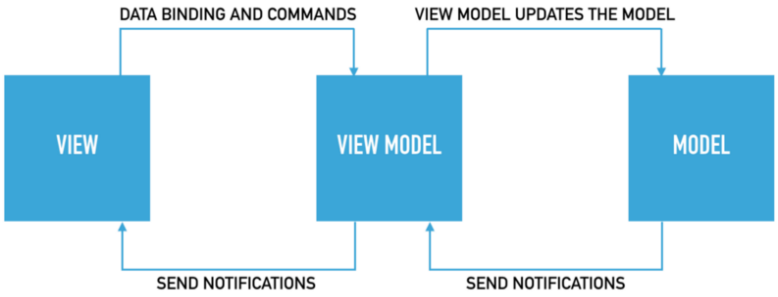
2.3 Mobilní aplikace

2.3.1 Architektura

Mobilní aplikace bude založena na architektuře MVVM[13], která se skládá (jak už napovídá název) ze tří hlavních komponent:

- **View** – informuje *ViewModel* o akcích uživatele
- **ViewModel** – poskytuje *View* data obvykle ve formě Observable objektů a aktualizuje *Model* na základě akcí uživatele
- **Model** – abstrakce zdroje dat

[13] Model-View-ViewModel

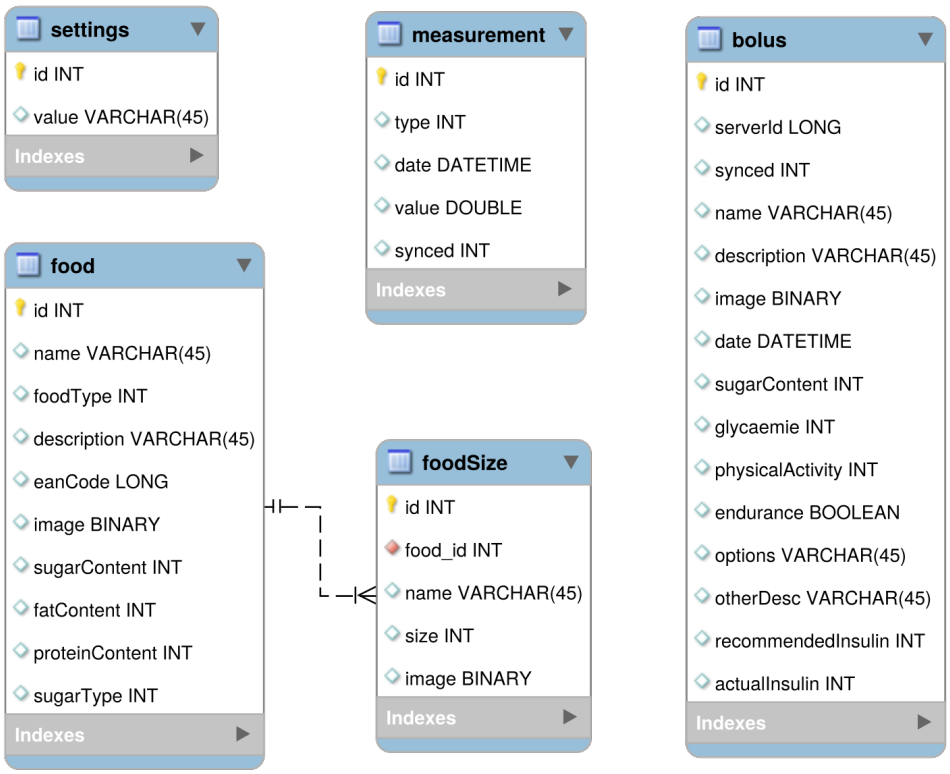


Obrázek 2.10: Architektura MVVM Zdroj:[19]

2.3.2 Datový model

Datový model mobilní aplikace je o poznání jednodušší než model backendu, a to především proto, že uchovává data pouze jednoho uživatele. Aplikace také nemusí řešit oprávnění uživatelů k jednotlivým akcím. Celý datový model ukazuje obr. 2.11.

Z modelu je asi nejzajímavější entita *Bolus*, která stejně jako na backendu, uchovává data o jedné aplikaci inzulinu. Má vazbu M:N s entitou *Food*, která říká, jaké jídlo uživatel konzumoval. Protože entitu *Food* může uživatel editovat, je množství konzumovaných sacharidů zkopírováno do *Bolusu* při vytvoření. *FoodSize* obsahuje nabízené velikosti porce pro každé jídlo a jejich hmotnost (např. krajíc chleba nebo střední brambora). *Settings* ukládá hodnoty parametrů synchronizovaných z backendu. A konečně *Measurement* stejně jako na backendu obsahuje naměřené hodnoty.



Obrázek 2.11: Datový model - Mobilní aplikace

Kapitola 3

Implementace

3.1 Mobilní aplikace

3.1.1 Výběr technologií

3.1.1.1 Programovací jazyk

Jelikož z požadavků vyplývá, že mobilní aplikace musí fungovat na operačním systému Android, připadají v úvahu následující programovací jazyky[14]:

Java

Java byla první oficiální jazyk, ve kterém bylo možné vyvíjet pro Android. Jedná se o nejpoužívanější programovací jazyk vůbec [23] a tomu odpovídá množství dostupných technologií, frameworků a materiálů. Nebudeme opakovat mnohokrát řečené a známé, takže pouze krátce: Java je objektově orientovaný, staticky typovaný, multiplatformní, do bytekódu kompilovaný jazyk. [24]

Kotlin

Kotlin je moderní, oficiálně podporovaný jazyk pro Android, vyvinutý společností JetBrains (který stojí také např. za oblíbeným IDE Intellij IDEA). Je stejně jako Java kompilovaný do bytekódu, který je následně interpretován Java Virtual Machine (nebo Android Runtime v případě Androidu). Mezi jeho hlavní přednosti patří čitelnost, stručnost[15], funkcionální přístup (jako alternativa k OOP) a úplná interoperabilita s Javou, takže je možné z Java kódu volat funkce napsané v Kotlinu a obráceně. Jelikož je Kotlin kompilovaný

[14] Existuje několik alternativních SDK jako např. Corona[20], PhoneGap[21] nebo Xamarin[22], ty ovšem nejsou oficiálně podporované a jejich použití může přinést nepředvídatelné problémy.

[15] Uvádí se, že kód v Kotlinu má o 40% méně řádek, než identický kód v Javě[25]

do Java bytekódu plynou z toho i určitá omezení. Výsledný program je vždy pouze podmnožinou toho co umožňuje JVM. [26]

C/C++

Jazyk C/C++ je možné použít v rámci Android NDK pro vývoj nativních součástí aplikace. Je to vhodné například pro urychlení náročných výpočtů, dosažení nízké doby odezvy popř. to umožňuje použít znovu již hotové komponenty implementované právě v C/C++[27]. My budeme všechny náročné výpočty provádět na backend serveru a nikoli na zařízení, který bychom tak mohli nadměrně vytěžovat a tím výrazně snížit např. životnost baterie. Android NDK proto nepoužijeme.

3.1.1.2 Databáze a databázová vrstva

Jelikož naše mobilní aplikace bude muset ukládat data lokálně, budeme potřebovat nějakou databázi. Jedinou nativně podporovanou databázi na platformě Android je SQLite.

Abychom nemuseli všechny činnosti spojené s prací s DB provádět „ručně“ (nebo např. pomocí SQLiteOpenHelper[28]), bude se nám hodit kvalitní DB vrstva, která nám může ušetřit hodně práce (a tím i odstranit potencionální zdroj chyb). Od DB vrstvy očekáváme hlavně tyto věci:

- Převod objektů na databázové entity a zpět – tj. objektově-relační mapování
- Přehledné psaní SQL dotazů
- Automatická aplikace migrací

Těmto požadavkům výborně vyhovuje knihova **Room Persistence Library**[29] vyvíjená přímo společností Google.

3.1.2 Uživatelské rozhraní

Platforma Android nám poskytuje několik základních stavebních bloků pro tvorbu uživatelského rozhraní:

- Activity[30] - Aplikace se může skládat z jedné i více Activit. Je to třída popisující jednu až několik obrazovek. Všechny viditelné součásti aplikace musí patřit do nějaké Aktivity[16].
- Fragment[31] - Menší komponenta, existující vždy v rámci nějaké Aktivity. Obvykle se jedná o několik logicky souvisejících prvků uživatelského rozhraní např. formulář nebo dialog.

[16] S výjimkou Widgetů

- View[32] - Prvek uživatelského rozhraní jako např. textové pole nebo tlačítko.

Naše aplikace se bude skládat z následujících aktivit, které budou používat fragmenty reprezentující jednotlivé obrazovky:

MainActivity

Hlavní aktivita, která slouží zároveň i jako vstupní bod do aplikace tj. zobrazí se uživateli po spuštění aplikace. Tato aktivita obsahuje logiku obsluhující přihlašování a odhlašování uživatelů. Důležitou součástí této aktivity je *NavigationDrawer* – hlavní menu (viz obr. 3.2), které vyjíždí z levé strany obrazovky. Obsahuje několik fragmentů (viz dále), mezi kterými může uživatel přecházet pomocí hlavního menu.

- DashboardFragment – Domácí obrazovka ukazující nejrůznější údaje (např. velikost poslední dávky) a grafy (predikce aktivního inzulinu v čase, fyzická aktivita) – obr. 3.1
- EntryListFragment – Seznam bolusů aplikovaných v minulosti – obr. 3.3
- FoodListFragment – Seznam vytvořených jídel – obr. 3.5
- WizardFragment – Průvodce výpočtem nového bolusu obsahující další tři fragmenty
  - FoodFragment – Konzumované jídlo – obr. 3.6
  - StateFragment – Údaje o glykémii a jiných okolnostech – obr. 3.7
  - RecommendationFragment – doporučená dávka – obr. 3.8
- SettingsFragment – Nastavení aplikace – obr. 3.9

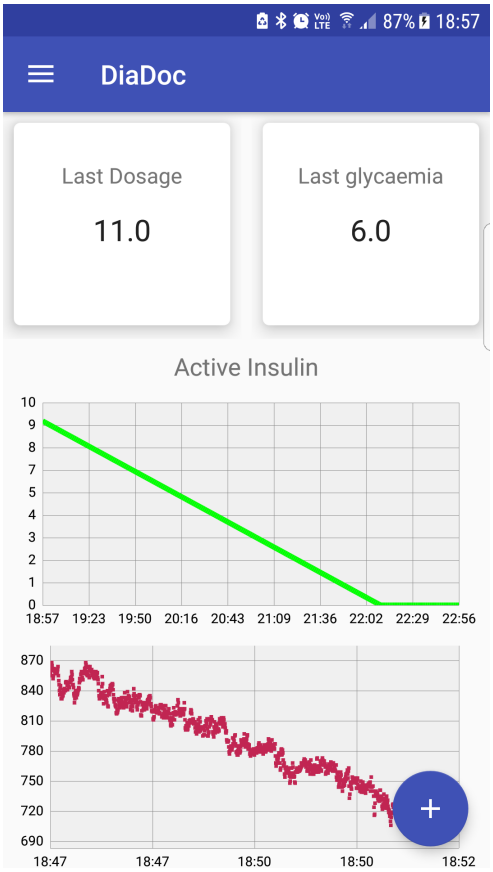
FoodActivity

Aktivita sloužící pro vytvoření a úpravu jídla (viz obr. 3.4). Uživatel zde může jídlo vyfotit, zadat jeho nutriční hodnoty, EAN kód atd.

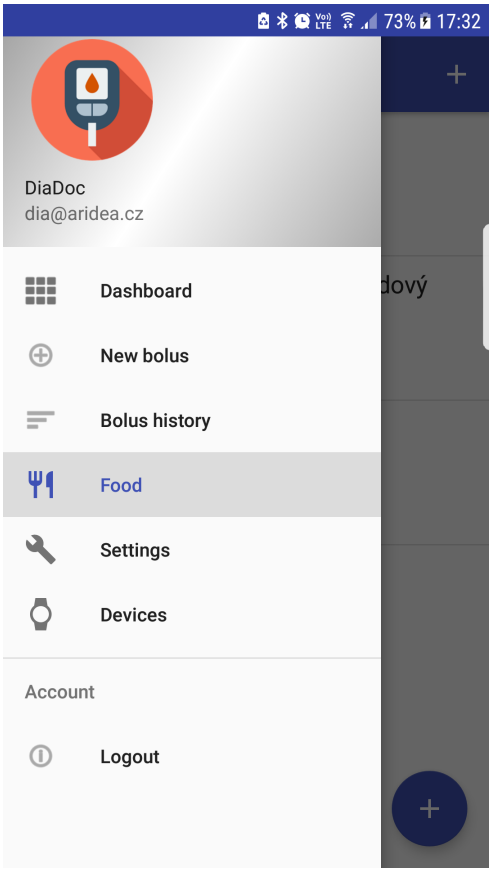
DeviceActivity

Tato aktivita se stará o správu zařízení používaných pro zaznamenávání fyzické aktivity a stresu. Obsahuje tři fragmenty:

- ScanFragment – Hledání a přidávání nových zařízení – obr. 3.10
- DeviceListFragment – Seznam známých zařízení – obr. 3.11
- DeviceDetailFragment – Detail připojeného zařízení – obr. 3.12



Obrázek 3.1: UI - Domácí obrazovka



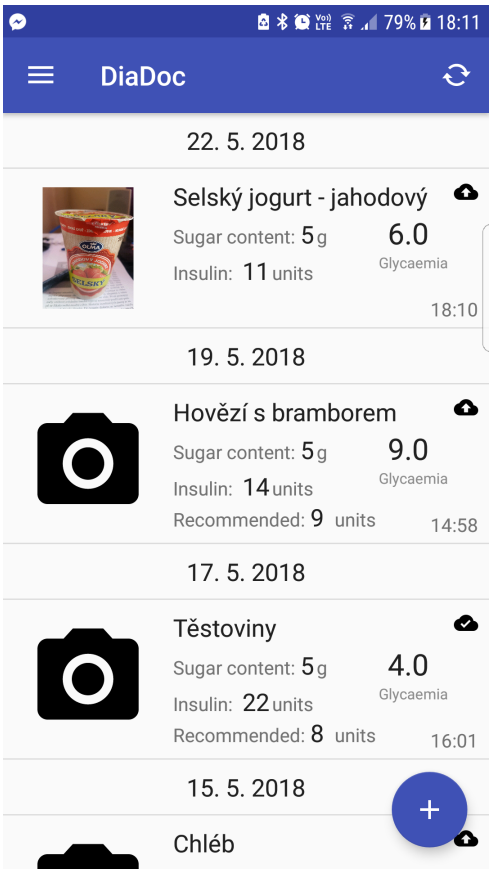
Obrázek 3.2: UI - Hlavní menu

3.1.3 Services

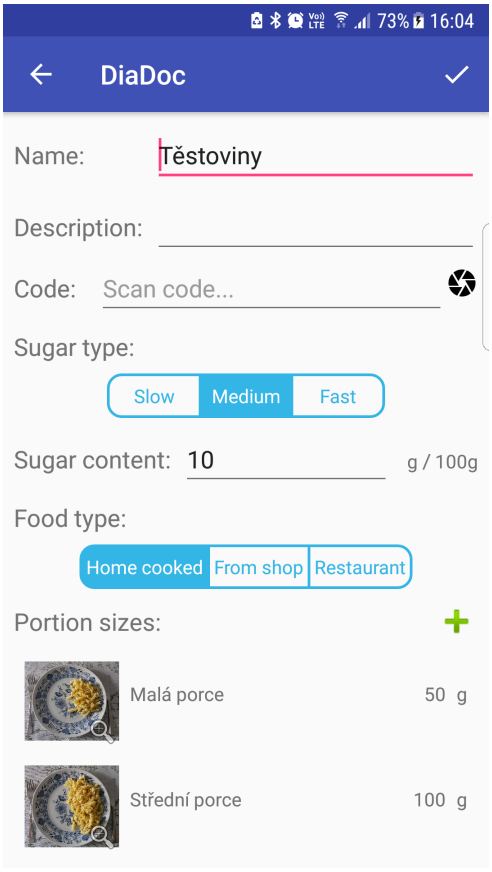
Jelikož aplikace musí zaznamenávat data z několika různých senzorů, i když není spuštěna, budeme potřebovat odpovídající služby běžící na pozadí. Služby mohou být dvojího typu: stále běžící nebo spuštěné podle potřeby, které se po vykonání práce zase vypnout.

Služby je vždy potřeba nastartovat, o což se v našem případě stará třída *AutostartReceiver*, což je *BroadcastRecievere*[17], nastavený tak, že se spustí po startu telefonu a také když aplikace byla aktualizována. Toto nastavení zajistí, že služby budou vždy běžet a budou aktuální. V případě služeb, které nemusí běžet stále, *AutostartReceiver* zajistí nastavení alarmu, který služby probudí v definovaných intervalech.

[17] Objekt sloužící k zachycení broadcastu



Obrázek 3.3: UI - Seznam bolusů

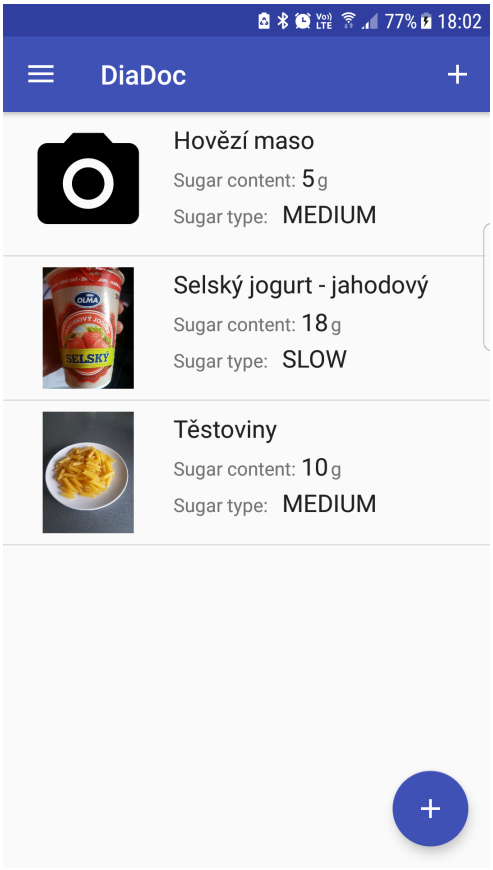


Obrázek 3.4: UI - Vytvoření jídla

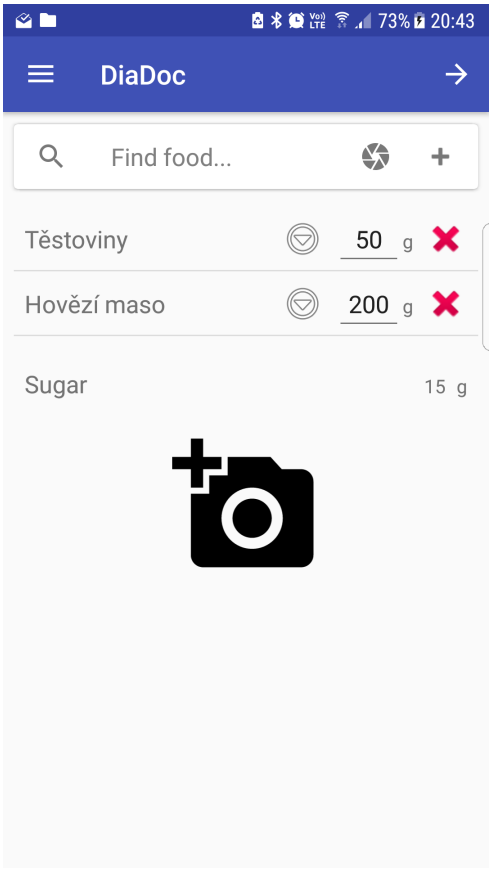
3.1.3.1 BluetoothLeService

Služba stále běžící na pozadí, která se stará o komunikaci se zařízeními připojenými přes Bluetooth Low Energy. Po startu služba z nastavení zjistí, zda jsou nakonfigurována nějaká zařízení. Pokud ano, tak se k nim pokusí připojit. Když se připojení podaří, nastaví zasílání notifikací při změně nějakého měřeného parametru. V případě neúspěšného připojení se ho pokusí navázat později. Při změně měřeného parametru obdrží služba novou hodnotu, kterou uloží do databáze. Podporovaná zařízení:

- Moodmetric Ring[14]
- MAXREFDES73# [13]
- Obecný senzor srdečního tepu



Obrázek 3.5: UI - Seznam vytvořených jídel



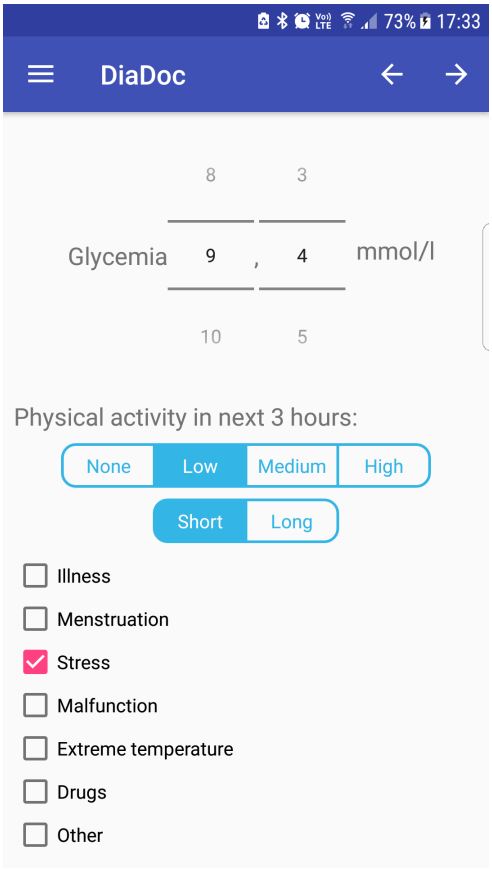
Obrázek 3.6: UI - Konzumované jídlo

3.1.3.2 SensorService

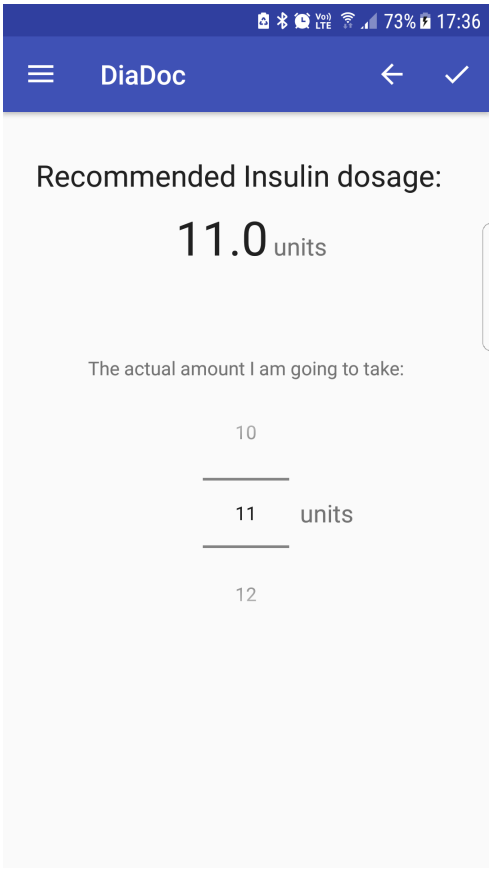
Služba spuštěná v pravidelném intervalu, která zaznamenává počet ušlých kroků za daný interval. Jelikož Android Sensor API[33] umožňuje pouze zjištění celkového počtu kroků od startu zařízení, budeme si muset vždy pamatovat minulou zjištěnou hodnotu a tu odečíst od aktuální. Tím získáme počet kroků od posledního měření. Ten je následně uložen do databáze a služba se ukončí.

3.1.4 Implementace architektury

Implementace architektury MVVM (viz 2.3.1) bude následující: *View* budou reprezentovat Activity resp. Fragmenty. Ty budou pomocí LiveData komunikovat s *ViewModely*. LiveData je implementace Observable, která je navíc Lifecycle-Avare tj. respektuje životní cyklus aktivit a fragmentů, takže se View aktualizují jen když je to možné a je to potřeba.



Obrázek 3.7: UI - Zadávání glykémie



Obrázek 3.8: UI - Doporučená dávka

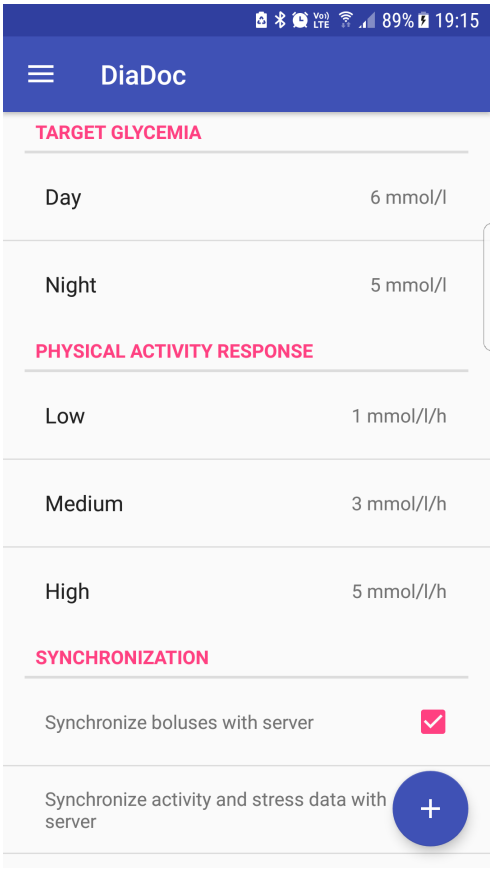
Samotné *ViewModely* budou instance třídy *android.arch.lifecycle.ViewModel*, která je taktéž Lifecycle-Avare a jsou navrženy tak, aby přežily změny konfigurace zařízení (jako např. změna orientace).

Model bude reprezentován Data Access Objekty, které získávají data z databáze (implementovány pomocí Room[29]). Data jsou získávána rovnou ve formě LiveData, takže *View* je upozorněno pokaždé, když dojde k nějaké změně. To je implementováno tak, že když se změní tabulka, na kterou je vázána instance LiveData, provede se dotaz znovu a aktualizovaná data se propagují přes *ViewModel* až do *View*.

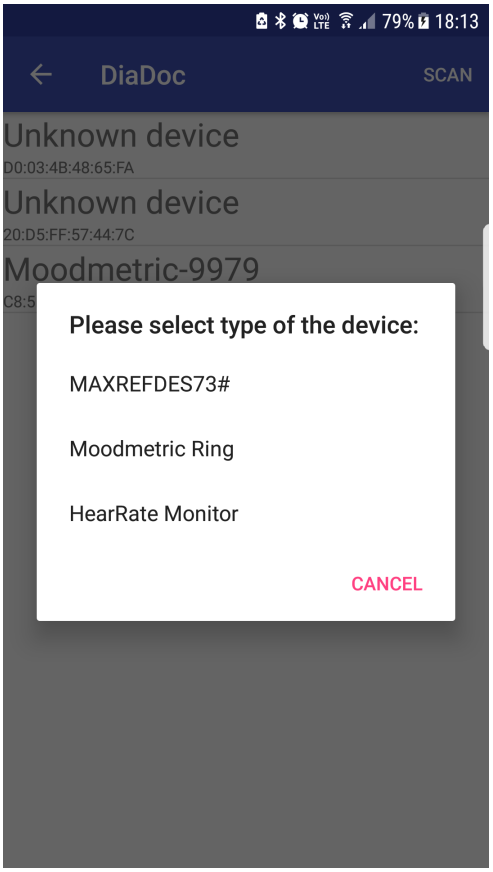
3.1.5 Implementace vybraných funkcí

3.1.5.1 Stanovení fyzické aktivity

Pro stanovení fyzické aktivity uživatele můžeme použít několik ukazatelů: počet kroků, srdeční tep, GSR a také zrychlení. Ale ne vždy budou všechny dostupné, protože uživatel obvykle např. nebude nosit senzor srdečního tepu,



Obrázek 3.9: UI - Nastavení

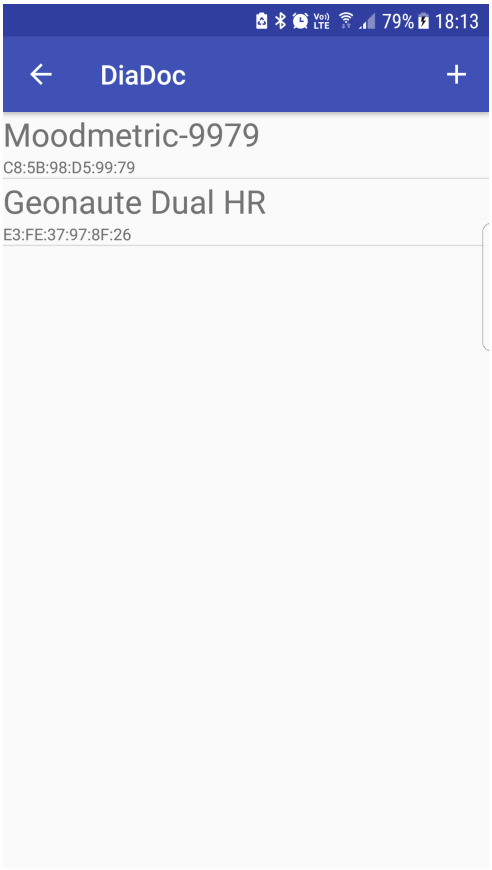


Obrázek 3.10: UI - Přidání senzoru

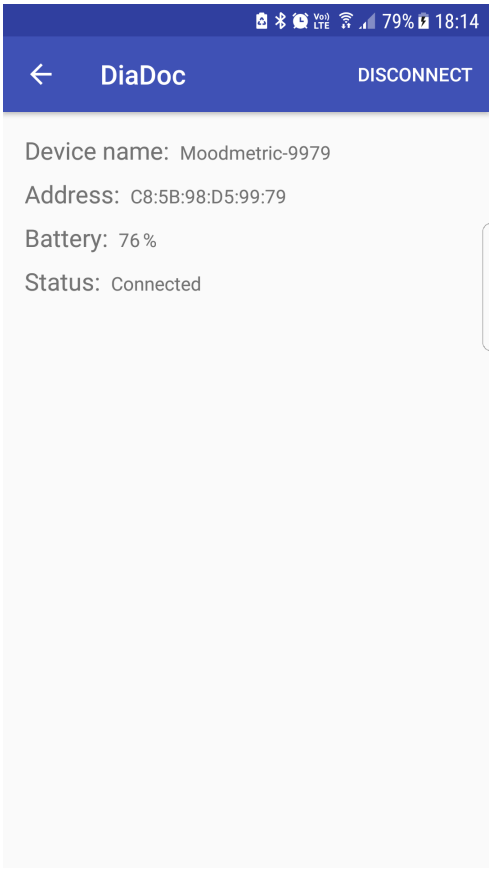
ale na sport si ho třeba vezme. Naproti tomu počet kroků bude znám vždy, ale nelze na něj zcela spoléhat, protože telefon nemusí mít uživatel přímo u sebe.

Z výše uvedených důvodů byl vytvořen interface *PhysicalActivityEstimator*, který bude abstrahovat jednotlivé ukazatele a dá nám odpověď ve formě jedné z hodnot popisujících intenzitu fyzické aktivity: NONE, LOW, MEDIUM a HIGH. Každý *Estimator* bude také schopen říct, do jaké míry je výsledek přesný, podle toho zda má dostatek naměřených údajů pro dané období. Samotný výběr nejpřesnějšího ukazatele bude probíhat podle priority se zohledněním aktuální přesnosti (pokud bude mít ukazatel s nejvyšší prioritou nízkou přesnost, použijeme další ukazatel v pořadí atd.).

```
public interface PhysicalActivityEstimator {
    Result estimate(Date now);
}
```



Obrázek 3.11: UI - Seznam senzorů



Obrázek 3.12: UI - Detail senzoru

```
public class Result {
        float accuracy;
        PhysicalActivity physicalActivity;
}
```

Jednotlivé ukazatele:

Srdeční tep Pokud známe maximální tepovou frekvenci, dokážeme určit v jakém tepovém pásmu se uživatel pohybuje.

Kroky V databázi si ukládáme v pravidelných intervalech kolik uživatel za danou dobu udělal kroků a tím pádem můžeme v každou denní dobu říct, zda se pohybuje průměrně (průměrný počet kroků za předchozí dny od půlnoci do aktuální denní doby je s nějakou odchylkou stejný jako počet dnes udělaných kroků), nebo podprůměrně resp. nadprůměrně.

GSR Zjistíme minimální a maximální hodnotu a vzniklý interval rozdělíme na 4 stejně velké pásma. Následně zjistíme průměrnou hodnotu za sledované období a zjistíme do jakého pásma spadá.

Akcelerace Spočítáme směrodatnou odchylku za sledované období a podle individuálně předdefinovaných intervalů zjistíme do jaké kategorie aktivita spadá.

3.1.5.2 Stanovení míry stresu

Pro určování míry stresu budeme používat ukazatel MM Level[15], který poskytuje Moodmetric Ring. V případě, že uživatel toto zařízení nebude používat, nebudeme míru stresu stanovovat automaticky, ale pouze na základě vstupu od uživatele při výpočtu bolusu.

MM Level je hodnota z intervalu 0 – 100, kdy 100 je nejvyšší míra vybuzení a 0 nula nejnižší. Ačkoli by hodnoty podle [15] měly být porovnatelné mezi uživateli, u každého diabetika má stres jiný vliv, a tak budeme potřebovat experimentálně zjištěný práh, nad kterým budeme započítávat stres do výpočtu dávky. V případě, že průměr za určité období překročí stanovenou mez, bude dávka navýšena o adekvátní[18] množství.

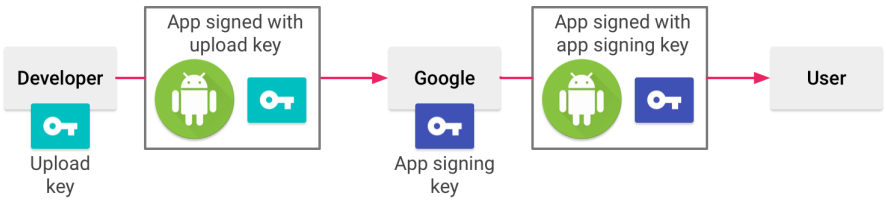
3.1.6 Nasazení

Nasazení na uživatelská zařízení bude probíhat přes službu Google Play[34]. Proces vytvoření a nasazení nového vydání probíhá následovně:

1. Ze zdrojových kódů aplikace pomocí build scriptu vytvoříme soubor APK[19].
2. APK podepíšeme svým upload klíčem.
3. Podepsané APK nahrajeme do Google Play pomocí Google Play Console[35].
4. Google Play ověří upload klíč[20] a následně podepíše APK distribučním klíčem
5. Aplikace je následně dostupná k instalaci pro uživatele Google Play. V případě, že takto vydáme novou verzi existující aplikace, aktualizace se postupně nainstaluje všem uživatelům starší verze.

Průběh podepisování aplikace ukazuje obrázek 3.13.

[18] také experimentálně zjištěné
[19] Android Package
[20] Podpis je ověřován proti klíči asociovanému s naší aplikací v Google Play



Obrázek 3.13: Průběh podepisování aplikace[36]

3.2 Backend server

3.2.1 Použité technologie

3.2.1.1 Programovací jazyky

Pro implementaci Backend serveru použijeme několik programovacích jazyků. Samotnou aplikační logiku budeme implementovat v jazyce Java. Jedná se o moderní multiplatformní jazyk, oblíbený nejen v podnikové sféře pro svůj rozsáhlý ekosystém knihoven, čitelnost[21] a rychlost vývoje. Nezanedbatelným benefitem také bude možnost sdílení kódů s mobilní aplikací. V neposlední řadě mám s tímto jazykem dlouholeté zkušenosti, takže vývoji nebude předcházet seznamování se s novým jazykem, které může často trvat velmi dlouho.

Pro implementaci funkcí na straně webového prohlížeče, jako je načítání dat z backendu na pozadí, zvolíme JavaScript. Tento interpretovaný jazyk podporují všechny moderní prohlížeče. Pro jednodušší práci s DOM elementy a daty zvolíme knihovnu jQuery[37]. Samotné webové stránky budou napsány v jazyce HTML5.

Na komplexní dotazování se nad daty v databázi použijeme jazyk SQL.

3.2.1.2 Knihovny

Protože není potřeba znovu vynalézat kolo, použijeme pro vývoj řadu knihoven, které nám umožní soustředit se na samotnou implementaci aplikace a nikoli na podpůrné funkce. Jako hlavní jmenujme:

- Spring framework[38] - framework s mnoha součástmi např. pro vývoj REST APIs, MVC a Dependency Injection
- jUnit[39] - testovací framework
- Twitter Bootstrap[40] - sada CSS usnadňující tvorbu HTML stránek
- Google API Client[41] - knihovna pro práci s Google APIs

[21] Syntaxe jazyka vychází z C/C++

3.2.1.3 Databáze

Jako úložiště dat použijeme databázi PostgreSQL[42] konkrétně ve verzi 10.4. Tato moderní, objektově relační databáze poskytuje komfort velkých databázových strojů jako jsou Oracle DB nebo MSSQL, avšak bez licenčních poplatků. V případě potřeby je možno databázi horizontálně škálovat.

3.2.1.4 Aplikační server

Naše aplikace bude potřebovat pro svůj běh aplikační server podporující Java EE technologie. Použijeme opensource server Apache Tomcat[43], který je možno k aplikaci přibalit během buildu, a tak efektivně vytvořit soubor JAR[22], který lze bez dalších závislostí spustit v Java Virtual Machine.

3.2.2 Implementaci architektury

Implementace zvolené MVC architektury bude následující:

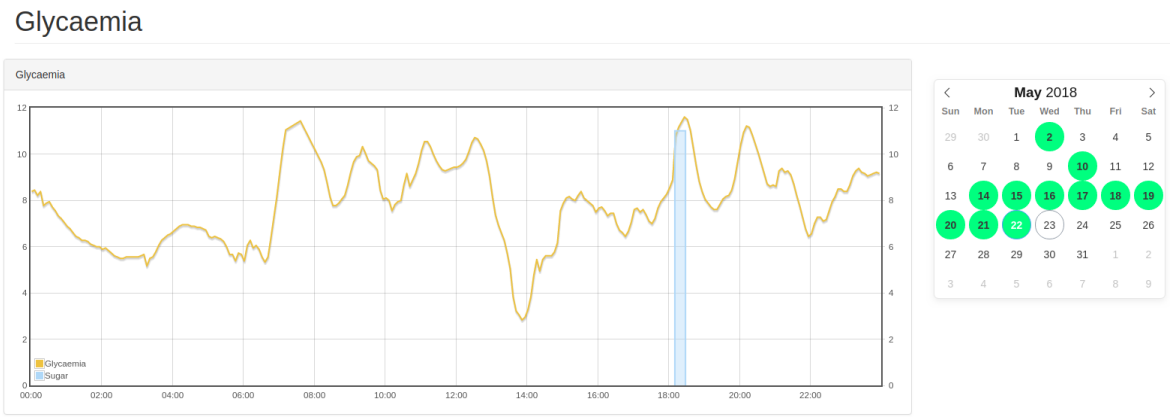
- *Model* bude reprezentován databázovou a servisní vrstvou tj. třídami se sufixem **Repository* nebo **Service*. *Repositories* budou Data Access Objekty přímo přistupující do databáze. Budou provádět typicky pouze *CRUD* operace. Komplexnější business logika bude obsažena v *Services*, které budou pro ostatní komponenty zprostředkovávat interakci s *Modelem*.
- *View* budou šablony HTML stránek, které bude *Controller* poskytovat uživateli naplněné příslušnými daty.
- *Controller* budou zastupovat třídy se sufixem **Controller*, které budou komunikovat s výše zmíněnými servisními třídami. Speciálním druhem pak budou třídy **RestController*, které nepoužívají *View* a klientům poskytují data ve formátu JSON.

3.2.3 Uživatelské rozhraní

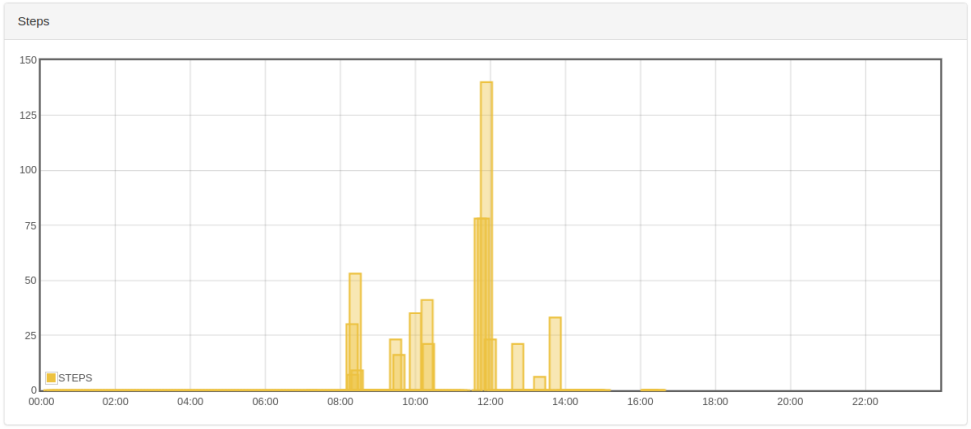
Uživatelské rozhraní webové aplikace je pro účely prototypu maximálně zjednodušeno a je orientováno primárně na zobrazení zaznamenaných dat. Vybíráme několik ukázek UI:

- Zobrazení glykémie sesazené s údaji o aplikovaných bolusech – obr. 3.14. Zelené pozadí dnu v kalendáři označuje, že pro tento den jsou dostupná data.
- Zobrazení počtu kroků – obr. 3.15
- Nastavení parametrů pacienta – obr. 3.16

[22] Java Archive



Obrázek 3.14: Zobrazení glykémie

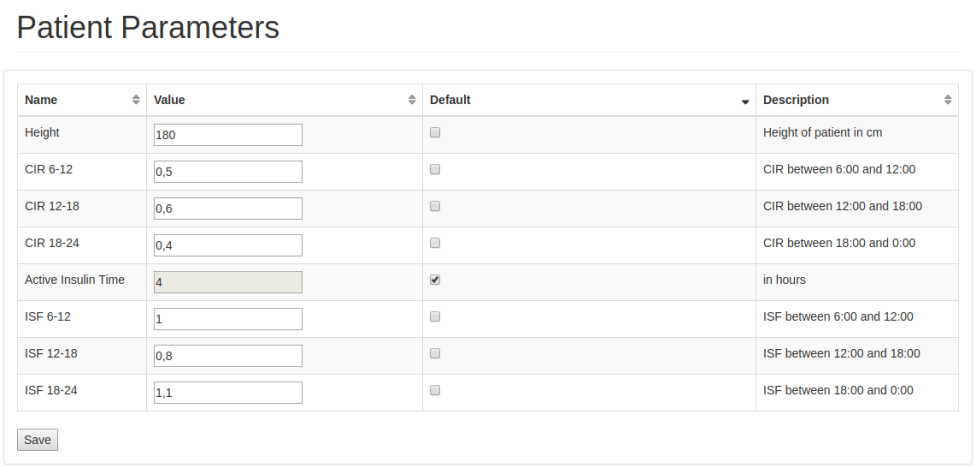


Obrázek 3.15: Zobrazení počtu kroků

3.2.4 Zabezpečení

3.2.4.1 Komunikace

Jak plyne z požadavku REQ18, veškerá komunikace se serverem musí být zabezpečena. Z tohoto důvodů se veškerá komunikace šifruje pomocí SSL/TLS, což zabraňuje odposlechu např. přihlašovacích údajů. Pro použití SSL musí být server vybaven vlastním certifikátem, kterým se komunikace podepisuje. Pro vývoj a testování byl vygenerován self-signed certifikát, který ale bude pro produkční nasazení nahrazen certifikátem vydaným certifikační autoritou.



Obrázek 3.16: Nastavení parametrů pacienta

3.2.4.2 Autentizace

Autentizace je proces ověření identity uživatele. Vzhledem k požadavku REQ12 musí aplikace podporovat přihlášení pomocí Google účtu. To je implementováno protokolem OAuth 2.0. Samotný proces přihlášení probíhá následovně:

1. Uživatel na přihlašovací stránce klikne na tlačítko *Přihlásit pomocí Google*
2. Uživatel je přesměrován na servery Google, který ověří, že je přihlášen ke svému Google účtu, popř. mu umožní se přihlásit
3. V případě, že se uživatel přihlašuje k aplikaci poprvé, je dotázán zda jí chce udělit potřebná oprávnění
4. Nakonec dojde k přesměrování zpět na web naší aplikace, která zkontroluje na základě předaného tokenu, že přehlášení bylo úspěšné
5. Uživatel je již jako přihlášený přesměrován na hlavní stránku aplikace

3.2.4.3 Autorizace

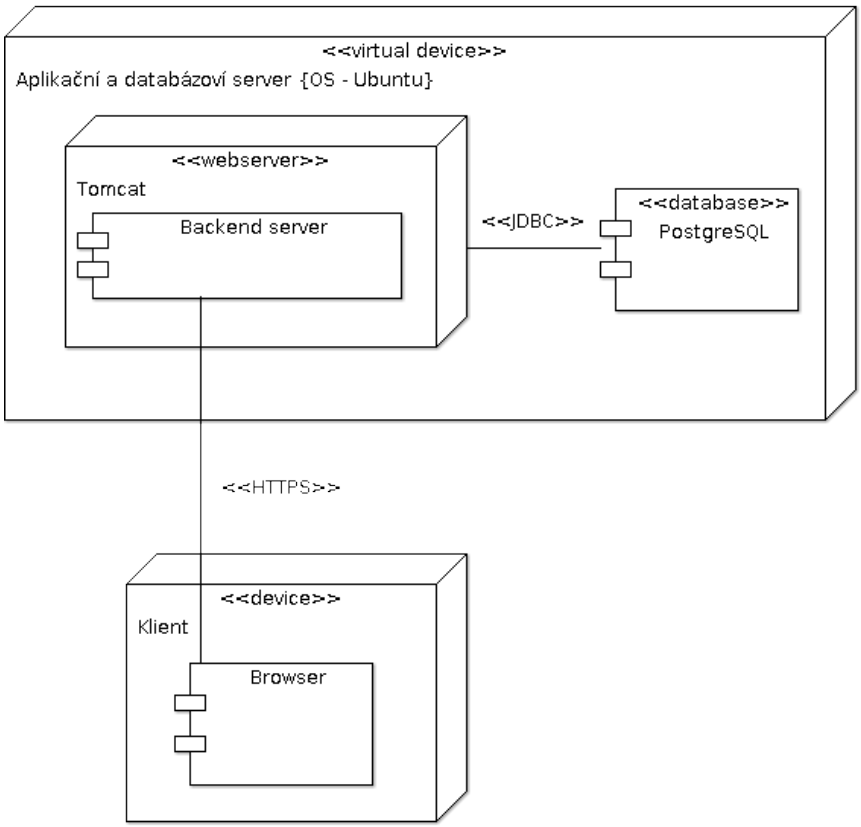
Autorizace je proces, který určuje, zda je uživatel oprávněn k vykonání konkrétní akce. Provádí se před vyřízením každého požadavku. Nejprve se ověří, zda je uživatel přihlášen. Pokud ne, je přesměrován na přihlašovací obrazovku. Pokud ano, je mu na základě uživatelského jména[23] přiřazena množina rolí,

[23] konkrétně použijeme email

která určuje, jaké akce může provádět. Pro jemnější kontrolu nad zdroji, ke kterým mohou lékaři přistupovat, existuje v databázi vazba 1:N mezi entitou *Lékař* a *Pacient*, podle které se určuje, zda je lékař oprávněn pracovat s daty konkrétního pacienta.

3.2.5 Nasazení

Obr. 3.17 ukazuje jak bude vypadat nasazení finální aplikace. Backend server bude nasazen na virtualizovaném serveru s operačním systémem Ubuntu 16.4. Na stejném serveru poběží i databáze PostgreSQL[42].



Obrázek 3.17: Diagram nasazení

Kapitola 4

Testování

4.1 Mobilní aplikace

Mobilní aplikace byla testována na několika úrovních. Po vytvoření logického celku[24] jsem jeho funkčnost ověřil ručně. Tímto způsobem je odhaleno největší množství chyb a jsou také nejrychleji opraveny. Zásadní nevýhodou takového testování je nemožnost automatizovaného opakování. Tuto nevýhodu odstraňuje nasazení automatických testů. Těch existuje mnoho druhů, které se liší množstvím testovaného kódu, běhovým prostředím, rychlostí běhu a v neposlední řadě také rozhraním, skrze které testy interagují s produkčním kódem.

Druhy automatických testů použitých při vývoji popisují části 4.1.1 - 4.1.4. Sekce 4.1.5 popisuje manuální testování měření fyzické aktivity.

4.1.1 Unit testy

Unit testy testují nejmenší části kódu, obvykle jednotlivé třídy nebo funkce. Testování probíhá v izolaci od ostatního produkčního kódu a všechny potencionální závislosti mockují tj. nahrazují se implementací s přesně definovaným chováním pro účely testu. Spouštění unit testů je obvykle velmi rychlé, a proto mohou být součástí každého sestavení aplikace. V případě vývoje pro Android OS běží unit testy v rámci JVM, na které probíhá samotné sestavení, takže lze testovat pouze kód bez závislostí na Android frameworku (pokud není možné je mockovat). [44] Samotné testy jsou implementovány v testovacím frameworku jUnit[39] a mokovacím frameworku Mockito[45].

4.1.2 Instrumentované testy

Instrumentované testy jsou druhem integračních testů(viz 4.2.2) a spouštějí se na reálném mobilním zařízení popř. v emulátoru, a proto mohou obsahovat závislosti na Android frameworku. Před samotným testování je vytvořeno

[24] Funkce, třída nebo např. obrazovka

APK s testy. To se nainstaluje na zařízení, na kterém se poté spustí. Tyto testy umožňují testovat všechny součásti aplikace včetně např. DAOs[25], která pracují s SQLite databází. Nevýhodou této třídy testů je potřeba přístupu k HW zařízení nebo emulátoru a pomalé provádění, způsobené nutností APK nainstalovat a spustit. [44]

4.1.3 UI testy

UI testy interagují s aplikací, na rozdíl od ostatních testů, pomocí uživatelského rozhraní, tedy stejně jako to budou dělat budoucí uživatele. Z tohoto důvodu se jedná o jedny z neužitečnější testů. Samotný test simuluje klikání (popř. jiné akce jako swipe) na elementy do uživatelského rozhraní a kontroluje jak aplikace reaguje. UI elementy jsou v testu identifikovány pomocí svých ID, takže např. změna umístění tlačítka test nerozbije. Testovací framework Espresso[46] dokáže kontrolovat i to, zda je element, na který chce test kliknout, skutečně viditelný a tak věrně napodobuje chování uživatele.

4.1.4 Testování migrací

Během vývoje aplikace je čas od času z důvodu změny požadavků nezbytné změnit databázové schéma a s ním i část dat. Protože chceme uživatelům zachovat stávající data, použijeme databázové migrace tj. sekvenci příkazů (obvykle v jazyce SQL), která provede požadované změny. Migrace obsahuje vždy verzi databáze, na které se spouští a výslednou verzi. Tedy např. Migrace (3,4) umí z databáze ve verzi 3 udělat verzi 4. Migrace je nutné psát pokaždé, když budeme vydávat novou verzi aplikace, ve které došlo ke změně schématu. Abychom věděli, jaké migrace máme spustit, použijeme pomocnou tabulku v databázi, ve které si budeme ukládat verzi databáze. Při spuštění aplikace tedy nejdříve zkontrolujeme tuto verzi a pokud je nižší než aktuální[26], spustíme odpovídající migraci. Pokud tabulka neexistuje, víme že se jedná o čistou instalaci a schéma vytvořme buď rovnou v aktuální verzi, nebo spustíme inkrementálně všechny dostupné migrace.

Protože používáme databázovou vrstvu Room[29], nemusíme se starat o samotné spouštění migrací. Stačí pouze dodat seznam dostupných migrací a aktuální verzi databáze. Room pokaždé před startem aplikace zkontroluje verze a v případě potřeby se postará o spuštění odpovídajících migrací. Pokud nenalezne potřebnou migraci, aplikace se ukončí s chybou. Room jde dokonce i dále a ukládá si při každé změně verze celé aktuální schéma, takže po provedení migrací dokáže zkontrolovat, že výsledek je stejný jako kdyby se databáze vytvářela z databázových entit.

Tato kontrola se provádí za běhu a v případě, že schémata nesouhlasí, dojde opět k ukončení aplikace. To je pro uživatele velice nepříjemné, a proto je

[25] Data Access Object

[26] Za aktuální verzi považujeme tu, kterou potřebuje aplikace ke svému běhu

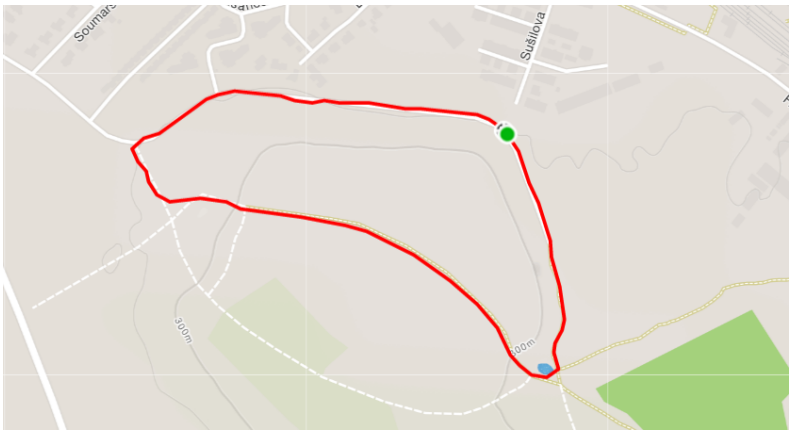
důležité migrace náležitě testovat. Room k tomuto účelu poskytuje třídu *MigrationTestHelper*, která nám umožňuje vytvořit databázi v konkrétní verzi, vložit do ní testovací data a spustit migraci na další verzi. Následně můžeme jednak zkontrolovat, že testovací data byla migrována korektně a Room zároveň provede kontrolu schématu (stejně jako za běhu). To nám umožňuje efektivně testovat databázové migrace, a tak eliminovat velkou část potenciálních chyb už při vývoji.

4.1.5 Testování měření fyzické aktivity

Testování měření senzorů probíhalo na okruhu, který ukazuje obr. 4.1. Jako první byla trasa absolvována pomalou chůzí v tempu 11:17 min/km tj. asi 5 km/h. Po pauze došlo k odpočinku, dokud se tepová frekvence nevrátila na původní hodnotu cca 85 tepů/min. Následoval další okruh, tentokrát rychlou chůzí v tempu 9:10 min/km (6,5 km/h). Poté další pauza a poslední část testu: běh v tempu 4:54 min/km (12 km/h). Jako poslední bylo provedeno referenční měření v relativním klidu při práci na PC.

- Parametry trasy:
  - Délka: 1,53 km
  - Převýšení: 7 m
  - Povrch: 50% asfalt, 50% tráva
- Použitý hardware:
  - Mobilní telefon: Samsung Galaxy S7 edge
  - Senzor srdečního tepu: Geonaute BluetoothSmart HRM Belt
  - MAXREFDES73#
  - Moodmetric Ring
  - GPS hodinky: Garmin vívoactive®
- Zaznamenávané údaje:
  - MM Level (Moodmetric Ring)
  - Srdeční tep
  - Počet kroků (měřeno telefonem)
  - Počet kroků (měřeno krokoměrem integrovaným v senzoru srdečního tepu)
  - GSR (Moodmetric Ring)
  - GSR (MAXREFDES73#)
  - Zrychlení (Moodmetric Ring)

- Testovací subjekt:
  - Pohlaví: Muž
  - Věk: 27 let
  - Váha: 70 kg
  - Trénovanost: Střední



Obrázek 4.1: Testovací okruh

4.1.5.1 Srdeční tep

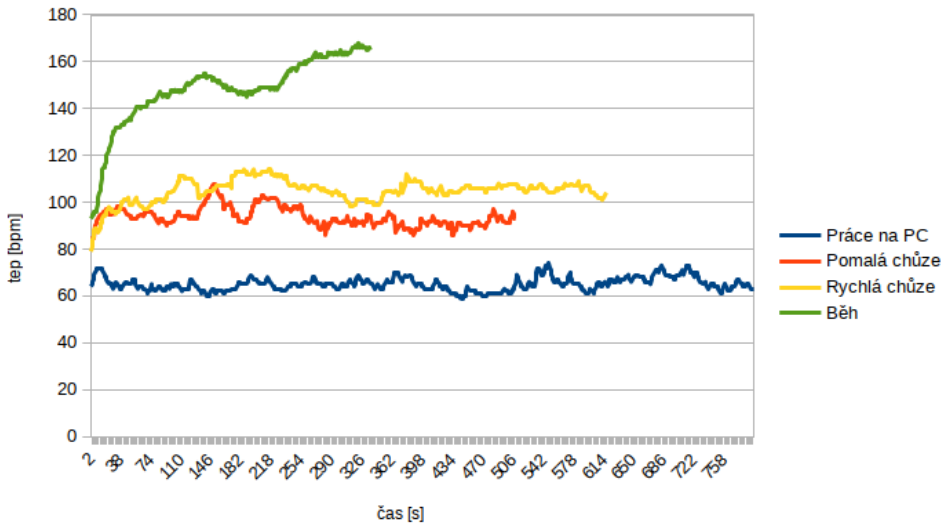
Jako referenční, pro stanovení skutečné fyzické námahy, probíhalo měření srdečního tepu hrudním pásem. Naměřené údaje ukazuje obr. 4.2. Data potvrzují očekávané. Náročnost fyzické aktivity je nejnižší u práce na PC a stupňuje se podle rychlosti chůze až k běhu.

Pozn. : U "pomalé chůze"došlo k výpadku senzoru před koncem měření, a proto několik minut dat chybí. Trend je ovšem jasný a měření tedy nebylo opakováno.

4.1.5.2 Kroky

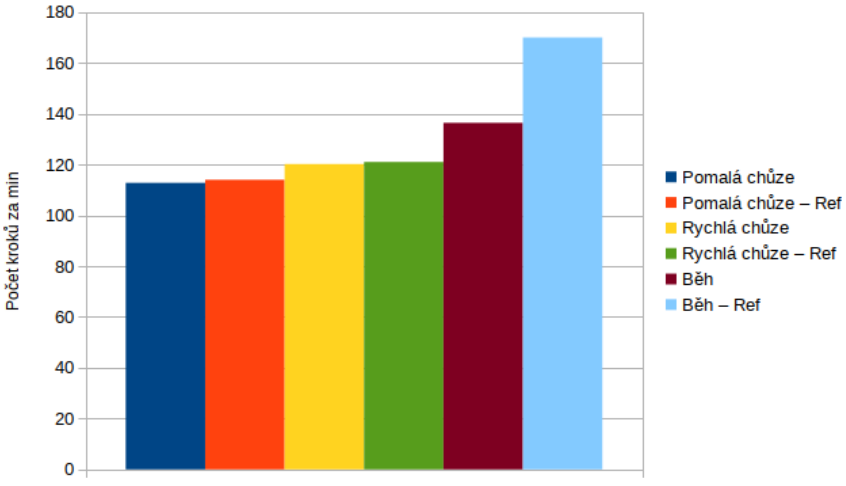
Měření kroků probíhá v intervalu 5-ti minut, kdy dojde k probuzení *SensorService*, který vyčte novou hodnotu ze Sensor API (viz 3.1.3.2). Naměřenou kadenci kroků ukazuje graf na obr. 4.3. Jako referenční byla použita data zaznamenaná tepovým senzorem.

Naměřené hodnoty se od referenčních liší v případě práce na PC, pomalé a rychlé chůze minimálně (v řádu jednoho procenta), ale v případě běhu je



Obrázek 4.2: Srdeční tep

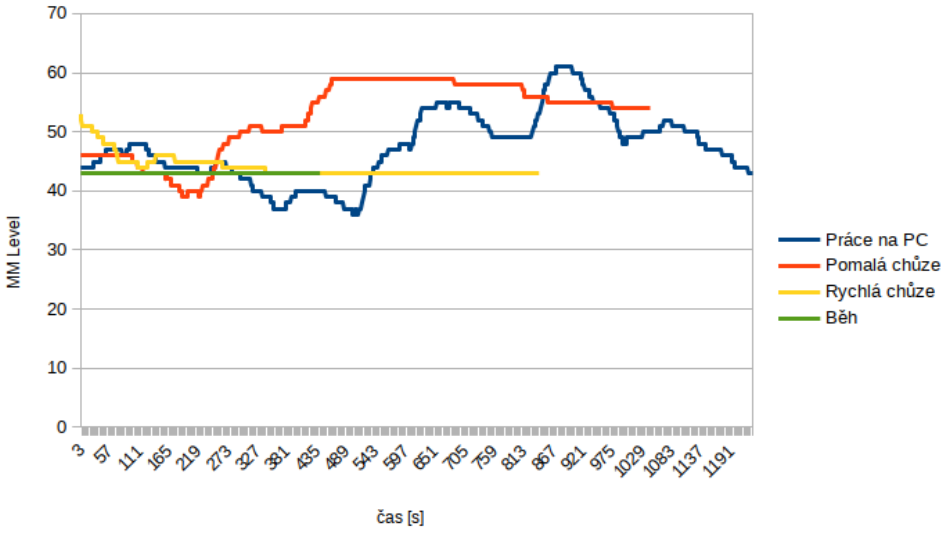
chyba skoro 20%. To je způsobeno právě dlouhým intervalem mezi měřeními, takže je 7-mi minutový běh pokryt dvěma intervaly, které ovšem začínají nebo končí před, respektive po běhu, a tak zahrnují i pauzy.



Obrázek 4.3: Kroky

4.1.5.3 MM Level

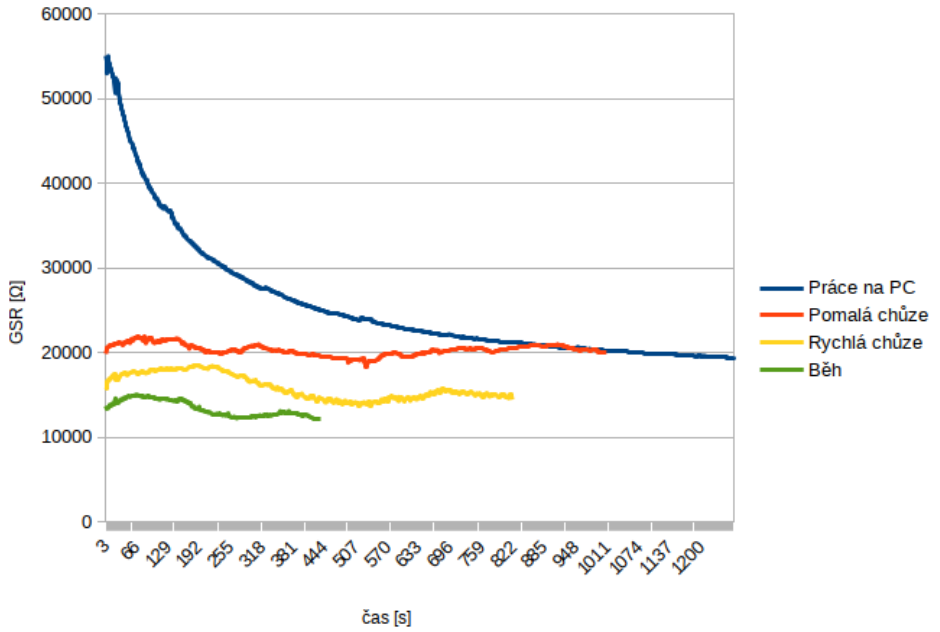
MM Level[15] je metrika určená primárně pro měření psychického stavu jedince (vysoké hodnoty indikují stres, nízké klid), a proto nepřekvapí, že naměřená data (obr. 4.4) na první pohled nijak nereflektují fyzickou aktivitu.



Obrázek 4.4: MM Level

4.1.5.4 GSR

Jak ukazují obr. 4.5 a 4.6 vliv fyzické námahy na galvanický odpor kůže je evidentní. Naměřené absolutní hodnoty se samozřejmě liší podle umístění senzoru (prst nebo zápěstí), ale trend je jasný. S vyšší fyzickou námahou klesá odpor, což je způsobeno vyšší aktivitou potních žláz. Na obou grafech můžeme pozorovat, že k ustálení hodnoty dojde až po několika minutách po změně činnosti. Na grafu 4.6 je také vidět, že u pomalé chůze došlo ke ztrátě kontaktu prstenu s kůží a tím pádem i ke skokovému nárůstu odporu. Po několika minutách se hodnota opět stabilizovala.



Obrázek 4.5: GSR - MAXREFDES73#

	Práce na PC	Pomalá chůze	Rychlá chůze	Běh
Směrodatná odchylka	9.0	20.5	24.2	65.1

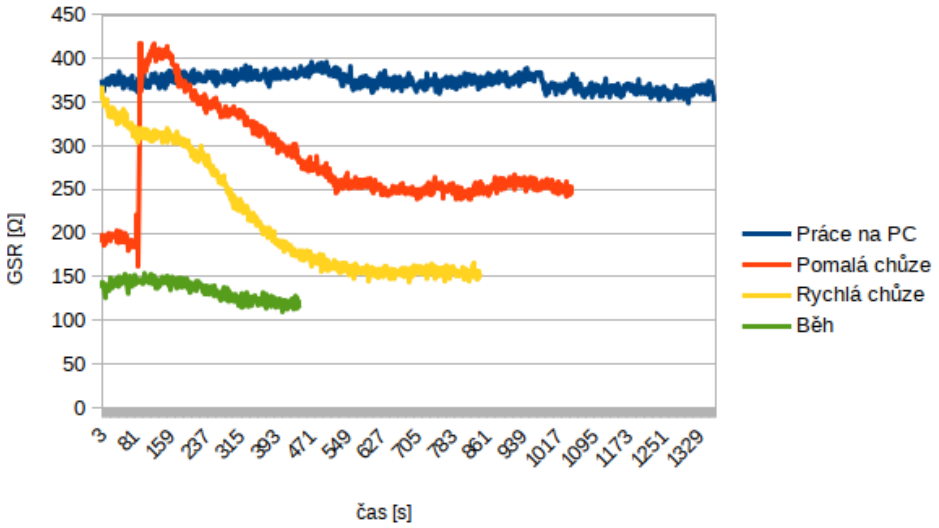
Tabulka 4.1: Směrodatná odchylka zrychlení

4.1.5.5 Zrychlení

Naměřené hodnoty zrychlení ukazuje obr. 4.7. Moodmetric Ring dokáže zaznamenávat zrychlení ve osách x, y, z, my ale pro zjednodušení budeme používat velikost vektoru získanou jako $\sqrt{x^2 + y^2 + z^2}$.

Z dat je jasně patrný vliv intenzity pohybu. Jako zajímavější, než samotné absolutní hodnoty zrychlení, se ukazuje jejich variabilita, jak můžeme vidět v tabulce 4.1. Stanovení intenzity pohybu ze zrychlení změřené pomocí prstenu Moodmetric Ring je tedy pro určitou množinu činností možné. Výjimku, se kterou je nutno počítat, budou představovat sporty jako např. spinning[27], při kterých nedochází k výraznému pohybu horních končetin.

[27] jízda na stacionárním kole



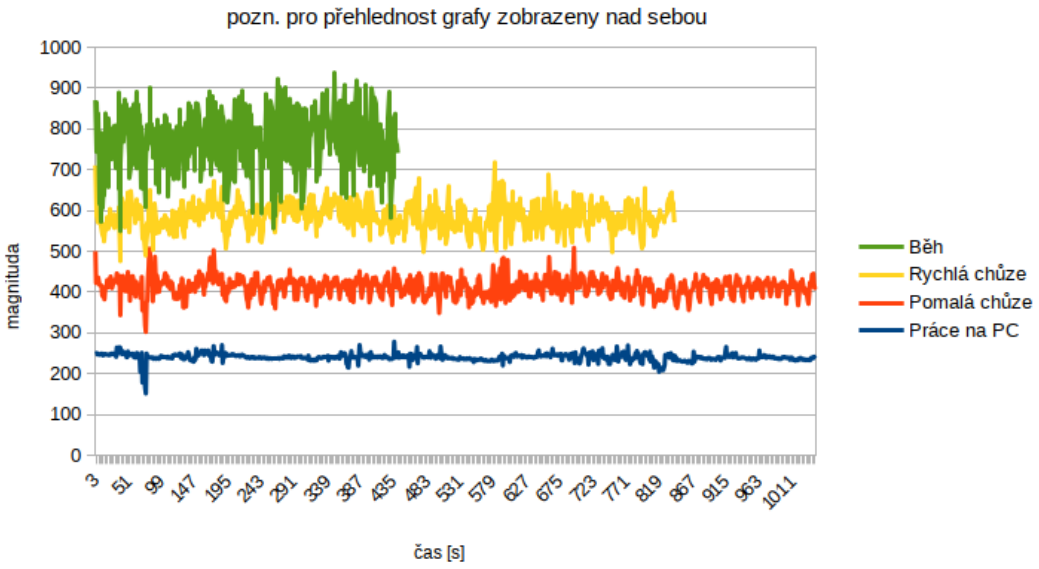
Obrázek 4.6: GSR - Moodmetric Ring

4.1.5.6 Závěry z testování

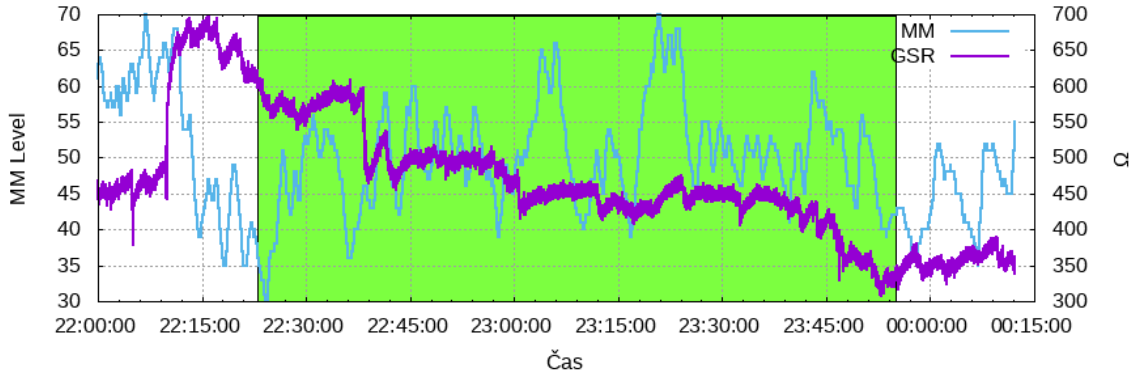
- Zrychlení, GSR a počet kroků se pro měření fyzické aktivity hodí a jejich použití v reálném nasazení nic nebrání. Naproti tomu MM level je pro měření nevhodný.
- Měření srdečního tepu je také více než vhodné, ale použití hrudního pásu je nepohodlné. V budoucnu by stálo za to otestovat optické senzory na zápěstí.
- Výdrž baterie *MAXREFDES73#* je nízká a nepřibližuje se 15 h, které udává výrobce. Reálně baterie vydržela maximálně 3 hodiny.
- Naproti tomu *Moodmetric Ring* reálně zvládne na jedno nabití bez problémů několik dní. Jedinou jeho nevýhodou je forma prstenu (a nikoliv drobného), která by mohla vadit mužům. Důležité je zvolit správnou velikost, protože pokud je prsten volný, měření často nefunguje korektně.

4.1.6 Testování měření stresu

Pro simulaci stresové situace jsem, z důvodů jasného časového rámce, zvolil sledování hororového filmu. Konkrétně se jednalo o snímky *Hory mají oči*[47] a *Kill List*[48]. Sledování probíhalo ve večerních hodinách ve známém prostředí. Filmy jsem viděl poprvé pro dosažení maximální úrovně stresu. Měřen byl MM Level a GSR, pomocí Moodmetric Ring. Naměřené hodnoty ukazují obr. 4.8, 4.9 a tabulka 4.2. Doba sledování je na grafech zvýrazněna zelenou barvou.



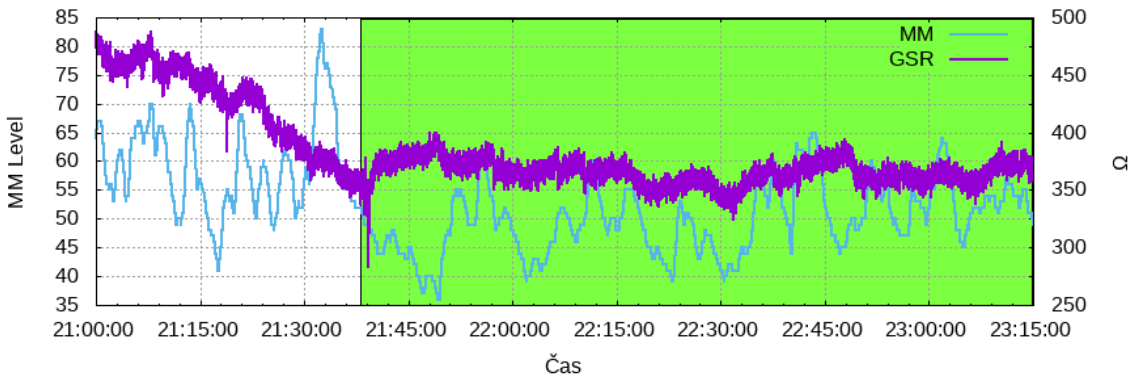
Obrázek 4.7: Zrychlení



Obrázek 4.8: Stres – Kill List

4.1.6.1 Závěry z testování

Testování poznamenalo několik faktorů. Zaprvé, bylo nemožné předem odhadnout reakci na konkrétní film, což se projevilo zejména u snímku *Kill List*,



Obrázek 4.9: Stres – Hory mají oči

Průměrný MM Level	Hory mají oči	Kill List
Před začátkem	44.9	50.0
Během filmu	52.8	49.3

Tabulka 4.2: Stres – MM Level

který až na několik scén, nebyl subjektivně vůbec děsivý. Film *Hory mají oči* byl v tomto ohledu „lepší“, ale ani tak jsem nezaznamenal výraznější strach.

Zadruhé, pro přesnější výsledky by bylo zřejmě vhodné, nechat hodnoty ustálit déle než půl hodiny před začátkem, jak se to stalo v případě tohoto měření.

Na obrázku 4.9 je jasně viditelná změna trendu GSR po začátku filmu. Ta ovšem neodpovídá očekávání, protože GSR by se mělo snižovat s narůstajícím stresem. Možné vysvětlení je, že ve stejné době došlo k ustálení po nasazení prstenu. Toto tvrzení je podpořeno i tím, že ustálená hodnota odpovídá údajům naměřeným pří práci na PC z obr. 4.6. Ani hodnoty MM Level nevykazují signifikantní změny během sledování filmu, jak je vidět v tabulce 4.2.

Testování můžeme tedy shrnout tak, že z výše uvedených důvodů, nebyl prokázán jasný vliv stresu na sledované ukazatele. Testování by bylo vhodné zopakovat v reálných stresových situacích a na vetším vzorku testovacích subjektů, což je ovšem nad rámec této práce.

4.2 Backend server

4.2.1 Unit testy

Pro jednotkové testy[49] backendu použijeme také framewory JUnit[39] a Mockito[45]. Samotné psaní testů probíhá stejně jako u mobilní aplikace. Unit testy jsou

opět součástí buildu, takže v případě dostatečného pokrytí hned uvidíme, když něco nebude fungovat.

4.2.2 Integrační testy

Integrační testy verifikují systém na úrovni spolupráce jednotlivých komponent. Použitý Spring Framework[38] v sobě obsahuje podporu pro psaní integračních testů ve formě třídy *SpringRunner*[28], která zajišťuje (mimo jiné) start aplikace před začátkem testu.

4.2.3 API Testy

Jelikož Backend poskytuje API mobilní aplikaci, musíme ho také testovat. Na to použijeme framework REST Assured[50], který nám umožňuje jednoduše nadefinovat přesnou podobu requestu a následně validovat odpověď serveru. REST Assured zároveň nabízí podporu Spring MVC, takže můžeme aplikaci nastartovat stejně jako v běžném integračním testu a následně kontrolovat chování API.

[28] Runner pro jUnit

Závěr

Tato práce si kladla za cíl vyvinout systém pro pacienty s diabetem a jejich lékaře, který bude pacientům doporučovat optimální dávku inzulinu a lékařům umožní přistup k nejrůznějším datům získaným od pacienta.

Byla implementována mobilní aplikace pro systém Android, která dokáže na základě konzumovaného jídla, fyzické aktivity a mnoha dalších parametrů určit, kolik inzulinu by si měl uživatel aplikovat. Aplikace spolupracuje s několika druhy hardwarových senzorů, které měří fyzickou a psychickou aktivitu uživatele.

Data o aplikovaných bolusech, konzumovaném jídle a údaje získané ze senzorů jsou synchronizována s webovou aplikací, kde je mohou lékaři prohlížet a sesazovat s daty z kontinuálních senzorů glykémie. Data mohou poté analyzovat a podle výsledků upravovat pacientům výpočet dávky.

Testování systému na reálných datech ukázalo, že měření fyzické aktivity funguje spolehlivě a mělo by být přínosem při kompenzaci diabetu. Naproti tomu na sledování stresu je potřeba ještě zapracovat aby pracovalo korektně.

V současnosti se rozbíhá testování s reálnými pacienty trpícími diabetem.

Možná zlepšení

Při tvorbě aplikace jsem narazil na mnoho funkcí, o které by se dala vylepšit, ale bohužel na to prozatím nebyl čas. V budoucnu bych aplikaci rád rozšířil o následující funkcionalitu:

- Využití strojového učení pro algoritmus, který doporučuje velikost dávky inzulinu. Pokud budou pacienti aplikaci aktivně využívat, mělo by být možné nasbíraná data využít pro pokročilejší analýzu reakcí na různé podněty a podle toho upravovat doporučenou dávku.
- Zahrnutí tuků a bílkovin do výpočtu.
- Integrace s nějakou existující databází potravin, která by zjednodušila zadávání jídla. Stačilo by vyhledat potravinu ze společné databáze, která by obsahovala kompletní a hlavně verifikované nutriční hodnoty. V současnosti probíhají jednání o spolupráci s databází NutriData[51].
- Podpora Bluetooth glukometrů, která by umožnila automaticky získat hodnotu aktuální glykémie a uživatel by ji nemusel zadávat.
- Integrace s Google Fit[52], které poskytují komplexní informace o fyzické aktivitě uživatele.

Literatura

[1] Stechova, K.: *Lecba inzulinovou pumpou, aneb, Kazdodenni zivot rodiny Novakovy : prirucka pro pacienty s diabetem*. Praha: Maxdorf, 2013, ISBN 978-80-7345-338-1.

[2] Stechova, K.: *Diabetes mellitus 1. typu : [pruvodce pro kazdodenni praxi*. Praha: Maxdorf, 2014, ISBN 978-80-7345-377-0.

[3] Olsovsky, J.: *Diabetes mellitus 2. typu : pruvodce osetrujiciho lekare*. Praha: Maxdorf, 2012, ISBN 978-80-7345-277-3.

[4] Medtronic: *Insulin Pump Therapy*. [cit. 2018-05-10]. Dostupné z: `https://www.medtronicdiabetes.com/treatments/insulin-pump-therapy`

[5] FatSecret: *FatSecret*. [cit. 2018-05-05]. Dostupné z: `https://www.fatsecret.com/`

[6] Sirma Medical Systems: *Diabetes:M – Your Diabetes Management App*. [cit. 2018-05-05]. Dostupné z: `https://www.diabetes-m.com/`

[7] Google: *Diabetes:M - Google Play*. [cit. 2018-05-05]. Dostupné z: `https://play.google.com/store/apps/details?id=com.mydiabetes&hl=en`

[8] Social Diabetes: *Social Diabetes*. [cit. 2018-05-05]. Dostupné z: `https://www.socialdiabetes.com/en/`

[9] Google: *Social Diabetes - Google Play*. [cit. 2018-05-05]. Dostupné z: `https://play.google.com/store/apps/details?id=com.socialdiabetes.android&hl=en`

[10] Live Science: *How Accurate Are Fitness Tracker Heart Rate Monitors?* [cit. 2018-05-05]. Dostupné z: `https://www.livescience.com/56459-fitness-tracker-heart-rate-monitors-accuracy.html`

[11] Geonaute: *Geonaute HR belt*. [cit. 2018-05-05]. Dostupné z: https://customercare.geonaute.com/hc/en-gb/sections/201651632-BluetoothSMART-HRM-belt

[12] Fitbit: *Fitbit Surge*. [cit. 2018-05-05]. Dostupné z: https://www.fitbit.com/us/surge

[13] Maxim Integrated: *MAXREFDES73: Wearable, Galvanic Skin Response System*. [cit. 2018-04-24]. Dostupné z: https://www.maximintegrated.com/en/design/reference-design-center/system-board/6147.html

[14] Moodmetric: *The Moodmetric ring*. [cit. 2018-04-24]. Dostupné z: http://www.moodmetric.com/

[15] Moodmetric: *How to interpret the Moodmetric data*. [cit. 2018-04-24]. Dostupné z: http://www.moodmetric.com/interpret-moodmetric-data/

[16] Empatica: *Empatica Embrace*. [cit. 2018-04-24]. Dostupné z: https://www.empatica.com/embrace/

[17] Empatica: *Empatica E4*. [cit. 2018-04-24]. Dostupné z: https://www.empatica.com/research/e4/

[18] Mozilla: *MVC architecture*. [cit. 2018-04-17]. Dostupné z: https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture

[19] Medium: *Model-View-View Model*. [cit. 2018-05-01]. Dostupné z: https://medium.com/@husayn.hakeem/android-by-example-mvvm-data-binding-introduction-part-1-6a7a5f388bf7

[20] Corona Labs Inc.: *Corona*. [cit. 2018-04-16]. Dostupné z: https://coronalabs.com/

[21] Adobe Systems Inc.: *Adobe PhoneGap*. [cit. 2018-04-16]. Dostupné z: https://phonegap.com/

[22] Microsoft Corporation: *Xamarin*. [cit. 2018-04-16]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/android/get-started/

[23] Tiobe: *TIOBE Index*. [cit. 2018-04-17]. Dostupné z: https://www.tiobe.com/tiobe-index/

[24] Wikipedia: *Java (programming language)*. [cit. 2018-04-17]. Dostupné z: https://en.wikipedia.org/wiki/Java_(programming_language)

[25] JetBrains: *FAQ - Kotlin Programming Language*. [cit. 2018-04-17]. Dostupné z: https://kotlinlang.org/docs/reference/faq.html

[26] JetBrains: *Kotlin Programming Language*. [cit. 2018-04-17]. Dostupné z: http://kotlinlang.org/

[27] Google: *Android NDK*. [cit. 2018-04-16]. Dostupné z: https://developer.android.com/ndk/guides/index.html

[28] Google: *SQLiteOpenHelper*. [cit. 2018-04-17]. Dostupné z: https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html

[29] Google: *Room Persistence Library*. [cit. 2018-04-17]. Dostupné z: https://developer.android.com/topic/libraries/architecture/room.html

[30] Google: *Activity*. [cit. 2018-05-01]. Dostupné z: https://developer.android.com/reference/android/app/Activity

[31] Google: *Fragment*. [cit. 2018-05-01]. Dostupné z: https://developer.android.com/reference/android/app/Fragment

[32] Google: *View*. [cit. 2018-05-01]. Dostupné z: https://developer.android.com/reference/android/view/View

[33] Google: *Motion sensors*. [cit. 2018-05-08]. Dostupné z: https://developer.android.com/guide/topics/sensors/sensors_motion

[34] Google: *Google Play*. [cit. 2018-04-26]. Dostupné z: https://play.google.com/store

[35] Google: *Google Play Console*. [cit. 2018-04-26]. Dostupné z: https://play.google.com/apps/publish

[36] Google: *Android Developers - Sign your app*. [cit. 2018-04-26]. Dostupné z: https://developer.android.com/studio/publish/app-signing

[37] The jQuery Foundation: *jQuery*. [cit. 2018-05-10]. Dostupné z: https://jquery.com/

[38] Pivotal Software: *Spring Framework*. [cit. 2018-05-01]. Dostupné z: https://spring.io/

[39] JUnit: *JUnit - About*. [cit. 2015-04-15]. Dostupné z: http://junit.org/

[40] Twitter: *Twitter Bootstrap*. [cit. 2018-05-01]. Dostupné z: https://getbootstrap.com/

[41] Google: *Google API Client*. [cit. 2018-05-01]. Dostupné z: https://github.com/google/google-api-ruby-client

[42] The PostgreSQL Global Development Group: *PostgreSQL*. [cit. 2018-05-10]. Dostupné z: https://www.postgresql.org/

[43] The Apache Software Foundation: *Apache Tomcat*. [cit. 2018-05-01]. Dostupné z: http://tomcat.apache.org/

[44] Google: *Android testing*. [cit. 2018-05-01]. Dostupné z: https://developer.android.com/studio/test/

[45] Mockito: *Mockito Framework*. [cit. 2018-05-01]. Dostupné z: http://site.mockito.org/

[46] Google: *Espresso*. [cit. 2018-05-01]. Dostupné z: https://developer.android.com/training/testing/espress

[47] ImDB: *The Hills Have Eyes (2006)*. [cit. 2018-04-24]. Dostupné z: https://www.imdb.com/title/tt0454841/

[48] ImDB: *Kill List (2011)*. [cit. 2018-04-24]. Dostupné z: https://www.imdb.com/title/tt1788391/

[49] Wikipedia.org: *Unit testing*. [cit. 2015-04-15]. Dostupné z: http://en.wikipedia.org/wiki/Unit_testing

[50] REST Assured: *REST Assured*. [cit. 2018-05-01]. Dostupné z: http://rest-assured.io/

[51] Fitsport-komplex s.r.o.: *NutriData*. [cit. 2018-05-01]. Dostupné z: https://nutridata.cz

[52] Google: *Google Fit*. [cit. 2018-05-01]. Dostupné z: https://developers.google.com/fit/android/

Příloha **A**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Muller_Jiri_2018.pdf` ..... text práce ve formátu PDF