Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Mobilní aplikace pro sběr každodenních aktivit uživatele pro webovou aplikaci BodyInNumbers

Plzeň 2018 Tomáš Ballák

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 28. června 2018

Tomáš Ballák

Abstract

Within my bachelor thesis I deal with the question of developing a mobile application for needs of the Body In Numbers system. This application is developed within the Department of Computer Science at the Faculty of Applied Sciences at the University of West Bohemia in Pilsen. The main goal is to create a clear and easy to use application with which users will be able to monitor their meals, movement and psychic fitness. Before any further steps one had to become familiar with the Body In Numbers system and with its structure and measurement process. I dealt with the possibilities of developing a mobile application and explored the available applications concerning themselves with healthy lifestyle. I also explored the possibilities of collecting data using wearable electronics. I described the development and functionality of the resulting mobile application in the implementation section. The application is created with the React Native framework and JavaScript. The server part of the application is then developed using SqlAlchemy, Flask and Python.

Abstrakt

V mé bakalářské práci se zabývám vývojem mobilní aplikace pro systém Body In Numbers. Tato aplikace je vyvíjena v rámci Katedry informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské univerzity v Plzni. Hlavním cílem je vytvoření přehledné a snadno ovladatelné aplikace, kterou uživatelé zaznamenávají svoji stravu, pohyb a psychický stav. V práci se nachází seznámení se systémem Body In Numbers, jeho strukturou a procesem měření. Práce se zabývá možnostmi vývoje mobilních aplikací a již vytvořenými aplikacemi pro monitorování životního stylu. Dále se zde nachází průzkum možností sběru dat nositelné elektroniky. V implementační sekci popisuji průběh vývoje aplikace. Aplikace je vystavěna na frameworku React Native s programovacím jazykem JavaScript. Serverová část aplikace je vyvíjena využitím technologií SqlAlchemy, Flask a programovacím jazykem Python.

Obsah

1 Úvod

Sledování fyzických aktivit a stravovacích návyků získává v poslední době na popularitě. Monitorování a následná úprava fyzických aktivit může mít zásadní vliv na život člověka zdravého i nemocného. Sledováním stravovacích návyků a pohybových aktivit si může každý upravit životní styl. Jeho zlepšením se může předejít psychickým i fyzickým chorobám, které dnes doprovázejí spoustu lidí. Tato data, která jsou sledována, mají potenciál být využita pro strojové učení, které by případně identifikovalo zdravotní stav uživatele.

V práci nejprve probíhá seznámení s komplexním systémem Body In Numbers, který zaznamenává a vizualizuje fyziologická data o uživatelích, kteří jsou jeho součástí. Dále jsou zde prozkoumané možnosti sběru dat za využití nositelné elektroniky, jako jsou chytré hodinky a náramky. Po dostatečném průzkumu je nastíněn průběh a výsledky návrhu celého software, kde se probírá, mimo jiné i návrh funkcionality. V této kapitole je také důkladná analýza dostupných prostředků pro vývoj mobilních aplikací a serveru. Potom jsou prozkoumány podobné, již vytvořené aplikace, které jsou zdarma a velmi populární. Porovnává se zde, jestli splňují dané požadavky. V další části je nastíněn podrobnější postup celé implementace. Popis implementace obsahuje výběr technologií, popis databáze nebo využité knihovny. V neposlední řadě se zde nachází popis struktury celého projektu. Po popisu implementace následovalo rozebrání výsledků uživatelských a jednotkových testů.

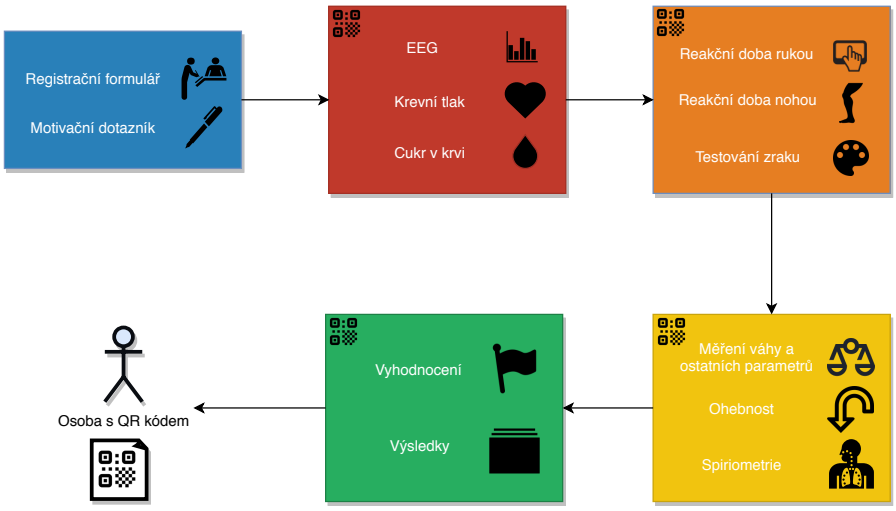
Hlavním cílem této práce je vytvoření mobilní aplikace, která bude umožňovat sledování životního stylu uživatele. Aplikace bude také zpřístupňovat vyplňování dotazníku pro členy Body In Numbers, jehož obsah je zaměřen na psychickou a fyzickou kondici. V rámci Body In Numbers budou v budoucnu tyto data využívána k určení toho, jak pohyb, strava a psychika ovlivňují mozkovou vlnu P300.

Aplikace je vyvíjena zejména pro účely programu Body In Numbers. Bude nabízet i testovací účet, který umožní uživateli aplikaci vyzkoušet. Aplikace by měla splňovat několik kritérií jako jsou: rychlost, přehlednost a rozšiřitelnost.

2 Sběr medicínských dat v aplikaci Body In Numbers

V rámci systému *Body In Numbers* se měří různorodá data člověka, jako například reflexy, tlak, váha, výška atd. Hlavní veličina, která je měřena, je mozková vlna *P300*. Tato vlna je měřena pomocí EEG[1]. Díky měření takto různorodých dat existuje možnost budoucího výzkumu toho, jaký tyto parametry mají vliv na mozkovou aktivitu člověka.

Měření probíhají na samostatných stanovištích. Na každém stanovišti se vkládají data do záznamového archu a mobilní aplikace. Webová stránka Pavla Šnejdara viz [13], která byla jeho diplomovou prací, následně zpracovává a vizualizuje tato data. Proces samotného měření je vidět na obrázku 2.1.



Obrázek 2.1: Diagram stanovišť (zdroj: vlastní tvorba).

[1]EEG Elektroencefalogram

2.1 QR kódy

Každému měřenému subjektu je přidělen jedinečný QR[2] kód, který je generován náhodně v hexadecimální podobě. Je rozšířen o řetězec „[n]", kde n značí pořadí vygenerované kódu. Každé provedené měření je přes tento kód propojeno s měřenými subjekty. Po měření si každý může zkontrolovat své výsledky, právě přes zmíněný QR kód.

2.2 Registrace

Měřený subjekt je registrován pomocí webové aplikace. Zadávají se zde informace jako je věk, pohlaví, dominantní ruka, číslo dotazníku a zda měřený subjekt souhlasí se zpracováním osobních údajů. O posílání dat na server z jednotlivých stanovišť se stará mobilní aplikace Davida Bohmana, která byla jeho bakalářskou prací viz [6].

2.3 Stanoviště

Data z každého stanoviště jsou zadávána do aplikace Davida Bohmana a zároveň vyplňována na papír. Každý měřený subjekt si s sebou nosí *QR kód*, který s nimi prováže naměřená data.

2.3.1 Motivační otazník

Motivační dotazník obsahuje 13 otázek. Jelikož psychický stav nelze změřit, existuje tento dotazník. Otázky jsou cíleny na fyzickou činnost, stravu a osobní život. Takto získaná dat mohou pomoci pro další analýzy.

2.3.2 Měření tlaku a glykémie

Krevní tlak je měřen digitálním tlakoměrem *Omron M6 Comfort IT*, měří diastolický a systolický tlak. Glykémie je měřena glukometrem *FORA Diamond Mini*.

Glykémie

Glykémie je koncentrace cukru v krvi. Zvýšená glykémie se také označuje jako hyperglykémie. Příčinou může být malá fyzická aktivita nebo špatná

[2]QR Quick Response

funkce slinivky břišní. Snížená glykémie neboli hypoglykémie, je způsobena například přehnanou fyzickou aktivitou, nebo vynecháním jídla.

2.3.3 Sběr mozkových aktivit

Mozkové aktivity jsou měřeny EEG ve speciální pro ni určené komoře. Po nasazení EEG čepice následují reakce měřeného subjektu na podněty definované vytvořeným scénářem. Tento scénář obsahuje několik činností: například zapnutí televize, rádia nebo přivolání pomoci.

EEG amplifier wamp

Pro měření je využíván EEG zesilovač. Dodaným softwarem lze získat data v *Brain Vision* formátu. Z nich lze dalším zpracováním získat latence vlny *P300*.

2.3.4 Reakční doba rukou

Měření reakční doby rukou probíhá na základě rychlostí a přesností reakcí měřené osoby. Na desce jsou rozmístěné čtyři LED panely, které se rozsvítí a měřený subjekt klikne na příslušné tlačítko, které je těsně u panelu. Zaznamená se čas uplynulý od stisku (reakční doba), počet zameškání a počet chyb. Viz časopis [9].

2.3.5 Reakční doba nohou

Podobně jako reakční doba rukou se měří i reakční doba nohou. Měřený subjekt stojí na měřící desce a před promítaným obrazem. Deska reaguje na čtyři směry šlápnutí, které jsou označeny šipkami. Měřený subjekt šlápne na příslušnou šipku podle pokynů na obrazovce. Čas uplynulý od stisku se zaznamená.

2.3.6 Pružnost a barvocit

Pružnost je měřena podle toho, jestli je subjekt schopný předklonu. Když se subjekt v předklonu dotkne prstu u nohou, tak je ohodnocen kladnými body, v případě neúspěchu získá body záporné.

Barvocit je měřen pomocí pseudoizochromatických tabulek. V tabulkách jsou obrázky, ve kterých jsou zobrazeny různé tvary, jako například čísla. Lidé s poruchou barvocitu tyto tvary nerozeznají.

2.3.7 Spiriometrie

Spiriometrie je úkon, který se zabývá měřením kapacity plic. Měří se usilovná vitální kapacita FVC[3], usilovný vydechnutý sekundový objem FEV1[4] a vrcholová výdechová rychlost PFE[5].

2.3.8 Výška a váha

Výška subjektu je měřena metrem. Hmotnost se měří chytrou vahou, která je schopna podle elektrického odporu, věku, pohlaví a výšky určit složky jako například tuk, BMI[6] a vodu v těle.

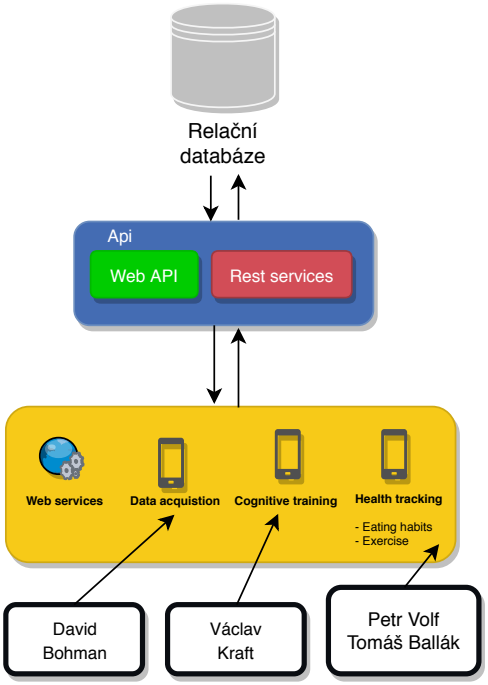
2.3.9 Vyhodnocení

Zpracování naměřených dat probíhá ve webové aplikaci Pavla Šnejdara, která umožňuje přehledné zobrazení těchto dat. Aplikace vizualizuje například reakci končetin nebo také tlak.

2.4 Architektura Body In Numbers

Systém je rozdělen do tří hlavních částí, které se dále dělí. První z nich je *relační databáze* uchovávající naměřená data. Další částí je API[7] rozhraní pro komunikaci s datovým úložištěm a ostatními moduly. Toto rozhraní obsahuje *WEB API* a *REST*[8] *services*. Poslední částí je skupina modulů komunikující přes API s relační databází. Hlavním modulem této části je webová stránka, která zpracovává a vizualizuje naměřená data, viz předchozí kapitola. Dalším modulem je mobilní aplikace pro kognitivní trénink viz [8]. Vizualizace architektury lze vidět na obrázku 2.2.

[3]FVC Forced Vital Capacity
[4]FEV1 Forced Expiratory Volume
[5]PFE Peak Expiratory Flow rate
[6]BMI Body Mass Index
[7]API Application Programming Interface
[8]REST Representational state transfer



Obrázek 2.2: Architektura Body In Numbers (zdroj: vlastní tvorba).

2.5 Vize

Celý systém by měl být v dohledné době upraven tak, aby byla splněna kritéria jako robustnost a zabezpečení. Jelikož Evropská Unie 25. května schválila nový zákon o ochraně osobních údajů nazvaný GDPR[9].

Tento zákon přináší několik pravidel, kterými by se měly řídit firmy nebo společnosti, které zachází s uživatelskými daty. Uživatelé by měli mít možnost svá data smazat nebo je alespoň anonymizovat. Je důležité, aby byli seznámeni s tím, jak se s jejich daty bude pracovat. Další věc, na kterou se musí dbát, je ochrana dat. K datům mají mít přístup jenom pověřené osoby. Práva těchto osob mohou být dána informačním systémem.

[9]GDPR General Data Regulation Protection

3 Možnosti sběru dat

Na trhu s elektronikou existuje mnoho zařízení, které umožňují měřit kroky, tep atd. Bude zde proveden průzkum toho, jaké zařízení bych mohl využít v k této práci. Zařízení budou hodnocena podle následujících kritérií.

- Měření kroků
- Měření tepu
- Získání polohy (GPS[1])
- Cena
- Operační systém
- Možnosti získání dat pomocí API

3.1 Počet kroků

Měřiče kroků využívají nejčastěji akcelerometr nebo vestavěný krokoměr. Pro měření kroků lze využít například chytré hodinky, náramky nebo telefon. Existuje zde i možnost měření pomocí externího krokoměru. Pokud však uživatel vlastní chytrý telefon, je toto zařízení zbytečné, právě díky přítomnosti akcelerometru v mobilním telefonu.

3.1.1 Akcelerometr

Akcelerometr se skládá ze tří hlavních částí. První je volná část (m), která se může volně pohybovat. Tato část je obklopena většinou piezoelektrickým materiálem (pm). K pm a m jsou zapojeny elektrody. Při rozpohybování se m přibližuje k pm a tím se zvyšuje proud v obvodu. Pomocí této hodnoty se určuje zrychlení. Toto je provedeno pro tři osy x, y a z. Zrychlení je zaznamenáno v jednotkách m/s^2. Tato data lze využít pro navigaci v budovách, kde není GPS signál.

Zařízení se systémem Android vrací hodnoty na ose x, y a z viz [4]. Stejně tak pracují i iOS zařízení.

[1] Global Positioning System

3.1.2 Krokoměr

Krokoměr obsahuje kyvadlo, které se při každém kroku vychýlí. Tato výchylka je zařízením detekována a přičtena do paměti.

3.2 Srdeční tep

Měření srdečního tepu může být provedeno přiložením prstu na zápěstí ruky a odpočítávání pomocí hodinek nebo zařízeními, které jsou pro to určené. Možnosti elektronického měření tepu jsou, prosvicování nebo vysílání elektrických pulzů do pokožky.

3.3 Spánek

Rozpoznání spánku může být na základě pohybu uživatele. Při žádné aktivitě systém detekuje spánek. Takto fungují například zařízení *Fitbit*. Některé zařízení dokáží detekovat i kvalitu spánku. Tyto data mohou být užitečná při detekci poruchy spánku, jako je například spánková apnoe.

3.4 Měření polohy

Pro měření polohy se využívá buď GPS, WiFi nebo datové připojení. Nejlepší je však kombinace GPS a WiFi nebo GPS a mobilních dat.

3.4.1 GPS

GPS umožňuje přesnou detekci pozice daného subjektu. Čip komunikuje s vícero satelity a nespotřebovává mobilní data, ale za to spotřebuje hodně energie. Ke zvýšení přesnosti se využívá GSM[2] nebo WiFi signál.

3.5 Zařízení v laboratoři

Neuroinforamtická laboratoř má několik nositelných zařízení. Nachází se zde chytré náramky *Fitbit Flex*, *HR* a chytré hodinky *LG watch 100*.

[2]GSM Global System for Mobile communications

3.5.1 Fitbit

Společnost Fitbit se zabývá zdravým životním stylem. Prodává chytré sportovní náramky a hodinky, které jsou propojeny s Fitbit účtem.

Fitbit Flex

Fitbit Flex je chytrý náramek, který je schopen sledovat základní aktivity, jako jsou kroky, spálené kalorie, nachozenou vzdálenost, čas, kdy je uživatel aktivní, kvalitu a délku spánku. Vzhled náramku je vidět na obrázku 3.1b. Náramek uživateli ukazuje v podobě malých diod, jak pokročil se splněním denního plánu počtu kroků.

Fitbit charge HR

Dokáže navíc oproti *Fitbit Flex* zobrazovat průměrný srdeční tep a počet vystoupaných schodů. Nabízí možnost zobrazování notifikací z telefonu, jako jsou hovory, ale stále zde chybí GPS. Náramek lze vidět na obrázku 3.1a.

(a) Fitbit HR (zdroj: cellucity.co.za).

(b) Fitbit flex (zdroj: img.yugster.com).

Obrázek 3.1: Chytré náramky

API

Jedna z největších výhod zařízení Fitbit, je možnost zdarma využívat API pro získávání dat ze zařízení nebo serveru. Pro získání práv na tato data, musí vývojář nejprve registrovat svou aplikaci. Při registraci se vyplňuje název aplikace, URI[3] mobilní aplikace, podmínky použití, zásady ochrany osobních údajů atd.

Pomocí přístupového tokenu uživatele a jeho id se daná služba autorizuje a získá přístupová práva k jeho datům z Fitbit účtu. Pro získání tokenu se uživatel musí přihlásit do služby Fibit z dané aplikace. Po přihlášení se autorizační token přenese na specifikovanou URI, kterou lze využít k otevření své aplikace.

3.5.2 LG Watch W100

Chytré hodinky LG Watch W100 umí monitorovat jen kroky. Výhodou oproti *Fitbit Flex*, je možnost získávat přehledně notifikace z mobilního telefonu. Velkou výhodou oproti *Fitbit Flex* zařízení je to, že zde běží Android Wear OS. Díky tomu lze využít aplikace, které jsou pro Android Wear OS napsané.

Obrázek 3.2: LG Watch 100 (zdroj: saymandigital.com).

3.6 Zařízení na trhu

Trhu s chytrými hodinkami nejvíce dominuje Samsung a Apple viz [11], statistika je však z posledního kvartálu 2016, poměr se tak mohl patrně změnit.

[3]URI Uniform Resource Identifier

3.6.1 Garmin

Garmin je velký výrobce zařízení jako jsou navigace, kamery, chytré hodinky a náramky. Jako *Fitbit* nabízí možnost získání dat přes API. Bohužel přístup k jejich datům je zpoplatněn.

Forerunner 645 Music

Hodinky mají v sobě GPS a snímač tepu. Umožňují zobrazit notifikace z připojeného telefonu. Cena se pohybuje kolem 10 000 Kč. Hodinky jsou vidět na obrázku 3.3a.

3.6.2 Samsung

Samsung patří mezi největší výrobce nositelné elektroniky. Existuje možnost získání dat pomocí jejich SDK[4]. Je doporučeno, aby vývojář aplikaci před registrací nejprve otestoval dodaným scénářem. Po otestování si může zažádat o schválení, které trvá zhruba dva týdny.

3.6.3 Gear S3 Frontier

V této době, mezi nejznámější hodinky od Samsungu patří model Gear S3 Frontier. Hodinky disponují senzorem pro měření tepu, GPS, vizualizací notifikací telefonu atd. Hodinky jsou propojeny s aplikací SHealth, kterou lze přirovnat k aplikaci Fitbit. Cena se pohybuje kolem 10 000 Kč.

[4]SDK Software Development Kit

(a) Forerunner 645 Music (zdroj: notebookitalia.it)

(b) Gear s3 (zdroj: samsung)

Obrázek 3.3: Chytré hodinky

3.6.4 Apple

Apple má podle statistiky z roku 2016 [11] největší podíl na trhu s chytrými hodinkami. Apple stejně tak jako Samsung a Fitbit poskytuje API přístup k datům o životním stylu. Jejich Health Records API umožňuje získat informace o zdravotním stavu, například o alergiích apod.

Apple Watch series 3

Apple Watch jsou chytré hodinky. Disponují GPS, měřičem tepu a zobrazování notifikací. Hodinky běží na operačním systému watchOS.Jejich vzhled lze vidět na obrázku 3.4. Cena přesahuje 10 000 Kč.

Obrázek 3.4: Apple Watch (zdroj: luminfire)

3.6.5 Vyhodnocení

Při pohledu na tabulku 3.1, lze vidět, že většina dražších chytrých hodinek má podobné funkce. Garmin však zaostává kvůli zpoplatnění API přístupu k datům. Fitbit HR poskytuje velmi dobrý poměr cena a počet funkcí.

Srovnání parametrů hodinek						
Název	Krokoměr	Tep	GPS	Cena [Kč]	OS	API
Fitbit Flex	✓	X	X	700	-	Zdarma
Fitbit HR	✓	✓	X	2 500	-	Zdarma
Forerunner 645	✓	✓	✓	10 000	-	Zpoplatněné
Gear s3	✓	✓	✓	10 000	tizen	Zdarma
Apple Watch	✓	✓	✓	10 000	watchOS	Zdarma

Tabulka 3.1: Srovnání chytrých hodinek

4 Existující aplikace

Existující aplikace často umožňují sledovat mnoho uživatelských dat. Málokterá společnost umožňuje bezplatný a snadný přístup k těmto datům, pomocí jednoduchého API. Požadavky pro aplikaci jsou následující.

- Vkládání a sledování pohybových aktivit
- Ukládání a sledování stravování
- Nastavení váhy a denního příjmu kalorií, sacharidů, tuků bílkovin a vody
- Kontrola přijatých a spálených kalorií
- Vizualizace počtu kroků
- Zobrazení délky spánku

4.1 Samsung Health

SHealth[1] spolu s *Fitbit* jsou jedny z nejkomplexnější fitness aplikací. Uživatel si zde může sledovat svojí aktivitu, běhání, pěší turistiku, stravování atd. Velká výhoda je, že aplikace umožňuje při sledování aktivity zobrazit mapy. Na obrázku 4.1a lze vidět, že aplikace umožňuje sledovat i pitný režim. Uživatelé si mohou přidávat informace o glukóze v krvi a krevním tlaku. Tato data jsou ručně zadávána uživateli do aplikace.

4.2 Fitbit

Aplikace od společnosti Fitbit viz obrázek 4.1b je velmi podobná *SHealth* aplikaci. Uživatel může dostat upozornění na nedostačující pohyb. Existují zde soutěže mezi uživateli a porovnání jejich výsledků. Jsou zde také připravené série cvičení, které je možné využít. Stejně tak jako v aplikaci *SHealth* lze přidávat stravu, aktivity, glukózu v krvi a krevní tlak.

[1] SHealth Samsung Health

4.3 Google Fit

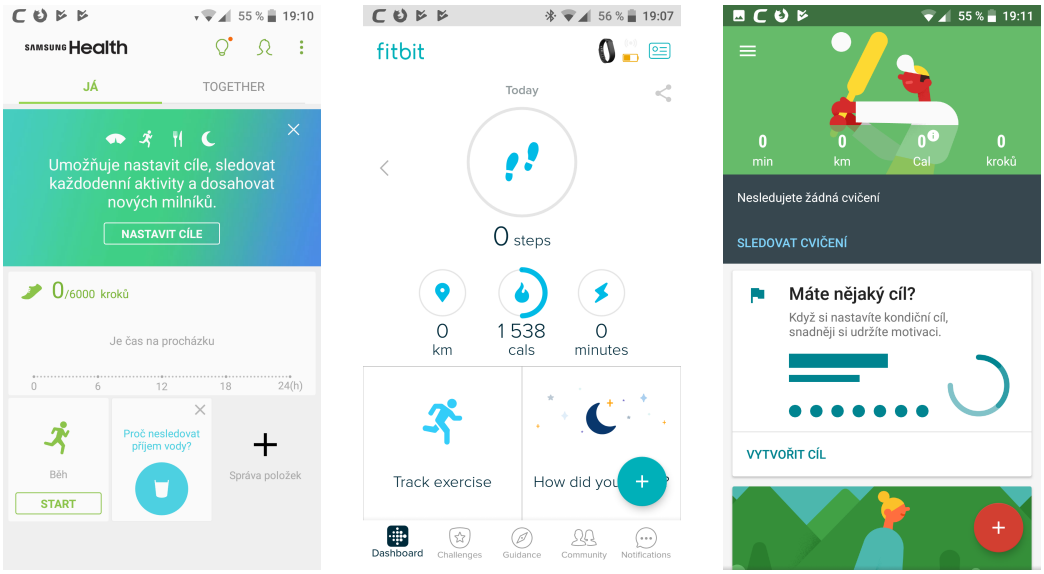
Tato aplikace je strohá, její jedinou funkčností je sledování aktivit viz obrázek 4.1c. Aplikace dokáže na základě pohybu a váhy vypočítat kalorie stejně tak jako aplikace *Fitbit* a *SHealth*. Není tu propojenost s jídelníčkem uživatele a sledování spánku.

4.4 Shrnutí

Výše zmíněné aplikace splňují požadavky, jen *Google Fit* je úzce zaměřen na sportování, ale stravování nebere v potaz. Výsledky porovnávání lze vidět na tabulce 4.1.

Splněné požadavky						
Aplikace	Stravování	Kalorie	Pohyb	Krokoměr	Spánek	Váha
SHealth	✓	✓	✓	✓	✓	✓
Fitbit	✓	✓	✓	✓	✓	✓
Google Fit	X	1/2	✓	✓	X	✓

Tabulka 4.1: Požadavky pro aplikaci



(a) SHealth (b) Fitbit (c) Google Fit

Obrázek 4.1: Aktuální aplikace (zdroj: vlastní)

5 Návrh aplikace

V této kapitole se nachází návrh celého software. Dále je zde průzkum architektury a technologií určených pro vývoj mobilní aplikace a serveru.

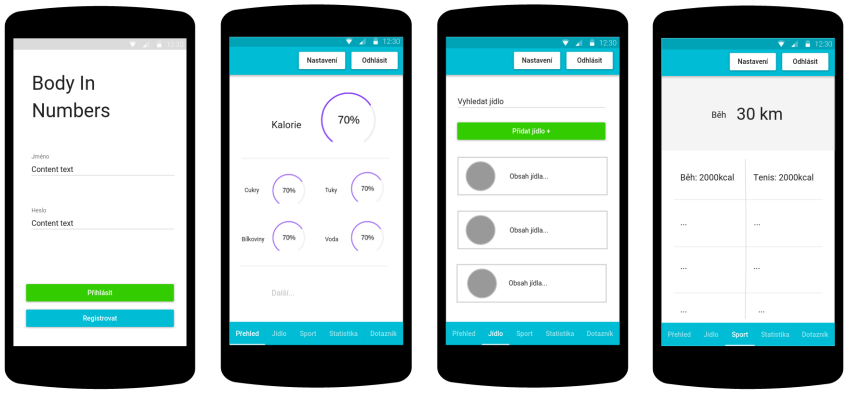
5.1 Funkcionalita

Hlavním cílem je vytvořit příjemnou aplikaci, která se bude snadno ovládat. Musí umět vkládat data uživatele do databáze pro účely sbírání dat v rámci systému *Body In Numbers*. Očekávají se od ní následující funkce:

- Vkládání a sledování pohybových aktivit
- Ukládání a sledování stravování
- Nastavení váhy a denního příjmu kalorií, sacharidů, tuků, bílkovin a vody
- Kontrola přijatých a spálených kalorií
- Vizualizace počtu kroků
- Zobrazení délky spánku

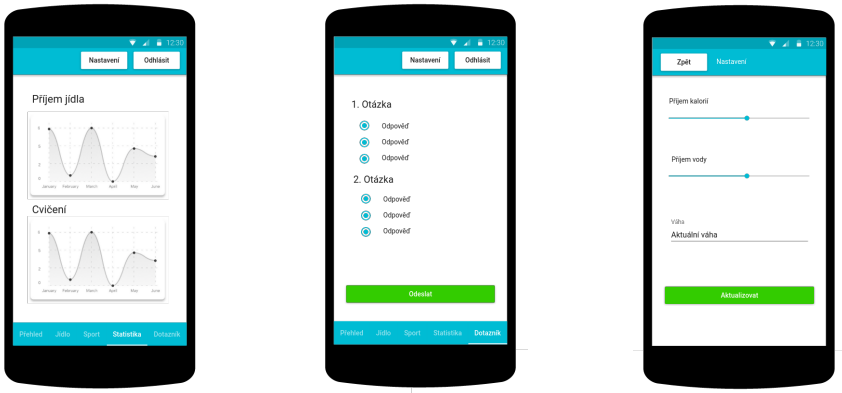
5.1.1 Vzhled

Vzhled aplikace by měl být takový, aby uživatele zaujal a zároveň aby se v ní dobře orientoval. Byla zde navržena základní logika toho, jak by měla aplikace fungovat a vypadat. Na skupině obrázků 5.1 jsou vidět čtyři základní obrazovky na kterých je vidět přihlášení, hlavní přehled, přehled jídla a základní informace o sportovních aktivitách. Další obrazovky obsahují vizualizaci jídla, dotazníku a denního nastavení, která lze vidět na obrázku 5.2.



(a) Přihlášení (b) Úvod (c) Jídlo (d) Sport

Obrázek 5.1: Návrh část první (zdroj: vlastní)



(a) Statistika jídlo (b) Dotazník (c) Denní nastavení

Obrázek 5.2: Návrh část druhá (zdroj: vlastní)

5.2 Technologie pro vývoj mobilních aplikací

Existuje mnoho frameworků pro vývoj mobilních aplikací. Kritéria pro tuto práci jsou rychlost vývoje a vytvořeného produktu. Důležitým požadavkem také je, aby daná technologie byla dlouho dostupná a podporovaná pro případné pozdější přidání funkcí.

5.2.1 React Native

React native je framework vytvořen společností Facebook. Vývoj zde probíhá v jazyku JavaScript. React Native je závislý na tzv. Bridge, který umožňuje volání JavaScriptu přes nativní kód platformy.

Základ React Native je snadná přenositelnost mezi platformami. Je zapotřebí dát si pozor i přes vysokou míru znovupoužitelnosti kódu napříč platformami na výjimky, které je zapotřebí na různých OS řešit individuálně.

Framework nevyužívá k zápisu klasické HTML[1] jako ve frameworku *Ionic*, nýbrž přichází s variací JSX[2], který svojí syntaxí lehce HTML připomíná. JSX umožňuje oproti HTML jednoduše vkládat JavaScriptový kód složenými závorkami viz kód 5.2.

Oproti nativnímu vývoji, přináší možnost sdílení velké části kódu pro obě platformy.

Kód v React Native je složen z jednotlivých komponent. Tyto komponenty obsahují svůj vnitřní stav (state), který lze vidět v ukázce kódu 5.1. Do tohoto stavu se ukládají data, které komponenta využívá. Předávání vnějších dat pro komponentu probíhá přes props[3]. Použití props lze vidět na ukázce kódu 5.2. Vzhled komponenty se definuje v metodě *render*, která zajišťuje její vykreslení.

```
1   constructor(props) {
2     super(props)
3     this.state = {
4       dataUvnitrStavu: [
5         {
6           objekt: 'retezec'
7         },
8         {
9           objekt: 'retezec_2'
10        },
11        ]
12    }
13  }
```

Kód 5.1: Stav komponenty

```
1     <View style={{
2       flexDirection: "row",
3       height: 100,
4       padding: 20,
5     }}>{this.callingFunction()}</View>
```

Kód 5.2: JSX komponenta

[1]HTML HyperText Markup Language
[2]JSX JavaScript Syntax Extension
[3]Props Properties

V React Native mají komponenty definovaný svůj vlastní životní cyklus. Metody těchto komponent jsou postupně volány v závislosti na tom, v jaké fázi životního cyklu se nachází. Nejdůležitější z nich jsou *componentWillRecievеProps*, *componentWillMount*, *componentDidMount*, *componentDidUpdate* a *componentShouldUpdate*.

ComponentWillRecieveProps

Tato metoda je volána pouze v tom případě, když se změní *props* dodávané do komponenty. Metoda se ale nezavolá v případě, když se komponenta vykresluje poprvé.

ComponentWillMount

Metoda ComponentWillMount je volána před prvotním vykreslení komponenty, tudíž po změně stavu nedojde k překreslení.

ComponentDidMount

Metoda je volána v momentě, když je komponenta připojena. Zavoláním *setState* metody způsobí další vykreslování.

ComponentDidUpdate

Tato metoda se zavolá v případě, že jsou aktualizovaná data v komponentě.

ComponentShouldUpdate

Metoda se využívá při určení toho, jestli se má překreslit celá komponenta. Touto metodou lze značně urychlit celý chod aplikace. Lze například porovnat nové a současné props a na základě toho pak rozhodnout, jestli se má komponenta překreslit.

Nativní kód a bridge

Množina komponent React Native je omezena, existuje ale mnoho open source knihoven. Některé komponenty si při implementaci nevystačí jen s poskládáním z existujících React Native komponent. Potřebují z důvodů přístupnosti k HW zařízení nebo výkonu být alespoň z části implementovány v nativním kódu platformy. Ke komunikaci mezi nativním kódem a JS slouží rozhraní nazvané Bridge. Implementace tohoto rozhraní se pro různé platformy liší. U Androidu se třídy u většiny případů dědí od *ReactContextBaseJavaModule*. Pro zavolání metody z JavaScriptu se musí tato metoda

označit anotací *@ReactMethod*. Potom se komponenta registruje v metodě *createNativeModules*.

Pokud se vytváří modul pro iOS je potřeba, aby objekt dědil od *RCTBridgeModule*. Pro přístupnost metod přes JavaScript se musí přidat makro *RCT_EXPORT_METHOD()*.

5.2.2 Ionic

Ionic je framework určený pro multiplatformní vývoj mobilních aplikací. Princip vývoje v Ionicu je takový, že aplikace běží v okně webového prohlížeče, díky tomu se ztrácí značná část výkonu. Výhodou je, že na všech zařízení poběží aplikace stejně, jelikož se nepřekládá do nativních komponent. Odpadá zde problematika ladění pro různé platformy.

Jsou zde různé možnosti jakým frameworkem psát kód aplikace, jako je například *Angular* nebo *VueJs*.

AngularJS

Angularjs je webový framework založený společností Google. Kód v AngularJS je psaný jazykem JavaScript a HTML. Můžou se zde použít libovolné komponenty, které HTML nabízí viz ukázka kódu 5.3. Jelikož je tento framework pod záštitou společnosti Google, je pravděpodobné že bude i nadále podporován.

```
1  <html ng-app>
2    <body>
3      <div>
4        <label>Name:</label>
5        <input type="text" ng-model="yourName"
6        placeholder="Enter a name here">
7        <hr>
8        <h1>Hello {{yourName}}!</h1>
9      </div>
10   </body>
11 </html>
```

Kód 5.3: AngularJS

VueJs

Další možnost psaní aplikace pro *Ionic* je VueJs. Využívá v základu HTML, díky tomu ztrácí možnost vkládání kódu přímo do šablony, která je v *React*

Native. Výkonem se však jako jeden z mála *React Native* přibližuje.

5.2.3 Nativní vývoj

Nativní vývoj je určený na míru dané platformy. Získá se tím možnost plného využití výkonu daného zařízení a mnohem lepší kompatibility, díky psaní kódu pomocí nativních komponent. Jako hlavní nevýhoda tohoto vývoje je nutnost psát minimálně dvě aplikace, jednu pro Android a druhou pro iOS. To zabere hodně času. Pro tuhle volbu by se mělo rozhodnout jen tehdy, pokud je potřeba využít maximální výkon zařízení. Tato situace může nastat například tehdy, když budu vyvíjet hru, která využívá náročné výpočty pro svůj chod.

Android

U platformy Android se vyvíjí aplikace v Android API, které je téměř nerozpoznatelné od Javy. Hlavním rozdílem mezi nimi je, že Java běží v JVM[4] a Android API v DVM[5].

IOS

Aplikace pro iOS jsou v současné době vyvíjeny v jazyce Swift ve verzi 4. Stejně tak jsou ve Swift programovány i desktop aplikace pro MacOS.

5.3 Technologie pro vývoj serverové části

V této části se nachází popis aktuálních technologií, který využívá systém Body In Numbers.

5.3.1 Python

Python je interpretovaný programovací jazyk. Jelikož backend systému Body In Numbers je napsán v Pythonu, konkrétně ve verzi 3.5.2, bude se s ním nadále pracovat. Pro ulehčení vývoje lze využít mini framework *Flask* a pro databázi *SqlAlchemy*.

[4]Java Virtual Machine
[5]Dalvik Virtual Machine

Flask

Flask je framework pro Python, který je určen pro vývoj webových stránek. Umožňuje jednoduché vytváření routs (přístupové body) pomocí anotace @APP.route('someEndpoint'). Framework lze dále využít pro rozšíření webové aplikace, například pro nahrání dat do databáze.

5.3.2 Docker

Části systému *Body In Numbers* jako je například webová aplikace a databáze, využívají Docker. Je to nástroj určený nejen pro vývoj software. Umožňuje zbavení se opakovaných konfigurací projektu na různých zařízení a vyvíjet na stejném prostředí jako běží produkce.

Využívá stejné jádro jako je hostující operační systém a je tak jeho další vrstvou. Docker je složen ze dvou hlavních částí: obrazu a kontejneru.

Obraz

Obraz je zachycení stavu systému. Tvoří základ ze kterého se tvoří kontejner.

Kontejner

Kontejner je spustitelný balíček. Obsahuje v sobě aplikační vrstvu a její závislosti. Jeho hlavní výhodou je, že běží stejně na jakémkoliv systému ať už Windows nebo Linux. Na rozdíl od virtual machine v sobě nemá zabudovaný OS. Využívá jádro, které hostující systém nabízí.

5.3.3 GitLab

Pro přehlednější vývoj jsem využil verzovacího systému Git. Verzovací systém ukládá změny projektu, registruje si je kým, kdy a proč byly provedeny. Další značnou výhodou verzovacího systému je větvení projektu a tím, že se vede verzování s každým nahráním na server se zálohují data.

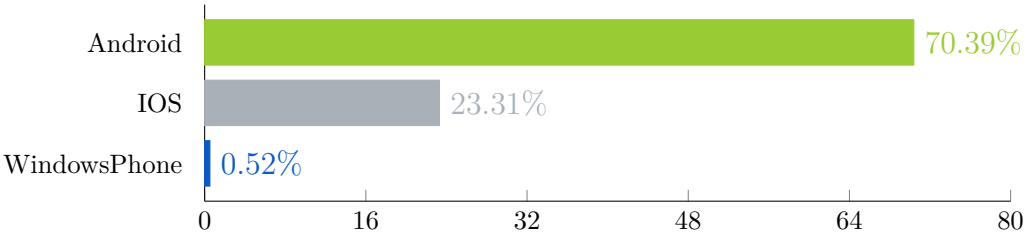
Pro Git repozitář existuje mnoho možností jako jsou GitHub, GitLab a Bitbucket. Pokud srovnám GitHub, Bitbucket a GitLab, tak hlavní výhodou GitLabu je možnost zdarma nastavit projekt na privátní. Kromě této výhody, mají tyto služby téměř stejnou funkcionalitu.

Serverovou část aplikace lze vidět na `https://gitlab.com/sloth-team/exercise-and-wellness`. Zdrojové kódy mobilní aplikace lze najít zde na `https://gitlab.com/neja/fitness-tracker`.

5.3.4 Platforma

Na poli chytrých telefonů a tabletů se dnes vyvíjí zejména pro platformy Android a iOS. Testovací subjekty ve většině případech vlastní zařízení s operačním systémem Android. I přesto, že React Native umožňuje vyvíjet pro více platforem, aplikace bude z časových důvodů fungovat jen pro Android, který pokrývá kolem 70% trhu viz Graf 5.3.

Vývoj je zaměřen na jednu platformu kvůli tomu, že ladění aplikace pro iOS by bylo velmi časově náročné hlavně proto že nevlastním zařízení s MacOS, kde bych mohl vývoj provést. Případný vývoj pro platformu WindowsPhone, kde ji navíc v roce 2017 společnost Microsoft definitivně ukončila, by neměl smysl.



Obrázek 5.3: Přehled operačních systémů na trhu pro rok 2017 (zdroj: [10])

5.4 Databáze

Pro práci s databází existují nejrůznější možnosti. Pro Python je nejznámější *Django ORM* a *SqlAlchemy*. Jelikož je v systému *Body In Nubmers* *SqlAlchemy* zavedeno, budu s ním i nadále pokračovat. Oproti *Django* je jednodušší pro naučení. Výhodou těchto technologií je, možnost přímo vkládat SQL[6] dotazy. V některých případech to může být nevýhoda kdy je požadována přenositelnost mezi různými databázemi.

5.4.1 Jídlo

Databáze by měla obsahovat nejméně tři tabulky pro jídlo. Jedna bude obsahovat referenční hodnoty pro jídla jako například kalorie, sacharidy a tuky. Tato tabulka bude přístupna všem. Další tabulka bude sloužit pro zaznamenání příjmu jídla, které bude pouze pro daného uživatele. Poslední tabulka by měla obsahovat jednotlivé části jídla.

[6]SQL Structured Query Language

5.4.2 Pohyb

Pro pohyb budou existovat stejně jako pro jídlo tři tabulky. Jedna tabulka bude referenční, kde se budou nacházet názvy sportů spolu s konstantami pro výpočet kalorií. Na rozdíl od jídla zde bude tabulka pro sledování pohybu navázána na tabulku s referenčními hodnotami. Pro pohyb bude určena ještě jedna tabulka, která bude uchovávat polohu, rychlost a dobu sledovaného sportu. Těmito daty pak budu moci určit spálené kalorie, nachozenou vzdálenost apod.

5.4.3 Dotazník

Pro dotazník budou existovat čtyři tabulky. Dále zde budou referenční tabulky pro odpovědi a otázky. Z těchto tabulek se budou kopírovat data a provážou se s odpověďmi uživatelů.

5.4.4 Testovací uživatel

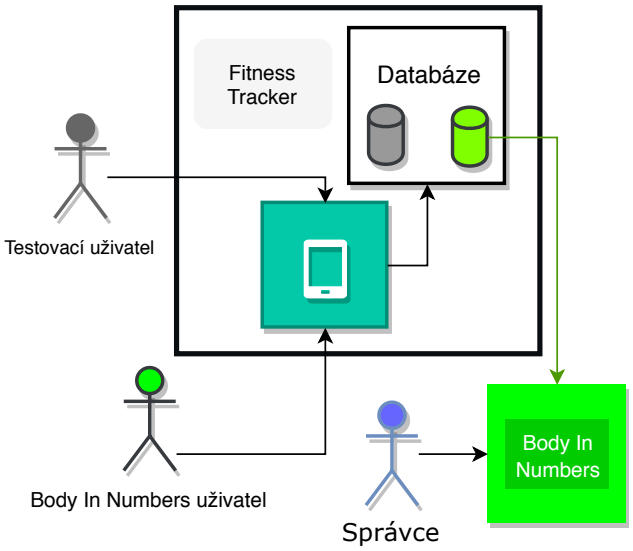
Databáze by měla obsahovat další tabulku s uživatelem, který chce aplikaci jenom vyzkoušet. V této tabulce budou základní údaje jako je pohlaví a váha, dále zde bude uložen token a doba jeho vypršení. Bude zde potřeba uchovávat i přihlašovací jméno a heslo.

5.5 Architektura

Jedna z nejdůležitější věcí u vývoje aplikací je její architektura. Výběr špatné architektury může být v budoucnu značně nepříjemný.

5.5.1 Rozdělení aplikace

Aplikace je primárně určena pro osoby zapojené do programu *Body In Numbers*. Jedním z požadavku bylo i to, že aplikaci budou moci využívat i lidé, kteří nejsou spojeni s tímto systémem, musí se tak vytvořit dva módy aplikace. Jeden pro *Body In Numbers* a jeden pro vyzkoušení aplikace. V testovacím módu, se kvůli ochraně osobních údajů, nebude propojovat daný uživatele pomocí jeho vlastních údajů. Přihlašovací jméno a heslo bude uživateli generováno. viz Obrázek 6.8a.



Obrázek 5.4: Dělení aplikace (zdroj: vlastní)

5.5.2 REST API

REST API je způsob komunikace mezi zařízeními, jejím úkolem je mít přehledný a uniformní způsob komunikace. Základem je určení jakých datových typů bude obsah požadavku a jaká bude odpověď serveru při úspěšné nebo neúspěšné akci. Pro REST API je rozdělena komunikace se serverem do několika typů požadavku viz [12].

POST

Tento typ se využívá, když je požadavek na přidání nového záznamu do databáze.

GET

Požadavek typu GET, se využívá tehdy, když se žádá o data, ale nic se neupravuje v databázi.

PUT

PUT se používá pro aktualizování dat v databázi. Podle [12] se může použít i v případě, když se vytváří data.

DELETE

DELETE je využíván pro mazání záznamů.

Odpovědi

Pro úspěšné požadavky je návratový kód *200* v případě *GET*. U *POST* je úspěšná odpověď *201*. Při neúspěšných požadavcích u *POST* lze získat *404*(Not Found) nebo *409* v případě, že záznam již existuje. U ostatních typů se používá jen *404*.

5.5.3 MVC

Aplikace by měla splňovat parametry *MVC*[7] architektury. Architektura umožňuje relativně snadnou implementaci a udržitelnost.

Model

Model se stará o přímou komunikaci s databází. Provádí výběr, mazání a přidávání záznamů.

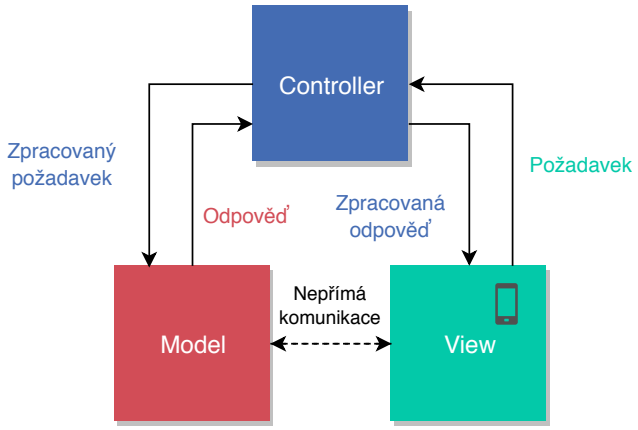
Controller

Controller zpracovává požadavky od *View*. Dále se také stará o zpracování dat, která jsou předána do *View*, v požadovaném formátu, například v JSON.

View

Hlavní funkcí View je zobrazování dat z *Controller*. Jak je vidět na obrázku 5.5, díky *Controller* komunikuje nepřímo *View* s *Modelem*.

[7]MVC Model View Controller



Obrázek 5.5: MVC architektura (zdroj: vlastní)

6 Implementace aplikace

Pro vývoj jsem si vybral framework React Native kvůli jeho výše zmíněným výhodám. Aplikace je vyvinuta primárně pro Android s možným rozšířením. Po úpravách kódu by bylo možné aplikaci zprovoznit i na iOS. V této kapitole popíši podrobněji vývoj celého software.

6.1 Důležité knihovny

K realizaci projektu bylo využito mnoho knihoven, ať už pro vizualizaci dat nebo k realizaci vnitřní komunikace aplikace. Nejdůležitější knihovny zde uvedu.

6.1.1 Redux

Redux je knihovna pro organizaci vnitřních dat aplikace. Funguje na principu úprav stavu, který vlastní data aplikace. Redux je složen ze čtyř hlavních částí: *global state*, *action*, *reducer* a *container*. V aplikaci se využívá pro tyto účely: ukládání dat na offline použití, připojení dat a metod k určitým obrazovkám. Připojení dat k jedné určité obrazovce zde bude popsáno.

Vytvořím si jeden hlavní container, který bude obalovat požadovanou obrazovku. Tomuto containeru přiřadím token, který je obsažen v globání stavu. Token propojím s obrazovkou pomocí metody *connect*. Lze přiřadit do containeru i určité actions, například přihlášení, které bude dodáno jen do přihlašovací obrazovky. Takto se pracuje i s dalšími obrazovky, tímto způsobem obrazovka dostane pouze to co potřebuje.

Architekturu komunikace mezi knihovnou Redux a vnitřní logikou aplikace lze vidět na obrázku 6.1. Další z výhod je to, že jde aktuální global state uložit do paměti a není potřeba při každém vstupu do aplikace opakovat přihlášení a získávat dat ze serveru.

Global state

Global state je stěžejním elementem celé knihovny. Tento stav využívá principu immutabillity, při každé modifikaci se stav nakopíruje a upraví. Tento stav je tak v principu využit jako úložiště pro data, která aplikace využívá.

Actions

Actions jsou definované JavaScriptové objekty, které obsahují identifikátor akce, příklad action je vidět na ukázce kódu 6.1.

```
1    {
2      type: TYPE_OF_ACTION,
3      data: dataOfAction
4    }
```

Kód 6.1: Actions

Reducer

Reducer se stará o samotnou úpravu Global state. Získá příslušnou action a podle jejího typu spustí příslušný Reducer, který dále modifikuje global state. Pomocí property spread notation Reducer stav nakopíruje viz ukázka kódu 6.2 a přidá nová data.

```
1      export default reducer = (state = {}, action) => {
2        switch (action.type) {
3        case ACTION_TYPE:
4          return {
5            ...state,
6            someData: action.data
7          }
8        default:
9          return state
10       }
11     }
```

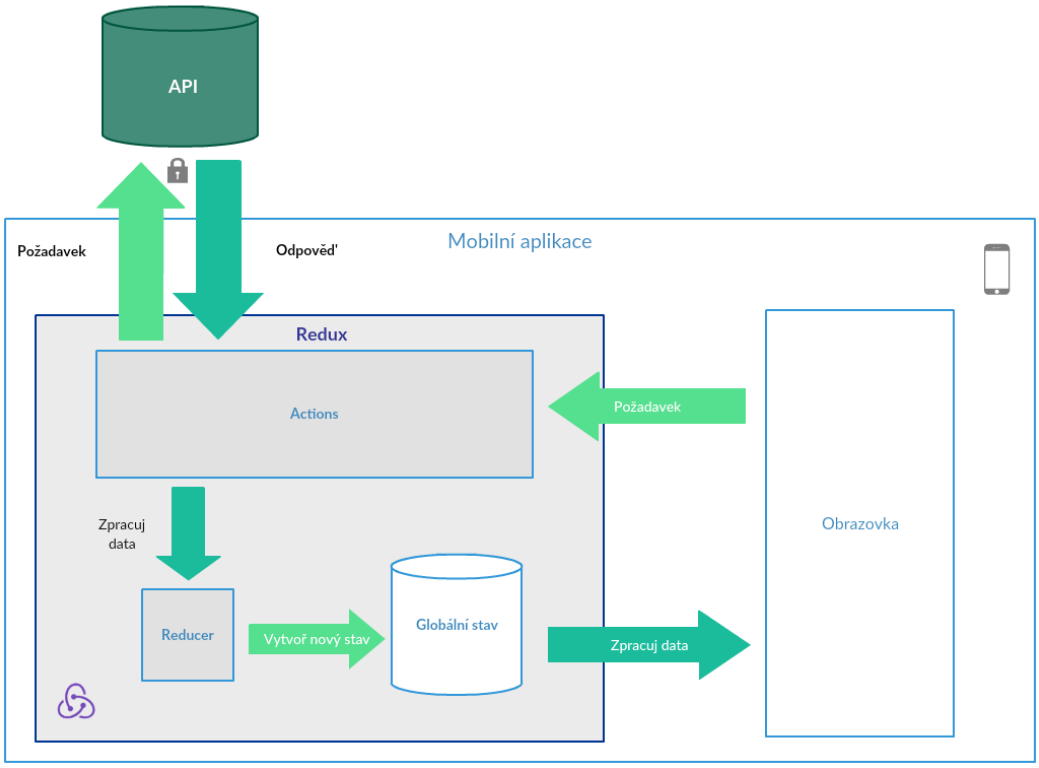
Kód 6.2: Reducer

Container

Container obaluje obrazovku. Dané obrazovce přidá jen ta data a metody, které následně využije. Data jsou získávána ze serveru nebo ze state. Přiřazování se provádí pomocí *mapDispatchToProps* a *mapStateToProps* viz ukázka kódu 6.3. *MapStateToProps*připojuje pouze data. K připojení metod, které modifikují globální stav slouží *mapDispatchToProps*. Veškerá data přiřazená pro daný container, jsou předávaná přes props dané obrazovky.

```
1   const mapDispatchToProps = (dispatch) => ({
2     addQustionnaier: (username, token, questionnaier) =>
          dispatch(questionnaierAddFetch(username, token,
          questionnaier)),
3     startup: () => dispatch(StartupActions.startup())
4   })
5
6   const mapStateToProps = (state) => ({
7     nav: state.nav,
8     questionnaier: state.questionnaier,
9     ...loadDefaultProps(state)
10  })
11
12  export default connect(mapStateToProps,
       mapDispatchToProps)(FormScreen)
```

Kód 6.3: Container



Obrázek 6.1: Architektura Redux komunikace (zdroj: vlastní)

6.1.2 React Navigation

Tato knihovna je využívána pro vytvoření navigace mezi obrazovkami. Obsahuje několik elementárních komponent důležitých pro konstrukci aplikace jako je *StackNavigator* nebo *TabNavigator*. Ukázka kombinace těchto dvou komponent je vidět na kódu 6.4.

TabNavigator

TabNavigator je komponenta, přes kterou probíhá základní navigace mezi jednotlivými obrazovkami. Obsahuje několik záložek, které při kliknutí změní stav aplikace a překreslí dílčí obrazovky.

StackNavigator

StackNavigator je určený pro případ, když chci přejít na určitou obrazovku, která je součástí stejné skupiny. Po přechodu na ní se ukáže tlačítko zpět, které umožní vrátit se na předchozí obrazovku, která je součástí předdefinované skupiny. Ze zásobníku se potom vyjme obsah. Vrátit se zpět lze i tlačítkem operačního systému.

```
1     const Tabs = TabNavigator({
2       Home: {
3         screen: StackNavigator({
4           Home: HomeScreen,
5           Settings: Settings,
6         }),
7       },})
```

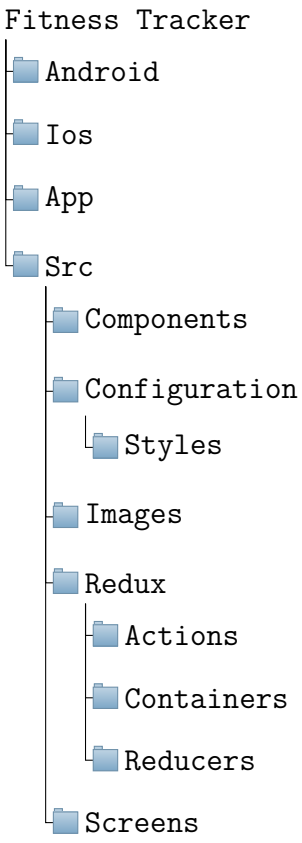
Kód 6.4: Použití React Navigation

6.2 Mobilní aplikace

V této části popíši implementační část mobilní aplikace.

6.2.1 Struktura

Pro vytvoření struktury jsem se inspiroval na [7]. Tu jsem si lehce upravil podle svého uvážení, to je vidět na obrázku 6.2.



Obrázek 6.2: Struktura mobilní aplikace (zdroj: vlastní)

Screens

Obrazovky se nacházejí v adresáři *src/screens*. Tyto komponenty jsou to, co vidí uživatel. Obsahují v sobě různé další moduly ať už mé vlastní, nebo převzaté.

Android a iOS

Zde probíhá modifikace aplikace pro platformy Android a iOS. V této aplikaci je využita jen složka android. IOS je ponechán pro pozdější rozšíření.

Src

V Src adresáři se nachází hlavní zdrojové kódy aplikace.

Components

Components jsou stěžejní součástí aplikace a nachází se ve složce *src/components*. Jejich hlavní vlastností je jejich znovupoužitelnost. Můžu je tedy využít, ve většině obrazovkách. Pokud mám například tlačítko, které využívám na více místech, vytvořím ho jako oddělenou komponentu, nezávislou na obrazovce a mohu ho tak použít na více místech.

Configuration

Tato složka obsahuje konfigurační soubory pro aplikaci. Nachází se zde jak složka se styly, tak i jednotlivé konfigurační soubory. Jsou tu například soubory pro konfiguraci navigace aplikace, obrazovek a konfigurační soubory pro props.

Styles

Tato složka obsahuje předdefinované konstanty barev nesoucí hexadecimální hodnotu a fonty textu, které jsou v aplikaci využívány.

6.3 Server

V této části je explicitně vyhrazený prostor pro aplikaci se jménem Fitness. Vnitřek je dále dělen podle kontextu dat, se kterými se manipuluje. Každý druh dat má vlastní Model a Controller. Uvnitř je kód rozdělen podle architektury MVC, kde View představuje mobilní aplikace.

6.3.1 Model

Model zde nepředstavuje jen přímou komunikaci s databází, ale i vytvoření samotných tabulek. V případě, že tabulka při spuštění serveru neexistuje, vytvoří se podle daného modelu nová viz obrázek 6.5.

```
1     class Food(Base, db.Model):
2     """Table for storing info about food"""
3     __tablename__ = 'food'
4
5     name = db.Column(db.String(256))
6     water = db.Column(db.Float)
7     calorie = db.Column(db.Float)
8     protein = db.Column(db.Float)
9     fat = db.Column(db.Float)
10    carbohydrate = db.Column(db.Float)
```

Kód 6.5: Model

6.3.2 Controller

Controller má za úkol dělat prostředníka mezi model a view. Zajišťuje i odpověď serveru, v tomto případě vytvoří JSON, jenž obsahuje požadovaná data. Na ukázce kódu 6.6 je vidět kontrola parametrů pomocí *parser.add_argument*. Při případném dodání nevhodných dat controller vrátí chybovou hlášku. Controller se stará i o autentizaci uživatele a autorizaci uživatele k určitým informacím.

```
1     class AddExercise(Resource):
2     parser = reqparse.RequestParser()
3     parser.add_argument('client_username', type=str, required
          =True)
4     parser.add_argument('token', type=str, required=True)
5     parser.add_argument('exercise_id', type=str, required=
          True)
6
7     def post(self):
8       data = self.parser.parse_args()
9       client_username = data.get('client_username', '')
10      token = data.get('token', '')
11      userID = authorize(client_username, token)
12      if userID:
13        exercise_id = data.get('exercise_id','')
14        exercise = Exercise.get_by_id(Exercise,exercise_id)
              [0]
15        if len(exercise) == 0:
16          return {'message': 'exercise does not exist'}, 422
17        id = ExerciseTracking.add_tracking(
18          userID,
19          exercise.id,
20          False
21          )
22        return {'exerciseTrackingId': id}, 200
23      else:
24        return {'message': 'authorization failed'}, 401
```

Kód 6.6: Controller

6.3.3 JSON

JSON je textový formát pro předávání dat. Jednou z hlavních výhod je jeho nezávislost na jazyku, který ho zpracovává. Data jsou ukládána vždy jako klíč hodnota viz kód 6.7, kde lze vidět přiřazení objektu pod klíčem *jsonObject*. Jako klíč se považuje vždy řetězec a hodnoty mohou být pole, objekt, číslo, řetězec a boolean. Jelikož JavaScript nativně obsahuje knihovnu pro zpracování Json můžu jí lehce využít viz [5]

```
1  "jsonObject": {
2    "array": [
3    {"data": null},
4    {"data": null}
5    ],
6    "object": {"content":"text"},
7    "booleanExample": false,
8    "number": 1
9  }
```

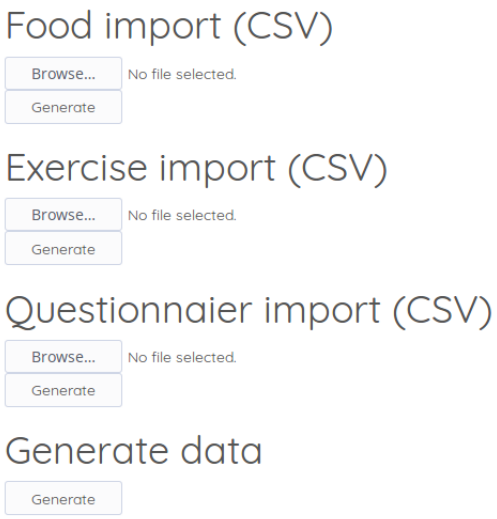
Kód 6.7: JSON struktura

6.4 Webová stránka Body In Numbers

Díky tomu, že má aplikace je úzce propojena se systémem Body In Numbers, musel jsem dodělat modul do webové aplikace.

6.4.1 Nahrání dat

V sekci *nastavení » food* jsem si vytvořil šablonu pro nahrání dat do databáze. Formuláře nacházející se v šabloně přijímají pouze *CSV*[1] soubory. Nahrává se zde jídlo, cvičení a dotazníky viz obrázek 6.3. Dále je tu možnost vygenerování dat pro testovací účet.



Obrázek 6.3: Rozšíření webu (zdroj: vlastní)

[1]CSV - Comma Separated Values

6.4.2 Jídlo

Vytvořil jsem si základní data-set pro jídlo. Vybral jsem pár nejběžnějších potravin a jejich parametry jsem si našel v kalorických tabulkách viz [1], kde tyto údaje vyplňují jednotlivý uživatelé. Jídlo je uloženo tak, že každý sloupec uchovává informaci o daném jídle jako je název, váha sacharidů, tuků, bílkovin a vody vždy na 100g.

6.4.3 Cvičení

Pro cvičení jsem si našel data kde jsem zkoumal u každého sportu při různých časech a hmotnosti jak se mění hodnota spálených kalorií. Tyto hodnoty jsem získal z [3]. S těmito daty jsem vypočítal konstanty, které pomocí toho vzorce 6.4 využívám k výpočtu kalorií. Každý sport disponuje rozdílnou konstantou.

$$spálené\ kalorie = hmotnost * čas * konstanta$$

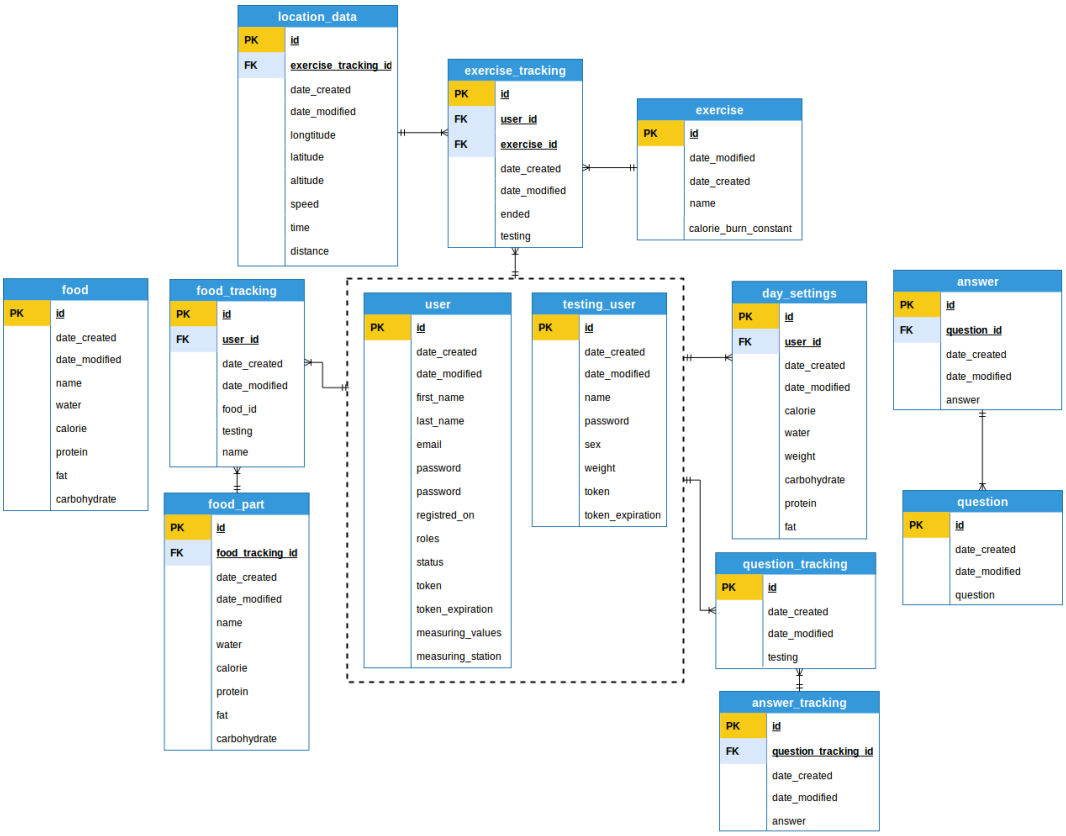
Obrázek 6.4: Výpočet kalorií (zdroj:vlastní)

6.4.4 Dotazník

Dotazník je již vytvořen, jen jsem ho přepsal do CSV pro snadnější importování. Struktura je jiná oproti jídlu, každý sloupec obsahuje otázku a pod ní se nachází odpovědi. Toto rozložení umožňuje snadnější editaci a celý dokument je tak přehlednější.

6.5 Databáze

Databáze je vytvořena technologií *SqlAlchemy*, což je open source technologie, jenž obaluje klasické SQL. *SqlAlchemy* je objektový relační mapovač (ORM), je to způsob jak vytvářet relace mezi objekty v aplikaci a jejich reprezentaci v relační databázi.



Obrázek 6.5: Databázový model (zdroj: vlastní)

Hlavní důvod proč používám tuto technologii, je, že v podstatě rozšiřuji už existující databázi Body In Numbers. Na obrázku 6.5 je vidět, že využívám celkem 13 tabulek z nichž dvě (*testing_user* a *user*) říkají to samé, jelikož se jedná v obou případech o uživatele. Jejich rozdílem je, že u *user* tabulky jsou uživatelé propojeni s Body In Numbers a u *testing_user* se nachází testovací uživatelé (bez vlastního jména nebo emailu), jejich přihlašovací údaje jsou vygenerována.

6.5.1 Základní Tabulka

Každá tabulka v základu má minimálně tři sloupce. Prvním je *date_created*, ten uchovává timestamp, kdy byla data vytvořena. Druhý sloupec je *date_modified* ten uchovává dobu, kdy byla data naposled upravena. A poslední je sloupec nesoucí *id* dané informace. Od této tabulky se dědí všechny sloupce, které získají ostatní tabulky.

6.5.2 Tabulka User

User tabulka kromě základních tří sloupců obsahuje dalších dvanáct navíc. Jedná se o tabulku měřeného uživatele, který je propojen s dílčími tabulkami. Ukládání jména probíhá do dvou sloupců *first_name* a *last_name*. Email se uchovává v kolonce *email*. Hesla jsou hashována a ukládána do sloupce *password*. Zda je uživatel zaregistrován se uchovává v *registred_on*. Role a status uživatelů jsou uloženy v *roles* a *status*. Vygenerovaný token, který je vyžadován při autorizaci je pak uchován ve stejnojmenném sloupci (stejně tak jeho doba zániku). Další jsou tu *measuring_values* a *measuering_station*.

6.5.3 Testing Person

Tato tabulka obsahuje testovacího uživatele, který není zapojen do systému Body In Numbers. Namísto tabulky *User*, která má sloupce *firstname* a *lastname*, Testing Person disponuje pouze sloupcem *name*. Je to primárně z toho důvodu, že jméno je generováno systémem, stejně tak jako heslo. Dále obsahuje navíc sloupec pro ukládání váhy *weight*.

6.5.4 Tabulka Exercise

Tabulka je určena pro ukádání jednotlivých cvičení, které si může uživatel zaznamenávat. Sloupec *name* uchovává název daného cvičení. Další sloupec *calorie_burn_constant* slouží k případnému výpočtu spálených kalorií.

6.5.5 Tabulka Exercise Type

Tabulka obsahuje jednotlivé typy sportů. Uchovává si v podstatě jen jejich název ve sloupci *name*.

6.5.6 Tabulka Exercise Tracking

Tato tabulka obsahuje záznamy provedených sportů. Jsou zde kromě několik základních sloupců, navíc i sloupce *user_id a exercise_id*, které propojují data tabulky *Exercise* a *User* nebo *Testing Person*. Sloupec *ended* nese informaci o tom, zda byla daná sledovaná aktivita ukončena. Díky tomuto tagu se zamezí přidávání dalších pozic do databáze, jelikož se smí přidávat pouze k takovému cvičení, které nebylo ukončeno. Další důležitý sloupec je *testing*, který říká, jestli daný sloupec patří testovacímu uživateli.

6.5.7 Tabulka Food

Food tabulka definuje jednotlivá jídla, která uživatel může využít pro přidání jídla. Uživatel není povinen tyto data využít, slouží spíše pro snadnější přidávání, jelikož uživatel nemusí navíc zadávat každé jídlo zvlášť. Tabulka má navíc 6 sloupců. Jedním z nich je název a ostatní slouží pro podrobnější popis, jako jsou kalorie nebo tuky.

6.5.8 Tabulka Food Tracking

Tato tabulka obsahuje data o snědeném jídle uživatele. Nachází se zde sloupec *photo_id*, který obsahuje id přidané fotografie, díky tomuto id je možnost k dané fotce přistoupit a propojit ji s příslušným jídlem. Dále je tu *name* sloupec, který určuje celkový název jídla. Pokud jsem snědl například zeleninový salát, tak tento sloupec bude obsahovat právě *ovocný salát*, ne výpis jednotlivých složek. Dále je tu sloupec *user_id*, pomocí kterého se propojuje s tabulkou uživatele, je to cizí klíč. Důležitým sloupcem je *testing*, který říká, jestli je propojený účet testovací.

6.5.9 Tabulka Food Part

V této tabulce se nachází dílčí části jídla. Pokud tedy mám výše zmíněný ovocný salát, pak zde se budou nacházet jednotlivé složky jako například jablko, pomeranč apod. Tato tabulka obsahuje kromě cizího klíče tabulky *Food Tracking* i název a ostatní podrobnější data o dané části jídla jako jsou kalorie, cukry, tuky, bílkoviny a voda.

6.5.10 Tabulka Day Settings

Tabulka v sobě uchovává denní nastavení cílů uživatele týkající se potravy. Jsou zde sloupce *calorie*, *water*, *carbohydrate*, *fat* and *protein*, které nesou informaci o nastavení denního příjmu těchto složek. Jsou zde i další informace v podobě váhy uživatele, kterou uchovává sloupec *weight*.

6.5.11 Tabulka Question

Question tabulka v sobě ukrývá jednotlivé otázky, které dohromady spolu s odpověďmi tvoří požadovaný dotazník. Tabulka obsahuje kromě základních sloupců také samotnou otázku, která je ve sloupci *question*.

6.5.12 Tabulka Answer

Odpovědi na otázky jsou uloženy v tabulce Answer. V tabulce se otázky propojí přes *question_id*, pak lze pomocí toho id získat odpovědi. Jednotlivé odpovědi jsou uložené ve sloupci *answer*.

6.5.13 Tabulka Question Tracking

Tato tabulka si udržuje otázky, na které uživatel odpověděl. Otázky jsou nakopírované z tabulky *Question*. Nejsou přímo navázány na id otázky kvůli tomu, kdyby se některé modifikovaly nebo smazaly. Všechny otázky jsou uloženy ve sloupci *question*.

6.5.14 Tabulka Answers Tracking

Answers Tracking si udržuje odpovědi uživatel. Jsou zde nakopírované odpovědi z tabulky *Answers*, podobně jako u *tabulky Question Tracking*. Odpovědi jsou uloženy ve sloupci *answers*.

6.6 Vývojářské nástroje

Pro vývoj aplikace je využito nejrůznějších nástrojů, které umožňují jednodušší, přehlednější a rychlejší vývoj. Pro zálohu a verzování vývoje jak serveru tak i mobilní aplikace jsem využil *GitLab*. Používám i nástroje: *Eslint* a *Yarn*.

6.6.1 Eslint

Eslint je nástroj, který kontroluje jakým způsobem člověk píše kód. Hlídá jeho ucelený vzhled a přehlednost. Dává možnost pár kliknutími zformátovat celý zdrojový kód a držet si jistou konvenci při jeho psaní. Využívám ho jen pro kód mobilní aplikace.

6.6.2 Yarn

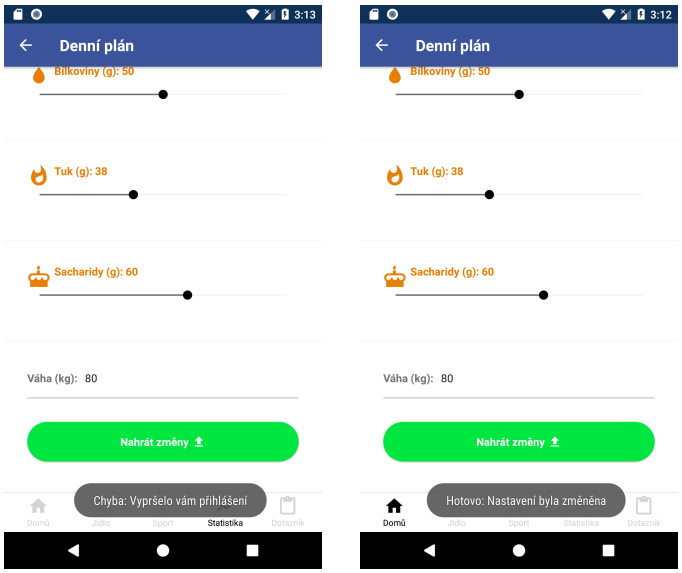
Yarn je balíčkovací systém pro Node (JavaScript platformu). Je to alternativa k velmi rozšířenému balíčkovacímu systému *NPM*. Oproti němu umožňuje paralelní instalaci a tím výrazně urychlit práci. Všechny balíčky, které lze nainstalovat pomocí *NPM*, lze nainstalovat i přes Yarn. Před každou instalací provádí kontrolní součet aby, zajistil kompletnost stahovaného balíku viz [2].

6.7 Funkcionalita

Aplikace má patnáct obrazovek. Dokáže pracovat i offline, ale jen v režimu čtení, kde nelze provádět úpravy. Pokud má telefon přístup k internetovému připojení, tak se při každém otevření obsah aplikace aktualizuje. V této sekci se zabývám podrobným popisem funkčností jednotlivých obrazovek.

6.7.1 Vizualizace hlášek aplikace

Při různých událostech, kdy je potřeba komunikovat se serverem, musí být zpětná vazba, jak daný požadavek dopadl. Ke zpětné vazbě je využito vyskakovacích zpráv *Toast*. Odpovědi jsou rozděleny na dvě skupiny: chyba (neúspěšný požadavek) a hotovo (úspěšný požadavek). Tyto dvě zprávy lze vidět na obrázcích 6.6a a 6.6b, jejich rozdílem je nadpis a obsah.

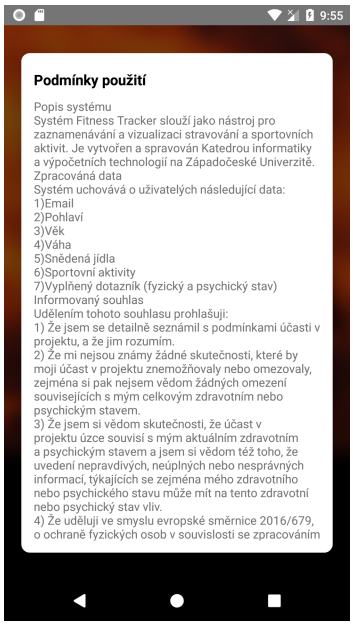


(a) Chybová hláška (b) Úspěšná hláška

Obrázek 6.6: Hlášení aplikace (zdroj: vlastní)

6.7.2 Podmínky pro použití aplikace

Kvůli GDPR a případné právní ochraně jsou v aplikaci uvedeny podmínky pro použití uživatelských dat. V aplikace je napsáno, že uživatel dává svá data dobrovolně a pro účely dalšího zpracování systémem Body In Numbers. Vzhled obrazovky lze vidět na obrázku 6.7.



Obrázek 6.7: Podmínky použití (zdroj: vlastní)

6.7.3 Výběr módu aplikace

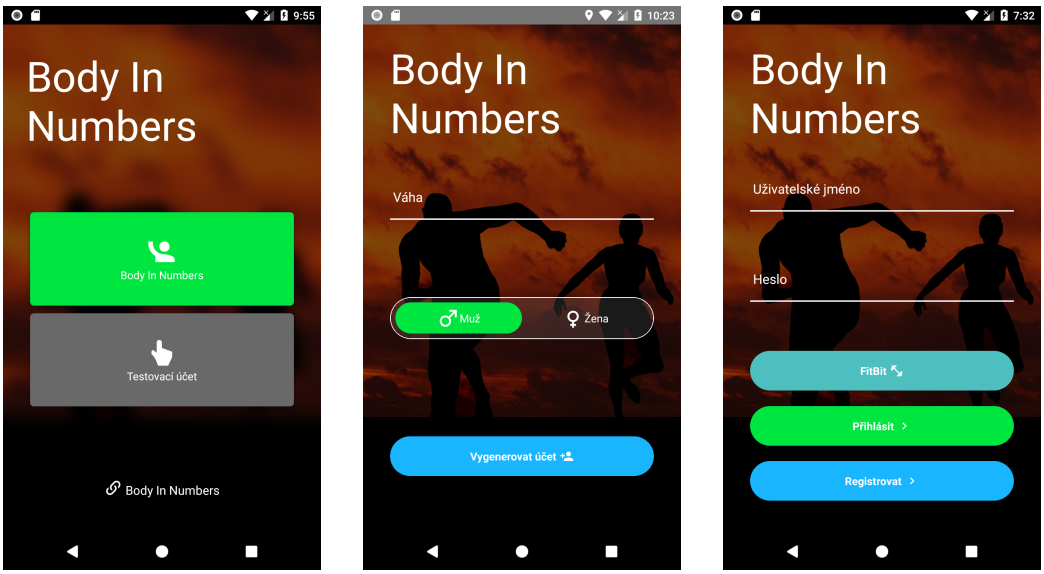
Na této obrazovce uživatel vybírá do jaké oblasti aplikace chce vejít. Může přejít do testovacího režimu nebo režimu Body In Numbers. Po přepnutí aplikace do Body In Numbers se uživatel může přihlásit pouze právě do tohoto systému, stejně tak jako v testovacím režimu. Rozdělení aplikace je vidět na obrázku 6.8a. Po prvním přihlášení, aplikace uchová údaje jako token a přihlašovací jméno, díky tomu se uživatel nemusí znovu přihlašovat, pokaždé když otevře aplikaci.

6.7.4 Registrace

Registrace probíhá pouze pro uživatele v testovacím režimu, jelikož registraci do systému Body In Numbers má na starosti webová aplikace Pavla Šnejdara. Na obrázku 6.8b je vidět jedno pole pro zadávání váhy. Další je tu skupina dvou tlačítek, která obsahují hodnotu pohlaví registrovaného uživatele. Po úspěšné registrace je uživatel přesměrován na obrazovku s přihlášením, kam se pošlou zadané údaje pro registraci. Přihlašovací údaje si musí uživatel zapamatovat. Heslo nejde zpětně získat, jelikož je uloženo jako hash.

6.7.5 Testovací přihlášení

Přihlášení testovacího uživatel probíhá podobně jako klasické přihlášení. Důležitý rozdíl je v tom, že se přistupuje do druhé tabulky *testing_person*, která je oddělena od tabulky *user*. Jako u klasického přihlašování, server generuje token a ukládá jej do databáze. Vzhled přihlášení je vidět na obrázku 6.8c. Stejně jako u uživatele Body In Numbers, i zde lze přidat Fitbit účet a případně si kontrolovat svůj spánek a počet kroků nachozených za den.



(a) Módy aplikace (b) Registrace (c) Testovací přihlášení

Obrázek 6.8: Úvodní obrazovky (zdroj: vlastní)

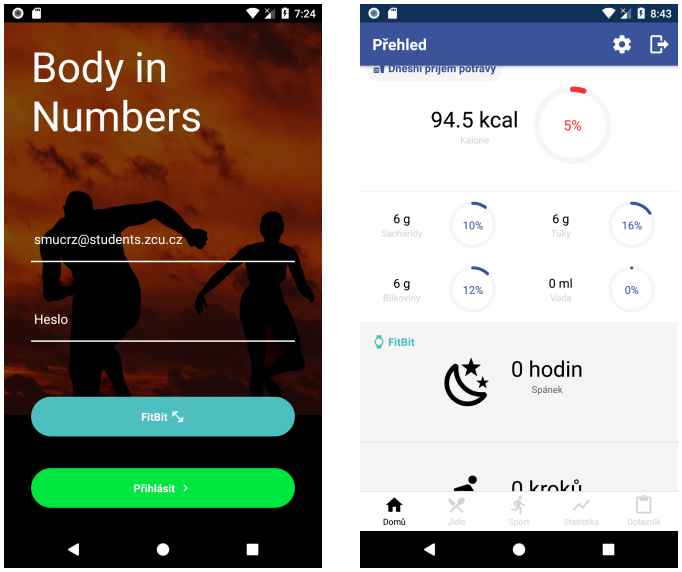
6.7.6 Přihlášení

Přihlášení do aplikace probíhá na základě emailu a uživatelského hesla. Při prvním spuštění aplikace se objeví přihlašovací obrazovka, kam uživatel zadá své údaje. O průběhu ověření údajů se uživatel dozví hlášením aplikace viz obrázek 6.6b. Vzhled přihlašovací obrazovky lze vidět na obrázku 6.9a. Dále aplikace umožňuje připojit uživateli Fitbit účet, díky kterému se potom zobrazí výše zmiňované informace.

6.7.7 Přehled

Tato část se stará o zobrazování přehledu, stravování, pohybu uživatele a jeho spánku. Přehled se zobrazí ihned po přihlášení, uživatel má tak možnost ihned zkontrolovat, jak se například v průběhu týdne zvyšuje jeho pohybová aktivita nebo příjem složek potravy pro aktuální den. Uživatel zde

také může vidět kolik hodin daný den spal a kolik ušel kroků, tyto informace získá za předpokladu, že provedl přihlášení do Fitbit účtu přes aplikaci. Vše je přehledně zobrazeno za pomocí grafů. Vzhled obrazovky je vidět na obrázku 6.9b.

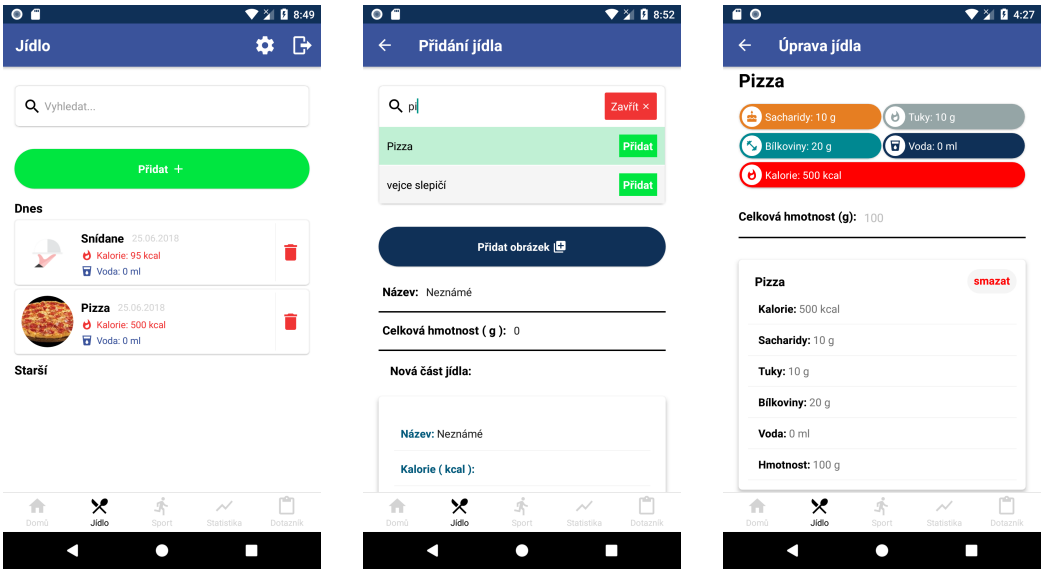


(a) Přihlášení (b) Přehled

Obrázek 6.9: Základní obrazovky (zdroj: vlastní)

6.7.8 Jídlo

V této sekci uživatel vidí svůj příjem jídla, jak dnešní tak i starší. Je zde možnost vyhledávání, přidání a úpravy jídla, které uživatel snědl. Po stisku tlačítka „přidat +" lze vložit snědené jídlo do seznamu přes obrazovku přidání jídla, kterou lze vidět na obrázku 6.10b. Na této obrazovce lze vyhledat již přidané jídlo, které se načte se všemi jeho částmi. Pro editaci jídla, které bylo přidáno uživatelem, stačí stisknutí příslušné položku a potom se otevře editační okno viz obrázek 6.10c. U editace jídla lze provést pouze smazání jednotlivých položek. Do editační části se lze dostat i pomocí vyhledávače, na obrazovce 6.10a. Po přechodu na nalezené jídlo se zobrazí stejné editační možnosti.

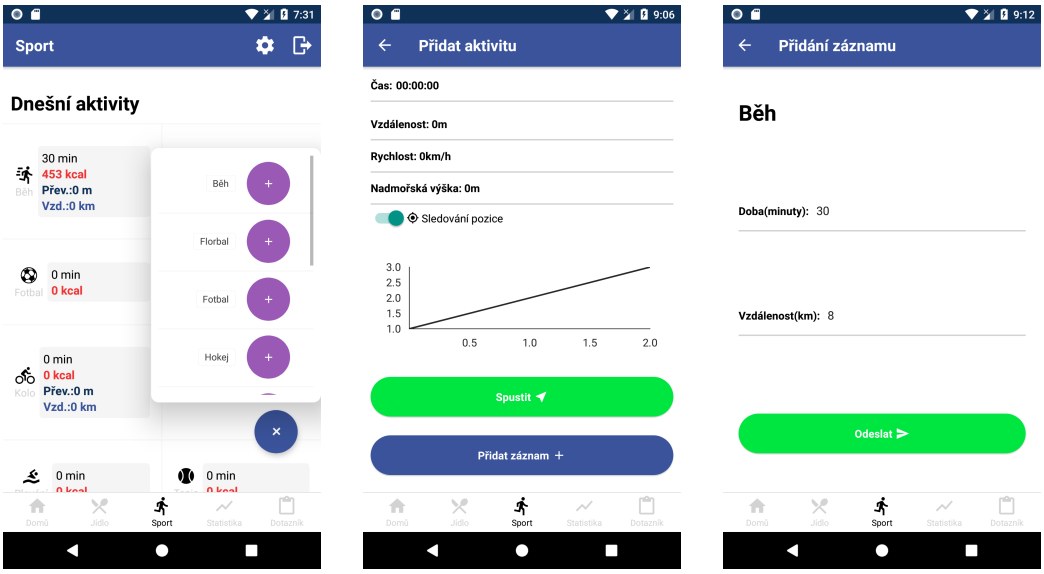


(a) Přehled jídla (b) Přidání jídla (c) Úprava jídla

Obrázek 6.10: Obrazovky jídla (zdroj: vlastní)

6.7.9 Sportovní aktivity

O přehled nad pohybovými aktivitami se stará obrazovka Sport. Lze zde vidět jak dlouho uživatel strávil času nad daným sportem a kolik při tom spálil kalorií. V pravé spodní části je tlačítko, které ukáže nabídku sportů, kde si lze sledovat aktivitu nebo přidat starý záznam. Tento přehled lze vidět na obrázku 6.11a. Sledování aktivity uživatele poskytuje zaznamenání rychlosti, doby a případně převýšení. Sledování sportu se spustí ihned po stisku tlačítka „Spustit" viz obrázek 6.11b. Aplikace po spuštění sledování sportu pravidelně posílá novou pozici na server. Pokud uživatel zapomněl nebo pouze nechtěl, sledovat aktivitu, lze ji nejpozději v ten den přidat manuálně viz obrázek 6.11c. Po přidání sportu aplikace aktualizuje data související s jeho sledováním.

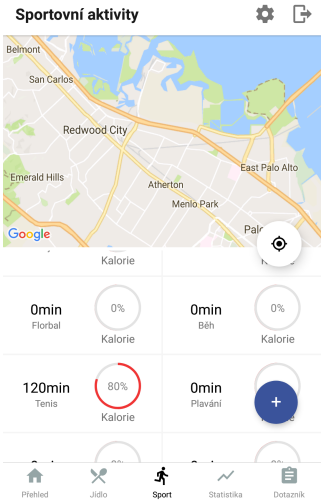


(a) Přehled aktivit (b) Přidání aktivity (c) Přidání záznamu

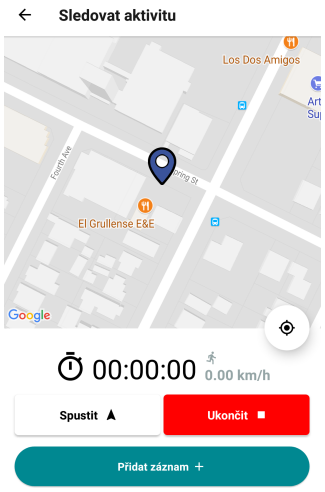
Obrázek 6.11: Sportovní obrazovky (zdroj: vlastní)

Zobrazování map

V předchozí verzi byla možnost zobrazování aktuálních sportovních aktivit na mapě. Bylo to umožněno díky knihovně, která využívala Google Maps. Bohužel v průběhu vytváření aplikace Google změnil své podmínky využití map. Rozhodl, že pokud chce vývojář využít mapy, musí mu dát k dispozici bankovní účet, pro případné strhnutí peněz, kdyby se aplikace stala produkční verzí. Google při vývoji hradí veškeré poplatky za vývojáře. Při přestoupení na produkční verzi by si účtovali měsíčně několik tisíc za jejich služby. Vzhled lze vidět na sadě obrázku 6.12



(a) Přehled aktivit



(b) Přidání aktivity

Obrázek 6.12: Staré sportovní obrazovky (zdroj: vlastní)

6.7.10 Vizualizace dat

Aplikace si vede podrobné informace o aktivitách uživatele. Tyto data se vizualizují na této obrazovce, jsou k nim přidány jednotky a popisky. Dále jsou tu data o tom jak uživatel sportoval, jaké přijímal dané složky potravy jako například sacharidy, tuky, bílkoviny, kalorie a množství vody, po dobu jednoho měsíce. Vizualizace je rozdělena podle druhů informací, pokud jde o příjem jídla nebo o pohyb. Vzhled obrazovky lze vidět na obrázku 6.13a.

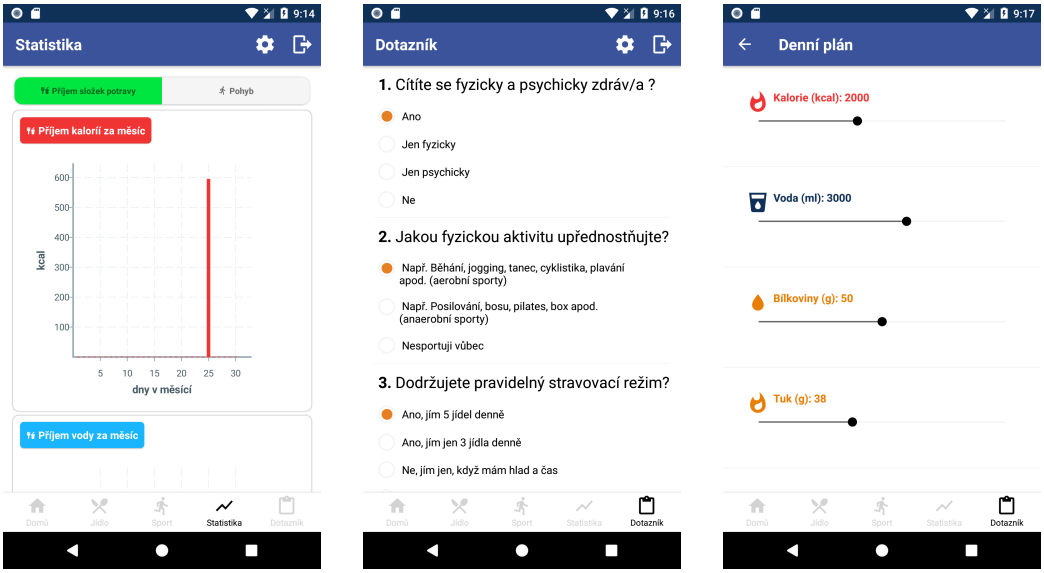
6.7.11 Dotazníková část

Uživatel bude každý měsíc vyplňovat jednoduchý dotazník. Obrazovka se stará o zobrazování dotazníku. Otázky se týkají aktuálního zdravý, stravovacích návyků, psychického rozpoložení atd. Data z dotazníku jsou důležitá kvůli získávání co nejvíce komplexních dat. Vyplňování je přístupné jen těm uživatelům, kteří jsou přihlášení přes účet Body In Numbers. Vzhled dotazníku lze vidět na obrázku 6.13b.

6.7.12 Denní plán

V části denní plán si uživatel nastavuje, kolik jednotlivých složek bude uživatel chtít přijmout. Nastavuje se zde pitný režim a příjem kalorií, sacharidů, proteinů a tuků. Zároveň se bude vždy při této akci aktualizovat váha o hodnotu uloženou ve vstupním poli viz obrázek 6.13c. Tyto změny ovlivňují

vzhled domovské obrazovky, jelikož se z těchto dat procentuálně vypočítává splnění denního příjmu složek potravy.



(a) Vizualizace dat (b) Dotazník (c) Denní plán

Obrázek 6.13: Další obrazovky (zdroj: vlastní)

7 Testování

Pro testování byly zvoleny dva způsoby, jeden z pohledu uživatele, kde byla třem lidem předána aplikace, aby vyzkoušeli její funkcionalitu. V další části bylo provedeno Jednotkové testovaní na serverové části, kde se kontrolovalo přidávání, získávání a mazání dat z tabulek.

7.1 Uživatelské testování

Aplikaci byla předána třem nezávislým lidem a studentům *ZČU*. Každý spolu s aplikací dostal testovací protokol, podle kterého se při testování měli řídit. Po otestování uživatelé vyplnili formulář, který jim byl předložen.

7.1.1 Testovací protokol

Protokol obsahuje sedm kroků k otestování aplikace. Ke každému bodu tester zaznamená výsledek.

- Registrace a přihlášení (Testovací Účet)
- Přidání jídla
- Smazání jídla
- Změna denního nastavení
- Zaznamenání sportu
- Sledování sportu
- Grafy (zkontrolovat po přidání hodnot)
- Připomínky

7.1.2 Tester 1

První tester aplikaci otestoval velmi důkladně. Aplikaci vyzkoušel na zařízení *CUBOT NOTE S* disponujícím Androidem ve verzi *5.1*. Přišel na pár chyb, kde při zkoušení sledování sportu, se neprovedlo uložení v případě vypnuté GPS. Dále měl připomínky k tlačítku „Přidat záznam". U přidání jídla mu přišlo matoucí tlačítko „Přidat", které je blízko tlačítka „Odeslat", nevěděl, že slouží k přidání položky jídla.

7.1.3 Tester 2

Druhý testovací subjekt , zkoušel aplikaci na zařízení *Xiaomi Redmi 4X* s verzí Androidu *7.0 Nougat*. Nenalezl žádné chyby kromě již zmíněné chyby prvním testovacím subjektem.

7.1.4 Tester 3

Poslední tester aplikaci testoval na telefonu *Xiaomi Redmi 3* s Androidem ve verzi *5.1 Lollipop*. Nenarazil na závažnější problém, jen se ukazovalo hlášení chyby i přesto, že vše proběhlo v pořádku.

7.1.5 Vyhodnocení

Po replikaci chyb nalezených uživateli byla aplikace opravena.

7.2 Jednotkové testy

U jednotkových testů byla testována validita výstupu jednotlivých metod pro získávání, přidávání a úpravu dat. Testování využívalo knihovnu unittest, která vychází z knihovny JUnit. Ukázku toho, jak testy vypadají lze vidět na kódu 7.1. Testováním prošla například registrace a přihlášení uživatele. To proběhlo tak, že se vytvořila nová testovací osoba. Metoda pro vytvoření osoby vrací její přihlašovací údaje, ty byly využity k přihlášení. Následně byl otestován token získaný z přihlášení.

7.2.1 Výsledky

Jak lze vidět z tabulky 7.1, serverová část aplikace funguje. Data se uchovávají tak, jak se předpokládalo, a bez problémů probíhá i získávání dat.

```
1   class TestTestingUser(unittest.TestCase):
2     def test_add_user(self):
3         new_user = TestingUser.add(TestingUser,'woman',80)
4         self.assertNotEqual(new_user, None)
```

Kód 7.1: Jednotkový test

Výsledky testování	
Název	**Výsledek**
Registrace uživatele	✓
Přihlášení uživatele	✓
Změna denního nastavení	✓
Přidání jídla	✓
Smazání jídla	✓
Vyhledání jídla načteného ze souboru	✓
Vyhledání jídla přidaného uživatelem	✓
Přidání sportu a pozice	✓
Získání denních informací	✓

Tabulka 7.1: Požadavky pro aplikaci

8 Závěr

V rámci mé bakalářské práce byly prozkoumány možnosti vývoje mobilních aplikací. Na základě průzkumu se rozhodlo, že aplikace bude vyvíjena primárně pro Android. Dále jsou zde popsány možné technologie pro vývoj, a u některých je vysvětleno jejich použití a vnitřní fungování. Byly zde zmíněny důležité knihovny, které se použily při vývoji mobilní aplikace a serverové části. Dále jsou zde popsány nástroje, které byly využity při tvorbě aplikace.

Implementační část zahrnovala výběr technologií, které se ukázaly jako nejvhodnější. Je zde popsána implementace databáze a účel jednotlivých tabulek. V další části je nastíněna funkcionalita jednotlivých ovládacích prvků uživatelského rozhraní. K této částí jsou přiloženy screenshoty jednotlivých obrazovek. Práce obsahuje popis architektury a struktury adresáře serveru. Dále obsahuje popis zpráv vyměňovaných mezi mobilní aplikací a serverem.

Aplikace splnila všechny požadované kritéria, která byla uvedena v zadání. Je schopna uživateli poskytnout dostatečný komfort pro zadávání jeho aktivit a stravovacích návyků. Při implementaci vznikl problém se společností Google, která změnila svojí politiku pro přístup k jejich mapám (Google Maps). V aplikaci tak musela být provedena náročná úprava, aby bylo možné nahradit vizualizaci map. Zrušila se vizualizace sportovních aktivit pomocí map, tímto bylo ztraceno spoustu času.

Backend aplikace na serveru byl otestován pomocí unit testů. Mobilní část prošla validací funkčnosti několika nezávislými uživateli. Po zjištění menších chyb, které jsem našel já nebo testeři, bylo vše opraveno. Testování proběhlo i na starších verzích Android jako je *Lollipop*, ale i na novější *8.0 Oreo*. Aplikace se testovala i pro různé velikosti displeje, tam se neukázali žádné nesrovnalosti.

Jako možné rozšíření aplikace do budoucna lze implementovat sledování a zobrazování aktivit pomocí map. Aplikace by se dala s několika úpravami vydat i pro platformu iOS.

A Literatura

[1] *Kalorické Tabulky* [online]. Kalorické Tabulky, 2018. [cit. duben 2018]. Dostupné z: `https://www.kaloricketabulky.cz/`.

[2] *Yarn* [online]. Yarn, 2018. [cit. duben 2018]. Dostupné z: `https://yarnpkg.com/en/`.

[3] *Calories Burned From Exercise* [online]. My fitness pal, 2018. [cit. duben 2018]. Dostupné z: `https://api.myfitnesspal.com/exercise/lookup`.

[4] Android. *Motion Sensors* [online]. Android, 2018. [cit. duben 2018]. Dostupné z: `https://developer.android.com/guide/topics/sensors/sensors_motion.html`.

[5] Atif Aziz, S. M. *An Introduction to JavaScript Object Notation JSON* [online]. microsoft, 2007. [cit. duben 2018]. Dostupné z: `https://msdn.microsoft.com/en-us/library/bb299886.aspx`.

[6] Bohman, D. Mobilní aplikace, pro sběr medicínských dat. Master's thesis, Západočeská univerzita v Plzni, 2017.

[7] Carli, S. *Organizing a React Native Project* [online]. Medium, 2016. [cit. duben 2018]. Dostupné z: `https://medium.com/the-react-native-log/organizing-a-react-native-project-9514dfadaa0`.

[8] Kraft, V. Systém pro distribuci a správu kognitivních her. Master's thesis, Západočeská univerzita v Plzni, 2017.

[9] Mouček, R. Software and hardware infrastructure. *Neuroinformatics*. 2014.

[10] StatCounter. *Mobile and Tablet Operating System Market Share Worldwide* [online]. StatCounter, 2018. [cit. duben 2018]. Dostupné z: `http://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/#monthly-201706-201802-bar`.

[11] Statista. *Market share of smartwatch unit shipments by vendor from the 2Q'14 to 4Q'16* [online]. Statista, 2016. [cit. duben 2018]. Dostupné z: `https://www.statista.com/statistics/524830/global-smartwatch-vendors-market-share/`.

[12] Todd Fredrich, P. e. *Using HTTP Methods for RESTful Services* [online]. restapitutorial, 2018. [cit. duben 2018]. Dostupné z: http://www.restapitutorial.com/lessons/httpmethods.html.

[13] Šnejdar, P. Vytvoření aplikace pro sběr medicínských dat v rámci projektových dnů. Master's thesis, Západočeská univerzita v Plzni, 2018.

B Uživatelský manuál

Ke spuštění aplikace je nutné mít zařízení s Androidem nebo alespoň jeho emulátor.

B.1 Instalace

Jelikož aplikace není na oficiálních stránkách *Obchod play*, musí uživatel před instalací povolit v telefonu instalaci z neznámých zdrojů. Potom lze velmi jednoduše aplikaci nainstalovat.

B.2 První spuštění

Při prvním spuštění je nutné odsouhlasit podmínky použití, jinak aplikaci nelze používat. Pak už stačí si jen vybrat příslušný mód a přihlásit se. Uživatelé, kteří nejsou součástí *Body In Numbers* mohou využít testovacího účtu, kde bude nejprve potřeba se registrovat.

B.3 Přidání Jídla

Pokud chce uživatel uložit snědenou potravinu do svého účtu, provede tak přesměrováním na obrazovku *Jídlo*, kde se tlačítkem *Přidat* dostane na daný formulář.

B.4 Sledování sportování

Pro sledování sportovních aktivit je nutné se dostat na obrazovku Sport. Tady, po kliknutí na plovoucí tlačítko +, se lze dostat na určitý sport, který lze sledovat nebo ručně přidat.

B.5 Dotazník

Pro vyplnění dotazníku v rámci *Body In Numbers* uživatel musí přejít na obrazovku *Dotazník*, kde si vybere možnosti podle svého uvážení. Takto vyplněný dotazník pošle tlačítkem *Odeslat*. Dotazník se zase objeví po měsíci od posledního vyplnění.

C Seznam obrázků

D Seznam kódů

E Seznam tabulek

F Seznam zkratek

- EEG: Elektroencefalogram
- QR: Quick Response (rychlá odpověď)
- FVC: Forced Vital Capacity (usilovná vitální kapacita)
- FEV1: Forced Expiratory Volume (usilovný vydechnutý sekundový objem)
- PFE: Peak Expiratory Flow rate (vrcholová výdechová rychlost)
- BMI: Body Mass Index (index tělesné hmotnosti)
- API: Application Programming Interaface (rozhraní pro programování aplikací)
- REST: Representational state transfer (rozhraní pro komunikaci se serverem)
- GDPR: General Data Regulation Protection (právní rámec ochrany osobních údajů v rámci EU)
- GPS: Global Positioning System (globální poziční systém)
- GSM:Global System for Mobile communications (globální systém pro mobilní komunikaci)
- URI: Uniform Resource Identifier (jednotný identifikátor zdroje)
- SDK: Software Development Kit (sada nástrojů pro vývoj software)
- UI: User Interface (uživatelské rozhraní)
- HTML: HyperText Markup Language (hypertextový značkující jazyk)
- JSX: JavaScript Syntax Extension (rozšíření syntaxe JavaScriptu)
- Props: Properties (vlastnosti, parametry komponenty)
- JVM: Java Virtual Machine (virtuální stroj Java)
- DVM: Dalvik Virtual Machine (virtuální stroj Dalvik)
- MVC: Model View Controller (model pohled kontroler)

- SQL: Structured Query Language (strukturovaný dotazovací jazyk)
- CSV: Comma Separated Values (hodnoty odděleny
- SHealth: Samsung Health