VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

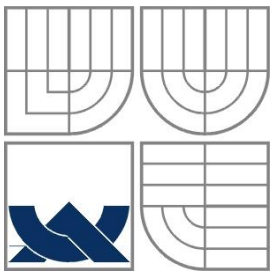
BEŽECKÝ TRENÉR PRO IPHONE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

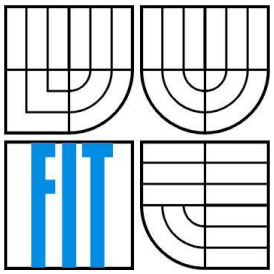
AUTOR PRÁCE
AUTHOR

Radek Doležal

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BEŽECKÝ TRENÉR PRO IPHONE

RUNNING COACH FOR IPHONE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Radek Doležal
AUTHOR

VEDOUCÍ PRÁCE — Ing. Vítězslav Beran, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Cílem práce je vytvořit aplikaci na iPhone fungující jako osobní trenér, která pomocí skloubení funkce mapování denního příjmu a výdeje energie a funkce krokoměru, bude uživatele motivovat k pohybu a upravení jeho denních návyků. Stěžejní částí práce je vlastní návrh aplikace a jejího uživatelského rozhraní. Práce dále pojednává o problematice vývoje aplikací na platformu iOS a zahrnuje základní informace o funkci metabolismu.

Klíčová slova

Osobní trenér, měření běhu, měření výdeje a příjmu energie, iPhone, iOS, uživatelské rozhraní, Core Motion, mobilní aplikace, krokoměr, metabolismus.

Abstract

The goal of my thesis project is development of personal trainer iPhone application. Through combination of daily intake and daily expenditure monitoring including also pedometer function, the user will be motivated to exercise and modify his or her daily habits. Key part of my work is actual application and its user interface design. Furthermore my thesis describe the iOS platform application development and includes a basic information about the metabolic function.

Keywords

Personal trainer, measurement of a run, monitoring of daily intake and expenditure, iPhone, iOS, user interface, Core Motion, mobile application, pedometer, metabolism.

Citace

Doležal Radek: Běžecký trenér pro iPhone. Brno, 2013, bakalářská práce, FIT VUT v Brně.

Běžecký trenér pro iPhone

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radek Doležal
8. května 2013

Poděkování

Děkuji vedoucímu mé práce panu Ing. Vítězslavu Beranovi, Ph.D. za odborné vedení mé práce, motivování v průběhu semestru a užitečné rady při návrhu postupu realizace celé práce. Dále bych chtěl poděkovat Tomáši Kavanovi za pomoc při testování aplikace.

Obsah

Úvod

V dnešní době, kdy správné stravovací návyky a tělesná aktivita ustupují spíše na pozadí lidských potřeb je stále těžší a těžší udržet si vyrovnaný a zdravý způsob života, což vede k tloustnutí lidské populace. Tento jev jistě souvisí i s tím, že v dnešní hektické době nemá člověk příliš času na to, aby se zamyslel nad tím, co vlastně jí a jestli jeho příjem potravin není větší, než jeho tělo ve skutečnosti potřebuje. Proto jsem se rozhodl ve své bakalářské práci věnovat tomuto problému.

Cílem mé práce je navrhnout a implementovat prototyp aplikace, která uživateli umožní jednoduše zmapovat svůj den, co se příjmu a výdeje energie týče a bude uživatele motivovat k určité aktivitě a pohybu, proto název běžecký trenér. Jako platformu, na kterou budu aplikaci vyvíjet, jsem zvolil operační systém iOS mobilních telefonů iPhone. Pro mobilní aplikaci jsem se rozhodl hlavně z důvodu aktuálního trendu rozmachu chytrých telefonů ve společnosti a z důvodu jednoduchého přístupu k aplikaci v telefonu, který má většina lidí stále u sebe. Hlavním rysem aplikace by měla být jednoduchost jejího použití, proto se v práci zaměřím hlavně na vhodný návrh uživatelského rozhraní.

Vlastní práce je logicky členěna na čtyři části. První z nich se věnuje teoretickým znalostem, potřebným pro úspěšné navrhnutí a realizaci aplikace, druhá část je věnována myšlenkám, vizím a jejich převedení do návrhu aplikace, třetí část popisuje postup realizace a podstatné implementační detaily. Poslední čtvrtou částí je závěr, ve kterém zhodnotím dosažené výsledky.

1 Teorie

Kapitola teorie je rozdělena na čtyři části, z nichž každá popisuje jeden směr teoretických znalostí, potřebných pro návrh a implementaci aplikace.

1.1 Úvod do iOS

iOS je mobilní operační systém, vyvinutý společností Apple. Vznikl roku 2007 a původně byl určený pro mobilní telefony iPhone a pro iPod Touch, postupem času se začal používat i na dalších zařízeních od Apple, a to na tabletech iPad a nejnověji i na Apple TV.

Operační systém iOS je systém UNIXového typu. Jedná se o odlehčenou verzi systému Mac OS X známého z počítačů firmy Apple. Protože se jedná o operační systém určený pro mobilní zařízení, neobsahuje veškerou funkčnost OS X, ale za to je doplněn o podporu dotykového ovládání. Jednoduché ovládání a přívětivé uživatelské rozhraní je jeho základním rysem. Rozhraní je postavené na konceptu přímé manipulace s objekty s vhodnými zpětnými vazbami a možnosti užití multi-touch gest. Systém jako takový je uzavřený, ale obsahuje některé open-source součásti.

Architektura iOS

Operační systém iOS je rozdělen do čtyř vrstev: Cocoa Touch, Media, Core Services, Core OS, kde každá zajišťuje určitou základní funkčnost a poskytuje vývojáři API a frameworky potřebné pro vývoj aplikací [1]. Vrstvy na nižších úrovních poskytují základní služby a technologie, na které spoléhají všechny aplikace. Vrstvy na vyšších úrovních obsahují více sofistikované služby a technologie.

Obr. 1 Vrstvy iOS [1]

Popis vrstev [1]:

Cocoa Touch

Tato vrstva obsahuje nejdůležitější frameworky při vývoji aplikací. Technologie dostupné v této vrstvě poskytují infrastrukturu pro implementaci grafického rozhraní aplikace a interakci s uživatelem

a poskytuje vysoko úrovňové systémové služby (např. multitasking, dotykové ovládání, lokální a push notifikace, rozpoznávání gest, atd).
Příklady frameworků: UIKit, Game Kit, Event Kit, Map Kit, Twitter, iAd.

Media

Tato vrstva umožňuje vytvářet graficky a zvukově propracované aplikace. Tyto technologie umožňují plynulé přehrávání animací, videí a zvuků.
Příklady frameworků: Core Graphics, OpenGL ES, Core Animation, Core Audio.

Core Service

Tato vrstva obsahuje základní systémové služby, které všechny aplikace využívají (např. práci s daty, automatická reference objektů, určování polohy zařízení, práci s akcelerometrem, atd). I když člověk nemusí využívat tyto služby přímo, stále na nich stojí spousta jiných součástí systému.
Příklady frameworků: Core Data, Core Foundation, Core Location, Core Motion.

Core OS

Tato vrstva obsahuje nízko úrovňové funkce, na kterých je většina ostatních technologií postavena (např. matematické výpočty, alokace paměti, standardní vstup/výstup, atd.).
Příklady frameworků: Accelerate Framework, Security Framework, External Accessory Framework.

Pro vývojáře je podstatné, že Apple a tudíž i iOS přináší většinu svých systémových rozhraní ve speciálních balících tzv. frameworcích. Framework je adresář, který obsahuje dynamické sdílené knihovny a zdroje (např. hlavičkové soubory, obrázky, pomocné aplikace, a tak dále) potřebné pro podporu této knihovny.

Vlastnosti iOS

Pro programátora je důležité uvědomit si, že iOS je operační systém pro mobilní zařízení a tak jsou jeho vlastnosti mírně odlišné od vlastností desktopových systémů.

Mezi nejdůležitější vlastnosti patří:

Omezené možnosti multitaskingu

Nejedná se o plnohodnotný multitasking. V popředí běží vždy jen jedna aplikace, která se v případě potřeby pozastaví nebo úplně vypne. Aplikace na pozadí se automaticky pozastavují, jen některé výjimky jako například aplikace na stahování, přehrávání zvuků, sledování polohy nebo aplikace přijímající VoIP hovory, mohou na pozadí běžet neomezenou dobu [2].

Pouze jedno okno

iOS umožnuje aplikaci vytvořit pouze jedno „okno“, se kterým může aplikace pracovat [3].

Omezený přístup

iOS přísně omezuje, k čemu se aplikace může dostat. Číst lze pouze ze souborů a zapisovat do nich lze pouze v části souborového systému, která byla vytvořena pro vaši aplikaci. Této oblasti se říká „pískoviště“ a aplikace do ní ukládá dokumenty, předvolby a všechna ostatní data, která potřebujete uložit [3].

Omezený čas odezvy

Kvůli způsobu využití musí být iPhone, potažmo iOS rychlý a to samé se očekává od aplikace. Když se program spustí, musí otevřít aplikaci, načíst předvolby a data a zobrazit hlavní okno na displeji, to mu nesmí trvat déle, než několik sekund. Program musí navíc počítat s tím, že jej může uživatel kdykoliv přerušit. Pokud uživatel stiskne tlačítko pro návrat do hlavního menu, aplikace musí rychle uložit všechna data a ukončit se. Pokud to trvá déle než pět sekund, bude aplikace ukončena násilně [3].

Omezená velikost obrazovky

Displej telefonu není tak velký, a proto není k dispozici tolik prostoru, jako na displeji moderního počítače [3].

ARC (Automatic Reference Counting)

ARC se stará o automatické uvolňování paměti. ARC vyhodnocuje dobu životnosti objektů a automaticky vkládá volání odpovídajících uvolňujících metod v době kompilace [4].

Vývoj na iOS

Je třeba říci, že náklady na vývoj pro platformu iOS nejsou nejmenší. Aplikace pro iOS je možné vyvíjet pouze na strojích od firmy Apple, a k vyvíjení je potřeba zřídit si u Apple developerský účet, jehož zřízení je sice zdarma, ale pokud chce člověk aplikaci distribuovat přes Appstore, testovat na fyzickém zařízení iPhone (ne pouze na simulátoru) a využívat nadstandardních nástrojů musí platit roční poplatek 99$.

Nespornou výhodou iOS, oproti ostatním operačním systémům pro mobilní zařízení, je pro vývojáře to, že se jedná o systém ušitý na míru HW zařízení od firmy Apple, tudíž vývojář nemusí řešit problémy ohledně kompatibility s různými přístroji (různé velikosti displeje, různé velikosti operační paměti, apod.). Další výhodou je rozsáhlý systém technické podpory od firmy Apple a možnost distribuovat aplikace skrze nejziskovější distribuční systém Appstore, který nabízí různé způsoby monetizace a pokrývá veškerou práci publisherů, distributorů a dalších prostředníku, díky čemuž vývojáři zůstane větší podíl ze zisku.

Před vystavením na Appstore musí aplikace projít schvalovacím procesem, díky kterému Apple zabraňuje distribuování nekvalitních, chybových a nepovolených aplikací.

Nástroje a základní přístupy k vývoji na iOS:

Vývojové prostředí Xcode

Apple zdarma poskytuje vývojové prostředí Xcode, které obsahuje nástroje pro tvorbu a ladění zdrojového kódu, pro tvorbu uživatelského rozhraní, kompilování aplikací a zlepšování jejich výkonu. Součástí Xcode je také simulátor, který umožňuje spouštět a testovat většinu aplikací pro iPhone přímo na počítači.

Jazyk Objective-C

Nativní jazyk pro systém iOS je jazyk Objective-C. Jedná se o vysokoúrovňový, objektově orientovaný jazyk implementovaný jako rozšíření jazyka C. Oproti C je do jazyka Objective-C přidán systém zasílání zpráv z jazyka Smalltalk, možnost definovat vlastnosti proměnných a možnost užívat tzv. bloky (uzávěry). Objective-C navíc vyžaduje oddělení rozhraní a implementace do samostatných bloků, eventuálně souborů.

Koncepce Model-Vzhled-Řízení (Model-View-Controller)

Každá správná aplikace pro iOS by se měla řídit koncepcí Model-Vzhled-Řízení (MVC) [3], [7].
Koncepce MVC rozděluje funkcionalitu aplikace do tří různých kategorií:

- Model: Třídy a objekty, které obsahují data aplikace. Někdy je model označován jako jádro aplikace.
- Vzhled (View): Tvoří okna, ovládací prvky a další komponenty, které uživatel vidí a se kterými může interagovat.
- Řízení (Controller): Spojuje model a vzhled a představuje logickou součást aplikace, která rozhoduje o tom, jak aplikace naloží se vstupem, který jí uživatel předá.

Cílem koncepce MVC je při programování zařadit každý objekt do jedné z těchto kategorií a aby téměř nebo vůbec neobsahoval funkcionalitu, která by spadala do některé z ostatních dvou kategorií.

1.2 Uživatelské rozhraní

Stěžejní částí mobilní aplikace je uživatelské rozhraní, aplikace musí být jednoduchá na použití a intuitivní. Pokud v dnešní době velkého množství mobilních aplikaci není uživatel schopen jednoduché interakce s aplikací, okamžitě aplikaci maže a volí jinou. Správnému návrhu aplikací a jejich uživatelských rozhraní se věnuje oblast známa pod názvem User experience design (zkráceně UXD) [8]. Vhodným návrhem uživatelského rozhraní, při kterém dodržujeme zásady a pravidla UXD ovšem jeho vývoj nekončí, možná ještě důležitější části vývoje je jeho následné otestovaní, vyhodnocení výsledků a případná další úprava či vylepšení. Dá se říci, že vývoj uživatelského rozhraní je iterativní proces.

Zásady při tvorbě uživatelského rozhraní

U mobilních zařízení a obzvlášť u iOS je jednou z nejdůležitějších věcí vzhled a intuitivnost ovládání, tomu se věnuje User experience design (UXD). Firma Apple, která je pověstná svým propracovaným User experience, na což se dá nahlížet jako na nadmnožinu UXD, která zahrnuje i pocity, jež má konkrétní produkt v uživateli vyvolat [9], dokonce vydala příručku nazvanou iOS Human Interface Guidelines, která se právě UX i UXD věnuje. Jsou v ní popsány i následující principy, kterých by se měl vývojář pro iOS při návrhu uživatelského rozhraní držet.

Principy návrhu uživatelského rozhraní [6]:

Estetická Integrita

Estetická integrita neboli celistvost, znamená, že vzhled aplikace by měl odpovídat její funkčnosti. Například v aplikaci, ve které uživatel řeší nějakou produktivní úlohu, by dekorativní prvky měli být pouze na pozadí a hlavní důraz by měl být kladen na ovládací prvky a vlastní úlohu. Naopak třeba u her uživatel očekává hezký a zábavný vzhled, tudíž může být upřednostněn. Zkrátka vzhled aplikace by měl uživateli dávat jasnou zprávu o jejím účelu.

Konzistence

Konzistencí v rozhraní se rozumí to, že by ovládací prvky aplikace měli splňovat určité normy a vzory, které lidé obecně znají. Tato konzistence, umožňuje uživatelům využít zkušenosti s ovládáním z jiných aplikací a snadnější užití. iOS Human Interface Guidelines detailně popisuje zažité příklady užití konkrétních ovládacích prvků.

Přímá manipulace

Uživatelé iOS jsou zvyklí manipulovat s objekty na obrazovce přímo, namísto použití samostatných ovládacích prvků. Napomáhá k tomu určitě vlastnost systému iOS a to možnost užívání multi-touch gest. Tento způsob manipulace dává uživatelům větší pocit kontroly nad objekty a uživatelé potom snáze porozumějí výsledkům svých činů. Příkladem přímé manipulace je užití gesta „špetka“ nad konkrétním místem na obrazovce při zoomování namísto kliknutí na ovládací prvek.

Zpětná vazba

Uživatel iOS vyžaduje okamžitou zpětnou vazbu, působí-li na ovládací prvek aplikace a ocení informaci o aktuálním stavu operace u déle trvajících operací. Za zpětnou vazbu může být považováno například zvýraznění prvku, decentní animace nebo zvuk, zvuk by ale neměl být užit jako jediná zpětná vazba, protože uživatel může aplikaci používat s vypnutými zvuky.

Metafory

Pokud jsou objekty a akce v aplikaci virtuálními metaforami objektů a akcí v reálném světě, uživatelé rychle pochopí, jak aplikaci používat. Klasickým příkladem softwarové metafory je složka, lidé používají složky v reálném světě k ukládání věcí a dokumentů, a tak okamžitě pochopí i myšlenku ukládání souborů do složek v aplikaci.

Uživatel řídí aplikaci

Iniciovat a kontrolovat činnost by měl uživatel, ne aplikace. Aplikace může navrhnout postup nebo varovat o nebezpečných důsledcích, ale neměla by dělat rozhodnutí bez zásahu uživatele. Aplikace by měla poskytovat dostatek možností jak zrušit operaci (ať už za běhu operace nebo před jejím začátkem) a měla by varovat uživatele před provedením potenciálně destruktivních akcí.

Testování uživatelského rozhraní

Při vývoji uživatelského rozhraní se vyplatí testovat jej již v průběhu návrhu a vývoje, předejde se tím zbytečnému plýtvání času a rozvíjení chybných návrhů do slepých uliček. Na vývoj uživatelského rozhraní tedy můžeme nahlížet jako na iterativní proces, ve kterém se opakuje návrh, implementace, testování a vyhodnocení výsledků.

Vhodným způsobem testování uživatelského rozhraní je uživatelské testování, při kterém uživatelům zadáme nějaký úkol a poté pomocí pozorování nebo dotazů získáváme zpětnou vazbu. Při uživatelském testování je důležité vybrat vhodný vzorek uživatelů a definovat si cíl a průběh testování. Průběh testování by měl být s každým uživatelem stejný, aby nedocházelo ke zkreslení výsledků testů. Proto je dobré si předem definovat protokol testování, jehož se v každém testu budeme držet. Testovaní uživatelé by měli být lidé odpovídající cílové skupině uživatelů aplikace.

Při uživatelském testování návrhu rozhraní lze využít jedné z následujících technik [12]:

Náčrtek na papíře

Hrubý náčrtek rozhraní na papíře, který ukážeme uživateli a pomocí otázek zjišťujeme porozumění jednotlivým prvkům. Tato technika je vhodná při testování prvotních koncepčních nápadů rozhraní.
Výhody: jednoduché na vytvoření.
Nevýhody: horší možnosti úprav, návrh je třeba celý překreslit.

Wireframe

Wireframe je „drátěný model“ rozhraní vytvořený v editoru na počítači. Jeho využití je podobné náčrtku na papíře. Wireframy doplněné o dotazník můžeme elektronicky šířit a získávat tak zpětnou vazbu od více uživatelů.
Výhody: snadné provádění úprav, možnost šířit elektronicky, kopírovat a tisknout.
Nevýhody: stále se jedná o oddělené náčrty, které je třeba ukazovat uživateli jeden za druhým.

Interaktivní prototyp

Jednoduchý prototyp aplikace zaměřený pouze na funkčnost uživatelského rozhraní, může se jednat o pouhé propojení více wireframů navzájem. Je možné simulovat pohyb skrze aplikaci. Díky interaktivitě prototypu již nemusí být hlavním motivem testování kladení otázek, ale můžeme uživateli pouze zadat úkol a sledovat jej při jeho plnění.

1.3 Energetická výměna v lidském těle

Jelikož cílem práce je vytvořit mobilní aplikaci v oblasti zdraví a fitness, je důležitou částí teoretických znalostí potřebných pro návrh a implementaci aplikace také znalost toho, jak tělo funguje. Konkrétně jak funguje energetická a látková výměna v lidském těle a jak souvisí s udržováním optimální váhy, popřípadě s hubnutím.

Metabolismus

Udržování optimální váhy, popřípadě redukce váhy je závislá na množství přijaté a spotřebované energie. Základními jednotkami energie jsou buď kilokalorie (kcal) nebo kilojoule (kJ), přičemž 1 kcal je zhruba 4,185 kJ.

Hodnota vydané energie potřebné pro látkovou přeměnu se jinak nazývá metabolismus. Pro určení metabolismu je nutné znát hodnotu bazálního (klidového) metabolismu. K němu pak připočteme energii, kterou vydáváme fyzickou i duševní činností během dne a výsledné číslo je množství vydané energie nutné pro pokrytí celkového denního metabolismu. Pokud zdravý člověk příjme množství energie zhruba odpovídající tomuto číslu, neměl by přibírat ani ztrácet váhu. Logicky tedy pokud člověk přijímá denně větší množství energie, než vydává, tak přibírá a naopak pokud společně s bazálním metabolismem vydává více energie, než přijímá, tak hubne. Co se týče hubnutí, tak se odborníci shodují, že optimální je hubnout něco kolem 2 kg za měsíc, což znamená denně přijmout přibližně o 485 kcal méně než vydat [10].

Bazální metabolismus

Bazální metabolismus (BM) vyjadřuje množství energie vydané tělem, funkcí orgánů, centrální nervovou soustavou apod. v klidovém stavu. Hodnota BM je u každého člověka individuální, výsledné číslo je ovlivněno věkem, tělesnou strukturou, pohlavím, výškou, váhou, poměrem aktivní (netukové tkáně) a pasivní hmoty (tukové tkáně). Energetický výdej je nejvyšší během dětství a dospívání. S rostoucím věkem hodnota BM klesá, což se u většiny lidí projevuje zvyšováním procenta tuku.

Na bazální metabolismus má vliv i svalová a duševní aktivita jedince a hladovění, svalová a duševní aktivita bazální metabolismus zvyšuje a hladovění naopak snižuje. Pokud se tedy člověk snaží zhubnout pouze omezením příjmu potravy, dochází k rapidnímu snížení hodnoty BM, což má za následek tak zvaný „Jojo efekt“. Člověk totiž hladověním nějaké to kilo shodí, ale vlivem poklesu BM ihned, jakmile opět začne jíst, tak jak byl zvyklý, rychle nabere na váze. Proto je vhodné hubnout spíše zvýšením tělesné aktivity, anebo alespoň vyváženou kombinací omezení příjmu a zvýšením tělesné aktivity.

Vzorce pro výpočet BM [11]:

muži: $BM = 66 + (13.7 \times hmotnost\ [kg]) + (5 \times výška\ [cm]) - (6.8 \times věk\ [roky])\ [kcal/den]$

ženy: $BM = 655 + (9.6 \times hmotnost\ [kg]) + (1.85 \times výška\ [cm]) - (4.7 \times věk\ [roky])\ [kcal/den]$

1.4 Existující řešení

Před vlastním návrhem aplikace je vhodné prostudovat již existující řešení na trhu. Jelikož cílem této práce je vytvořit aplikaci fungující současně jako běžecký trenér a výživový poradce, zaměřím se v následující kapitole na existující aplikace právě těchto dvou typů.

Aplikace typu výživový poradce

Tento typ aplikací slouží k mapování uživatelova příjmu a výdeje kalorií. Aplikace dle zadaného cíle (hubnutí nebo udržení váhy) vypracuje uživateli určitý plán na zadanou délku období a uživatel po té den co den do aplikace jednoduše zadává, co snědl, popřípadě vykonal. Aplikace mu každý den ukazuje, kolik mu ještě zbývá k dosažení denních limitů a vede statistiky o průběhu sledovaného období.

Jednotlivé aplikace tohoto typu se od sebe příliš neliší. Některé sice obsahují pár funkcí navíc jako třeba možnost ukládání receptů a podobné, ale o praktičnosti těchto rozšíření si může udělat obrázek každý sám. Stěžejními prvky těchto aplikací a jedinou možností jak se odlišit, je způsob výběru a vložení jídla/aktivity do denního souhrnu a přehlednost zobrazení denního průběhu a statistik. Proto se při popisu existujících aplikací zaměřím převážně na tyto vlastnosti.

MyPlate Calorie Tracker - LIVESTRONG.COM

Zajímavě řešená aplikace mapující příjem a výdej kalorií. Hlavním rysem je rozdělení denního příjmu kalorií na jednotlivé denní chody, jako je snídaně, oběd, večeře a svačiny, které se musí naplnit, čímž motivují uživatele ke správným stravovacím návykům. Výběr konkrétního jídla/aktivity je řešen pomocí vyhledávácí lišty do které uživatel vpisuje jméno zadávaného jídla/aktivity a lišta vrací všechny jídla/aktivity odpovídající zadanému řetězci. Tento způsob výběru je sice velice rychlý, ale jeho nedostatkem je, že pokud nevíme pod jakým názvem je konkrétní jídlo/aktivita v aplikaci uložena, tak je nemožné ji najít. Zobrazení denního průběhu je řešeno vkusně a přehledně.

Calorie Counter & Diet Tracker by MyNetDiary

Vlivem soustředění všech funkcí aplikace na jednu domovskou obrazovku je aplikace nepřehledná. Zadávání jídel je opět rozděleno na denní chody, ovšem oproti MyPlate Calorie Tracker postrádá toto rozdělení nějakou přidanou motivační hodnotu a ani nějak nezpřehledňuje zadávání jídla. Aplikace umožňuje vkládat jídla buď vyhledáním pomocí vyhledávací lišty, nebo pomocí sejmutí čárového kódu z konkrétního výrobku. Zobrazení denního průběhu se ztrácí v nepřeberném množství možností na hlavní úvodní obrazovce.

Calorie Counter: diets & activities

Co se týče vzhledu, velice příjemně řešená aplikace. Díky přehledné a logicky správně uspořádané liště záložek je pohyb skrze aplikaci jednoduchý a uživatel se v aplikaci téměř nemůže ztratit. Výběr jídla je tentokrát řešen pomocí hierarchických tabulek, kdy si uživatel nejprve vybere skupinu jídel, do které hledané jídlo patří a v ní následně konkrétní jídlo. Tento způsob výběru je sice o něco pomalejší, než způsob s vyhledávací lištou, ale zase naopak nehrozí, že by uživatel jídlo nenašel.

Aplikace typu pedometr

Z vývojářského hlediska lze mobilní pedometry dělit na dvě skupiny, na pedometry využívající data z GPS a na pedometry nebo spíše krokoměry využívající dat z akcelerometru. Oba přístupy s sebou nesou jisté výhody a nevýhody. Výhodou pedometrů pracujících s GPS jistě je, že si můžete prohlédnout mapu běhu a zaznamenaná délka a rychlost je o něco přesnější, ale tato výhoda se rychle maže, pokud není k dispozici kvalitní GPS signál, proto se dá říci, že tyto pedometry nejsou tak spolehlivé jako ty s akcelerometrem. Naopak krokoměry pracující s akcelerometrem, měří za každých podmínek a dají se použít i při měření běhu na běžeckém pásu, ale nejsou tak přesné.

Z uživatelského hlediska se co do funkčnosti pedometry příliš neliší, některé sice nabízejí různé rozšíření, jako například přehrávání hudby, možnosti interakce se sociálními sítěmi a podobně, ale hlavní vlastností aplikace je opět spíše vzhled a přehlednost.

Nike+ Running

Aplikace má příjemný vzhled a uživatelské rozhraní. Nabízí možnost soutěžit s přáteli, kdo naběhá víc, obsahuje zabudovaný přehrávač hudby a možnosti sdílení výsledků na sociální sítě. Co se týče měření běhu, je aplikace vystavěna jako kombinace obou v úvodu zmíněných přístupů. Implicitně pracuje s GPS, ale pokud je špatný signál, přejde na měření běhu pomocí akcelerometru.

Pedometer FREE GPS +

Hlavním rysem a zároveň nevýhodou je dle mého názoru nerovnoměrné využití záložek, kdy veškerá důležitá funkčnost je soustředěna na jednu obrazovku, což vede k jejímu přeplácání a ostatní záložky obsahují pouze doplňkové funkce, které uživatel málo kdy využívá. Co se týče měření běhu, aplikace opět využívá kombinace obou způsobů získávání dat.

Footsteps - Pedometer Free

Vcelku příjemný jednoduchý krokoměr s vhodně navrženým uživatelským rozhraním. Pracuje čistě na bázi akcelerometru, což určuje jeho přesnost a spolehlivost.

Endomondo

Jedna z nejznámějších a neoblíbenějších aplikací na měření běhu. Jedná se o něco komplexnější aplikaci, která nabízí možnosti měření i jiných sportů a aktivit. Co se týče uživatelského rozhraní, aplikace nabízí možnost upravení si aplikace na míru. Uživateli zobrazuje právě to, co mu vyhovuje.

Měření běhu je realizováno pomocí dat z GPS, tudíž na přesnost měření má vliv okolní prostředí a kvalita signálu.

2 Návrh

Tato kapitola, se věnuje vlastnímu návrhu aplikace. Popisuje základní vizi a cíl řešení, jeho návrh a nastiňuje postup testování v závislosti na předešlých teoretických kapitolách. Kapitola je rozdělena do pěti podkapitol, které jsou seřazeny od abstraktnějších návrhů a vizí až po konkrétní návrh GUI a testování aplikace.

2.1 Vize a cíl

V závislosti na poznatcích získaných při studování existujících aplikací v oblasti zdraví a fitness, jsem si definoval cíl projektu. Rozhodl jsem sem vytvořit aplikaci, která v sobě spojuje funkčnost dvou typů aplikací. Aplikací sloužících k mapování příjmu a výdeje energie a aplikací na zaznamenávání a měření pohybových aktivit jako je chůze nebo běh (dále jen běh). Vedly mě k tomu dva důvody, za prvé jsem nenašel aplikaci, která by toto spojení dvou relativně souvisejících funkcí nabízela a za druhé si myslím, že vhodné propojení těchto dvou funkcí, kdy se výsledky denních stravovacích návyků promítají do tréninkové části, a naopak výsledky běhů promítají do souhrnu denního příjmu a výdeje kalorií, může vést k větší motivaci uživatele aplikaci používat a přizpůsobovat svůj životní styl ke zdravějšímu.

Zmínil jsem slovní spojení „vhodné propojení“, tím rozhodně nemyslím pouhé spojení dvou aplikací do jedné. Takové spojení by vedlo pouze k navýšení složitosti aplikace a vlastní aplikaci by nepřidalo žádnou přidanou hodnotu. Jednalo by se pouze o soustředění více funkcí do jedné aplikace a to dle mého názoru spíše uživatele od aplikace odradí, ať už pro již zmíněnou nepřehlednost nebo proto, že uživatel vždy nalezne kombinaci dvou specializovaných aplikací, které budou splňovat jeho požadavky lépe než jedna aplikace, ve které by mu třeba jedna část úplně nevyhovovala. Proto jsem se při návrhu mé aplikace zaměřil jen na určité funkce dvou zmíněných typů aplikací a pomocí nich jsem vytvořil nový typ jednoduché aplikace.

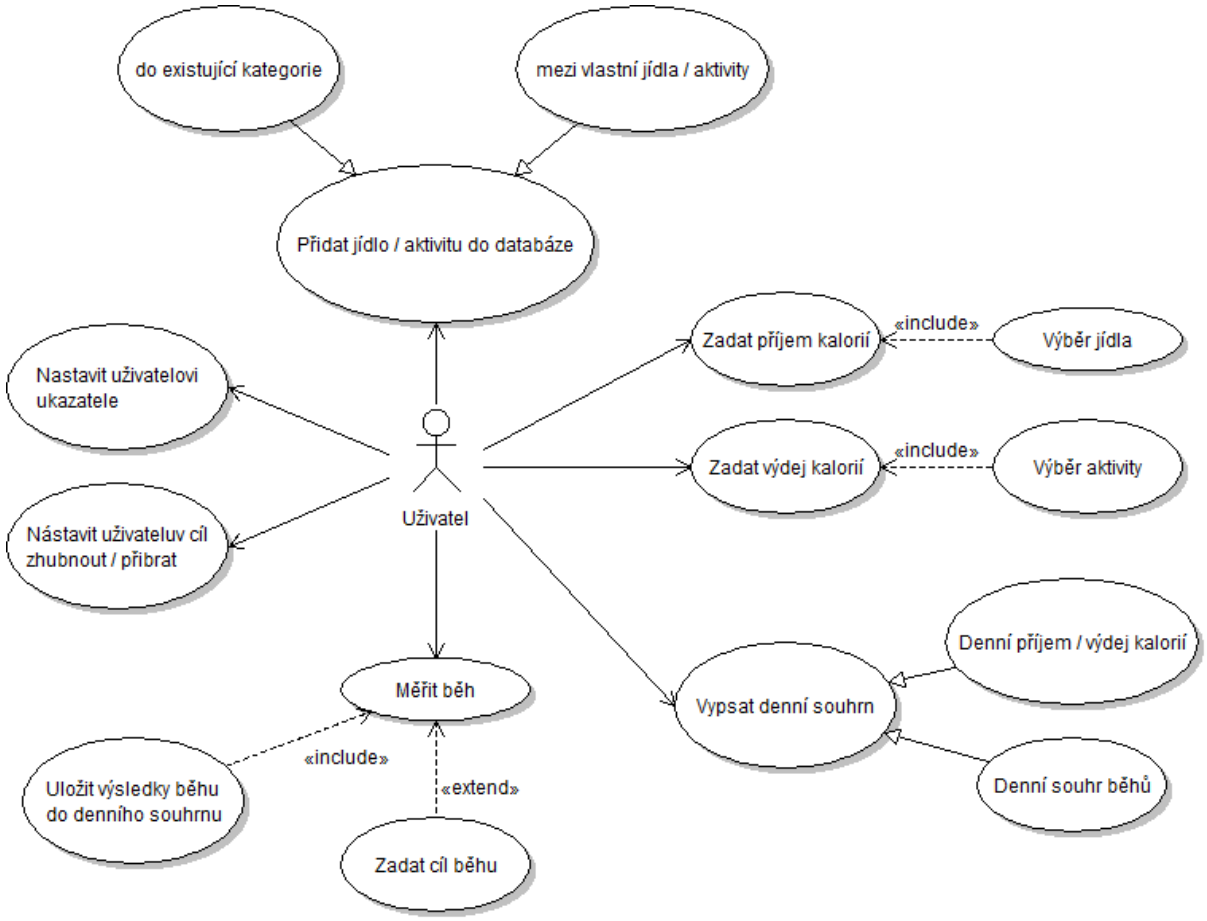
Cílem a hlavní vizí mého projektu, je vytvořit aplikaci, která by uživateli po jednom dni používání ukázala nedostatky v jeho stravovacích a denních návycích, a naznačila co je potřeba změnit k vyváženějšímu způsobu života. Cílovou skupinou aplikace by měl být obyčejný člověk, který chce pouze zjistit, jak zlepšit svůj denní režim. V nadsázce lze říci, že aplikace bude cílit například i na lidi, kteří jen chtějí zjistit, za jak dlouho vyběhají přebytek kalorií po nedělním rodinném obědě. Nebude se jednat o odbornou aplikaci popisující metody hubnutí a vysvětlující správnou skladbu jídelníčku. Proto je v aplikaci kladen důraz spíše na jednoduchost použití a přehlednost. Díky tomu,

že aplikace je zaměřena jen na jeden právě probíhající den, nebude uživatel nucen k zakládání uživatelského profilu, na který by se ukládali jeho dlouhodobé statistiky. Proto je aplikace plně funkční i bez přístupu k internetu.

2.2 Popis užití

Jak již bylo nastíněno v předchozí podkapitole, aplikace je zaměřena na sledování průběhu jednoho dne. Uživatel si během dne bude do aplikace ukládat, co za aktivity vykonal, popřípadě jaké jídlo snědl a na konci dne bude vědět, kolik kalorií mu přebývá nebo chybí k dosažení vyváženého denního poměru příjmu a výdeje energie. Ten se bude odvíjet od cílových hodnot pro denní příjem a výdej energie, které budou nastaveny podle uživatelova cíle. Cílem může být buď udržení stávající váhy, nebo hubnutí, v případě hubnutí může uživatel ještě upravit, kolika procenty chce hubnout zvýšením denních aktivit a kolika procenty omezením příjmu potravy.

V průběhu dne si uživatel v aplikaci může kdykoliv spustit běžecký trénink, kterému může nastavit určitý cíl. Například, že chce spálit kalorie, které má ten den navíc nebo rozdíl mezi aktuální a cílovou hodnotou denního výdeje energie, anebo prostě jen cílový čas nebo délku běhu.
Všechny funkce aplikace a možnosti použití se pokusím demonstrovat na obr. 2.



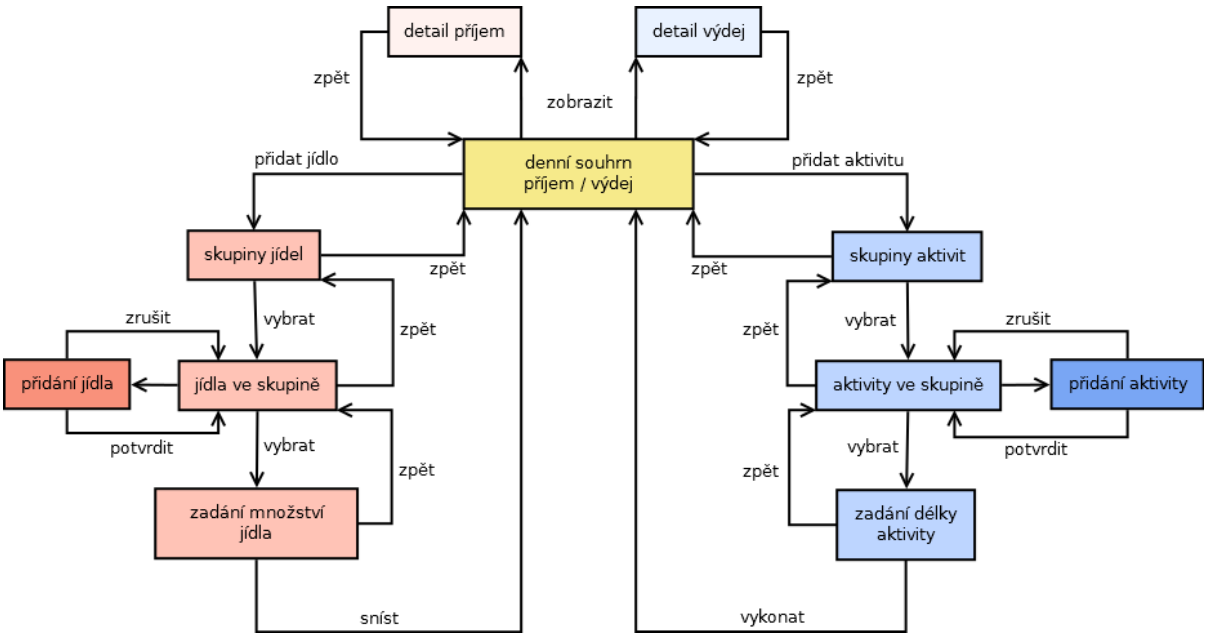
Obr. 2 Diagram užití celé aplikace

2.3 Návrh aplikace

V této kapitole je popsána architektura vlastní aplikace a způsob řešení jejich jednotlivých části. Aplikace je logicky a vlastně i vizuálně, pomocí lišty záložek, členěna na tři části. Část věnující se mapování příjmu a výdeji energie, část věnující se běhům a část sloužící jako nastavení aplikace.

Mapování příjmu a výdeje energie

Už v kapitole o existujících řešeních jsem zmínil, že stěžejními prvky tohoto druhu aplikací jsou způsob zobrazení denního souhrnu a způsob výběru a zadávání příjmů a výdeje energie. Proto jsem se rozhodl, že denní souhrn budu zobrazovat již na úvodní obrazovce této záložky, která v této části aplikace slouží jako výchozí a centrální prvek. Kam se z něj může uživatel dostat a jak se lze v této části aplikace pohybovat jsem demonstroval ve stavovém diagramu na obr. 3.



Obr. 3 Stavový diagram popisující část aplikace mapující denní příjem a výdej energie

Způsob zobrazení denního souhrnu příjmu a výdeje energie

V denním souhrnu aplikace uživateli zobrazuje aktuální denní hodnoty příjmu a výdeje energie, doporučený denní limit příjmu energie, doporučený denní limit energie vydané tělesnou aktivitou a hodnotu nazvanou odchylka. Tato odchylka uživateli zobrazuje, kolik mu ještě chybí, popřípadě přebývá energie pro vyvážený poměr příjmu a výdeje. V případě, že uživatel bude aplikaci užívat s nastaveným cílem zhubnout, bude v odchylce zahrnut i doporučený rozdíl mezi příjmem a výdejem 485 kcal popsaný v kapitole 1.3 Energetická výměna v lidském těle, který je vhodný pro hubnutí 2 kg za měsíc [10].

Vzorec pro výpočet odchylky při snaze o udržení váhy:

$$odchylka = aktuální\ denní\ příjem - aktuální\ denní\ výdej$$

Vzorec pro výpočet odchylky při snaze o hubnutí:

$$odchylka = aktuální\ denní\ příjmem - aktuální\ denní\ výdej + 485$$

Vzorce pro výpočet odchylky

Doporučené denní limity jsou vypočteny dle uživatelových parametrů promítnutých ve spočítané hodnotě bazálního metabolismu a cíle užívaní aplikace nastaveného v nastavení aplikace.

Vzorce pro výpočet doporučených denních limitů při snaze o udržení váhy:

$$limit\ příjem = BM + energie\ vydaná\ tělesnou\ aktivitou$$

$$limit\ výdeje\ tělesnou\ aktivitou = hodnota\ přijaté\ energie\ převyšující\ BM$$

**BM = bazální metabolismus*

Vzorce pro výpočet doporučených denních limitu při snaze o hubnutí:

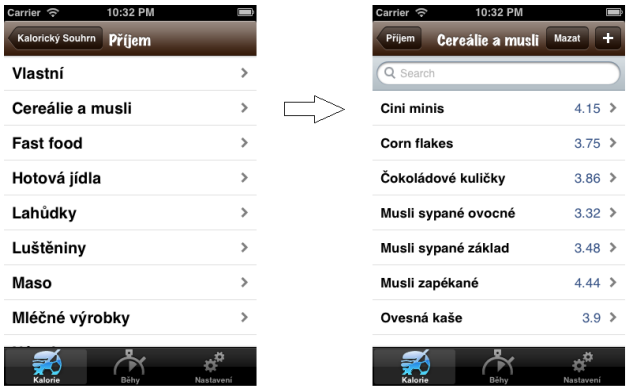
$$limit\ příjem = BM + energie\ vydaná\ tělesnou\ aktivitou\ převyšující\ limit - 485 \times PHP$$

$$limit\ výdeje\ tělesnou\ aktivitou = hodnota\ přijaté\ energie\ převyšující\ limit + 485 \times PHA$$

**BM = bazální metabolismus*
PHP = poměr hubnutí pomocí omezení příjmu
PHA = poměr hubnutí pomocí zvýšení tělesné aktivity

Vzorce pro výpočty denních limitů

Způsob výběru příjmu či výdeje energie



Obr. 4 Způsob výběru pomocí hierarchických tabulek

Co se týče způsobu výběru jídla či aktivity v aplikaci, zvolil jsem řešení pomocí hierarchických tabulek, kdy jednotlivá jídla a aktivity jsou členěny do skupin příbuzných jídel či aktivit. V jednotlivých skupinách je navíc možné vyhledávat pomocí vyhledávací lišty, čímž bych chtěl uživateli umožnit jak rychlý výběr, tak spolehlivost, že jídlo či aktivitu najde, i když nebude znát přesné jméno, pod kterým je v aplikaci uložena. V jednotlivých skupinách může uživatel mazat či přidávat vlastní jídla a aktivity.

Obr. 5 Způsob zadání porce

Způsob zadání porce či délky aktivity

Po výběru konkrétního jídla či aktivity přichází na řadu způsob zadání kvantity. Což je část aplikace, která nejvíce ovlivní rychlost zadání jednotlivých položek a vlastní příjemnost používání aplikace.

Co se týče zadávání porce sněženého jídla, nabídne aplikace uživateli dva způsoby. Buď zadání pomocí přesné gramáže jídla, tato možnost by měla být využívána převážně u jídel, která nemají stanovenou určitou velikost běžné porce nebo u jídel, která do aplikace přidal sám uživatel a nevyplnil volitelnou položku, gramáž běžné porce. Druhým způsobem zadávání je zadání pomocí počtu běžných porcí. Kde všechna jídla, u kterých lze váhu běžné porce určit, mají tuto hodnotu defaultně zadanou a uživatel při zadávání bude pracovat rovnou s touto hodnotou. Bude ji moci mírně poupravit pomocí posuvníku. Také bude moci upravit, kolik takových porcí snědl. Tento způsob je výchozím způsobem zadávání jídla a po pochopení principu by měl uživateli usnadnit a urychlit práci, což je i součástí testování.

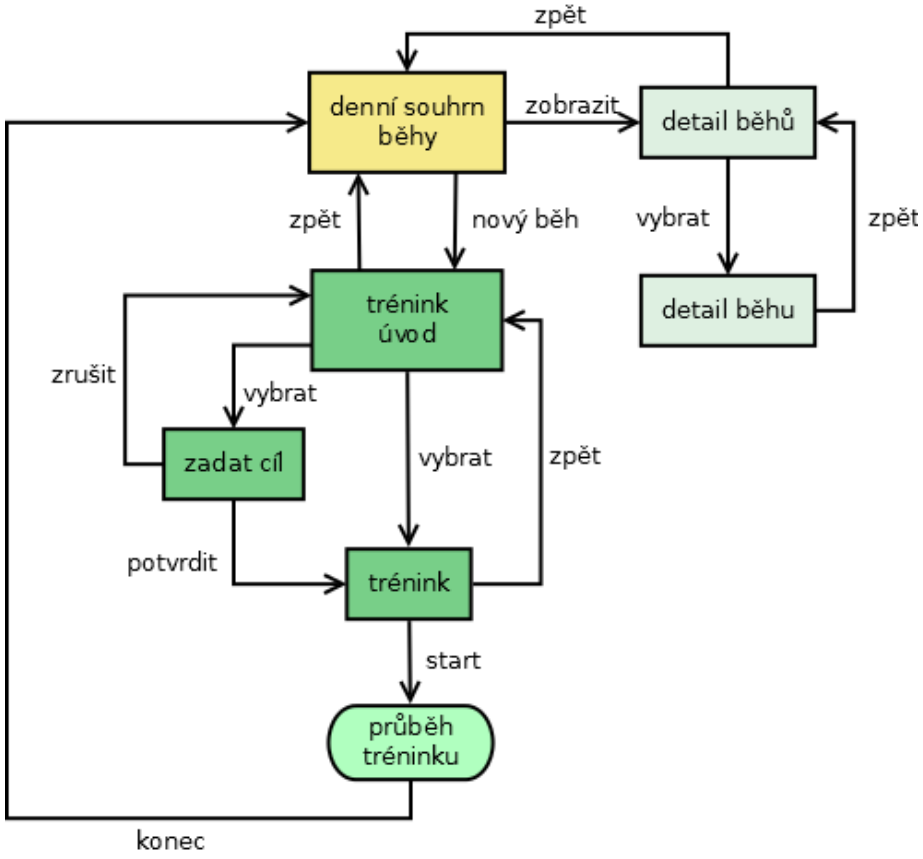
Co se týče zadávání délky aktivity, aplikace nabízí pouze jeden způsob a to jednoduché nastavení délky aktivity v hodinách a minutách na dvou rámcích pro výběr (tzv. pickerech).

Způsob detailního zobrazení a editace denního přijmu a výdeje

Aplikace nabízí možnost vypsat si detailně všechny jídla potažmo aktivity, které uživatel ten den vykonal a informace o nich. Toto zobrazení je implementováno dvěmi přehlednými tabulkami (jednou pro příjem a druhou pro výdej), které může uživatel editovat a tím upravovat svůj denní souhrn.

Mapování pohybových aktivit

Abych udržel určitou konzistenci mezi jednotlivými částmi aplikace, tak úvodní obrazovkou a centrálním prvkem této části je opět denní souhrn. Vlastní architekturu celé této části aplikace demonstruji na stavovém diagramu na obr. 6.



Obr. 6 Stavový diagram popisující část aplikace mapující pohybové aktivity

Způsob zobrazení denního souhrnu pohybových aktivit (dále jen běhů)

V denním souhrnu běhů se uživateli zobrazují statistiky o uskutečněných bězích onoho dne, jako je celková uběhlá vzdálenost, celkový čas, počet kroků za celý den, průměrná rychlost všech běhů, počet spálených kalorií a počet jednotlivých běhů.

Možnost zadání cíle běhu

Před vlastním spuštěním měření běhu, si uživatel vybere, jestli si chce spustit pouze volný trénink bez určitého cíle nebo zda chce svému běhu nastavit nějaký cíl. Pokud zvolí běh s cílem, nabídne mu aplikace tři varianty cíle běhu. Buď si může nastavit cílovou vzdálenost běhu, nebo cílový čas a nebo cílový počet spálených kalorií. Počet spálených kalorií muže zadat buď konkrétní cílovou hodnotu a nebo pomocí funkce aplikace, která spočítá počet kalorií, který uživateli chybí k dosažení doporučeného denního limitu pro výdej energie.

Průběh běhu

Po spuštění měření běhu, je možné měření pozastavit, opět spustit (pokračovat) nebo resetovat. Aplikace měří a zobrazuje aktuální uběhlou vzdálenost, počet kroků, počet spálených kalorií, čas a průměrnou rychlost běhu. Při měření běhu se zadaným cílem aplikace po dosažení cíle, uživatele upozorní vibrací telefonu a změnou barvy zobrazení hodnoty, která přesáhla cíl. Po ukončení měření běhu, se běh uloží jak do denního souhrnu běhů, tak do denního souhrnu výdeje energie v druhé části aplikace.

Algoritmus krokoměru

Stěžejní části krokoměru je algoritmus pro detekci kroků. Jelikož jsem se krokoměr rozhodl implementovat nad akcelerometrem, pracuje se v algoritmu se snímáním a vyhodnocením otřesů telefonu. Po celou dobu měření běhu je snímána jedna osa akcelerometru telefonu, dle polohy telefonu. Diskrétně periodicky snímané hodnoty z akcelerometru jsou nejprve přefiltrovány přes dolní propust, aby došlo k vyhlazení nasnímaného signálu. Vlastní vyhodnocení nasnímaných hodnot probíhá za pomocí dvou citlivostí, minimální a průběžné, přičemž průběžná citlivost se dynamicky přizpůsobuje intenzitě běhu, ale nikdy neklesne pod hodnotu minimální citlivosti.

Způsob detailního zobrazení běhů

Aplikace uživateli umožňuje kromě výpisu souhrnných statistik běhů na úvodní obrazovce také detailní výpis všech běhů. Všechny běhy zobrazí v přehledné tabulce, seřazené podle času, kdy byly aplikací změřeny. Běhy je možné v této tabulce editovat. Po výběru konkrétního běhu z tabulky aplikace uživateli zobrazí detail naměřených statistik vybraného běhu.

Nastavení aplikace

V nastavení aplikace je možné nastavit uživatelovi parametry a cíl užívání aplikace. U hodnot délek jednotlivých typů kroku aplikace umožňuje hodnoty dopočítat z aktuální výšky uživatele.

V aplikaci je možné nastavit následující parametry:

1. Osobní parametry pro výpočet bazálního metabolismu:
   Pohlaví, věk, míra aktivity, výška, váha.
2. Tělesné parametry pro správnou funkci krokoměru:
   Délka kroku při chůzi, délka kroku při běhu, délka kroku při sprintu.
3. Cíl užívání aplikace:
   Udržet váhu, zhubnout (omezením příjmu potravy [%], zvýšením tělesné aktivity [%]).

2.4 Návrh GUI

Při návrhu uživatelského rozhraní jsem vycházel z principů popsaných v kapitole 1.3 Zásady při tvorbě uživatelského rozhraní. Hlavním rysem navrženého GUI je snaha o co nejjednodušší a nejpřehlednější zobrazení všech prvků.

V rámci zachování určité konzistence a usnadnění ovládání aplikace, jsem stěžejními prvky pro navigaci a pohyb skrze aplikaci zvolil standardní ovládací prvky, jako je lišta záložek a navigační lišta. Pro větší přehlednost a intuitivnost je v rámci celé aplikace dodržena určitá konzistence v barvách, které jsou spjaty s každým logickým celkem aplikace.

Obr. 7 Návrh GUI jednotlivých logických celků

2.5 Návrh testovaní

Testování celé aplikace se dělí na testování přesnosti krokoměru a testování uživatelského rozhraní. V následující podkapitole popisuji navrhnutý průběh obou testování, co bylo v testech měřeno a co jsem chtěl testováním zjistit.

Testování krokoměru

Při testování krokoměru jsem se zaměřil na přesnost naměřených hodnot. Cílem testování bylo zjistit odchylky naměřených hodnot a vliv upevnění telefonu na přesnost měření.

Testování probíhalo na předem změřeném úseku, na kterém jsem měřil chůzi a běh. Nejprve jsem provedl několik sérií měření s telefonem umístěným v kapse a po té několik sérií měření s telefonem umístěným ve speciálním pouzdře na běhání. Více sérií jsem zvolil proto, abych zjistil, jestli je měření dostatečně stabilní a výsledky testování lze považovat za relevantní. Naměřené hodnoty jsem porovnával s hodnotami získanými na stejném úseku existujícím krokoměrem, konkrétně aplikací Pedometer Free GPS+. Hodnoty, které lze určit přesně, jako je vzdálenost, počet kroků a průměrná rychlost jsem také porovnal s reálnými hodnotami.

Z měření jsem učinil závěr o přesnosti naměřených hodnot a vlivu upevnění telefonu na odchylku.

Testování uživatelského rozhraní

Testování uživatelského rozhraní bylo zaměřeno na dva faktory, které výrazně ovlivňují příjemnost používání aplikace, konkrétně na intuitivnost uživatelského rozhraní a naučitelnost navržených ovládacích postupů.

Testování je rozděleno do dvou testů, na komplexní test A a praktický test B. Test A se skládá z otázkové části a následných praktických úkolů. Test B obsahuje pouze praktické úkoly. V testu A jsem se v otázkové části zaměřil na intuitivnost konkrétních prvků uživatelského rozhraní a v následných praktických úkolech jsem sledoval osvojení jejich významu a praktické užití. V testu B, ve kterém byl uživatel postaven před praktické úkoly ještě před prozkoumáním celé aplikace, jsem sledoval intuitivnost použitých ovládacích postupů. V obou testech jsem měřil časy strávené nad úkolem a úspěšnost jejich řešení. Výsledky obou testů jsem porovnal a učinil závěr o naučitelnosti navržených ovládacích postupů a vlivu osvojení významů jednotlivých prvků uživatelského rozhraní na délku času stráveného plněním úkolu.

Test A byl proveden na jedné polovině testovaných uživatelů a test B na druhé polovině. V obou polovinách byli zastoupeni stejně staří a zkušení uživatelé. Aby nedocházelo ke zkreslení získaných výsledků, probíhali testy s každým uživatelem ve stejných podmínkách dle následujících postupů.

Průběh testu A:

1. Nejprve byl uživatel v krátkosti seznámen s aplikací, s tím co je jejím cílem a co by měla dělat. Co se týče obsahu seznámení, bylo informačně na úrovni popisu aplikace v Appstore, ve kterém se uživatel chce dozvědět, co za aplikaci si stahuje.

2. Uživatel si spustil aplikaci a společně jsme procházeli vybranými částmi aplikace s tím, že jsem uživateli pokládat otázky ke konkrétním prvkům uživatelského rozhraní. Po vyslovení uživatelova názoru, jsem uživateli řekl, zda odpověděl správně nebo špatně a popřípadě mu řekl správnou odpověď. Během této části testování jsem si u každé otázky značil správnost odpovědi. Odpovědi měly odhalit chyby v navrženém uživatelském rozhraní.

3. Po seznámení uživatele s aplikací přišli na řadu praktické úkoly. Zadání úkolů bylo bez podrobností, zadány byly pouze konečné požadavky. Jednalo o jednoduché úkoly, se kterými by se uživatel při používání aplikace denně setkával. Při každém úkolu jsem si značil chybné cesty uživatele a konkrétní prvky, na kterých k nim došlo. Dále jsem se zaměřil na sledování času, který zabralo řešení úkolu. Naměřené časy jsem při vyhodnocení testu porovnával s naměřenými časy z testu B.

4. Na závěr testování jsem se uživatele ještě zeptal na zpětnou vazbu, jaký měl z používání aplikace pocit a co by v aplikaci udělal jinak. Tyto informace budu moci využít při dalším vývoji aplikace.

Průběh testu B:

1. Nejprve byl uživatel v krátkosti seznámen s aplikací, s tím co je jejím cílem a co by měla dělat. Co se týče obsahu seznámení, bylo informačně na úrovni popisu aplikace v Appstore, ve kterém se uživatel chce dozvědět, co za aplikaci si stahuje.

2. Po seznámení uživatele s aplikací jsem rovnou zadával praktické úkoly. Jednalo se o stejné úkoly jako v praktické části testu A aby bylo možné porovnat výsledky obou testů. Při každém úkolu jsem si opět značil chybné cesty uživatele a konkrétní prvky, na kterých k nim došlo. Dále jsem se zaměřil na sledování času, který zabralo řešení úkolu. Naměřené časy jsem při vyhodnocení testu porovnával s naměřenými časy z testu A.

3. Na závěr testování jsem se uživatele ještě zeptal na zpětnou vazbu, jaký měl z používání aplikace pocit a co by v aplikaci udělal jinak. Tyto informace budu moci využít při dalším vývoji aplikace.

3 Realizace

Následující kapitola se věnuje vlastní realizaci projektu. Nejprve se zaměřím na popis implementace aplikace, ve kterém rozeberu její architekturu a podstatné implementační detaily. Poté se zaměřím na popis testování aplikace a vyhodnocení testů. V závěru kapitoly nastíním možnosti dalšího rozvíjení aplikace.

3.1 Popis implementace aplikace

Aplikace je implementována v nativním jazyce Objective-C. Vývoj probíhal ve vývojovém prostředí Xcode za využití nástroje Storyboard, který lze popsat jako vizuální editor, sloužící pro stavbu a propojení uživatelského rozhraní a ovládacích prvků. Aplikace je vystavěna nad základními frameworky UIKit, Foundation a CoreGraphics, které poskytují dostatečné aplikační rozhraní pro práci s daty a standardními ovládacími prvky [1]. Kromě těchto základních frameworků využívá aplikace ještě frameworku CoreMotion, nad kterým je implementován krokoměr aplikace.

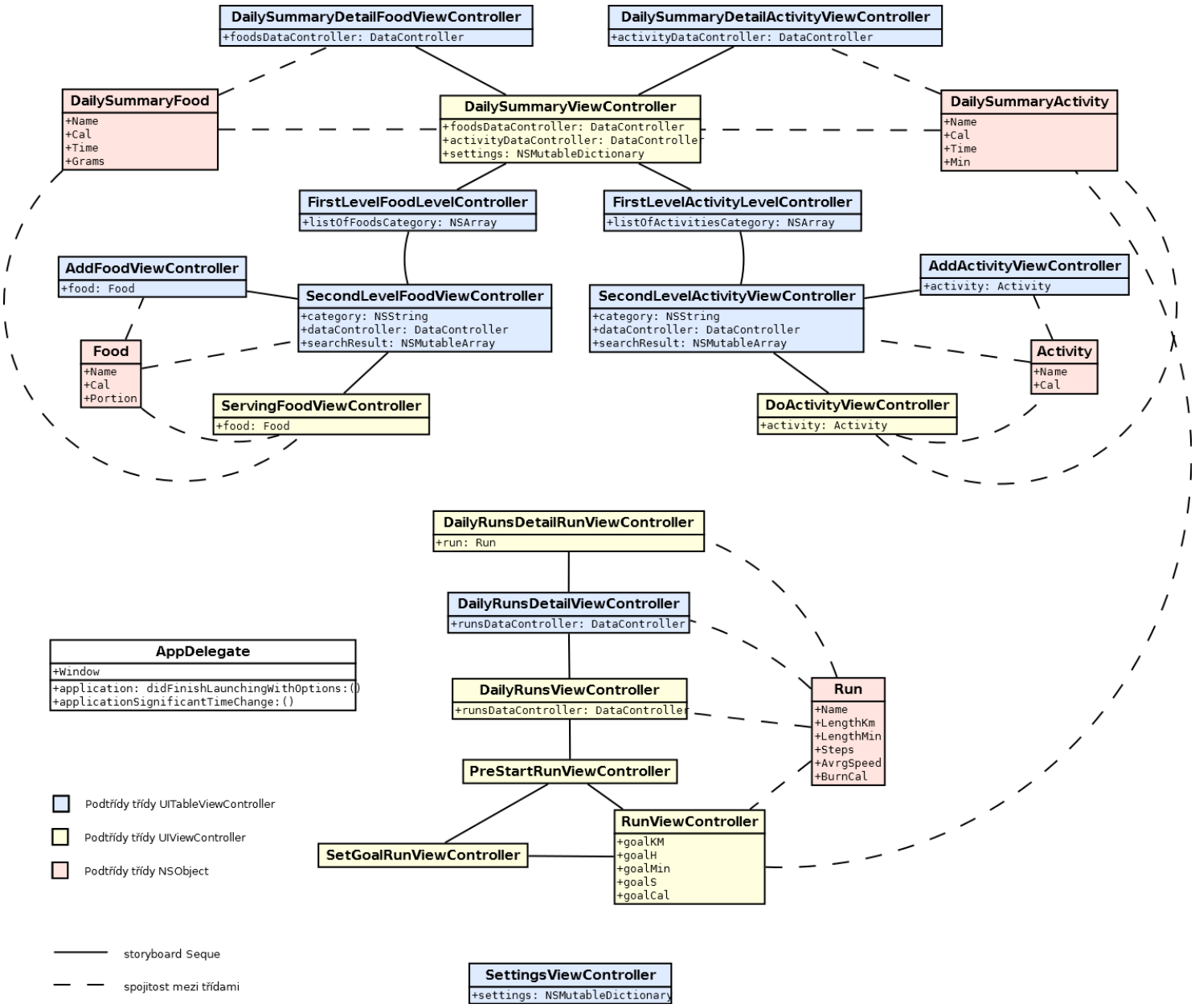
Architektura aplikace

Kostrou aplikace je soustava podtříd třídy *UIViewController* nebo *UITableViewController*, které reprezentují vlastní obrazovky aplikace a starají se o ovládání a zobrazování dat a prvků uživatelského rozhraní. Protože jsem aplikaci vyvíjel za pomocí Storyboard, jsou tyto třídy propojeny pomocí *UIStoryboardSeque*, přes které mezi sebou komunikují a předávají si důležité parametry.

Jednotlivé objekty, se kterými se v aplikaci pracuje, jako jsou jídla, aktivity a běhy, jsou v aplikaci reprezentovány jako instance jim odpovídajících tříd *Food, Activity* a *Run*, popřípadě *DailySummaryFood* a *DailySummaryActivity*. Kde *Food* a *Activity* představují jídla a aktivity v hierarchických tabulkách a nesou informace podstatné pro zadávání jídel a aktivit do denního souhrnu. A *DailySummaryFood* a *DailySummaryActivity* představují jídla a aktivity vykonané během dne a nesou informaci o vlastních porcích jídel či aktivit. O práci se seznamy se zmíněnými objekty v rámci aplikace se stará třída *DataController*.

Načítání aplikace a komunikaci s operačním systémem obstarává třída *AppDelegate*, která spravuje singletonový objekt *UIAplication* reprezentující celou aplikaci. V této třídě je implementováno vytvoření perzistentních souborů pro ukládání dat po prvním spuštění aplikace a také přepisování souborů obsahující denní souhrny při každé změně data. Toto přepisování je implementováno v metodě *applicationSignificantTimeChange:* která je volána vždy jako reakce na systémové hlášení o změně data.

Celou architekturu aplikace demonstruji na diagramu tříd na obr. 8.



Obr. 8 Diagram tříd celé aplikace

Způsob perzistentního uložení dat

Jednou ze stěžejních částí aplikace je způsob uložení dat. Co se týče perzistentního uložení dat v iOS aplikacích mezi nejběžnější způsoby patří využití seznamů vlastností, archivování objektů, SQLite či nástroje Core Data. Všechny čtyři způsoby trvalého uchování dat sdílí jednu důležitou vlastnost a to, že pracují se složkou *Documents* vlastní aplikace. Každá aplikace totiž může načítat a zapisovat data pouze do této složky [3].

Jelikož se v mé aplikaci nepracuje se složitými modely dat a nad daty není potřeba vytvářet relace a různé typy agregací, zvolil jsem implementaci perzistentního uložení dat pomocí seznamu vlastností a archivů objektů. Konkrétně seznamem vlastností implementuji uložení nastavení aplikace a archivování využívám při ukládání vlastních seznamů objektů, jako například seznamů jídel, aktivit či seznamů s denními souhrny.

Co se týče strategie ukládání dat do souboru, zvolil jsem způsob uložení dat ve více souborech, kdy každý soubor reprezentuje příslušný seznam. Tento přístup umožňuje načítat pouze data vyžádaná uživatelem, což vede k šetření paměti.

Seznam vlastností

Seznamy vlastností jsou praktické, protože je lze upravovat manuálně v prostředí Xcode. Pokud obsahují pouze specifické serializované objekty, lze do nich také ukládat instance tříd *NSDictionary* a *NSArray*. Serializovaný objekt je objekt konvertovaný na tok bajtů. Ačkoliv lze pro serializaci připravit kterýkoliv objekt, do kontejnerů jakými jsou objekty tříd *NSDictionary* nebo *NSArray* lze umístit jen některé z nich. A to konkrétně objekty následujících tříd: *NSArray*, *NSMutableArray*, *NSDictionary*, *NSMutableDictionary*, *NSData*, *NSMutableData*, *NSString*, *NSMutableString*, *NSNumber* a *NSDate*. Vlastní kontejner obsahující správné objekty se do souboru zapíše pomocí metody *writeToFile:atomically*. Nevýhodou využití seznamů vlastností je, že do nich nelze serializovat vlastnoručně navržené objekty [3].

Archivování objektů

Technika archivování objektů modelu umožňuje jednoduše zapisovat komplexní objekty do souborů a opět je z nich načítat. Jedná se o obecnější způsob serializace dat, kterým můžeme serializovat i vlastnoručně navržené objekty. Pokud je každý atribut, který třída nebo objekt obsahuje skalární veličina nebo instance třídy, které odpovídá protokolu *NSCoding*, může být objekt kompletně archivován.

Protokol *NSCoding* deklaruje dvě metody, které jsou v implementaci archivované třídy obě povinné. Jedna z nich zapisuje objekt do archivu, druhá z archivu načíta a vytvoří nový objekt. Oběma metodám se předá instance třídy *NSCoder*, která zapisuje a načítá z archivu mechanismem klíč-hodnota.

Vytvoření archívu z objektů odpovídajících protokolu *NSCoding*, je relativně jednoduché. Nejprve se vytvoří instance třídy *NSData*, která bude uchovávat „zakódovaná“ data, a potom instance třídy *NSKeyedArchiver* (jež je podtřídou *NSCoder*), která bude objekty do instance třídy *NSData* archivovat. Při obnově objektů z archivů se postupuje obdobně, pouze namísto s třídou *NSKeyedArchiver* se pracuje se třídou *NSKeyedUnarchiver*, pomocí které se načítají objekty z archivovaných dat třídy *NSData* [3].

Implementace krokoměru

Jak již bylo mnohokrát řečeno, krokoměr jsem se rozhodl implementovat nad akcelerometrem telefonu. Součástí iPhone je tříosý akcelerometr, který dokáže detekovat pohyb nebo působení gravitační síly v třídimenzionálním prostoru. Akcelerometr měří zrychlení, takže když vrátí hodnotu 1.0, znamená to, že v určitém směru detekoval zrychlení 1 g.

Co se týče přístupu k datům akcelerometru, Apple poskytuje framework Core Motion. Core Motion framework je primárně zodpovědný za přístup k syrovým datům akcelerometru a gyroskopu telefonu, avšak navíc využívá jedinečných algoritmů pro zpracování těchto syrových dat a umožňuje tak vývojáři přístup k mnohem jemnějším informacím. Toto zpracování probíhá v samostatném vlákně frameworku. Přístup k datům poté funguje jednoduše tak, že Core Motion doručuje události o pohybu přímo do aplikace, která o ně zažádá [13].

Centrálním přístupovým bodem frameworku Core Motion je třída *CMMotionManager*. V aplikaci by měla být vytvořena pouze jedna instance této třídy. Po vytváření instance třídy *CMMotionManager* se upřesní interval aktualizace, a poté už jen stačí spustit zasílaní dat s příslušným řízením dodání každé události o pohybu.

Core Motion události o pohybu jsou reprezentovány třemi datovými objekty, z nich každý zapouzdřuje jedno či více měření.

- Objekt *CMAccelerometerData* zachycuje zrychlení podél každé z prostorových os.
- Objekt *CMGyroData* zachycuje rychlost otáčení kolem každé ze tří prostorových os.
- Objekt *CMDeviceMotion* zapouzdřuje několik různých měření, převážně se jedná o, určitým způsobem, zpracované či filtrované měření rychlosti otáčení a zrychlení, které umožňuje přístup k detailnějším informacím o měření [14].

V rámci své aplikace využívám datových objektů *CMDeviceMotion* a *CMAccelerometerData*. *CMDeviceMotion* využívám k měření polohy telefonu, pro určení osy, na které bude aplikace měřit otřesy. *CMAccelerometerData* využívám k vlastnímu snímání zrychlení z příslušné osy akcelerometru a následné detekci kroků.

Měření kroků

S každou přijatou aktualizací nad daty *CMAccelerometerData* se provádí vyhodnocení naměřeného zrychlení na vybrané ose, které určí, zda byl proveden krok. Před vlastním vyhodnocením je naměřené zrychlení upraveno dolní propustí, aby nedocházelo ke zkreslení vlivem diskrétního snímání spojitého signálu.

Vyhodnocení probíhá na základě porovnání naměřeného zrychlení z hodnotou *tempSens*, reprezentující aktuální citlivost krokoměru a sledování aktuálních hodnot boolovských proměnných *step* a *miss*. Proměnná *step* slouží k identifikaci, že sejmutá hodnota zrychlení spadá stále ještě do

provádění stejného kroku jako předchozí sejmutá hodnota. Proměnná se aktivuje, jakmile aktuální hodnota zrychlení je vyšší než hodnota *tempSens* a nuluje, jakmile aktuální hodnota zrychlení klesne pod hodnotu *minSens*, reprezentující neměnnou minimální citlivost krokoměru. Při každém nulování proměnné *step*, dochází k inkrementaci počtu kroků a k upravení hodnoty *tempSens*. Boolovská proměnna *miss* pracuje na podobném principu jako proměnná *step*, ale s tím rozdílem, že se aktivuje, již když aktuální hodnota zrychlení vzroste nad hodnotu *minSens*. Tato proměnná slouží k úpravě *tempSens* směrem dolů, *tempSens* je upravena, pokud aktuální hodnota zrychlení klesne pod hodnotu *minSens* ale v předchozím průběhu nepřekonala hodnotu *tempSens*. Zároveň proměnná *miss* slouží k připočtení prvního kroku po „minutí" hodnoty *tempSens*, před jejím vlastním upravením. Pro lepší pochopení algoritmu uvedu jeho zjednodušený kód.

```
Hodnota aktuálního zrychlení překročila hodnotu tempSens, to značí začátek kroku.
if (g  >  tempSens && !step){   //g = aktuální hodnota zrychlení
      step = YES;
      miss = NO:
      missCount = 0;
}
Hodnota aktuálního zrychlení překročila hodnotu minSens, pokud nedojde k
překročení hodnoty tempSens dojde k "minutí" a bude potřeba upravit hodnotu tempSens.
if ( g > minSens && g < tempSens && !step){
      miss = YES;
}
Hodnota aktuálního zrychlení se snížila pod hodnotu minSens, to značí konec kroku nebo "minutí".
if(g < minSens && ( step || miss)){
      if( miss ) missCount++;
      if( step || ( miss && missCount < 2 )){
            //inkrementace počtu kroků ...
      }
      step = NO;
      miss = NO;
      //přepočet nové hodnoty tempSens ...
}
```

Komentovaný pseudokód demonstrující algoritmus detekce kroků

Měření vzdálenosti

S každým naměřeným krokem se k dosavadní vzdálenosti připočte vybraná délka kroku. Délka kroků je vybrána na základě maximální intenzity otřesu (hodnoty zrychlení) naměřené v celém kroku. Je na výběr ze tří délek kroku, konkrétně délka kroku při chůzi, délka kroku při běhu a délka kroku při sprintu, jejichž hodnoty je možné nastavit v nastavení aplikace.

Měření času

Měření času je v aplikaci implementováno pomocí instance třídy *NSTimer*, která ve smyčce v intervalech po jedné vteřině počítá a zobrazuje časový interval od spuštění běhu.

Měření počtu spálených kalorií

S každým naměřeným krokem je k dosavadní hodnotě počtu spálených kalorií připočtena hodnota počtu spálených kalorií za jeden krok při běhu určitou rychlostí, odpovídající aktuální průměrné rychlosti.

$$Počet\ kroků\ za\ minutu\ = \frac{Vzdálenost\ uběhlá\ za\ minutu}{Délka\ kroku}$$

$$Počet\ spálených\ kalorií\ za\ jeden\ krok = \frac{Počet\ kalorií\ spálených\ za\ minutu}{Počet\ kroků\ za\ minutu}$$

**Při výpočtech se počítá s hodnotami odpovídajícími konkrétní rychlosti běhu (počet kroků za minutu při běhu určitou rychlostí, vzdálenost uběhlá za minutu při běhu určitou rychlostí, délka kroku při běhu určitou rychlostí, počet kalorií spálených za minutu při běhu určitou rychlostí a počet spálených kalorií za jeden krok při běhu určitou rychlostí).*

Odvození počtu kalorií spálených v jednom kroku ze známých hodnot počtu kalorií spálených za minutu z kalorických tabulek

Měření průměrné rychlosti

Průměrná rychlost je počítána z aktuální naměřené vzdálenosti a času běhu. Hodnota je aktualizována každou vteřinu.

Vzorec pro výpočet průměrné rychlosti:

$$Průměrná\ rychlost = \frac{Aktuální\ vzdálenost}{Aktuální\ čas}$$

Anotace dat

Aktuální prototyp aplikace obsahuje základní vzorek dat, dostačující pro otestování uživatelského rozhraní. Každé jídlo obsahuje informaci o jeho názvu, počtu kalorií v jednom gramu a gramáži průměrné porce. Každá aktivita obsahuje informaci o jejím názvu a počtu kalorií spálených při výkonu aktivity za jednu minutu na kilogram uživatelovi váhy. Ještě před možným uvedením aplikace na trh se počítá s rozšiřováním databáze.

3.2 Vyhodnocení testování

V následující kapitole uvádím výsledky testování aplikace. Vlastní otestování se skládá ze dvou nezávislých částí. Z testování krokoměru a testování uživatelského rozhraní.

Testování krokoměru

Cílem testování krokoměru bylo zjistit přesnost naměřených hodnot a vliv způsobu upevnění telefonu na měření. Postup testování byl popsán v kapitole 2.5 Návrh testování. V následující kapitole se zaměřím na popis naměřených hodnot a jejich vyhodnocení.

Měření přesnosti krokoměru s telefonem umístěným v kapse

V rámci testování jsem provedl několik sérií měření chůze a běhu na předem změřeném půlkilometrovém úseku s telefonem umístěným v kapse.

Reprezentativní vzorek dat při měření chůze:

	Počet kroků	Vzdálenost	Průměrná rychlost	Spálené kalorie
Krokoměr	540	415	4.4	26.64
Pedometer Free GPS+	559	512	5.1	28.00
Reálné hodnoty	550	500	4.98	--

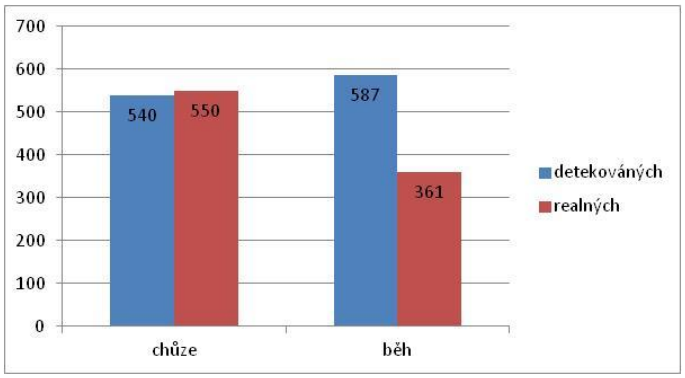
Reprezentativní vzorek dat při měření běhu:

	Počet kroků	Vzdálenost	Průměrná rychlost	Spálené kalorie
Krokoměr	587	451	9.5	53.39
Pedometer Free GPS+	342	475	8.5	35.7
Reálné hodnoty	361	500	8.95	--

Vyhodnocení testu:

Z naměřených hodnot vyplývá, že nestabilní poloha umístění telefonu má velký vliv na přesnost měření. Tento jev lze vysvětlit tím, že se telefon při běhu kromě pohybu kopírujícího pohyb uživatele ještě pohybuje sám v kapse, čímž dochází k detekování neexistujících kroků. Tento jev bohužel nelze odstranit pouhou změnou algoritmu pracujícího nad akcelerometrem telefonu, protože akcelerometr nedokáže rozlišit čím je pohyb telefonu způsoben. Jedním z možných řešení tohoto problému je korekce naměřených dat za pomocí dat naměřených z GPS.

Při chůzi, kdy otřesy telefonu nezpůsobují pohyb vlastního telefonu v kapse, je naopak dosaženo dostatečné přesnosti detekce kroků. Odchylka oproti reálnému počtu kroků se pohybuje průměrně kolem 2% celkového počtu kroků.



Obr. 9 Graf zobrazující přesnost detekce kroků telefonem v nestabilní poloze

Měření přesnosti krokoměru s telefonem umístěným v běžeckém pouzdře

Běžecké pouzdro se upíná na paži a zajišťuje pevné uchycení mobilního telefonu k ruce běžce. Nemůže tak dojít k nežádoucímu volnému pohybu telefonu tak, jako v případě umístění telefonu v kapse. V rámci testování jsem provedl několik sérií měření chůze a běhu na předem změřeném půlkilometrovém úseku s telefonem umístěným v běžeckém pouzdře.

Reprezentativní vzorek dat při měření chůze:

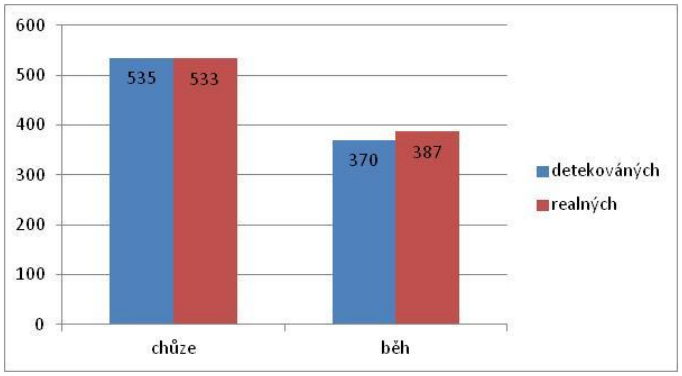
	Počet kroků	Vzdálenost	Průměrná rychlost	Spálené kalorie
Krokoměr	535	411	4.9	27.35
Pedometer Free GPS+	534	494	4.7	30.3
Reálné hodnoty	533	500	4.76	--

Reprezentativní vzorek dat při měření běhu:

	Počet kroků	Vzdálenost	Průměrná rychlost	Spálené kalorie
Krokoměr	370	284	6.5	30.53
Pedometer Free GPS+	334	520	8.2	45.3
Reálné hodnoty	387	500	7.88	--

Vyhodnocení testu:

Z naměřených hodnot vyplývá, že při stabilním umístění telefonu dosahuje algoritmus detekce kroků uspokojivé přesnosti. Konkrétně při chůzi se dokonce průměrná odchylka smrštila do 1% celkového počtu kroků a při běhu byla průměrná odchylka kolem 5% celkového počtu kroků. Testování odhalilo nepřesnost v měření vzdálenosti, od které se odvíjí přesnost hodnot na ni závislých. Jedná se o špatně zvolené hraniční hodnoty citlivostí, dle kterých se vybírá délka právě přičítaného kroku. V dalším vývoji aplikace je třeba se na tento problém zaměřit a za pomocí testování určit vhodné hraniční hodnoty citlivosti.



Obr. 9 Graf zobrazující přesnost detekce kroků telefonem ve stabilní poloze

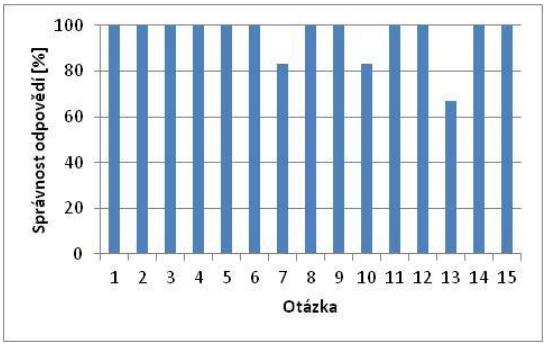
Testování uživatelského rozhraní

Cílem testování uživatelského rozhraní bylo odhalit logické chyby v jeho návrhu, zjistit jak je intuitivní a zda jsou navrhnuté postupy ovládání aplikace dostatečně osvojitelné a zapamatovatelné. Testování je rozděleno do dvou testů, na komplexní test A a praktický test B. Testování bylo provedeno na 12 uživatelích, z nichž 6 provedlo test A a 6 provedlo test B. Průběh obou testu je popsán v kapitole 2.5 Návrh testování, tudíž se v následující kapitole zaměřím spíše na výsledky a vyhodnocení testů.

Komplexní test A

Tento test se skládal z otázkové a praktické části. Cílem otázkové části bylo zjistit, jak jsou jednotlivé prvky uživatelského rozhraní intuitivní. Praktická část sledovala osvojení významu jednotlivých prvků a jejich praktické použití. Otázková část obsahovala 15 otázek ke konkrétním prvkům uživatelského rozhraní. Praktická část obsahovala 8 úkolů. Přesné znění otázek i úkolů se nachází v příloze 1 Protokoly testování.

Vyhodnocení testování otázkami v rámci testu A

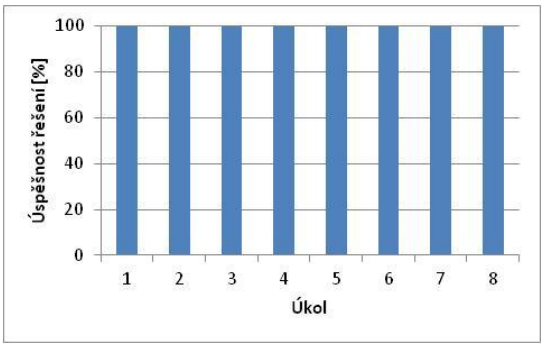


Obr. 10 Graf znázorňující správnost odpovědí

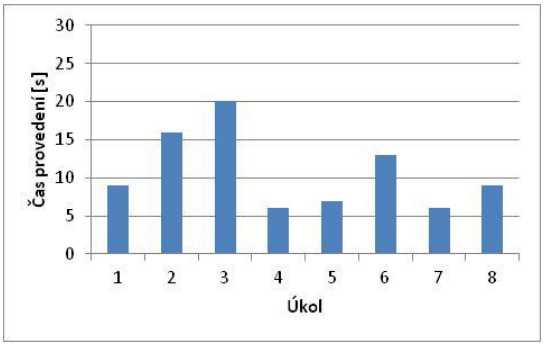
Z výsledků otázkové části vyplývá, že většina prvků uživatelského rozhraní je intuitivní a rozpoznání jejich funkčnosti nečiní uživatelům problémy. Mírné problémy činila uživatelům otázka číslo 13. *„Kterým tlačítkem na obrazovce ukončíte běh?“* Kde dva uživatelé vlivem nepozornosti a špatně zvoleného popisku na tlačítku sloužícímu k pozastavení běhu, volili na místo tlačítka *„Ukončit běh“* tlačítko *„Stop“*. To mě vedlo k opravení chyby a přejmenování tlačítka *„Stop“* na *„Pauza“*. Chyby na otázkách číslo 7 *„K čemu slouží posuvník pod gramáží porce při zadávání porce jídla?“* a číslo 10 *„Co se skrývá za tlačítkem běhy na obrazovce s denním souhrnem běhů?“* byly způsobeny spíše nepozorností uživatelů a v rámci praktických úkolů na zmíněných prvcích nedošlo k zaváhání.

Vyhodnocení praktických úkolů v rámci testu A

Z výsledků praktických úkolů plněných v průběhu komplexního testu A vyplývá, že osvojení významu jednotlivých prvků nečiní uživatelům problémy. Důležitým poznatkem je, že i uživatelé, kteří v předchozí části testu odpověděli na některé otázky špatně, v následném praktickém testu inkriminované prvky použili správně. Co se týče času stráveného nad úkoly, byli jednotlivé časy dosti individuální dle zkušeností uživatelů s mobilními aplikacemi. Z průměrných časů ale vyplývá, že krátké seznámení uživatele s aplikací vede k rapidnímu zvýšení rychlosti plnění běžných úkonů s aplikací a to značí, že jednotlivé navrhnuté ovládací postupy jsou snadno pochopitelné a naučitelné.



Obr. 11 Graf znázorňující úspěšnost řešení praktických úkolů v rámci testu A



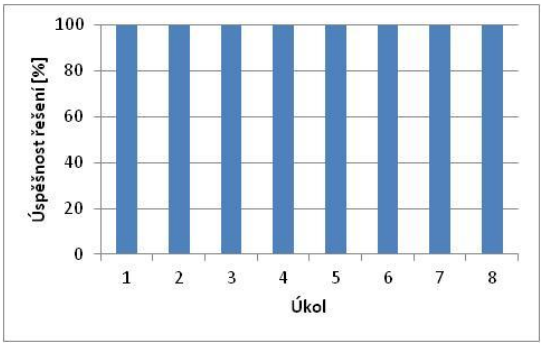
Obr. 12 Graf znázorňující průměrnou délku časů provedení praktických úkolů v rámci testu A

Praktický test B

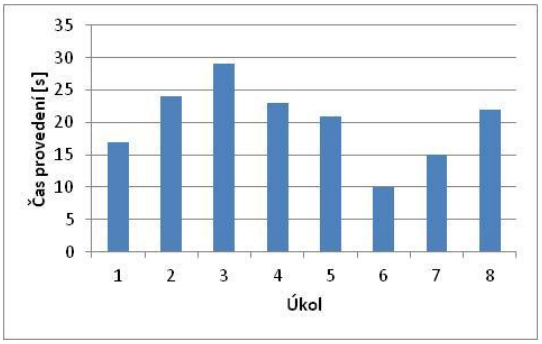
Test B se skládá z 8 úkolů. Jedná se o stejné a stejně zadané úkoly jako v praktické části testu A, ale oproti testu A je plní uživatele, kteří s aplikací ještě nejsou seznámení a neznají její ovládací postupy. Cílem testu je zjistit, jak jsou navrhnuté ovládací postupy intuitivní a zda uspořádání ovládacích prvků neobsahuje logické chyby, které by uživatele mátly.

Vyhodnocení praktických úkolů v rámci testu B

Z úspěšnosti řešení úkolů plyne, že navržené ovládací postupy jsou dostatečně intuitivní na to, aby je uživatel zvládl pochopit sám. Co se týče časů, byly opět individuální dle zkušeností jednotlivých uživatelů. Průměrné časy ovlivnili zkušení uživatele iPhone, kteří si s některými úkoly, ve kterých se dalo využít zažitých ovládacích postupů z většiny aplikací na iPhone, jako například mazání položek v tabulce pomocí gesta swipe, poradili rychleji i než někteří uživatelé z testu A. Na druhou stranu na prodloužení průměrných časů měla vliv určitá nervozita testovaných uživatelů a čas potřebný pro zorientování se v nové aplikaci. Z výsledků testu lze také vypozorovat, že se doba provádění úkolů snižuje v závislosti na pochopení principu ovládání z předchozích podobných úkolů. To lze pozorovat například na úkolech 5 a 6, kdy v pátém úkolu má uživatel spustit měření volného běhu a v šestém úkolu má spustit měření běhu se zadaným určitým cílem. I když je šestý úkol o něco složitější, vlivem pochopení, jak se v aplikaci dostat k měření běhu je vykonán v o polovinu rychlejším průměrném čase. V rámci testu B jsem sledoval i to, jak si uživatelé v praxi poradí s prvky, které někteří účastnici testu A určili špatně. V rámci testu B všichni účastnící použili inkriminované prvky správně v kontextu s řešenou úlohou.



Obr. 13 Graf znázorňující úspěšnost řešení praktických úkolů v rámci testu B



Obr. 14 Graf znázorňující průměrnou délku časů provedení praktických úkolů v rámci testu B

3.3 Možnosti dalšího vývoje

V následující kapitole se pokusím popsat možnosti rozšíření aplikace a cesty, kterými se v budoucnu může projekt ubírat. Nejprve se zaměřím na rozšíření aktuální verze prototypu aplikace, která jsou nutná před vlastním nasazení aplikace na Appstore a poté se pokusím nastínit možnosti dalšího rozvoje celého projektu.

Jelikož Appstore nabízí snadnou distribuci aplikace po celém světě, je před vlastním nasazení aplikace na trh podstatné rozšířit aktuální prototyp o možnosti lokalizace celé aplikace do všech významných jazyků. Dalším nutným rozšířením je rozšíření aktuální skromné databáze jídel a aktivit. Vzhledem k tomu, že na Appstore aplikaci nejvíce prodávají screenshoty, bylo by také dobré vytvořit k aplikaci propracovanější grafiku, která bude obsahovat určité signifikantní znaky, které budou spojeny s aplikací jako značkou.

Co se týče budoucího rozvoje projektu, bylo by vhodné vytvoření druhé rozšířené verze aplikace. Aktuální typ aplikace bude sloužit jako určitá free trial verze schopná pracovat offline a ve které bude hlavní důraz kladen na jednoduchost použití. K ní bude vytvořena placená aplikace rozšířená o určité prémiové funkce, jako je například ukládání si denních výsledků do svého profilu, možnosti interakce se sociálními sítěmi, ale třeba i rozšíření krokoměru o GPS funkce, jako je zobrazení mapy, či mapování převýšení. Tato kombinace dvou aplikací vede k oslovení širšího okruhu uživatelů a dává uživatelům možnost vybrat si aplikaci, která jim bude vyhovovat. Navíc pokud bezplatná trial verze uživatelé osloví, může uživatele přesvědčit ke koupi prémium verze, což vede k větším ziskům.

4 Závěr

Cílem bakalářské práce bylo navrhnout a implementovat prototyp aplikace na iPhone z oblasti zdraví a fitness. Před vlastním návrhem aplikace bylo potřeba nastudovat specifika platformy iOS a vytvořit si dostatečnou informační základnu ohledně metabolismu lidského těla. Jelikož stěžejní částí aplikace mělo být jednoduché a intuitivní uživatelské rozhraní, bylo důležitou součásti práce i studium návrhu uživatelského rozhraní.

Z načerpaných vědomosti se podařilo vytvořit prvotní návrh aplikace, který byl po uživatelském testování za pomocí náčrtku na papír upraven do verze popsané v kapitole 2. Návrh, tedy aplikace skládající se ze dvou částí, a to z části mapující denní příjem a výdej energie a části fungující jako krokoměr. Před vlastní implementací aplikace bylo důležité prostudovat existující frameworky poskytované firmou Apple a vybrat vhodné, nad kterými aplikace implementovat. Vlastní aplikace je implementována nad základními frameworky UIKit, Foundation a CoreGraphics, které jsou ještě rozšířeny o framework CoreMotion, nad kterým je implementován krokoměr aplikace. Vývoj algoritmu krokoměru, do podoby v jaké se teď nachází, byl iterativní proces, ve kterém se opakoval návrh, implementace a testování při kterém se měřila přesnost krokoměru.

Testováním aplikace bylo prokázáno, že při stabilním upevnění telefonu, dosahuje krokoměr, co se detekce kroků týče, uspokojivých výsledků. Testování odhalilo nepřesnost v měření vzdálenosti, na které je třeba se v dalším vývoji aplikace zaměřit. Z testování dále vyplynulo, že navržené uživatelské rozhraní vyhovuje požadavkům na intuitivnost a jednoduchost použití aplikace.

Výsledkem práce je funkční prototyp aplikace sloužící k mapování příjmu a výdeje energie uživatele a měření jeho pohybových aktivit (chůze a běhu) v průběhu jednoho dne.

Literatura

[1] Apple Inc, iOS Technology Overview, online, 2012.
URL:http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf

[2] Fraser Speirs, How iOS multitasking really works, online, 2012.
URL: http://www.macworld.com/article/1164616/how_ios_multitasking_really_works.html

[3] Dave Mark, Jeff LaMarche, iPhone SDK, Computer press, a.s., Brno, 2010.

[4] Apple Inc, Transitioning to ARC Release Notes, online, 2012.
URL: http://developer.apple.com/library/ios/releasenotes/ObjectiveC/RN-TransitioningToARC/RN-TransitioningToARC.pdf

[5] Apple Inc, iPhone App Programming Guide, online, 2012.
URL: https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/iPhoneAppProgrammingGuide.pdf

[6] Apple Inc, iOS Human Interface Guidelines, online, 2011.
URL: http://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf

[7] Apple Inc, Streamline Your App with Design Patterns, online, 2012.
URL: https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/StreamlineYourAppswithDesignPatterns/StreamlineYourApps/StreamlineYourApps.html

[8] Jiří Komár, User Experience, online, 2011.
URL: http://www.seo-slovnik.cz/slovo/user-experience/

[9] Martin Malý, User Experience pro vývojáře, online, 2011.
URL: http://www.zdrojak.cz/clanky/user-experience-pro-vyvojare/

[10] Radek Mirovský, Metabolismus, výdej a příjem energie, online, 2007.
URL: http://www.bud-fit.cz/zdrava-vyziva/metabolismus,-vydej-a-prijem-energie/

[11] Bazální metabolický výdej [online], poslední aktualizace 14.11.2005 02:47,
URL:http://cs.wikipedia.org/wiki/Baz%C3%A1ln%C3%AD_metabolick%C3%BD_v%C3%BDdej

[12] Adam Fendrych, Uživatelské testování návrhů webu, online, 2009.
URL: http://www.lupa.cz/clanky/uzivatelske-testovani-navrhu-webu/

[13] Apple Inc, Event Handling Guide for iOS, online, 2013.
URL: http://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/EventHandlingiPhoneOS.pdf

[14] Apple Inc, Core Motion Framework Reference, online, 2011.
URL: http://developer.apple.com/library/IOS/documentation/CoreMotion/Reference/CoreMotion_Reference/CoreMotion_Reference.pdf

Seznam příloh

Příloha 1

Protokol testování - komplexní test A

1. Otázková část

1) Identifikujte stisknutelné tlačítka na obrazovce „Kalorický souhrn“.

Správně Špatně

2) Co se skrývá za tlačítkem „detail“ na obrazovce „Kalorický souhrn“?

Správně Špatně

3) Co se skrývá za tlačítkem „+“ na obrazovce „Kalorický souhrn“?

Správně Špatně

4) Co znamená „Odchylka“ na obrazovce „Kalorický souhrn“?

Správně Špatně

5) Pomocí kterého ovládacího prvku se přidá jídlo do existující kategorie jídel?

Správně Špatně

6) Jak se uživatel přepne mezi módy zadávání jídel?

Správně Špatně

7) K čemu slouží posuvník pod gramáží porce na obrazovce se zadáváním porce jídla?

Správně Špatně

8) Jak uložíš zadanou porci do denního souhrnu?

Správně Špatně

9) Jak odstarníš jídlo z detailního výpisu denního souhrnu?

Správně Špatně

10) Co se skrývá za tlačítkem „běhy“ na obrazovce „Souhrn běhů“?

Správně Špatně

11) Co se skrývá za tlačítkem „nový běh“ na obrazovce „Souhrn běhů“?

Správně Špatně

12) Kterým tlačítkem se spustí měření běhu?

Správně Špatně

13) Kterým tlačítkem se ukončí měření běhu a běh se uloží do denních souhrnu?

Správně Špatně

14) Jak zobrazíš detailní statistiky běhu na obrazovce „Detailní výpis běhů“?

Správně Špatně

15) Jak se dostaneš do nastavení aplikace?

Správně Špatně

2. Praktická část

1) Na snídani si snědl banán, zaznamenej jej do aplikace.

Správně Špatně Čas:

2) Na oběd sis dal 300 gramový steak z hovězí roštěné, zaznamenej jej do aplikace.

Správně Špatně Čas:

3) Po obědě sis šel 30 minut zahrát golf, zaznamenej to do aplikace.

Správně Špatně Čas:

4) Smaž z aplikace zaznamenaný banán.

Správně Špatně Čas:

5) Vydal si se jen tak si zaběhat, spusť si měření běhu.

Správně Špatně Čas:

6) Rozhodl jsi se, že chceš uběhnout minimálně 1 kilometr. Spusť si běh s cílem 1 kilometr.

Správně Špatně Čas:

7) Chtěl by si aplikaci užívat s cílem zhubnout, a to konkrétně z 40% omezením přijmu potravy a z 60% zvýšením tělesných aktivit. Proveď potřebné nastavení.

Správně Špatně Čas:

8) V aplikaci chybí tvůj oblíbený pokrm, pokus se jej do aplikace přidat.

Správně Špatně Čas:

Protokol testování – praktický test B

Viz Praktická část komplexního testu A

Příloha 2

Obsah přiloženého CD

- Plakát
- Video
- Zdrojové kódy aplikace
- Technická zpráva