VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

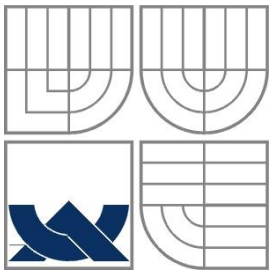
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SPORTOVNÍ ASISTENT PRO PLATFORMU IOS

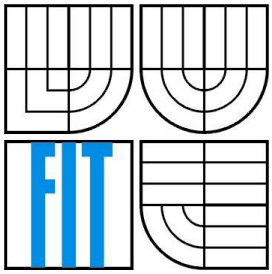
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Jakub Fišer
AUTHOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SPORTOVNÍ ASISTENT PRO PLATFORMU IOS

SPORT ASSISTANT FOR IOS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jakub Fišer
AUTHOR

VEDOUCÍ PRÁCE — Ing. Martin Hrubý, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Cílem této práce bylo vytvořit mobilní aplikaci pro platformu iOS, která by sloužila k zaznamenávání veškeré sportovní činnosti člověka. Aplikace umí zaznamenat pohyb člověka pomocí GPS nebo počítat jeho cviky manuálně, v časovém intervalu, či snímáním pohybu. Pro zpracování dat při snímání pohybu byla vytvořena knihovna Motion Recognizer. Aplikace využívá modul HealthKit. Aplikace také nabízí možnost plánování sportovní činnosti využitím trenéra. Pro práci s daty využívá aplikace modul Core Data a data zálohuje pomocí služby iCloud.

Abstract

The aim of this thesis was to create a mobile phone application for iOS, that would be used for recording all sport activities of a person. The application can record activity using GPS or count the exercise manually, automatically at time intervals or by motion sensor. For recognition of moves while using motion sensor was developed a library called Motion Recognizer. The application uses HealthKit framework. The application also offers an opportunity of planning a sport activity by using Trainer objects. It uses Core Data framework while working with data and it backs up the data on iCloud.

Klíčová slova

iOS, Apple, iPhone, sport, asistent, aplikace, GPS, HealthKit.

Keywords

iOS, Apple, iPhone, sport, assistant, application, GPS, HealthKit

Citace

Fišer Jakub: Sportovní asistent pro platformu iOS, bakalářská práce, Brno, FIT VUT v Brně, 2015

Sportovní asistent pro platformu iOS

Prohlášení

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

……
Jakub Fišer
19. května 2015

Poděkování

Chtěl bych poděkovat svému vedoucímu panu Ing. Martinu Hrubému, Ph.D. za vedení při řešení této práce.

Obsah

1 Úvod

Dříve patřila fyzická námaha mezi běžnou denní záležitost lidí. Mluvíme zde například o chůzi do schodů, pěší chůze mezi vesnicemi o vzdálenosti několika kilometrů, ruční práce a práce na polích apod. Dnes má většina lidí práci sedavého charakteru. Pohyb je ale nezbytnou součástí života a proto jej tito lidé později vyhledávají jinde ve svém volném čase. Když už nějakou fyzickou činnost vykonávají, rádi si ji měří a zaznamenávají výsledky z důvodu pozdějšího uveřejnění na sociální síti, porovnání s výsledky lidí jiných nebo sledování pokroku. Právě to se snaží umožnit mobilní aplikace, která je náplní této práce.

Aplikace nese název **SportAssistant** a je určena pro zaznamenávání všech sportovních činností uživatele a podporu ve sportovních aktivitách formou trenéra. Uživatel může zaznamenat aktivity jak exteriérové (např. běh, jízdu na kole, chůzi), tak i interiérové (např. silové cvičení). Exteriérové aktivity zaznamenávají trasu přímo vykreslovanou na mapu, čas, vzdálenost a rychlost. Interiérové sportování nabízí počítání provedených cviků, kde si uživatel může vybrat mezi manuálním počítáním pomocí dotyku obrazovky (např. dotyk nosem při provádění kliků), automatickým počítání v zadaném časovém intervalu, nebo počítání s pomocí pohybového senzoru mobilního zařízení (např. při skoku nebo dřepu). Po zaznamenání exteriérové i interiérové aktivity aplikace vypočte množství spálených kalorií v závislosti na typu cvičení a uloží tuto hodnotu do systémové databáze HealthKit. Mezi těmito dvěma typy aktivit si uživatel vybírá hned v úvodním menu, kde má na výběr ještě zobrazit historii cvičení, nebo přejít do sekce trenérů. Trenér je objekt, který uživatele vybízí k provozování sportu formou systémových notifikací. Pokud uživatel nechce být systémovými notifikacemi vyrušován, může je vypnout, trenér jej poté ale nemůže dále upozorňovat na naplánované cvičení. V historii cvičení jsou záznamy rozdělovány do uživatelem vytvořených kategorií, kde v aplikaci vždy existuje kategorie s názvem „Default“ pro případy, kdy si uživatel svoji kategorii nevybere. Veškerá práce s daty je spravována knihovnou Core Data a data jsou zálohována na externím úložišti iCloud. Detail záznamu je možné sdílet na sociální síti Facebook.

V Kapitole 2 si nejdříve zdokumentujeme již existující a mezi uživateli oblíbené aplikace se zaměřením na sledování sportovní činnost uživatele. Dále zde budou popsány základy vývoje aplikací pro platformu iOS, kde najdeme vedle popisu, jak začít aplikace vyvíjet, i stručnou charakteristiku některých v aplikaci použitých knihoven. Také bude popsán teoretický základ trénování. Kapitola 3 obsahuje návrh komponent aplikace potřebných k její implementaci. Jedná se o datový model, u kterého najdeme detailní popis všech entit a jejich vzájemný vztah, základy uživatelského rozhraní a v rámci práce vytvořenou knihovnu Motion Recognizer. Detaily implementace podstatných částí aplikace najdeme v Kapitole 4. Zaměříme se především na již zmíněnou knihovnu Motion Recognizer a popíšeme si i práci s externí knihovnou pro vykreslování grafů. V Kapitole 5 najdeme informace týkající se testování aplikace. V závěru práce se Kapitola 6 věnuje možnému dalšímu rozšíření aplikace.

2 Rozbor použitých technik

V této kapitole si nejdříve zdokumentujeme podobné aplikace na trhu. Dále vysvětlím, co je potřeba k programování aplikací pro iOS, jak se tvoří uživatelské rozhraní aplikace, a přiblížím systémové knihovny, které byly při vývoji mé aplikace použity. Na závěr kapitoly se budeme věnovat základům trénování.

2.1 Aplikace podobného charakteru

Na trhu existuje mnoho sportovních aplikací. Největším průkopníkem v oblasti sportovních aplikací je společnost Nike, která ve spolupráci se společností Apple má svoji vlastní kategorii záznamů ve frameworku HealthKit, který bude popsán v Kapitole 2. Pro využívání aplikací je nutné přihlášení pomocí Facebook nebo speciálního Nike+ účtu, skrze který jsou data ukládána i na serveru dostupném z internetových stránek, kde si lze například zobrazit historii cvičení či statistiky. První z aplikací, **Nike+ Running**, je zaměřena pouze na exteriérové sportování. Další aplikace, **Nike+ Training Club**, nabízí trénink. Uživatel má možnost okamžitého cvičení podle plánů, nebo si vybrat cvičební plán na několik týdnů, kde má možnost plán v malé míře upravovat. K zobrazení videí ke cvikům je potřeba internetové připojení. Poslední aplikací je **Nike+ Fuel**, která zpětně shromažďuje informace o všech aktivitách z náramku Nike+ FuelBand a zobrazuje k nim statistiky. Díky vytvořenému účtu získává uživatel komplexní přehled o svých cvičeních i porovnání s výsledky ostatních uživatelů.

Mezi jiné známé aplikace patří například i **SportsTracker**, který začínal již na systému Symbian[1]. Jeho funkcí je zaznamenávat exteriérové pohybové aktivity jako je běh, jízda na kole, cyklistika, apod. V aplikaci si lze dokoupit další funkcionality, jako například stanovení dlouhodobějšího cíle, podrobnější statistiky nebo stínové zaznamenávání pohybu. V aplikaci jsem se inspiroval při výběru zobrazovaných dat při cvičení i v jeho detailu. Aplikace také nabízí možnost zobrazení několika záznamů různých uživatelů ze stejné aplikace. Další velice dobře známou aplikací v této oblasti je **Runtastic**. Oproti předchozí aplikaci ale nabízí navíc i několik tréninkových plánů, které je ovšem potřeba si v aplikaci zakoupit. Volně dostupné jsou zde pouze tzv. „intervalové tréninky“. V aplikaci se mi líbí animace.

Aplikace pro interiérové cvičení nejsou natolik známé, jako ty pro exteriérové sportování, existuje jich ale daleko větší množství. Je to způsobeno především tím, že se vývojáři zaměřují na konkrétní typ cvičení. Jako příklad můžeme použít aplikaci **Virtual Trainer Dumbbell** od společnosti Virtual Trainer, která slouží pro posilování s činkami, nebo další aplikaci stejné společnosti **Virtual Trainer Gym Ball** pro posilování na míči. Tyto aplikace obsahují i videa, která mají uživateli názorně ukázat, jak cviky provádět. Cvičení v aplikaci probíhá podle předem nastaveného časovače, který se uživatel snaží dodržovat. Vedle takových existují i aplikace poskytující pouze časomíru bez ohledu na její využití. V takových aplikacích lze zpravidla nastavit délku cvičení, délku pauzy a počet opakování cvičení.

Všechny tyto aplikace jsou jednoúčelové, tedy jejich zaměření bývá buďto přímo na konkrétní typ cvičení nebo na určité spektrum sportovní činnosti.

[1] Operační systém Symbian. zdroj: http://licensing.symbian.org/

2.2 Úvod do programování v iOS

Pro programování aplikací pro platformu iOS je zapotřebí splňovat určitá kritéria. V první řadě to je vlastnit zařízení se systémem **OS X**. Dále je potřeba vytvořit vývojářský účet u společnosti Apple a mít nainstalované vývojové prostředí **Xcode**, které je po vytvoření účtu zdarma ke stažení. Pokud chceme aplikaci testovat na fyzickém zařízení nebo po vývoji vydat, musíme přikoupit rozšíření našeho vývojářského účtu.[2] Univerzitám Apple nabízí speciální program, v rámci kterého můžou studenti aplikace testovat na fyzickém zařízení, stále ale nemohou aplikace publikovat. Základní typ účtu umožňuje aplikaci testovat pouze v simulátoru zařízení, které je součástí vývojového prostředí Xcode a jeho možnosti jsou omezeny. Text předpokládá základní znalosti programování v jazyce Objective-C a systému iOS.

2.2.1 Základní vlastnosti aplikací

Před vývojem aplikace je vhodné se blíže seznámit se zavedenými postupy a možnostmi v systému iOS. Nerespektování některých vlastností systému může mít neblahý vliv na funkcionalitu aplikace. Při psaní podkapitoly jsem čerpal z knihy *iPhone SDK – Průvodce vývojem aplikací pro iPhone a iPod Touch* [1].

Z bezpečnostních důvodů mají aplikace v systému iOS omezený přístup k souborovému systému. Pro každou aplikaci je při její instalaci v souborovém systému vytvořena samostatná část nazývaná **sandbox**, do které má aplikace přímý přístup. Aplikace má přímý přístup pouze do této části, kde uchovává veškerá svá potřebná data. S výjimkou přístupu např. ke kontaktům nebo hudbě, nemá aplikace do zbylých prostor souborového systému přístup.

Dále existují i jistá časová omezení. Aplikace se musí spustit a zobrazit úvodní uživatelské rozhraní v časovém limitu. To samé platí i pro její ukončení. Při nedodržení těchto omezení může dojít ke ztrátám uživatelských dat.

Každá aplikace také musí definovat veškeré požadavky na cílové zařízení, od verze operačního systému po nezbytný hardware k chodu aplikace. Ty jsou uloženy v souboru **info.plist**, který automaticky vytváří vývojové prostředí při zakládání nového projektu. Vývojář je ovšem zodpovědný za jeho obsah a měl by jej před vydáním aplikace na App Store zkontrolovat a doplnit. Informace jsou poté využívány při rozhodování, které aplikace koncovému uživateli nabídnout podle jeho aktuálního zařízení.

Další zajímavostí pro tento systém je **delegace**. Je to způsob, kdy jeden objekt v programu jedná jménem, nebo ve spolupráci s jiným objektem, na kterého si uchovává referenci. Tomuto objektu poté zasílá zprávy o tom, že se chystá reagovat na nějakou vzniklou událost. Delegát může na zprávu reagovat například změnou vzhledu některých svých prvků, nebo vrátit hodnotu indikující, jak na vzniklý stav reagovat. Jako příklad si uvedeme okno aplikace a k němu delegáta. Když se chystáme okno zavřít, delegát je o stavu informován a vrací hodnotu typu *BOOL*, zda může být okno zavřeno. Například pokud existují neuložená data, může vrátit hodnotu NO a zobrazit výzvu k jejich uložení.

2.2.2 Tvorba uživatelského rozhraní

Na názor uživatele na aplikaci má značný vliv její uživatelské rozhraní. Zde nabízí společnost Apple jednoduché řešení v podobě frameworku **UIKit**, na kterém jsou založeny všechny aplikace pro platformu iOS. Tento Framework poskytuje stěžejní infrastrukturu potřebnou k vybudování a správě

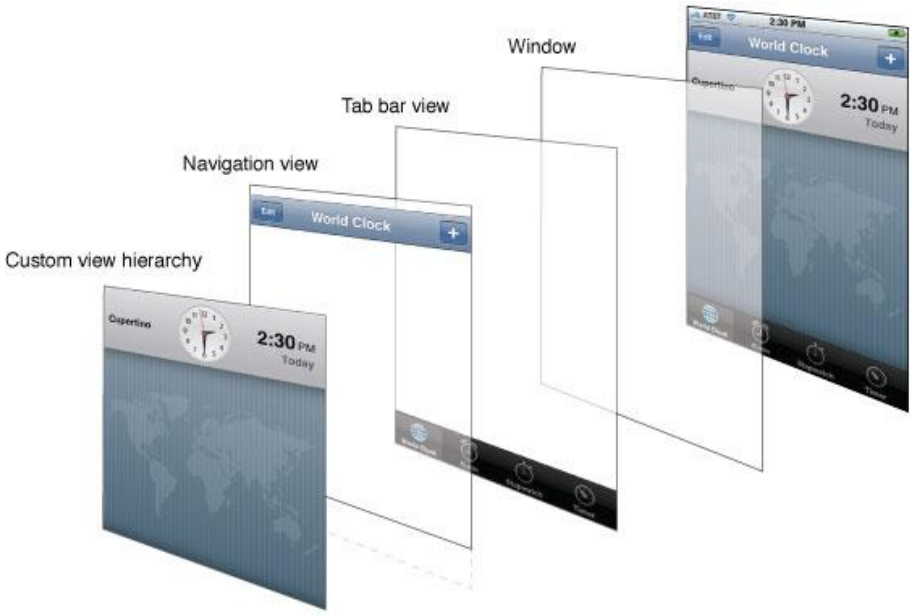
[2] Přehled programů na https://developer.apple.com/programs/

aplikací. Mezi její hlavní funkce patří vykreslování na obrazovku, zpracování událostí vyvolaných uživatelem a vytváření běžných interaktivních grafických prvků. Společnost Apple také vydala dokument, který má vývojáře vést při tvorbě uživatelského rozhraní [2]. Při psaní podkapitoly bylo dále využito programových příruček [3] a [4].

Základním prvkem pro tvorbu uživatelského rozhraní je **ViewController.** Tvoří spojovací vrstvu mezi daty a jejich zobrazením. V aplikacích se jich běžně používá více, kde každý vlastní část uživatelského rozhraní. Jeden například spravuje list uložených záznamů a další zobrazení těchto záznamů po výběru. ViewController spravuje zobrazovaný obsah a s ním spojené příchozí události od uživatele, jako třeba otočení displeje nebo zmáčknutí tlačítka. Taktéž obstarávají hierarchii mezi jednotlivými zobrazeními. Uživateli ovšem není viditelný tento ViewController, nýbrž s ním spojená kolekce komponent zvaných View.

View definuje část obrazovky a zajišťuje její vykreslování. Framework nabízí třídy specifické přímo pro zobrazení textu, obrázku, tlačítek, navigační lišty, seznamu, apod. Zanořováním View vzniká stromová hierarchie. Vnořené View se nazývá **subview** a **superview** je View, do kterého bylo nové View vloženo. Superview si udržuje reference na všechny subview v uspořádaném poli, kde pořadí v poli určuje pořadí při vykreslování těchto prvků nebo také při zasílání událostí o interakci uživatele. Změnou některých vizuálních vlastností superview jsou ovlivněna i všechna jeho subview. Takovými změnami jsou například změna velikosti, skrytí nebo nastavení alfa kanálu[3].

Na Obrázku 2.1 je znázorněna možná hierarchie aplikace.



Obrázek 2.1: Hierarchie zobrazení

[3] Více o alfa kanálu na https://developer.apple.com/library/ios/documentation/GraphicsImaging/Conceptual/drawingwithquartz2d/dq_color/dq_color.html

2.2.3 Přehled nástrojů Apple využitých v aplikaci

Nyní si popíšeme knihovny HealthKit a Core Data použité v aplikaci SportAssistant. Někdy se také setkáme s názvy framework nebo modul. Při zpracovávání podkapitoly věnující se knihovně HealthKit jsem čerpal z referenčního manuálu [6], o knihovně Core Data z programové příručky [5].

2.2.3.1 Modul HealthKit

Společnost Apple vytvořila tento modul pro sjednocení veškerých dat o uživatelově zdravotním stavu. Je dostupný na telefonech iPhone s operačním systémem iOS 8 a výše. V zařízeních disponujících nejméně M7 pohybovým senzorem shromažďuje informace z krokoměru. Umí také shromažďovat data z dostupných Bluetooth kompatibilních zařízení například značky iHealth[4]. V USA se začal používat v několika nemocnicích ke vzdálenému monitorování stavu svých pacientů [7].

Jelikož zde mohou být uložena i důvěrná data, uživatel musí aplikacím požadujícím přístup k datům explicitně povolit čtení nebo zápis pro jednotlivé typy dat. Existují pevně stanovené použité datové typy záznamů a měrných jednotek a vývojáři nemohou vedle těchto vytvořit žádné nové typy záznamů ani měrných jednotek. Z bezpečnostních důvodů, pokud aplikace nedostane povolení ke čtení, tváří se framework, že pro daný typ žádná data neexistují. Nastavení přístupu k datům lze později změnit v nastavení zařízení. Veškerá data jsou uživateli snadno přístupná v systémové aplikaci, kde si lze všechny záznamy prohlížet, přidávat i mazat. Na Obrázku 2.2 vidíme, jak systémová aplikace Health vypadá.

Obrázek 2.2: Systémová aplikace Health

Výhoda toho frameworku spočívá v tom, že data v něm uložená jsou přístupná více aplikacím. Vývojář tedy nemusí implementovat sdílení takovýchto dat s jinou aplikací a naopak některá data může jednoduše získat z přístroje uživatele. Toto sdílení například umožňuje jednoduchou spolupráci nutričních a fitness aplikací.

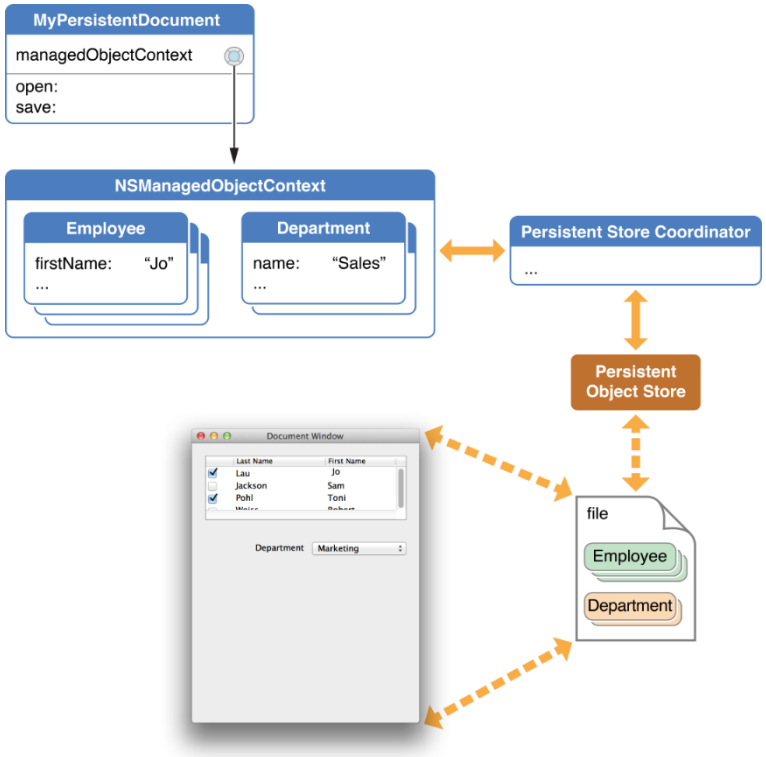
[4] Domovská stránka http://www.ihealthlabs.com/

2.2.3.2 Modul Core Data

Core Data je robustní objektový kontejner pro uchování dat. Je to nástroj určený k návrhu datových modelů od společnosti Apple, který dle dokumentace ušetří vývojáři 50-70% práce při psaní kódu. Navíc je zde jistá záruka funkčnosti a optimalizace garantovaná společností, jelikož je framework denně používán miliony aplikacemi.

Tento framework nabízí jednoduché vytvoření datového modelu a jeho správu. Rozděluje data do dvou částí – data v dočasné paměti a data v souborovém úložišti. Toto rozdělení umožňuje rychlejší práci. Apple nabízí možnost vybrat si typ souboru pro ukládání dat v datovém modelu, kde každý disponuje jinými vlastnostmi. Ve výběru najdeme XML soubor, který je v porovnání s ostatními v práci pomalejší, SQLite nebo binární soubor.

Kolekce objektů na Obrázku 2.3 se nazývá **persistence stack** a spravuje převod objektů v aplikaci do souborů a obráceně. Z pohledu vývojáře je jeho nejdůležitější částí **managed object context (MOC)**, pomocí které aplikace přistupuje k datům.



Obrázek 2.3: Princip práce Core Data

MOC funguje jako objektový prostor, do kterého jsou nahrána požadovaná data z datového modelu a ve kterém uživatel provádí veškeré změny. Dokud tyto změny nebudou uloženy, data v souborovém úložišti zůstávají nezměněna. Díky sledování provedených změn v kontextu jsme schopni provádět operaci vrácení změn nebo jejich opětovné provedení. Následující kód slouží jako ukázka práce s MOC.

```
NSManagedObject *newEmployee = [NSEntityDescription
    insertNewObjectForEntityForName:@"Employee"
    inManagedObjectContext:context]; // vložení nového objektu

[aContext deleteObject:aManagedObject]; // smazání objektu
```

K načítání dat slouží **fetch request**. Je to žádost o načtení, která musí obsahovat typ požadovaného objektu. Volitelnými parametry žádosti jsou predikáty (filtry výběru) a způsob řazení nalezených objektů. Žádost je předána MOC, který vrací odpovídající objekty.

Za mapování objektů do souborů v souborovém systému je zodpovědný **persistent object store**. Prakticky se s ním setkáváme pouze při jeho prvotní inicializaci, většina interakce s tímto frameworkem probíhá pomocí MOC.

Aby framework pracoval správně, je zapotřebí popsat veškeré objekty, se kterými budeme pracovat. **Managed object model** je schéma poskytující takový popis. Typicky se vytváří v grafickém editoru vývojového prostředí Xcode, což usnadňuje vývoj modelu. Schéma obsahuje názvy objektů, jejich atributy a vztahy s dalšími objekty v modelu.

2.3 Zásady tréninku

Trénink je dlouhodobý proces rozvoje specializované výkonnosti sportovce. Např. dosáhnout vysokého výkonu při zvedání závaží není otázkou jen jednoho týdne. Jedná se o dlouhodobý proces.

Trénink se skládá z několika složek, které působí v komplexu. Podle [8] rozlišujeme tělesnou, morální, technickou a taktickou přípravu. Morální přípravu lze chápat jako trénink, kdy jsou na sportovce kladeny stále vyšší nároky. Tělesná příprava je fyzická příprava lidského těla k podávání lepších výsledků, jak silových, tak i vytrvalostních.

U začátečníků a sportovců nižší výkonnosti, do které předpokládáme, že patří i koncový uživatel aplikace, musí však trénink přinášet i osvěžení, neboť může být aktivním odpočinkem po práci. Nesmí proto člověka odrazovat. Pokud člověk trénuje sám, bez trenéra nebo kolektivu, je nezbytné, aby měl potřebné základní vědomosti o metodice jednotlivých cviků, o zásadách životosprávy, o pravidlech sportu atd.

Má-li se sportovec ve své výkonnosti postupně zlepšovat, je nutné, aby jeho trénink neprobíhal rok od roku stále stejně, stereotypně, ale aby se v průběhu tréninku měnil rozsah i obsah tréninkové práce, střídalo se zatížení různého stupně s aktivním odpočinkem.

Tréninková lekce mívá podle [9] ustálenou strukturu. Na počátku lekce se běžně provádí rozcvičení, jehož úkolem je rozehřát organismus a připravit jej ke zvýšené námaze. Slouží jako prevence poranění. Správné rozcvičení bývá často podceňováno nebo úplně ignorováno. Dále se provádí vlastní cvičení. Bezprostředně po ukončení hlavní aktivity následuje zklidnění. Je to skupina lehkých cviků, které mají sloužit k přípravě organismu k přechodu ze zátěže do optimální tepové frekvence. Vynecháním poslední fáze se vystavujeme zkrácení svalů.

Důležitou součástí tréninku je ovšem i správné stravování, aby nedocházelo k úbytkům svalové hmoty a tím ke ztrátě energie. Výživa musí sportovci zajistit dodání všech energetických látek, potřebných jak k činnosti organismu, tak i k jeho růstu a vývoji. Nenahraditelné jsou potraviny obsahující hojně bílkovin, hlavně živočišných, které jsou důležité pro obnovu tkání po svalové práci.

2.3.1 Energetická rovnováha člověka

Energetická rovnováha je poměr přijaté a vydané energie člověka. Jakákoliv změna tělesné hmotnosti je výrazem výkyvu v energetické rovnováze. V ideálním stavu by měl být tento poměr roven 1:1, tedy množství přijaté energie za jeden den by se mělo rovnat množství vydané energie v tentýž den. Přijatá energie je energie přijatá ze stravy. Vydanou energií rozumíme veškerou fyzickou činnost, kterou člověk vykonává, v součtu s bazálním metabolismem člověka.

Bazální metabolismus člověka (BMR) je množství energie potřebné k chodu životně důležitých funkcí lidského těla, což je např. udržování tělesné teploty, tep srdce, atd. Liší se podle pohlaví jedince, jeho tělesné výšky, tělesné váhy a věku. K jeho výpočtu existuje několik matematických rovnic. Jednou z nich je Mifflin-St Jeor:

$$BMR = \frac{10 * hmotnost}{1\ kg} + \frac{6.25 * výška}{1\ cm} + \frac{5 * věk}{1\ rok} + s \qquad (2.1)$$

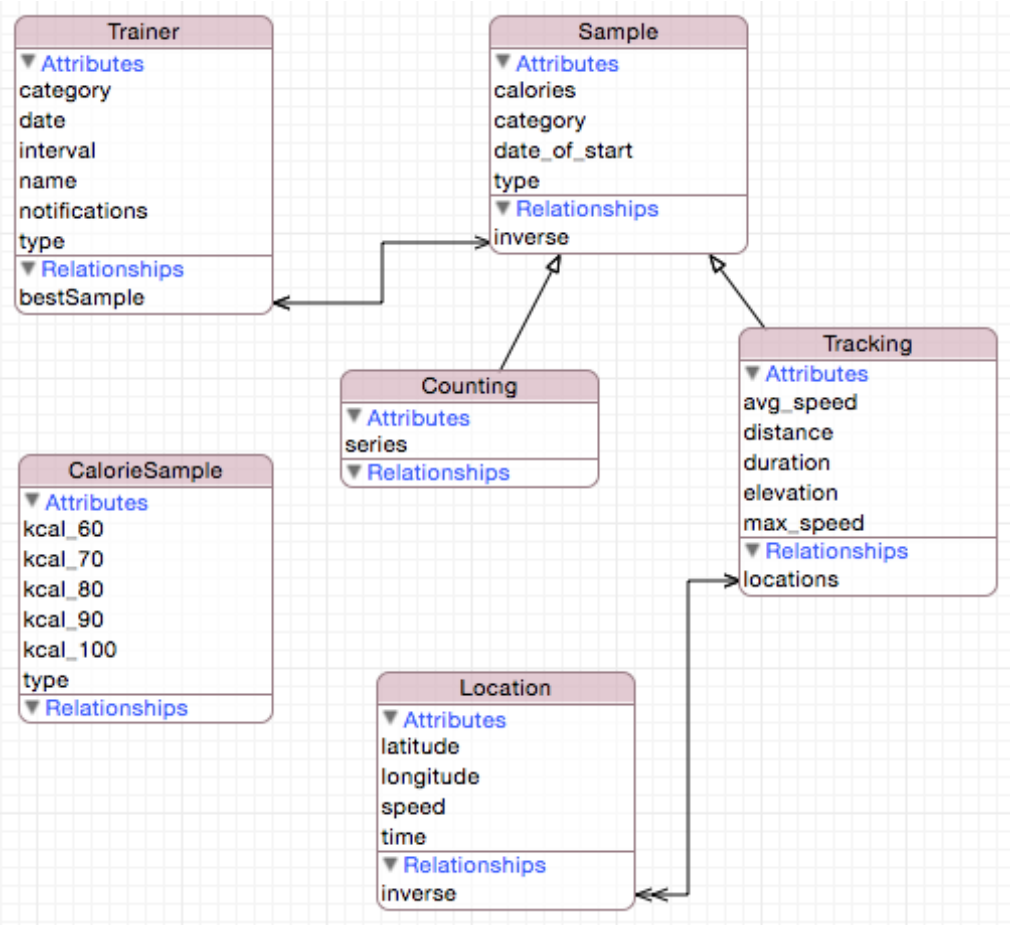
Výraz s v Rovnici 2.1 je roven +5 pro muže a -161 pro ženy. Výsledná hodnota je udána v jednotkách kcal/den. K výpočtu množství kalorií přijatých formou stravy a vydaných fyzickou aktivitou slouží kalorické tabulky. Některé tabulky obsahují již hodnoty bez BMR pro různé váhové kategorie. V aplikaci použijeme právě takové tabulky, abychom uživateli usnadnili její využívání, a to z internetových stránek *NutriStrategy* [10].

3 Návrh aplikace SportAssistant

Před implementací aplikace je potřeba nejdříve navrhnout datový model vhodný pro naši aplikaci a navrhnout také základní hierarchii uživatelského rozhraní. Obě části jsou popsány v této Kapitole, společně s návrhem knihovny pro rozlišení typu pohybu uživatele.

3.1 Datový model aplikace

Vývojové prostředí Xcode nabízí velice jednoduchý způsob tvorby datového modelu formou grafického editoru, který vytvořený model ihned prováže s aplikací a jeho grafickou podobu převádí do XML[5] souboru k uložení. Vývojář tedy nemusí obstarávat při navrhování žádný kód. Výhodou editoru je jeho okamžitá zpětná grafická vazba, která poskytuje lepší představu o celém modelu a provázání jeho jednotlivých prvků. Na Obrázku 3.1 je zobrazen návrh datového modelu pro aplikaci SportAssistant.



Obrázek 3.1: Návrh datového modelu aplikace

[5] XML – extensible markup language

3.1.1 Entity datového modelu aplikace

Entity jsou v aplikaci reprezentovány třídami se stejnými názvy. Všechny třídy s výjimkou CalorieSample jsou vytvářeny uživatelem za chodu aplikace. Třída **CalorieSample** slouží jako záznam v kalorické tabulce, který obsahuje neměnné energetické hodnoty nabízeného typu sportu v aplikaci a proto je vytvářena na pozadí při prvotním spuštění aplikace.

Třída obsahuje hodnoty pro pět váhových kategorií. Atributy třídy CalorieSample jsou (ve formátu název, datový typ a krátký popis):

- **Type (NSString)** – název aktivity, pro kterou jsou uložené hodnoty platné
- **Kcal_60/70/80/90/100 (NSNumber)** – počet kalorií pro jednotlivé váhové kategorie

Stěžejní třídou aplikace je třída s názvem **Sample**, která tvoří abstraktní základ několika dalším třídám. Využívá se zde tedy dědičnost atributů a to:

- **Calories (NSNumber)** – počet kalorií v jednotkách kcal spálených při vykonávání sportovní činnosti
- **Category (NSString)** – název kategorie, v níž je záznam uložen
- **Date_of_start (NSDate)** – datum začátku sportovní činnosti
- **Type (NSString)** – typ sportovní činnosti
- **Inverse (Trainer)** – zpětná reference na trenéra, v rámci kterého je záznam vytvořen

První třídou, která je potomkem Sample, je třída **Counting** reprezentující interiérovou sportovní aktivitu založenou na počítání provedení. Ke zděděným atributům obsahuje jeden následující atribut:

- **Series (NSString)** – počet vykonaných cviků

Při počítání cviků budeme pracovat s datovým typem pole, takový typ ovšem Core Data pro atributy nenabízí. Typ NSString je zvolen z důvodu jednoduchého převodu mezi těmito dvěma typy a také jako usnadnění při vypisování těchto výsledků při zobrazení detailu záznamu.

Druhou třídou, která dědí ze Sample, je **Tracking**. Ta je vytvořena po zaznamenání exteriérové sportovní aktivity založené na zaznamenávání polohy. Třídu Sample rozšiřuje o tyto atributy:

- **Avg_speed (NSNumber)** – průměrná rychlost
- **Distance (NSNumber)** – naměřená vzdálenost
- **Duration (NSNumber)** – doba měření
- **Elevation (NSNumber)** – výškový rozdíl
- **Max_speed (NSNumber)** – nejvyšší naměřená rychlost
- **Locations (Location)** – set všech naměřených poloh

Třída **Location** slouží především k zaznamenání trasy z důvodu jejího pozdějšího zakreslení do mapy a vypočítání některých parametrů záznamu při zobrazení jeho detailu. Třída se vytváří při zaznamenávání polohy, kde u konce zaznamenávání jsou všechny vytvořené třídy navázány k jediné třídě Tracking, jedná se tedy o vztah N:1. Nese atributy:

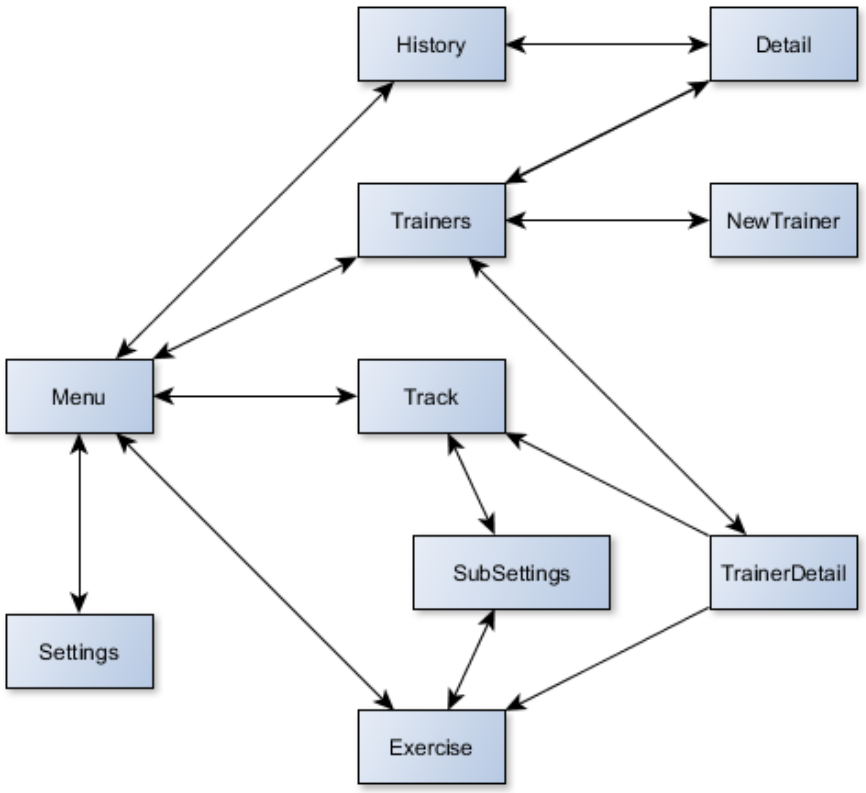
- **Latitude (NSNumber)** – zeměpisná šířka
- **Longitude (NSNumber)** – zeměpisná délka
- **Speed (NSNumber)** – aktuální rychlost zaznamenaná v dané lokaci
- **Time (NSDate)** – čas záznamu
- **Inverse (Tracking)** – zpětná reference na třídu Tracking

Dosud nezmíněnou třídu **Trainer** vytváří uživatel a po vytvoření se nabízí úprava pouze atributu Notifications. Zbylé atributy jsou neměnné nebo je mění aplikace na základě výpočtů při používání aplikace. Třída je charakterizována těmito atributy:

- **Name (NSString)** – název trenéra
- **Category (NSString)** – název kategorie k ukládání záznamů
- **Date (NSDate)** – datum příštího naplánovaného cvičení
- **Interval (NSNumber)** – interval mezi cvičeními v počtu dní
- **Notifications (BOOL)** – indikace povolení zasílání notifikací o naplánovaném cvičení
- **Type (NSString)** – typ aktivity
- **bestSample (Sample)** – nejlepší zaznamenaná aktivita pro daného trenéra

3.2 Uživatelské rozhraní aplikace

Při navrhování uživatelského rozhraní mé aplikace jsem se snažil vytvořit intuitivní a uživatelsky příjemné prostředí. Využil jsem jediného navigačního prvku a základního zobrazovacího prvku view a jeho rozšíření table view. Rozvržení jednotlivých obrazovek je znázorněno na Obrázku 3.2.



Obrázek 3.2: Návrh uživatelského rozhraní (označení obrazovek aplikace)

Aplikace po spuštění zobrazí uživateli **Menu**, ze kterého uživatel vybírá další akci. Nabízí se mu zobrazení obrazovek Settings, History, Track, Exercise a Trainers. V Settings je základní nastavení aplikace. Obrazovka **History** obsahuje historii cvičení, kde uživatel vidí veškeré své záznamy rozdělené do kategorií. Ty lze srolovat a tak skrýt všechny záznamy v dané kategorii. Po výběru záznamu se zobrazí obrazovka **Detail** s detailními informacemi.

Obrazovka **Track** slouží k zaznamenávání polohy uživatele a **Exercise** k počítání cviků. Při výběru zobrazení Track má uživatel možnost přejít do obrazovky **SubSettings**, kde nastavuje parametry pro uložení záznamu. Tato možnost je i v sekci Exercise.

V sekci **Trainers** je list všech dostupných trenérů, kde se po výběru konkrétního trenéra zobrazí obrazovka **TrainerDetail** s jeho detailními informacemi. Také je zde možnost zvolit přidání nového trenéra, které se uskutečňuje na obrazovce **NewTrainer**. Z detailu trenéra lze přímo přejít na zobrazení pro zaznamenávání zvoleného typu aktivity. Zde nastává jediný případ v celé aplikaci, kdy se uživatel nevrací do původního zobrazení, ale chová se, jako by se vstup do sekce uskutečnil z obrazovky Menu.

Ikony použité v aplikaci na Obrázku 3.3 pocházejí z volně dostupného projektu icons8. V licenci tohoto projektu je podmínkou použití uvedení odkazu na projekt v aplikaci. Tato podmínka je splněna v globálním nastavení v aplikaci. Ikona aplikace byla vytvořena sloučením dvou ikon z projektu icons8 a přidáním barevného pozadí.

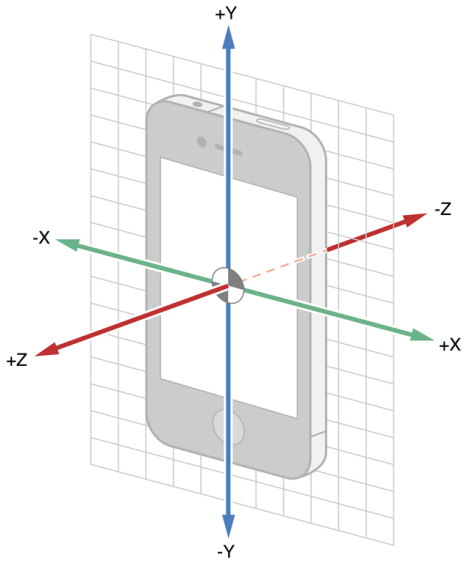
Obrázek 3.3: Použité ikony v aplikaci

3.3 Motion recognizer

V rámci této práce jsem navrhnul knihovnu pro rozpoznávání pohybu uživatele s využitím zabudovaného akcelerometru v mobilních zařízeních iPhone 4. generace a výše. Následující informace pocházejí z příručky o zpracování událostí [11].

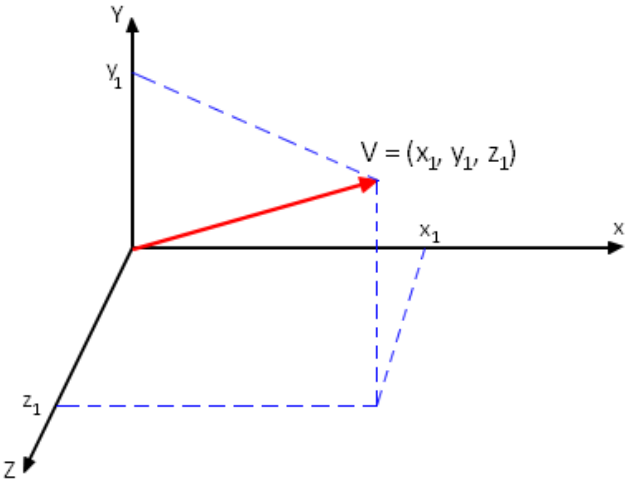
Zařízení snímá pohyb uživatele v trojrozměrném systému. Rozložení os v systému je znázorněno na Obrázku 3.4. Naměřené hodnoty jsou zaznamenány ve veličině **g-force**[6].

[6] g-force – základní měrná veličina akcelerometrů, více na http://en.wikipedia.org/wiki/G-force



Obrázek 3.4: Souřadnicový systém zařízení

Při každém záznamu o pohybu dostáváme tedy celkem tři hodnoty. Z nich spočteme celkovou akceleraci jako vektor, u kterého se nebudeme zabývat směrem pohybu, ale pouze jeho velikostí.

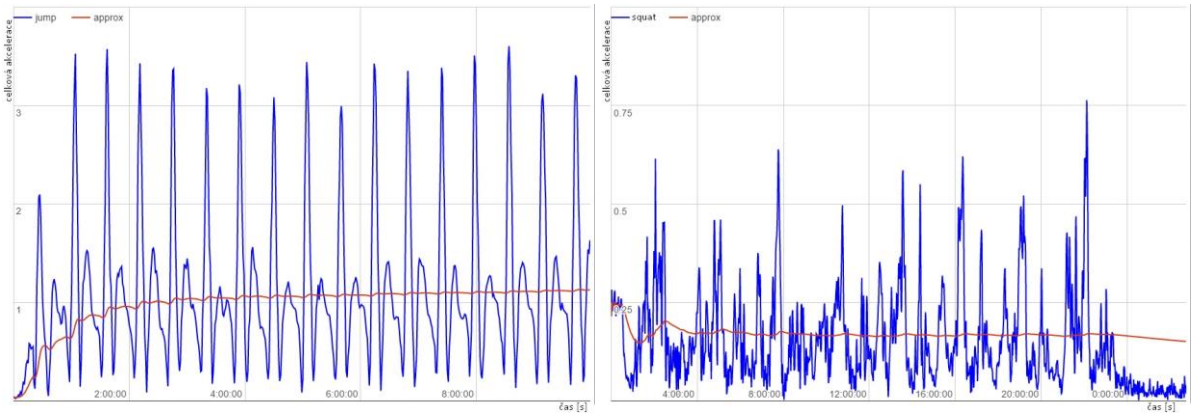


Obrázek 3.5: Vektor celkové akcelerace

Vektor je znázorněn na Obrázku 3.5, je charakterizován třemi souřadnicemi x_1,y_1,z_1 a pro výpočet jeho velikosti použijeme Euklidovskou normu. Výsledný vzorec je v Rovnici 3.1.

$$celkováAkcelerace = \sqrt{{x_1}^2 + {y_1}^2 + {z_1}^2} \text{ [g-force]} \qquad (3.1)$$

Nyní můžeme s naměřenými hodnotami pracovat bez ohledu na směr pohybu a zaměřit se pouze na intenzitu. V aplikaci budeme rozpoznávat dva pohyby – skok a dřep. K jejich rozpoznání si stanovíme mezní hodnoty, které musí uživatel překročit k detekování daného pohybu. Také budeme využívat průměrnou naměřenou hodnotu z naměřených hodnot při kalibraci a časový rozdíl mezi zaznamenáními pohybu, který využijeme pro detekování dřepu. Naměřené hodnoty si pro lepší představivost vyneseme do grafů, které vidíme na Obrázku 3.6.



Obrázek 3.6: Grafické zobrazení naměřených hodnot při skoku (vlevo) a dřepu (vpravo)

Nyní je potřeba v těchto datech nalézt vzorec pro detekci správného pohybu. U skoků nám postačí sledovat pouze překročení mezní hodnoty pro jejich zaznamenání. U dřepů se takových hodnot vyskytuje více, proto budeme zaznamenávat i počet záznamů mezi překročením mezní hodnoty. Nejprve tedy hledáme překročení nějaké hranice, po kterém zaznamenáme, že byl proveden pohyb. Delegátovi ale zašleme zprávu až při klesnutí pod průměrnou hodnotu, abychom jeden pohyb nedetekovali vícekrát.

V následujícím algoritmu použijeme tyto proměnné:

- **acc** – aktuální celková akcelerace získaná z rovnice 3.1
- **approx** – průměrná hodnota celkové akcelerace získaná při kalibraci
- **t_delta** – uplynulý čas od posledního zaznamenání pohybu
- **didJump** / **didSquat** – proměnné pro zaznamenání detekce typu pohybu

```
inicializace didJump a didSquat;
výpočet acc, zvýšení t_delta;

if ( acc < NO_MOVE_LIMIT)
   return žádný pohyb nevykonán;
else if ( acc > approx * RATIO )
{
   if ( acc > JUMP_LIMIT )
      didJump = YES;
      vynulování t_delta;
   else if ( t_delta > TIME_LIMIT )
      didSquat = YES;
      vynulování t_delta;
}
else if ( acc < approx )
{
   if ( didJump == YES )
      return vykonán skok;
   else if ( didSquat == YES )
      return vykonán dřep;
}
```

Veškeré konstanty byly zvoleny experimentální cestou při vývoji a testování knihovny.

4 Implementace aplikace

V rámci této kapitoly si popíšeme implementaci knihovny Motion Recognizer. Dále si vysvětlíme způsob počítání cviků v rámci trenéra, práci s kaloriemi a správu lokálních notifikací. Bude zde také vysvětleno použití externích knihoven v aplikaci.

4.1 Knihovna Motion Recognizer

Tato knihovna byla vytvořena pro rozpoznání skoků a dřepů. Využívá údajů získaných z akcelerometru telefonu, které v systému iOS zprostředkovává knihovna Core Motion, konkrétně třída `CMMotionManager`. Motion Recognizer tuto třídu zaobaluje a její používání je uživateli transparentní.

Třída definuje nový protokol, pomocí kterého informuje svého vlastníka (delegáta) o zaznamenaném pohybu. Pokud není při vytvoření delegát přiřazen, jeví se MotionRecognizer neaktivní.

```
@protocol MotionRecognizerDelegate <NSObject>
- (void)didUpdateMotion:(CMDeviceMotionType)motionType;
@end
```

Aplikace zaznamenává pohyby dva – skok a dřep. K jejich rozpoznání byl vytvořen jednoduchý výčtový typ enum.

```
typedef NS_ENUM(NSInteger, CMDeviceMotionType) {
   CMDeviceMotionTypeNone,
   CMDeviceMotionTypeJump,
   CMDeviceMotionTypeMoveUp,
};
```

Detekování pohybů zmíněné v Kapitole 3.3 formou pseudokódu probíhá v metodě

```
- (CMDeviceMotionType)scanMotion:(CMAcceleration)sample
```

Této metodě se parametrem předává aktuální naměřená hodnota akcelerace. Návratovou hodnotou je typ detekovaného pohybu. Tato metoda je využívána při zpracování příchozího záznamu z akcelerometru v bloku předaném třídě `CMMotionManager` při invokaci metody

```
- (void)startDeviceMotionUpdatesToQueue:(NSOperationQueue *)queue
withHandler:(CMDeviceMotionHandler)handler
```

V tomto bloku je poté, na základě zaznamenaných hodnot metodou `scanMotion`, předána zpráva delegátovi, na něhož si Motion Recognizer uchovává referenci. Před zahájením snímání pohybu je nutno nejdříve provést kalibraci voláním metody `calibrateTheDevice`. Pokud je kalibrace vynechána, třída se opět jeví jako neaktivní. Navíc v případě, kdy delegátem je objekt třídy `UIViewController`, je zobrazeno varovné okno. V případě, kdy chce uživatel provést opětovnou kalibraci, musí vypnout a opětovně zapnout možnosti snímání pomocí pohybového senzoru v zobrazení SubSettings nebo opustit a znovu otevřít obrazovku Exercise.

4.2 Počítání cviků (zobrazení Exercise)

Při cvičení prováděném v rámci trenéra je počítání cviků dekrementální, což by mělo sloužit jako motivace uživatele překonat své předchozí výsledky. Výsledky z nejlepšího cvičení v rámci daného trenéra jsou po celou dobu uloženy v poli `trainerSeries` a pro rozpoznání směru počítání slouží proměnná `countIsNegative` typu *BOOL*. Při vykonání stejného počtu cviků, jako je v poli `trainerSeries`, je tato hodnota zobrazena a dále již dochází při počítání k inkrementaci. Při ukládání probíhá v závislosti na hodnotě proměnné `countIsNegative` buď přímé ukládání, nebo odečtení od cílového počtu cviků. Počítání cviků probíhá ve funkci `clicked_count`:

```
-(IBAction)clicked_count {
    NSInteger addNumber = [count.text integerValue];
    if (trainerSeries && [trainerSeries count]>[series count] &&
        !countIsNegative) {
        addNumber -= 1;
        if (addNumber < 1) {
            countIsNegative = YES;
            addNumber = [[trainerSeries objectAtIndex:[series count]]
integerValue];
        }
    }
    else {
        addNumber += 1;
    }
    count.text = [NSString stringWithFormat:@"%ld", addNumber];
}
```

Pokud uživatel po započetí cvičení v sekci Exercise přejde do nastavení záznamu a v nastavení změní umístění do kategorie, typ cvičení, zapne nebo vypne snímání pohybu nebo automatické počítání, jeho dosavadní provedené cvičení se anuluje, jelikož je aplikace navržena k počítání jednoho typu cviků, pro který se většinou používá jeden způsob zaznamenávání. Například při skoku přes švihadlo neočekáváme, že uživatel v pauze změní způsob zaznamenávání ze snímání pohybu na manuální počítání, tedy dotyk obrazovky.

```
if (![category isEqualToString:cat] || ![t isEqualToString:type] ||
    mot!=motionActive || (a>0 && !autoCount) || (autoCount>0 && !a)) {
        settingsChanged = YES;
}
```

4.3 Kalorie

Při spuštění aplikace se ve funkci `applicationDidFinishLaunchingWithOptions` inicializuje vše potřebné pro běh aplikace. Je to MOC, Health Store, povolení notifikací a jiné náležitosti potřebné ke správné funkci aplikace. Probíhá zde také kontrola na přítomnost datových záznamů CalorieSample, pomocí funkce `countForFetchRequest` objektu MOC. Funkce `saveTableOfCalories` je volána v případě, kdy v datovém modelu nejsou žádné záznamy CalorieSample a ve funkci jsou následně vytvářeny. Tímto je zajištěno, že jsou záznamy vloženy pouze

při prvním spuštění aplikace a jsou vždy přítomny. Při zobrazení sekce Exercise nebo Track je poté proveden dotaz na MOC o požadovaný záznam CalorieSample dle zvoleného typu cvičení, který je později použit při ukládání záznamu o cvičení.

```
if (refKcal) {
   double result;
   if (usersWeight < 65)
      result = [refKcal.kcal_60 doubleValue];
   else if (usersWeight < 75)
      result = [refKcal.kcal_70 doubleValue];
   else if (usersWeight < 85)
      result = [refKcal.kcal_80 doubleValue];
   else if (usersWeight < 95)
      result = [refKcal.kcal_90 doubleValue];
   else
      result = [refKcal.kcal_100 doubleValue];

   result *= [[self countSeries:series] doubleValue]/3600;
   c.calories = [NSNumber numberWithDouble:result];
}
else
   c.calories = @0;
```

4.4 Notifikace

Aplikace využívá lokálních notifikací k upozornění na naplánované cvičení v rámci trenéra. Vytváření notifikací probíhá již při vytvoření objektu trenéra a to v nastavený čas nejbližšího dne.

```
UILocalNotification *localNotification = [[UILocalNotification alloc]
   init];
localNotification.fireDate = trainer.date;
localNotification.alertBody = [NSString stringWithFormat:@"%@: You have
   an exercise planned!", trainer.name];
localNotification.timeZone = [NSTimeZone defaultTimeZone];
localNotification.soundName = UILocalNotificationDefaultSoundName;
localNotification.alertTitle = @"SportAssistant";
[[UIApplication sharedApplication]
   scheduleLocalNotification:localNotification];
```

Nová lokální notifikace se vytváří i v případě, kdy uživatel provede cvičení v rámci trenéra. K aktuálnímu datu je přičten interval v počtu dní a je zachován původní čas. Pokud cvičení probíhá před naplánovaným časem, je potřeba zrušit již naplánovanou lokální notifikaci.

```
NSMutableArray *allNotifs = [NSMutableArray arrayWithArray:
   [[UIApplication sharedApplication] scheduledLocalNotifications]];
for (UILocalNotification *aux in allNotifs) {
   if ([aux.alertBody isEqualToString:[NSString stringWithFormat:
      @"%@: You have an exercise planned!",trainer.name]]) {
      [allNotifs removeObject:aux];
   }
}
[UIApplication sharedApplication].scheduledLocalNotifications =
   (NSArray*)allNotifs;
}
```

Rušení již naplánovaných notifikací probíhá i při smazání objektu trenéra, kde je potřeba zajistit, aby aplikace již nedostávala žádné notifikace od tohoto trenéra. Abych rozlišoval jednotlivé notifikace, vložil jsem název trenéra do těla notifikace a při mazání trenéra v cyklu procházím všechny naplánované notifikace a hledám právě notifikace s názvem trenéra ke smazání.

4.5 Další použité knihovny

Pro vykreslení grafu jsem využil externího frameworku CorePlot[7]. Je použit při zobrazování detailu cvičení založeném na zaznamenávání polohy, kde pro přepínání zobrazení mezi detaily a grafem byl vložen do navigační lišty segmented control. Graf by měl zobrazovat větší množství nepříliš rozdílných údajů, proto byl otočen o 90°, čímž jsme dosáhli většího prostoru. Pro správné fungování frameworku CorePlot je důležité, aby objekt, jenž spravuje vykreslování grafu, rozuměl protokolu *CPTPlotDataSource*, jenž vyžaduje implementaci alespoň funkce

```
- (NSUInteger)numberOfRecordsForPlot:(CPTPlot *)plot
```

V aplikaci je ze stejného protokolu použita i následující metoda, která vrací hodnotu konkrétního záznamu pro obě osy grafu:

```
- (id)numberForPlot:(CPTPlot *)plot field:(NSUInteger)fieldEnum
recordIndex:(NSUInteger)idx
```

Graf je vykreslován ve čtyřech metodách, kde každá z nich zastává jinou úlohu. Tyto metody byly převzaty z [13]. Jejich obsah byl mírně modifikován pro účely potřebné v aplikaci.

```
- (void)configureHost
- (void)configureGraph
- (void)configurePlots
- (void)configureAxes
```

Další informace o knihovně najdeme v dokumentaci [12].

[7] Ke stažení na https://github.com/core-plot/core-plot

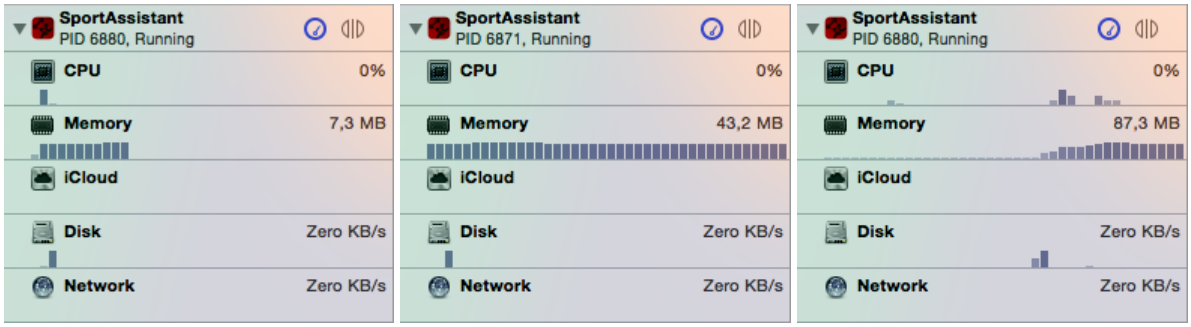
5 Testování

Při testování jsem se nejdříve zaměřil na měření hardwarových požadavků aplikace při jednotlivých úkonech. Dále byla testována knihovna Motion Recognizer. V rámci testování je v aplikaci možnost přidání záznamů simulujících každodenní používání aplikace po dobu dvou let. Záznamy jsou z počítání cvičení z důvodu složitého generování trasy. Vložení testovacích záznamů probíhá na obrazovce Settings.

Testování probíhalo na mobilním zařízení iPhone 5S s operačním systémem iOS 8.3. Jedinou výjimku tvoří testy na hardwarové nároky při zaznamenávání polohy, které byly prováděny v simulátoru vývojového prostředí Xcode.

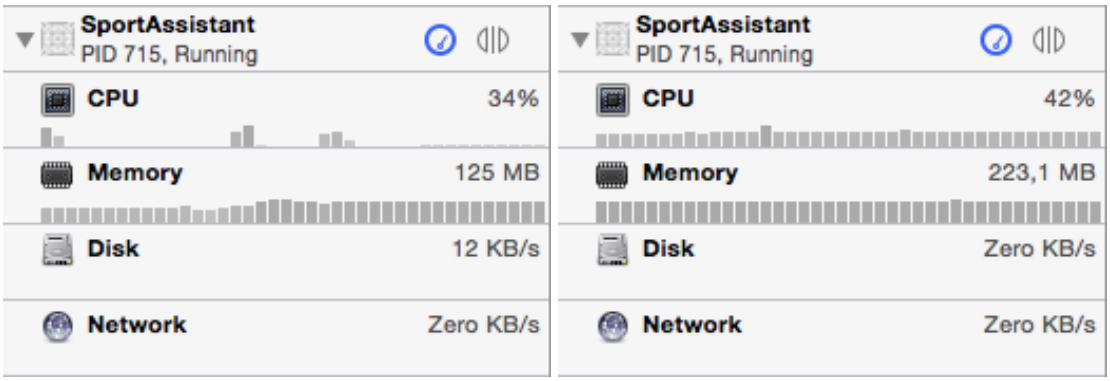
5.1 Hardwarové nároky aplikace

Obrázek 5.1 ukazuje nároky aplikace na hardware. Po spuštění má aplikace nejnižší nároky na paměť. Nároky se zvýší po procházení různými obrazovkami, zejména obrazovkami s mapou. Největší nárůst je při načtení obrazovky Track, kde probíhá zobrazení aktuální polohy na mapě, což je příčinou zvýšené alokace paměti.



Obrázek 5.1: Nároky aplikace na hardware po spuštění (vlevo), po zobrazení všech obrazovek (střed), při zobrazení obrazovky Track (vpravo)

Na Obrázku 5.2 je zaznamenáno hardwarové využití zařízení před startem a před ukončením měření a ukládání polohy uživatele. Test byl vykonáván v simulátoru vývojového prostředí, jeho časové trvání bylo přesných 10 minut 24 sekund a naměřená vzdálenost činila 20.7 km.



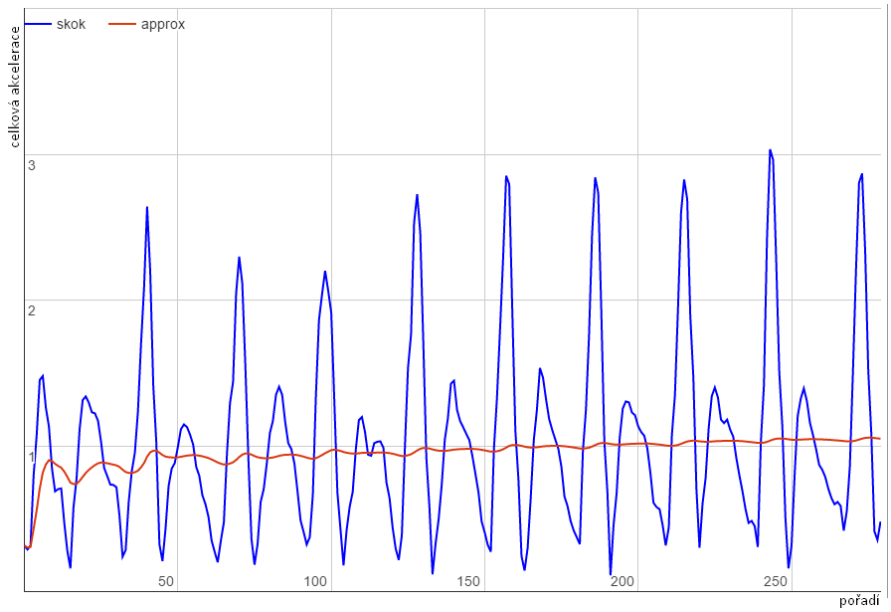
Obrázek 5.2: Nároky aplikace na hardware při startu zaznamenávání polohy (vlevo) a před ukončením (vpravo)

5.2 Testování knihovny Motion Recognizer

Knihovna Motion Recognizer vyžaduje ke svému chodu počáteční kalibraci. Od té se odvíjí kvalita zaznamenávání pohybu. Jakákoliv pohybová nečinnost při kalibraci tedy ovlivňuje výslednou úspěšnost snímání. V následujících podkapitolách si ukážeme rozdíly snímání skoku při správné a chybné kalibraci zařízení.

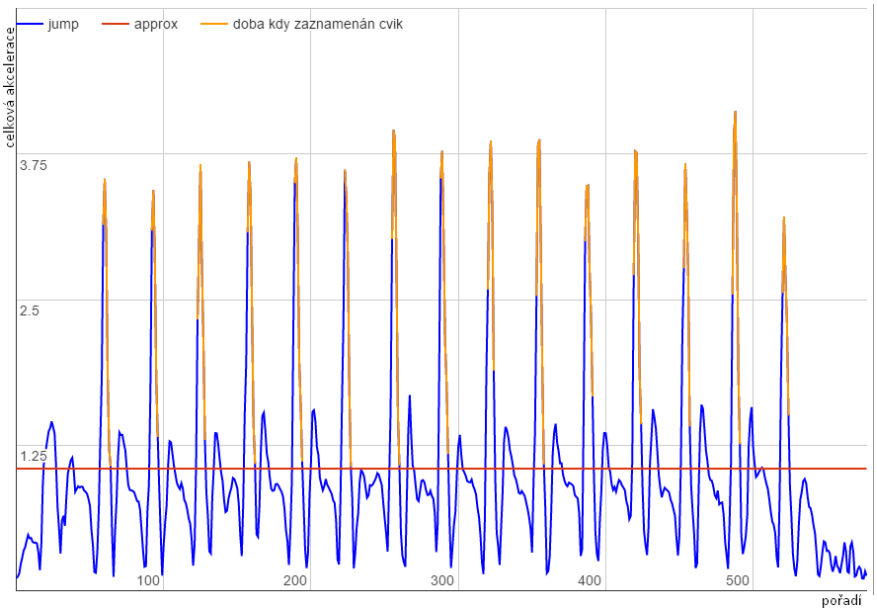
5.2.1 Očekávaná kalibrace

Při kalibraci zařízení se počítá průměrná hodnota ze všech naměřených celkových akcelerací, která později slouží k určení hranice pro zaznamenání aktivity. Očekáváme tedy, že uživatel bude pohybově aktivní po celou dobu kalibrace, která trvá 5 sekund. Na Obrázku 5.3 je grafické znázornění správné kalibrace v rámci tohoto testování.



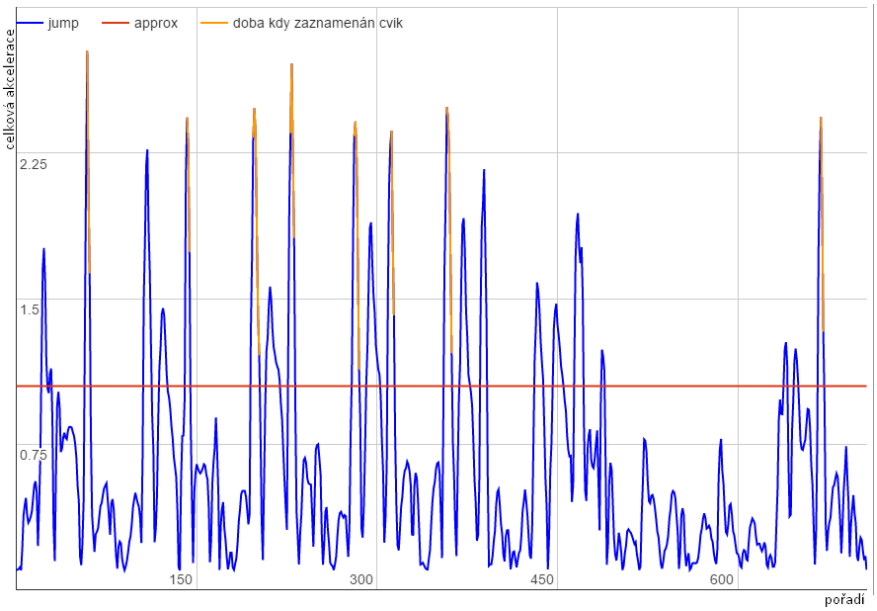
Obrázek 5.3: Grafické zobrazení skoku při kalibraci

Po kalibraci má průměrná hodnota všech celkových akcelerací hodnotu 1.049. Na Obrázku 5.4 je ukázán graf obdobného skákání, jaké bylo při kalibraci. Úsek **doba kdy zaznamenán cvik** je doba mezi zaznamenáním překročení hranice pro skok až po odeslání zprávy delegátovi. Knihovna zde správně rozpoznala všechny provedené skoky.



Obrázek 5.4: Grafické znázornění provádění nepřetržitého skákání

Chybné rozpoznávání se ovšem vyskytuje v případě, kdy uživatel mezi skoky vkládá krátké pauzy. Takové počínání je vidět na Obrázku 5.5. Knihovna v takovém případě zaznamenává více skoků, než je ve skutečnosti prováděno.

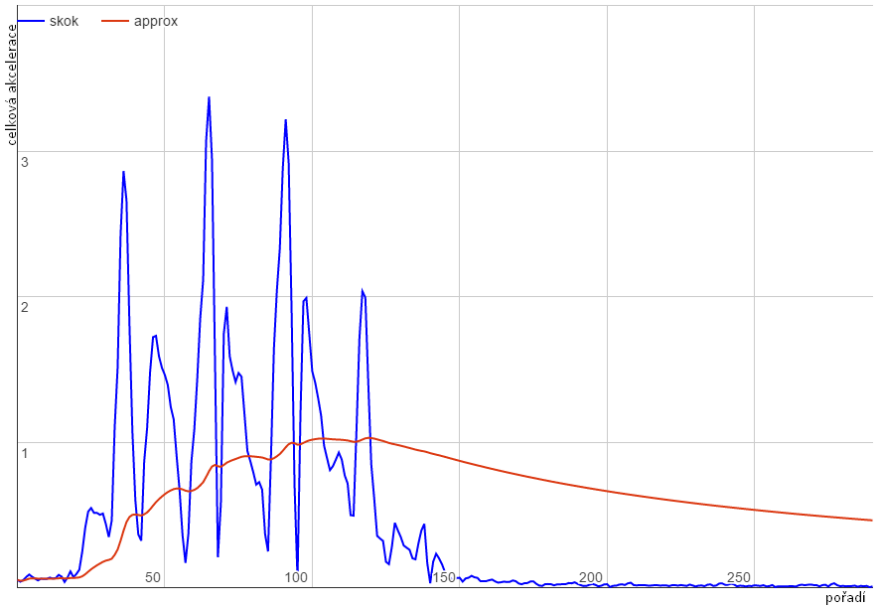


Obrázek 5.5: Graf při vkládání pauzy mezi skoky

5.2.2 Chybná kalibrace

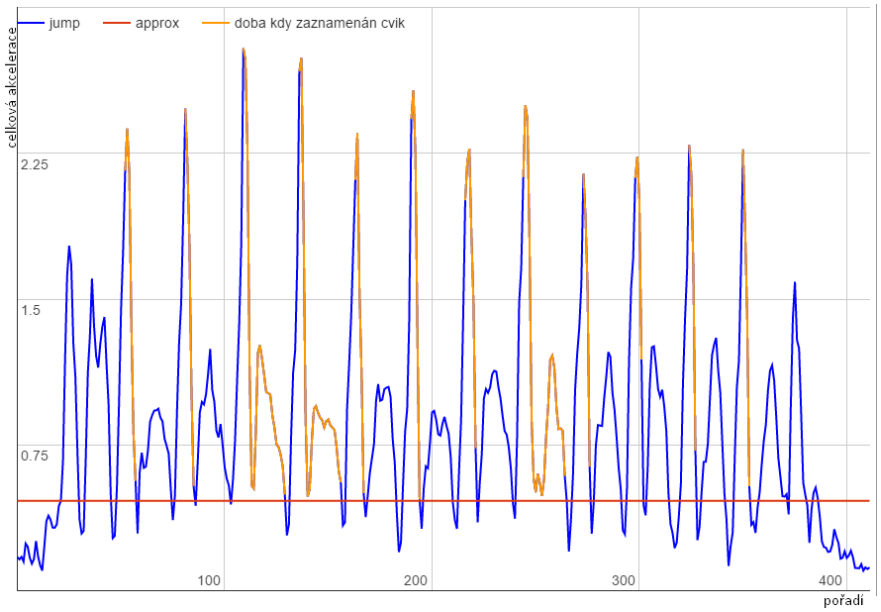
Nyní si ukážeme vliv chybné kalibrace na rozpoznávání skoků. Chybnou kalibrací rozumíme takovou kalibraci, při které uživatel není pohybově aktivní po celou dobu kalibrace. Následkem je výrazné

snížení průměrné hodnoty celkových akcelerací a tím i snížení hranice pro rozpoznávání pohybu. Taková kalibrace je znázorněna na Obrázku 5.6.



Obrázek 5.6: Grafické znázornění možné chybné kalibrace při skákání

Rozpoznávání pohybu po takové kalibraci je vidět v grafu na Obrázku 5.7, kdy úsek **doba kdy zaznamenán cvik** je výrazně delší než tomu bylo při správné kalibraci. Během tohoto testu nedošlo ke špatnému rozpoznání, ovšem na obrázku si můžeme všimnout, že se celková akcelerace několikrát téměř nesnížila pod průměrnou hodnotu, což by mělo za následek vynechání skoku při rozpoznávání.



Obrázek 5.7: Graf zaznamenávání skoků po chybné kalibraci

6 Závěr

Cílem této práce bylo vytvořit mobilní aplikaci pro platformu iOS, která má sloužit jako osobní asistent. Výsledkem je aplikace **SportAssistant**, která nabízí zaznamenávání sportovních aktivit uživatele, zobrazení detailů uložených záznamů a vytvoření motivace ke sportovním aktivitám pomocí objektu Trenéra. Aplikace využívá lokální notifikace, datový model Core Data. Použitím modulu HealthKit nabízí možnost spolupráce dalším aplikacím zaměřeným na fitness a zdraví. Výraznou částí aplikace je vytvořená knihovna Motion Recognizer pro rozpoznání pohybu uživatele.

Tvorba aplikace pro mne byla cenná zkušenost, kdy jsem byl zodpovědný za kompletní životní cyklus aplikace. Velkou výhodou je získání zkušeností s programovacím jazykem Objective-C a seznámení se s důležitými knihovnami využívanými v aplikacích. Také jsem získal povědomí o chování aplikací v systému iOS a nároků společnosti Apple.

Při dalším vývoji by aplikace mohla být rozšířena o další formy motivace, například získávání odměn formou odznaků za množství provedených cvičení ve stanoveném období. Také by mohla být přidána podpora zaznamenávání výškových rozdílů při exteriérových aktivitách. K podání lepší zpětné vazby o cvičeních pro uživatele by aplikace mohla být rozšířena i o statistiky ze všech cvičení, které by sledovaly například frekvenci cvičení uživatele v jednotlivých dnech v týdnu nebo v části dne.

Literatura

[1] Mark, D., LaMarche, J.: *iPhone SDK – Průvodce vývojem aplikací pro iPhone a iPod Touch*, Computer Press a.s., 2010, ISBN 978-80-251-2820-6, p. 480

[2] iOS Human Interface Guidelines, *iOS Developer Library* [online], Apple Inc., 2015 [cit. 13.05.2015], Dostupné na: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/

[3] View Controller Porgramming Guide, *iOS Developer Library* [online], Apple Inc., 2012 [cit. 07.05.2015], Dostupné na: https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/

[4] View Programming Guide for iOS, *iOS Developer Library* [online], Apple Inc., 2014 [cit. 07.05.2015], Dostupné na: https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/

[5] Core Data Programming Guide, *iOS Developer Library* [online], Apple Inc., 2004-2014, aktualizováno 2014-07-15 [cit. 06.05.2015], Dostupné na: https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/CoreData/

[6] HealthKit Framework Reference, *iOS Developer Library* [online], Apple Inc., 2015 [cit 05.05.2014], Dostupné na: https://developer.apple.com/library/ios/documentation/HealthKit/Reference/HealthKit_Framework/

[7] Higgins, T.: *Apple's HealhKit Linked to Patients at Big Los Angeles Hospital*, BloombergBusiness [online], 2015 [cit 05.05.2015], Dostupné na: http://www.bloomberg.com/news/articles/2015-04-26/apple-s-healthkit-linked-to-patients-at-big-los-angeles-hospital

[8] Perič, T., Dovalil, J.: *Sportovní trénink*, Grada Publishing a.s., Praha, 2010, ISBN 978-80-247-2118-7, p. 157

[9] Strakoš, J., Valouch, V.: *Osobní trenér II*, Grada Publishing a.s., Praha, 2004, ISBN 80-247-0475-7

[10] Calories Burned During Exercise, Activities, Sports and Work, *NutriStrategy* [online], 2015 [cit. 03.05.2015], Dostupné na: http://www.nutristrategy.com/caloriesburned.htm

[11] Event Handling Guide for iOS, *iOS Developer Library* [online], Apple Inc., 2015 [cit. 07.05.2015], Dostupné na: https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/

[12] *Core Plot (iOS) Documentation* [online], Dostupné na: https://core-plot.googlecode.com/hg/documentation/html/iOS/index.html

[13] Baranski, S.: *How to Draw Graphs with Core Plot, Part 1* [online], 2012 [cit 09.05.2015], Dostupné na: http://www.raywenderlich.com/13269/how-to-draw-graphs-with-core-plot-part-1

Seznam příloh

Příloha 1

Obsah CD

- /SportAssistant/ - adresář obsahující zdrojové soubory projektu aplikace
- /bachelory/ - adresář s textem práce ve formátu pdf i ve zdrojové podobě