ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Fitness aplikace komunikující s Experience
Student:	Tomáš Bedřich
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

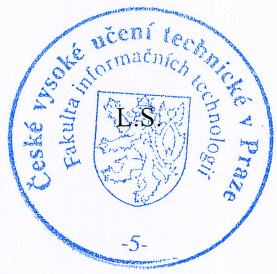
Pokyny pro vypracování

Cílem práce je navrhnout a implementovat první verzi aplikace pro Android a iOS, která bude zpracovávat fitness data z měřícího zařízení Experience přijímaná přes Bluetooth Low Energy a prezentovat je uživateli. Aplikace bude hybridní s využitím Ionic frameworku a AngularJS, publikovaná jako open source na Githubu. Zadavatelem je společnost JSTB International, s.r.o.

- Analyzujte min. čtyři podobné aplikace pro Android a iOS z hlediska jejich funkcí a struktury.
- Určete přesné funkce a strukturu aplikace na základě předchozí analýzy a specifických požadavků zadavatele.
- Navrhněte prototyp aplikace.
- Vhodně otestujte prototyp na budoucích uživatelích (např. testy použitelnosti).
- Upravte prototyp dle výsledků testování.
- Implementujte klíčové funkce aplikace.
- Navrhněte možná vylepšení do budoucna.

Seznam odborné literatury

Dodá vedoucí práce.



L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 10. listopadu 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Fitness aplikace komunikující s Experience

Tomáš Bedřich

Vedoucí práce: Ing. Jiří Hunka

15. května 2016

Poděkování

Děkuji kolegovi a kamarádovi Bc. Filipu Richterovi, se kterým jsem spolupracoval na tomto projektu. Jeho odborné znalosti a dobré rady mě několikrát posunuly vpřed z míst, kde jsem jenom slepě tápal. Děkuji také Gabriele Telekové a své rodině, kteří mě podporovali po celou dobu studia i práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2016 .....

Odkaz na tuto práci

Bedřich, Tomáš. *Fitness aplikace komunikující s Experience*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce řeší analýzu a vývoj mobilní aplikace pro zpracování dat měřených pomocí speciálního hardwarového zařízení. Data jsou přenášena bezdrátově, pomocí Bluetooth Low Energy, v aplikaci jsou zobrazena v reálném čase a zároveň ukládána a následně prezentována pomocí různých grafů a tabulek. Analyzoval jsem čtyři podobné aplikace, stanovil funkční a nefunkční požadavky, zvolil vhodné procesy a nástroje, prošel 10 iterací programování a navrhl API pro komunikaci s online službou pro synchronizaci dat.

Klíčová slova hybridní mobilní aplikace, Android, iOS, fitness, Bluetooth Low Energy, JavaScript, Ionic, Cordova, AngularJS

Abstract

This thesis is about analytics and development of a mobile application for processing a data measured by a specialized hardware device. The data are transferred wirelessly using Bluetooth Low Energy, displayed in real time in an application and also saved for later presentation using various charts and tables. I have analysed four similar applications, determined functional and non-functional requirements, picked appropriate processes and tools, walked through 10 iterations of development and designed an API for communication with online service for data synchronization.

Keywords hybrid mobile application, Android, iOS, fitness, Bluetooth Low Energy, JavaScript, Ionic, Cordova, AngularJS

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V současné době existuje mnoho různých fitness trackerů více či méně specializovaných na jedno sportovní odvětví [26]. K těmto fitness trackerům existuje také mnoho aplikací pro mobilní telefony. Některé se soustřeďují na spolupráci pouze s jedním specifickým HW zařízením, některé naopak podporují širokou škálu různých senzorů.

Tato práce vznikla díky externímu zadavateli a jeho záměru na výrobu nového fitness trackeru. Jeho návrh a výrobu si vzal na starost kolega Bc. Filip Richter z ČVUT FEL a mým cílem bylo navrhnout a implementovat mobilní aplikaci pro jeho ovládání, zobrazování naměřených dat a další manipulaci s nimi. Práce na obou částech bylo třeba koordinovat a operativně řešit nové požadavky, které přicházely v průběhu implementace.

Struktura práce

V práci jsem se snažil zachovat chronologickou souslednost myšlenkových postupů a zpracovávaných úkolů. V první kapitole se věnuji motivaci a vytyčenému cíli tak, jak jsem jej viděl při prvním seznámení a jak byl stanoven při podpisu smlouvy se zadavatelem.

Následuje hlubší analýza, kde se zpočátku věnuji výzkumu měřených dat, který jsme prováděli společně s kolegou. Dále pak rozebírám podobné fitness aplikace pro účely stanovení přesnějších funkčních požadavků, které sumarizuji na konci analýzy společně s hrubým strukturálním konceptem aplikace (wireframe).

V části zabývající se implementací napřed stanovuji vhodné nástroje a procesy a pak se věnuji detailům implementace iterativním způsobem – po verzích. Vložena je také sekce popisující návrh rozhraní pro komunikaci s online protějškem aplikace.

Kapitola 1

Motivace, cíl práce

Zadavatelem práce je externí společnost JSTB International, s.r.o. (dále pouze společnost) zabývající se novým specifickým fitness odvětvím: jumpingem [3]. Společnost se zabývá výrobou speciálních trampolín pro tento sport, školením instruktorů, organizací akcí a dalšími aktivitami spojenými s tímto odvětvím.

Přibližně v polovině roku 2015 jsem byl osloven jejím zástupcem, Markem Bartošem, za účelem vytvoření speciálního fitness trackeru, který by dovoloval měřit aktivitu uživatele při jumpingu a zaplnil tak díru na trhu podobných zařízení. Projekt se měl skládat ze dvou částí:

1. fyzického zařízení na měření dat,
2. a mobilní aplikace určené k jeho ovládání, zobrazování naměřených dat, sdílení a sociální interakci mezi uživateli.

Po domluvě s kolegou Filipem Richterem, který přislíbil spolupráci na projektu ze strany vývoje HW zařízení, jsem se společností uzavřel smlouvu, která velmi volně stanovila požadavky na výsledný produkt.

1.1 Funkční požadavky

Zařízení dle smlouvy mělo být schopno měřit zrychlení ve 3 osách, dále předzpracovávat naměřené hodnoty, hodnotit pohyb dle požadavků společnosti a odesílat data do aplikace. Způsob předzpracování a hodnocení dat byl předem neznámý a v rámci projektu bylo třeba jej vymyslet.

Funkční požadavky byly následující:

F1: Komunikace s fyzickým zařízením – aplikace bude bezdrátově komunikovat se zařízením a přijímat naměřená data.

F2: Výpočet skóre – na základě přijímaných dat bude aplikace vypočítávat jednoduchou číselnou hodnotu (*skóre*) určující sportovní výkon uživatele.

F3: Sdílení skóre – aplikace dovolí uživateli sdílet vypočtenou hodnotu skóre na sociálních sítích.
F4: Udržování profilu uživatele – aplikace umožní uživateli vytvoření a udržování profilu s jeho osobními údaji.
F5: Zaznamenávání historie měření – aplikace bude sledovat a zaznamenávat historii měření.

1.2 Nefunkční požadavky

Ty byly kladeny hlavně na aplikaci z hlediska kompatibility s mobilními operačními systémy.

N1 Podpora Android – aplikace bude spustitelná v systému Android ve verzi 4.4 a vyšší.
N2 Podpora iOS - aplikace bude spustitelná v systému iOS ve verzi 8 a vyšší.

Zvolené verze cílových platforem by měly v případě Androidu pokrýt 73,8% uživatelů [12] a v případě iOS 95% uživatelů [5].

1.3 Integrační požadavky

Dále byl stanoven integrační požadavek napojení aplikace na webový informační systém Jumping World Team, který měl zajišťovat možnost přihlášení existujících uživatelů, registrace nových účtů a porovnání vlastního skóre s ostatními uživateli pomocí účasti na tzv. *události*. Rozhraní pro komunikaci aplikace a informačního systému v době podpisu smlouvy nebylo definováno, takže jeho návrh se později také stal předmětem práce.

Kapitola 2

Analýza

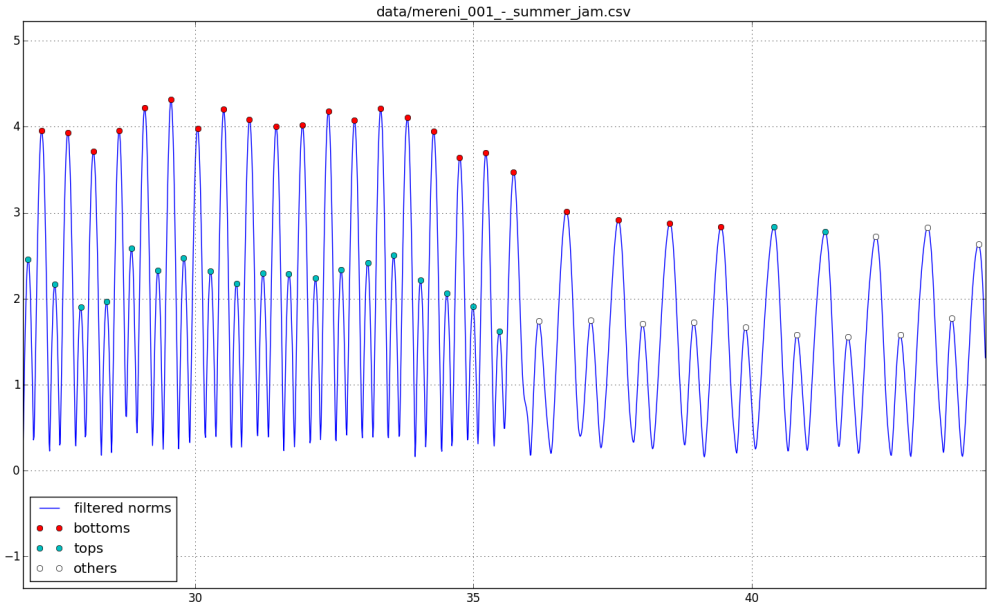
2.1 Měřící zařízení

První práce na projektu probíhaly zejména na měřícím zařízení. Bylo nutné sestavit prototyp a změřit pomocí něj dostatek typických průběhů lekcí jumpingu. Dále bylo třeba tato data analyzovat a pokusit se najít vhodný algoritmus výpočtu skóre. Prvních několik měsíců jsem tedy spolupracoval s kolegou na ladění prototypu a podpůrném SW, který bych zde rád zmínil zejména proto, že se jednalo o prvotní analýzu zpracovávaných dat, na které jsem stavěl různé další úsudky - tedy aby bylo vidět, že další rozhodnutí jsou založena na určitém výzkumu a nabytých zkušenostech, nikoliv neopodstatněných domněnkách.

Jako hardware kolega použil Arduino Pro Mini [6], které rozšířil o možnost měření zrychlení pomocí akcelerometru a o sériovou komunikaci po klasickém BT (nikoliv BLE). Softwarová část sestávala z několika skriptů napsaných v Pythonu s použitím ZMQ a návrhového vzoru publisher/subscriber [27], což zajišťovalo modulární architekturu a možnost síťové komunikace. Napsali jsme skripty pro:

- přijímaní dat posílaných po RS-232 přes BT a jejich přeposílání do ZMQ socketu (pracovně „BT proxy“),
- přijímání dat ze ZMQ socketu a jejich zapisování do souboru (pracovně „logger“),
- přijímání dat ze ZMQ socketu a jejich vykreslování grafu (pracovně „plotter“),
- filtrování dat a hledání extrémů (v kontextu problémové domény se dá mluvit o hledání skoků na trampolíně),
- klasifikaci extrémů a jejich skórování
- a v pozdější fázi vývoje jsem doplnil ještě skript pro přijímání dat posílaných před Websocket a jejich přeposílání do ZMQ socketu (pracovně „Websocket proxy“).

Při měření pomocí prvního prototypu jsme nahráli celkem *172,8 MB textových dat* a *1,46 GB videozáznamu*. Tato data jsme pak analyzovali a snažili se navrhnout algoritmus, který by z nich odfiltroval šum, našel špičky zrychlení a klasifikoval jejich směr (kontakt s trampolínou - dolní úvrať, horní úvrať, jiné).



Obrázek 2.1: Příklad nasbíraných dat. Na ose X je čas ve vteřinách, na ose Y je norma vektoru zrychlení. Vyznačeny jsou body klasifikované jako úvrati.

Dále jsme utvořili vzorce pro výpočet skóre pomocí klasifikovaných špiček zrychlení. Při návrhu těchto vzorců jsme opět spolupracovali se zadavatelem, kde probíhalo experimentální ladění výpočtu tak, aby reflektoval aktivitu uživatele při jumpingu - tedy získané skóre bylo přiměřené fyzickému výkonu.

Na základě analyzovaných dat a vytvořených skórovacích vzorců jsme se rozhodli zpracovávat a hodnotit data v rámci firmware fyzického zařízení a do aplikace přenášet až hotové skóre. Doufali jsme, že to zajistí mnohem menší spotřebu baterie jak zařízení, tak mobilního telefonu, vzhledem k řádově nižší nutnosti komunikace. Pro porovnání: 100 Hz je vzorkovací frekvence zrychlení, 1-10 Hz jsou klasifikované špičky zrychlení a změna skóre nastane přibližně s frekvencí 0.1-0.05 Hz.

V tomto okamžiku už jsme téměř přesně věděli: co budeme měřit, jak to budeme měřit, jaká data z toho vzejdou a jak často je bude třeba přenášet. Zbývalo tedy navrhnout vhodné technologie a architekturu pro finální řešení, které je následující:

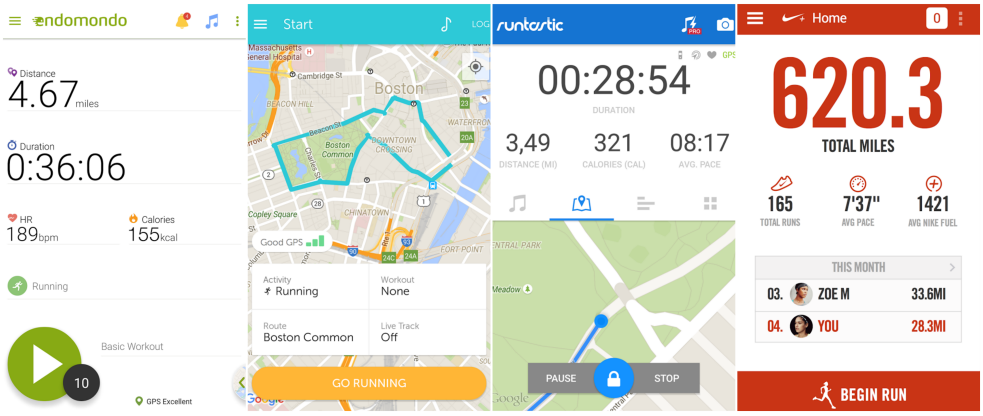
- jednočipový procesor Nordic Semiconductor nRF51822-QFAC-R7, 32-bitový ARM s integrovanou podporou BLE [11]

- 9-osý čip pro měření pohybu (IMU) MPU-9250 [10]
- filtrování dat, klasifikace a skórování na úrovni firmware
- zápis skóre do GATT charakteristiky, samotný přenos dat řeší procesor automaticky pomocí BLE mechanismu notifikací
- příjem notifikací, ovládání zařízení (začátek / konec měření, spárování, ...), prezentace a ukládání skóre na straně aplikace

Po těchto rozhodnutích se naše práce výrazněji oddělila a kolega začal pracovat na návrhu a výrobě finální verze zařízení a programování firmwaru, zatímco já jsem se začal zabývat aplikací.

2.2 Funkce a struktura podobných aplikací

Prvním úkolem bylo zanalyzovat funkce a strukturu alespoň čtyř podobně zaměřených aplikací. Konkrétní aplikace jsem vybíral s ohledem na cílové platformy, tedy aby existovaly vždy ve verzi pro Android i iOS a daly se porovnat.



Obrázek 2.2: Analyzované aplikace. Zleva: Endomondo, Runkeeper (po redesignu), Runtastic, Nike+ Running [7].

2.2.1 Endomondo

- registrace uživatelského profilu přes G+, FB, nebo e-mail
- menu vlevo (vyjížděcí, schované pod ikonkou „hamburgeru“ [15])
  - v záhlaví rozšířená lišta s profilem
  - měsíční statistiky a historie všech cvičení jsou jako záložky jedné položky menu
- úvodní obrazovka
  - několik číselných hodnot (čas, vzdálenost, kalorie atd. - možno zvolit jaké)
  - velké tlačítko Start / Pauza (k tomu malé Stop) - ikonkami

- motivační funkce
  - možnost vytvořit si tréninkový plán (pouze pro sebe)
  - možnost stanovit si cíle a sledovat jejich plnění (společně s přáteli)
  - možnost vytvářet výzvy pro přátele nebo celý svět (výzva obsahuje přihlášené účastníky a jejich žebříček)
- nastavení
  - uživatelský profil
  - metrické / imperiální jednotky
  - spojení s G+, FB, Twitterem, Google Fit
  - varování při ztrátě signálu GPS
  - připojení ke BLE zařízením
- zajímavosti
  - možnost synchronizace dat s online službou
  - nabízí statistiky a sdílení po ukončené aktivitě
  - v menu je odkaz na e-shop

Rozdíly ve verzi pro iOS jsou:

- nefunguje vysunutí menu tažením zleva
- nastavení je v hlavním menu, místo ikonky v navigační liště
- celkově aplikace používá více ikonek (které jsou také významově zřetelnější) - záložky jsou spíše ikonkové
- prvky GUI nemají stíny
- primární akce v rámci jednotlivých aktivit jsou umístěny jako hranaté tlačítko v horní třetině obrazovky
- sekundární akce jsou jako tlačítka dole, pod hlavním obsahem aktivity
- při spuštění sportovní aktivity se zamyká obrazovka (v rámci aplikace)

2.2.2 Runkeeper

- registrace profilu přes G+, FB, nebo e-mail
- menu *nahoře* (záložky)
  - profil (v něm jsou statistiky i historie dohromady), přátelé, trénink
  - přibližně v polovině listopadu 2015 předěláno na menu vlevo, stejně jako všichni ostatní
- úvodní obrazovka
  - na pozadí mapa (vizuálně atraktivní) s překryvem nastavení aktivity, tréninkového plánu, ...
  - velké tlačítko Start - slovy
- obrazovka aktivity
  - nahoře číselné údaje (bez možnosti nastavení)
  - uprostřed sloupcový graf aktivity za posledních X minut (s fixním počtem sloupečků)
  - v druhé záložce lze vidět mezičasy

  - tlačítka Stop a Pauza
- motivace
  - cíle a výzvy v rámci záložky profilu
  - možnost vytvořit tréninkový plán (vlastní, vygenerovaný pro mě, předdefinovaný)
- nastavení
  - profil, notifikace, soukromí
  - spojení s FB, Twitterem, Google Fit
- zajímavosti
  - nabízí sdílení a statistiky po ukončené aktivitě (s možností zapnutí automatického sdílení)
  - možnost synchronizace dat s online službou
  - možnost povzbuzování od kamarádů pomocí FB
  - velmi přívětivá možnost spojení s HR senzorem - na úvodní obrazovce začne skákat ikonka a stačí na ni kliknout
  - iniciativně nabízí ohodnocení aplikace na Google Play
  - iniciativně nabízí spojení s FB pro načtení přátel

Rozdíly ve verzi pro iOS jsou:

- velmi odlišná struktura aplikace
- záložky jsou umístěny dole, s ikonkami a barevně odlišené, nastavení je samostatná záložka
- na záložce Start je rozšířená horní lišta, kde je nastavení aktivity, pod ní lze vybrat hudbu a teprve ve spodní polovině je mapa s tlačítkem start
- při ukončení chce potvrzení
- tlačítka primárních akcí jsou umístěna v horní liště

2.2.3 Runtastic

- registrace profilu přes G+, FB, nebo e-mail
- menu vlevo
  - v záhlaví menu rozšířená lišta s profilem
  - historie a statistiky jsou dvě rozdílné položky
- úvodní obrazovka
  - několik číselných hodnot (možnost zvolit jaké)
  - tlačítko start (možnost výběru poslouchané hudby, typu tréninku, ...)
  - při běhu ukazuje mezičasy (číselně) nebo mapu
- motivace
  - žebříček mezi přáteli (které načítá z FB)
  - možnost nastavit a plnit tréninkový plán
- nastavení
  - profil, jednotky, notifikace

  - spojení s G+, FB, Twitterem, Google Fit
  - odkaz na fórum podpory
- zajímavosti
  - možnost synchronizace dat s online službou
  - zasílá týdenní reporty aktivity na mail
  - nabízí sdílení na sociálních sítích po každé ukončené aktivitě (lze vypnout v nastavení)

Rozdíly ve verzi pro iOS jsou:

- v GUI vícekrát použito zobrazení obsahu v překryvu nad rozostřeným obrázkem
- struktura aplikace je více „plochá“ - stačí méně kliknutí pro dosažení ovládacích prvků, není tolik využíváno rozbalovacích menu, ale spíše samostatných aktivit
- pauza sportovní aktivity se provádí tahem do strany, jak je zvykem pro odemčení obrazovky u iOS zařízení
- tlačítko primární akce je umístěno v navigační liště

2.2.4 Nike+ Running

- registrace profilu přes FB nebo e-mail
- menu vlevo
  - nahoře rozšířená lišta s profilem
  - historie a statistiky cvičení schované jako záložky (taby) jedné položky menu
- úvodní obrazovka
  - několik číselných hodnot (spíše statistického charakteru - historie)
  - tlačítko start (přejde na nastavení cíle běhu - vzdálenost nebo čas)
- obrazovka aktivity
  - podobná úvodní - několik čísel k aktuálnímu běhu (nemožno měnit)
  - tlačítko Pozastavit => Znovu spustit / Ukončit
- motivace
  - uživatel má určitý „level“ odlišený barvou na základě historie
  - žebříček mezi přáteli (načítá z FB)
  - možnost plnit výzvy (předdefinované nebo vlastní)
- nastavení
  - profil, jednotky, notifikace
  - notifikace připomínající pravidelná cvičení
  - soukromí při sdílení (co sdílet)
  - spojení s FB, Twitterem, Google Fit
  - odkaz na podporu
- zajímavosti
  - možnost synchronizace dat s online službou

  - aplikace řeší přerušení cvičení příchozím hovorem
  - v menu je odkaz na e-shop

Rozdíly ve verzi pro iOS jsou:

- úplně chybí statistiky a notifikace
- nastavení je v hlavním menu, místo ikonky v navigační liště
- v nastavení jsou častěji používána přepínací tlačítka místo zaškrtávátek, ačkoliv význam mají stejný

2.2.5 Android - společné znaky

Všechny analyzované aplikace pro Android mají následující společné znaky:

- umožňují registraci uživatelského profilu přes G+, FB, nebo e-mail
- mají tlačítko „Start / Stop / Pauza“
- při měření aktivity zobrazují číselnou hodnotu/y
- umožňují synchronizaci s online službou
- umožňují nastavení nějaké formy motivace (výzva, tréninkový plán, upomínky)
- zobrazují žebříčky mezi přáteli na FB
- dovolují propojení s FB, Twitterem a Google Fit
- udržují uživatelský profil se jménem, váhou a nastavenými jednotkami

3 ze 4 aplikací navíc:

- mají v menu rozdělenou historii a statistiku
- nabízí sdílení výsledků po ukončení aktivity
- mají menu vlevo s horní rozšířenou lištou s profilem

2.2.6 iOS - společné znaky

Ve většině věcí jsou aplikace pro iOS skoro stejné jako jejich protějšky pro Android, až na vzhledové detaily. Liší se hlavně v těchto věcech:

- všechno je méně hierarchicky zanořené, na méně kliknutí
- menu nelze otevírat gestem posunutí prstu ze strany obrazovky
- aplikace používají více ikonek, méně textu
- GUI používá plochý design (žádné stíny, přechody, ...)

3 ze 4 aplikací navíc:

- mají položku nastavení v levém menu
- tlačítko primární akce umísťují do horní lišty
- zamykají obrazovku (v rámci aplikace) při sportovní aktivitě

	Endomondo	Runkeeper	Runtastic	Nike+ Running
registrace pomocí FB	✓	✓	✓	✓
registrace pomocí G+	✓	✓	✓	✗
registrace pomocí e-mailu	✓	✓	✓	✓
vyzkoušení bez registrace	✗	✗	✓	✗
tlačítko Start / Stop	✓	✓	✓	✓
číselné hodnoty při měření	✓	✓	✓	✓
grafy při měření	✗	✓	✗	✗
mapa	✓	✓	✓	✓
sdílení po ukončení měření	✓	✓	✓	✓
porovnání s kamarády	✓	✓	✓	✓
tréninkový plán	✓	✓	✓	✓
cíle	✗	✓	✗	✗
výzvy	✓	✓	✗	✓
závazky	✓	✗	✗	✗
notifikace	✗	✓	✓	✓
povzbuzování od kamarádů	✗	✓	✓	✗
uživatelský profil	✓	✓	✓	✓
nastavení jednotek	✓	✓	✓	✓
spojení s Google Fit	✓	✓	✓	✓
připojení k BLE zařízením	✓	✓	✓	✗
synchronizace s online službou	✓	✓	✓	✓
týdenní reporty na e-mail	✗	✗	✓	✗
odkaz na e-shop	✓	✓	✓	✓
odkaz na podporu	✗	✗	✓	✓
nabízí hodnocení aplikace	✗	✓	✓	✗

Tabulka 2.1: Porovnání funkcí podobných aplikací (verze pro Android).

2.2.7 Zajímavosti

Na analyzovaných aplikacích mě vyloženě zaujalo několik věcí:

- velmi jednoduché připojení HR senzoru v Runkeeperu
- menu nahoře v podobě záložek v Runkeeperu – velmi přehledné
- zasílání týdenních reportů o aktivitě na e-mail
- motivace uživatele pomocí systému levelů v Nike+ Running
- odkazy na e-shop v hlavním menu

2.3 Funkce a struktura aplikace

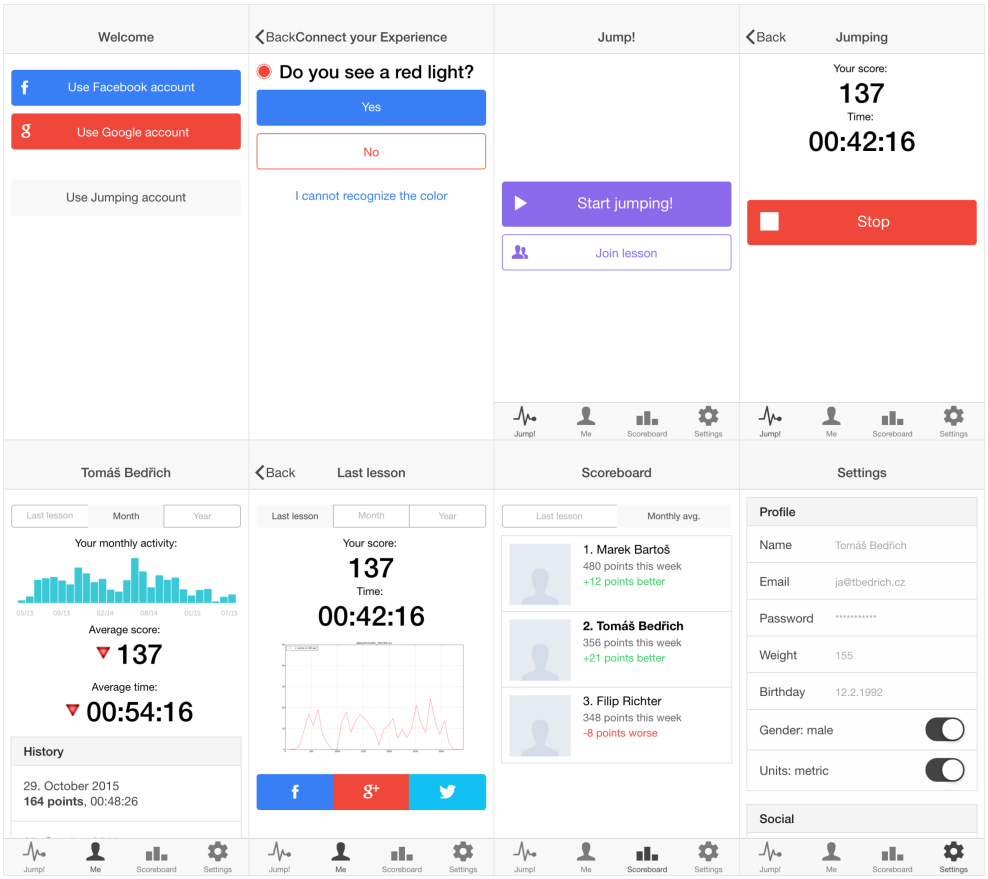
Na základě společných rysů Android a iOS aplikací a dle preference zadavatele jsem se rozhodl zvolit strukturu záložek. Celá aplikace tak měla být jednodušší, něž kdyby používala hlubokou hierarchii nebo rozsáhlé menu. První (domovská) záložka měla sloužit pro spuštění a ukončení aktivity, položky historie a statistiky jsem se rozhodl spojit do druhé záložky, třetí jsem vyhradil přátelům z FB a poslední nastavení. V tomto rozdělení jsem se inspiroval u aplikace Runkeeper. Při prvním spuštění aplikace se měl zobrazil „průvodce“, který by uživatele provedl přihlášením a spárováním zařízení.

Dále jsme se zadavatelem konfrontovali předchozí funkční požadavky s funkcemi analyzovaných aplikací, z čehož vzešla jejich bližší podoba. Finálně se tedy funkční požadavky rozšířily / doplnily o následující:

F6: Registrace uživatelského profilu – aplikace bude dovolovat registraci přes G+, FB, nebo e-mail.
F7: Párování fyzického zařízení – aplikace bude zobrazovat průvodce, který jednoduchou formou provede uživatele párováním jeho zařízení.
F8: Spuštění / ukončení měření velkým tlačítkem – pro tyto úkony bude použito velké tlačítko „Start / Stop“ podobně jako v aplikaci Endomondo.
F9: Zobrazení času a skóre – v průběhu měření aktivity bude aplikace zobrazovat čas od začátku a aktuální skóre.
F10: Zobrazení statistik – aplikace bude obsahovat statistiky ve formě průměru skóre, průměru času a sloupcového grafu, který bude znázorňovat sumu skóre pro jednotlivé dny v týdnu, dny v měsíci a měsíce v roce.
F11: Procházení historie – v aplikaci bude možno procházet seznam lekcí a kliknutím otevřít jejich detail, ve kterém bude zobrazen průběh pomocí tzv. „diff grafu“.
F12: Zobrazení žebříčku kamarádů – aplikace bude obsahovat seznam kamarádů z FB a G+, společně s jejich skóre z poslední lekce / týdne / měsíce.

S použitím shromážděných materiálů jsem pomocí nástroje Ionic Creator vytvořil interaktivní wireframe. Velká výhoda tohoto nástroje je, že wireframe lze zákazníkovi rovnou prezentovat na telefonu, takže si dovede dobře představit finální produkt. Také z něj lze exportovat kostru budoucí aplikace, která

v ideálním případě může posloužit jako základ implementace. Koncept byl zákazníkem schválen s drobnými připomínkami, které jsem hned zapracoval. Jeho struktura však byla schválen kompletně a bez výhrad.



Obrázek 2.3: Některé z obrazovek interaktivního wireframe. Zleva ze shora: uvítací a registrační obrazovka, párování zařízení, spuštění měření, probíhající měření, statistika + historie, detail lekce, žebříček kamarádů, nastavení.

Z hlediska softwarového inženýrství se tento okamžik dá označit jako milník Life Cycle Objectives (LCO) [20] - tedy bod, kdy jsem byl již schopen říci, co konkrétně se bude dělat. Věděl jsem, jaké jsou funkční i nefunkční požadavky, jaký je rozsah projektu, s kým bude třeba spolupracovat. Byla stanovena cena a předběžný harmonogram a všechny tyto náležitosti byly zadavatelem odsouhlaseny. Tím se dala počáteční analýza označit za uzavřenou a mohl jsem se přesunout k cyklu iterativní implementace a testování.

Kapitola 3

Implementace

3.1 Nástroje

V první řadě je třeba si uvědomit, že framework Ionic je pouze nadstavba ekosystémem hybridního mobilního frameworku Cordova. Jádro Cordovy zajišťuje zobrazení webové stránky jako mobilní aplikace a možnost volání nativního kódu pomocí JS běžícího v dané webové stránce. Typicky se aplikace nad touto platformou implementují jako single-page aplikace (SPA). Samotné jádro však neposkytuje téměř žádnou další funkcionalitu, proto Apache Software Foundation dále vydává různé pluginy, které se dají libovolně kombinovat a postupně rozšiřují funkce aplikace. To je dobrý základ, ale pro splnění hlavní myšlenky hybridní aplikace, tedy psát kód pouze jednou pro všechny platformy, chybí ještě řešení pro GUI.

Tímto řešením pak je mimo jiné Ionic framework, který nad Cordovu přidává další vrstvu vlastního JS (ve formě frameworku AngularJS) pro snazší implementaci bussiness logiky. Ve formě znovupoužitelných AngularJS direktiv také přináší předpřipravené GUI komponenty, které se pomocí CSS a JS přizpůsobují cílové platformě dle jejího style-guide. Mimo jiné také definuje doporučené rozdělení souborů v projektu, což usnadňuje počáteční orientaci.

Pro verzování jsem použil distribuovaný verzovací systém GIT. Jako editor jsem po celou dobu práce na projektu používal pouze Atom, nikoliv IDE. Pro nasazování aplikace do testovacích telefonů mi posloužil nástroj `ionic-cli` distribuovaný s frameworkem. Automatizační nástroj Gulp pak pro kompilaci SASS na CSS, minifikaci JS a v pozdějších fázích i pro kompilaci HTML šablon a automatické injektování JS souborů do SPA aplikace. Skvělou funkcí je také „živý náhled“, který automaticky obnoví stránku okamžitě po uložení souboru v editoru – což funguje díky kombinaci příkazu `watch` v Gulpu, dočasné aplikaci vygenerované a nasazené pomocí `ionic-cli` (která všechny požadavky z telefonu přesměrovává na lokální Node.js server, kde ve skutečnosti běží moje SPA) a volitelně ještě přesměrování portů pomocí `adb` (které umožňuje obejít nutnost síťové komunikace a přenáší data po USB). Pokud tedy spus-

tím příkaz `ionic run android --live` v terminálu, tak se vše výše uvedené *zcela automaticky nastaví a spustí* a po chvíli mohu interagovat s aplikací na fyzickém telefonu. Pro ladění v telefonu jsem pak používal Chrome Developer Tools integrované v prohlížeči Chrome (případně Safari Web Inspector pro iOS), které dovolují připojení přes `adb` a následně vzdálené ladění, debugování, profilování, atd.

3.2 Procesy

Vzhledem k nutnosti spolupráce a koordinace některých činností (typicky vývoj firmware a aplikace) jsme se s kolegou a zadavatelem dohodli na používání Trella jako centrálního komunikačního kanálu [13]. Na Trellu se úkoly organizují jako karty, které se dají přesouvat mezi seznamy a lze k nim přikládat přílohy, zaškrtávací seznamy, zodpovědné osoby, štítky, termíny atd. Pro naše účely jsem vytvořil několik seznamů:

- úkoly: pro naplánované, nebo částečně dokončené úkoly, na kterých momentálně nikdo nepracuje,
- právě probíhá: pro úkoly, které momentálně zpracováváme,
- hotovo: archiv úkolů, které jsou dokončené,
- schůzky: karty se zápisy ze schůzek.

Tento systém se mi již několikrát osvědčil na předchozích projektech a i zde si ho všichni zúčastnění pochvalovali.

Celkovou koncepci vývoje jsem po počáteční analýze pojal agilně / iterativně. Testování tedy probíhalo tak, že na konci každé iterace jsem aplikaci nainstaloval na iPhone, který jsem odevzdal. Zadavatel pak testoval jednak sám a příležitostně také na budoucích uživatelích. S pomocí Trella pak jeho zástupce jednoduše hlásil problémy nebo požadavky do dalších iterací, které jsem průběžně zpracovával. Tím jsem byl schopen velmi pružně reagovat na požadované změny a zadavatel často ovlivňovat směr projektu.

Pravidelné testování na běžných uživatelích jsme neprováděli, jelikož vývoj projektu probíhal utajeně. Bylo nežádoucí, aby se širší veřejnost dozvěděla o připravovaném zařízení nebo aplikaci. Proto testoval primárně zadavatel a jeho zaměstnanci, kteří však jako fitness trenéři také představovali významnou část cílové skupiny.

3.3 0. iterace

Jako první bylo třeba implementovat elementární funkčním požadavek: bezdrátovou komunikaci se zařízením. Dá se tedy mluvit o jakési nulté iteraci, při které probíhalo vytvoření počáteční kostry, nastavení vývojového prostředí, seznámení s novými technikami programování v JS (zejména „promises“ [14])

a počáteční výběr vhodných pluginů. V této iteraci jsem do projektu začlenil klíčový plugin `cordova-plugin-ble-central`, z něhož jsem napřed využil:

- metodu `scan()`, která vrací BLE zařízení v okolí
- metodu `connect()`, která se připojí na zařízení dle jeho MAC adresy
- metodu `startNotification()` pro čtení hodnot pomocí BLE notifikačního mechanismu
- metodu `write()` pro zápis hodnot (příkazů) do zařízení

Pro převod dat přenášených po BLE do/z JS jsem použil typovaná pole (typed arrays), např. `Uint8Array` [22]. Podařilo se mi navázat komunikaci pomocí BLE a postupně jsem začal budovat vlastní vrstvu v podobě AngularJS služby (service) pro zapouzdření veškeré nízkoúrovňové komunikace.

Dále jsem se potýkal s problémem logického rozdělení aplikace na kontrolery. Zvolil jsem návrh, kde jeden kontroler odpovídá jedné aktivitě (v AngularJS se místo aktivity používá termín „stav“). V rámci přechodu mezi stavy bylo třeba řešit také mapování na URL – jelikož se stále jedná o webovou SPA. To je standardně v Ionicu řešené pomocí modulu AngularUI Router, což jsem zachoval, ale bylo třeba se s ním řádně seznámit.

3.4 Verze 0.1.0

V první verzi, která měla být prezentovaná zákazníkovi, jsem konečně začal zpracovávat funkční požadavky.

3.4.1 Přihlašování

Zpočátku jsem řešil implementaci přihlášení pomocí sociálních služeb (FB a G+), pro které jsem kvůli neznalosti napřed chtěl použít protokol OAuth. Tento přístup se mi sice podařilo zprovoznit, nicméně pro uživatele byl nekomfortní v tom ohledu, že nevyužíval autentizace přes nativní aplikace nainstalované v operačním systému. Pokud tedy například uživatel měl nainstalovanou mobilní aplikaci Facebook, kde již byl přihlášen a vyvolal přihlášení „pomocí Facebooku“ v rámci mojí aplikace, tak očekávané bylo automatické přihlášení – což se však s použitím OAuth nedělo. Bylo tedy nutné přejít k použití nativních SDK vydávaných firmami Facebook a Google pro tyto účely a integrovat je do aplikace. Tak vyvstala potřeba instalace dalších pluginů: `cordova-plugin-facebook4` a `cordova-plugin-googleplus` (po několika vyzkoušených alternativách se tyto jevily jako nejlepší). Bohužel, pluginy se po delších zkušenostech zdají být velmi problémové, nestabilní a jejich nastavení je zdlouhavé a nepohodlné. Ke dni 7. dubna 2016 jsem například hlásil vzájemnou nekompatibilitu obou pluginů [18, 17]. Podobné chyby extrémně znesnadňují ladění, protože pluginy se zdají fungovat správně, ale *pouze pokud jsou nainstalovány odděleně.*

S řešením přihlašování také vyvstala potřeba perzistentního ukládání dat + správná volba okamžiku uložení a obnovení. Pro tyto účely jsem nainstaloval plugin `cordova-sqlite-storage` do Cordovy a `ngstorage` do AngularJS. První zmiňovaný dovoluje vytvořit a používat v rámci aplikace perzistentní SQLite databázi a vznášet nad ní klasické SQL dotazy, používající agregační funkce, spojení a jiné výhody relačních DB. Druhý zajišťuje tenkou vrstvu pro přístup k local storage pomocí AngularJS, což je také perzistentní úložiště, avšak nedovoluje žádné pokročilé dotazy nad daty – jedná se prakticky pouze o key-value storage. Napřed jsem experimentoval s ukládáním dat až při ukončení aplikace, ale pak se zapisování rovnou do perzistentního úložiště jevilo méně problémové – proto, jsem zvolil okamžitý zápis. Záměrem bylo použít SQLite DB do budoucna pro ukládání naměřených dat a local storage pro ostatní data (typicky JSON). Po úspěšné instalaci pluginů a zapouzdření do vlastní služby jsem tedy rovnou využil local storage pro ukládání dat o přihlášeném uživateli, takže se stačí autentizovat jednou, po prvním spuštění aplikace.

Kontrolorem přihlašování jsem pak na základě událostí vyvolaných stiskem tlačítka v pohledu pouze volal metody přihlašovací služby a v případě úspěchu přecházel do stavu skenování / párování.

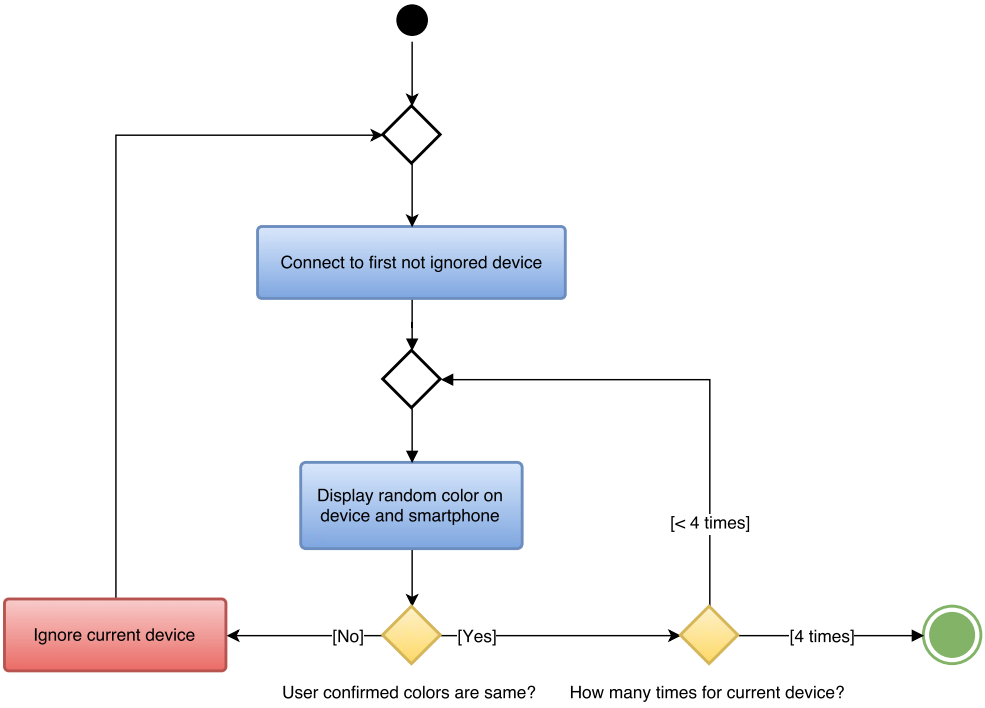
3.4.2 Párování fyzického zařízení

Druhým funkčním požadavkem byl proces párování. Vzhledem k tomu, že zařízení nemá žádný displej ani tlačítka a jeho jedinou možností interakce s uživatelem je RGB LED dioda, nebylo možné použít proces párování známý z jiných Bluetooth produktů – držení tlačítka, opisování číselného kódu... Bylo třeba vymyslet zcela nový způsob.

Společně s kolegou jsme tedy navrhli proces párování, který používá RGB LED. Na začátku procesu se v aplikaci vygeneruje náhodná sekvence 4 z 6 možných barev, která se postupně synchronně zobrazuje na telefonu a fyzickém zařízení. V každém kroku je na uživateli aby potvrdil, že se barva na telefonu shoduje s barvou svítící na zařízení. Pokud 4 krát po sobě odpoví kladně, je zařízení spárováno. Pokud kdykoliv odpoví, že se barva neshoduje, párovací proces je přerušen, zařízení je ignorováno a začíná se znovu s dalším nalezeným v pořadí. Pokud by tedy náhodou nastala situace, že více uživatelů vedle sebe bude párovat více zařízení ve stejný okamžik, tak tímto postupem je zajištěna pravděpodobnost spárování špatného zařízení:

$$\frac{1}{V_4(6) - 1} = \frac{1}{\frac{(6-4)!}{6!} - 1} = \frac{1}{359}$$

Vymysleli jsme i alternativní způsob párování s ohledem na barvoslepé uživatele, který jsem však zatím neimplementoval. Alternativně je plánováno vygenerovat v rámci 10 sekundového intervalu 4 náhodné „pulzy“ dlouhé jednu vteřinu, které nebudou následovat bezprostředně po sobě. V těchto pulzech



Obrázek 3.1: Párovací proces

bude LED svítit (nezávisle na barvě) a ve zbytku intervalu bude zhasnutá. Sekvence se pak synchronně spustí na zařízení a telefonu a uživatel dostane za úkol sledovat, zdali obě zařízení blikají ve stejné okamžiky. Pokud ano, párování bude dokončeno – pokud ne, proces se bude opakovat.

Proces párování řídil kontroler, který v pohledu zobrazoval barvy a přijímal přes něj odpovědi uživatele. Na pozadí pak volal vlastní službu nízkoúrovňové BLE komunikace, pomocí které zapisoval do LED charakteristiky RGB kód požadované barvy. V případě úspěšného spárování zařízení si zapsal jeho MAC adresu do perzistentního úložiště (key-value – local storage) a přešel na hlavní obrazovku aplikace. Při každém spuštění aplikace se pak pomocí tohoto úložiště kontroluje, zdali je nějaké zařízení spárováno a podle toho se zobrazí buď úvodní obrazovka s přihlášením a následně párováním, nebo hlavní obrazovka se záložkami.

3.4.3 Měření

Dále jsem implementoval samotné měření. Na nižší vrstvě spuštění probíhá tak, že se ve fyzickém zařízení vynulují současné hodnoty skóre (sledujeme více složek skóre, přičemž každá složka skóre má vlastní BLE charakteristiku), zaregistrují se notifikace na změnu těchto hodnot a do ovládací charakteristiky se zapíše příkaz pro začátek měření. Posledním zmiňovaným krokem se zařízení

přepne do „aktivního módu“ [1], kdy zapne senzory a spustí algoritmy pro vyhodnocování skóre. Zároveň se spuštěním měření je do DB zapsán začátek lekce.

Na tomto místě bych rád ještě popsal strukturu SQLite databáze – to proto, že potřeba jejího návrhu vyvstala právě při prvotní implementaci měření. Strukturu pro přehlednost uvádím v kompletní podobě, ačkoliv tím předbíhám až do verze 0.9.0.

```
CREATE TABLE lesson (
    start DATETIME PRIMARY KEY,
    end DATETIME,
    type TINYINT NOT NULL,
    event INT
);

CREATE TABLE score (
    start DATETIME NOT NULL,
    time DATETIME,
    score FLOAT NOT NULL,
    type TINYINT,
    PRIMARY KEY (start, time)
);
```

DB se tedy skládá ze dvou tabulek, kde jedna udržuje záznamy o lekcích a druhá o skóre. Sloupce `lesson.start`, `lesson.end` a `score.time` obsahují časy začátku lekce, konce lekce a čas příjmu hodnoty skóre. Sloupce `lesson.type` a `lesson.event` ve verzi 0.1.0 nebyly přítomny, nicméně v poslední verzi obsahují typ lekce (jumping, běh, cyklistika) a cizí klíč události, pod kterou daná lekce spadá (hodnota dává smysl pouze na serveru, v aplikaci se databáze událostí neudržuje). Sloupec `score.score` je samotná hodnota skóre přijatého ze zařízení a `score.type` pak typ (neboli označení složky) skóre. Mezi tabulkami `lesson` a `score` je vztah 1:N, přičemž cizím klíčem je `score.start`. Ke každé lekci tedy typicky existuje několik stovek řádků v tabulce skóre.

Zpět k měření: v jeho průběhu jsou tedy přijímaná data zpracovávána voláním metody obsluhující BLE notifikace. Každé takové volání pak data dekóduje pomocí JS typovaných polí a vloží je do DB. Při ukončení měření je v DB nalezena lekce s prázdným časem konce (za normálních okolností je vždy právě jedna) a řádek je doplněn.

Z hlediska kontroleru a pohledu pak bylo třeba vytvořit tlačítko pro spuštění a ukončení měření společně s jeho obsluhou. Ta spočívala kromě výše

[1] Je nezbytné pracovat s aktivním vs. pasivním módem a nepotřebné součástky dočasně vypínat. Dle měření provedeného kolegou se totiž odběr procesoru nRF51822 dohromady se senzorem MPU-9250 v aktivním módu *zvýší přibližně 140 krát!*

uvedeného ve spuštění časovače, který aktualizuje dobu měření (stopky). Při každém tiknutí stopek se pomocí agregační funkce vybrala z databáze aktuální suma skóre pro právě probíhající lekci (což bylo *extrémně* neoptimální, ale postačující pro první iteraci) a po ukončení stačilo pouze zastavit časovač.

3.4.4 Přípravy vydání

Před vydáním první verze jsem ještě pozměnil strukturu aplikace pro lepší oddělení vlastního kódu a používaných knihoven. To mělo sice pozitivní dopad na výslednou velikost aplikace, ale na druhou stranu jsem musel upravit podpůrné skripty. Pozměnil jsem tedy konfigurační soubor pro automatizační nástroj Gulp a přidal kopírování nutných JS souborů na správná místa. V pozdějších verzích se ještě zmíním o dalších podstatných vylepšeních tohoto souboru... Napsal jsem také krátký shell skript pro automatizované sestavení a podepsání balíčku `.apk` pro Android.

Aplikace ještě tak nějak mimochodem dostala nový design, který sice nebyl požadován, ale účelem bylo poněkud ji personalizovat oproti strohému vzhledu interaktivního wireframe. Tento design se ujal až nečekaně dobře a používal jsem jej až do verze 0.9.0.

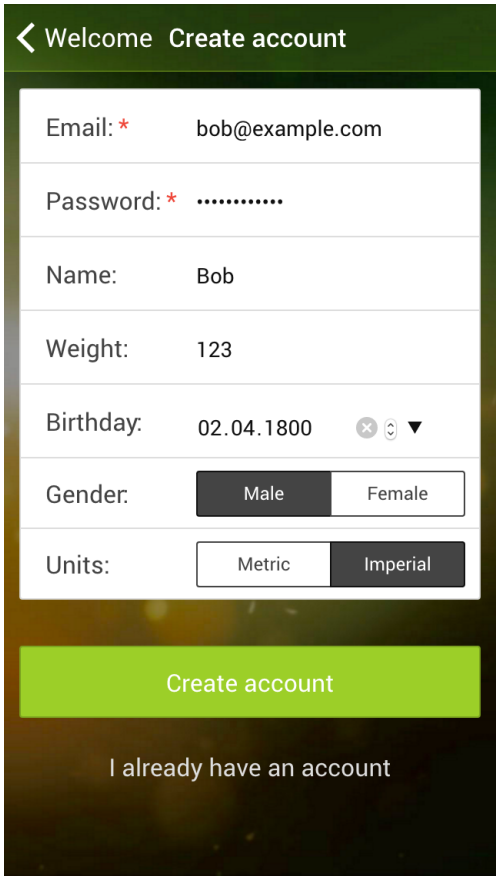
Jako poslední uživatelsky skrytou, ale přesto důležitou funkci, jsem v této iteraci přidal „vzdálené logování“. Použil jsem knihovnu `stacktrace.js` společně se staženou a upravenou AngularJS službou [25]. Účelem bylo, aby případné chybové stavy v aplikaci, kterou budu předávat zadavateli, byly rovnou odesílány na server (samozřejmě pouze při aktivním internetovém připojení). Tuto funkci jsem později *velmi* ocenil při několikerém hledání nahlášených chyb.

3.5 Verze 0.2.0

Druhou iteraci jsem začal refactoringem. Bylo vidět, že aplikace se bude rozrůstat, a proto nemohla stačit souborová struktura připravená Ionic frameworkem, která počítala s jedním souborem pro všechny kontrolery apod. Začal jsem tedy logickým rozčleněním aplikace na jednotlivé soubory pro kontrolery a služby.

Dále jsem vyřadil z verzovacího systému složku s knihovnami, upravil soubory `package.json` + `bower.json` kde jsem definoval závislosti tak, aby je balíčkovací nástroje Bower a `npm` mohly automaticky stáhnout a přidal vlastní soubor `README`, který obsahuje návod pro instalaci. Tím jsem kompletně vyřadil z GIT repozitáře verzování cizího kódu. Pomocí souboru `.gitignore` jsem také zakázal verzování generovaného kódu jako je kompilované SCSS.

V této iteraci jsem také vytvořil Python skript „websocket proxy“ zmiňovaný výše a jeho nutnou JS protistranu, která po vytvořila websocket a začala do něj přeposílat všechna data přijímaná po BLE. Pro účely nastavení parametrů websocketu jsem přidal položku v záložce nastavení.



Obrázek 3.2: Design aplikace vytvořený ve verzi 0.1.0

Z dalších věcí jsem se například zabýval chováním formuláře pro registraci pomocí e-mailu, zejména pak zobrazením chybových stavů (nesprávný e-mail apod.) a jejich vzhledem tak, aby byly pro uživatele co nejintuitivnější. V grafickém editoru jsem vytvořil ikonu a startovací obrazovku aplikace a naprogramoval jsem první verzi systému pro udržování BLE spojení, která jenom jednoduše ověřila konektivitu vždy při spuštění měření a případně se pokusila znovu připojit. Tento systém jsem později výrazně vylepšil.

3.5.1 Historie a detail lekce

Dalším funkčním požadavkem je historie lekcí + jejich detail. Přidal jsem tedy kontroler a pohled historie, který napřed velmi jednoduše zobrazoval všechny lekce jako textový seznam (pomocí služby pro komunikaci s DB).

Po kliknutí na jednu z lekcí je uživatel přesměrován do samostatného kontroleru, který také vybere data z DB – v tomto případě však bylo nutné data předzpracovat pro knihovnu `Chart.js` (kvůli kreslení diff grafu), kterou jsem

pro tento účel integroval do projektu. Předzpracování (agregace dat) probíhá jednak na úrovni SQL dotazu a finálně i na úrovni JS. Kromě grafu je v detailu lekce také vypočtená délka a celková suma bodů.

3.6 Verze 0.3.0

Kromě přidání několika animací GUI a odkazu na e-shop jsem se v této verzi zaměřil spíše na technická vylepšení, než na zpracovávání funkčních požadavků.

3.6.1 Kompatibilita, výkon

Po prezentaci aplikace 0.2.0 zadavateli začala být vidět potřeba řešení výkonu a kompatibility. Např. po kliknutí na různá tlačítka nebyly odezvy úplně optimální, položky pohledů se postupně objevovaly („donačítaly“), zobrazení v různých verzích Androidu a iOS bylo odlišné. Zejména mezi různými verzemi Androidu byl velký rozdíl, jelikož každá používala vlastní verzi vykreslovacího jádra Webkit, ve kterém běžela moje SPA uvnitř Cordova obalu [19].

Tento problém by měl řešit projekt Crosswalk, který vykreslovací jádro v moderní verzi přibalí do aplikace. Tím sice vznikne prostředí konzistentní pro všechny platformy, bohužel ale také dramaticky naroste velikost výsledného balíčku (přibližně o 20MB) [8]. I s vědomím tohoto problému jsem se rozhodl Crosswalk nasadit a vyzkoušet, což se doposud zdá jako fungující volba.

Pro zlepšení výkonu vykreslování pomocí frameworku AngularJS jsem se rozhodl použít službu kešování šablon. Tato služba umožňuje rychlejší vykreslení pohledů díky tomu, že při změně stavu nenačítá šablony ze souborů, ale jsou již uložené (nakešované) v JS. Problémem bylo, že šablony jsem měl umístěny v souborech HTML a vzhledem k navržené struktuře aplikace jsem to nechtěl měnit. Použil jsem tedy doplněk `gulp-angular-templatecache` a doplnil do `gulpfile.js` automatické generování JS souboru s šablonami. V rámci tohoto generování jsem také začal injektovat proměnné sestavení, specificky tedy GIT SHA aktuálního commitu, aktuální GIT branch, apod. Díky tomu pak bylo možné kdykoliv přesně zjistit s jakou verzí aplikace právě pracuji v telefonu.

3.6.2 Local storage

V rámci drobných oprav jsem také začal intenzivněji pracovat s local storage, do kterého jsem postupně ukládal více informací o uživateli.

Zajímavým rozšířením přidaným v této iteraci je možnost ukončení aplikace (dokonce i násilného ukončení / pádu) bez ztráty dat o měřené lekci. Vzorky skóre přijímané při měření jsou ukládány duálně, rovnou do DB a zároveň do local storage. Neukončené lekce jsou pak v DB ignorovány a aplikace se prioritně rozhoduje podle dat v local storage, pokud jde o spuštění

nového měření / navázání již běžícího. V okamžiku korektního ukončení lekce jsou najednou dočasná data v local storage vymazána a záznam v DB dokončen. Tento problém by se dal podobně řešit i s výhradním použitím SQLite, nicméně přístup k local storage je jednodušší a pro malé objemy dat i rychlejší. Dalo by se říci, že toto byla pouze příprava a funkci navázání běžícího měření jsem pak dodělal až později.

3.7 Verze 0.3.1

Vzhledem k tomu, že se snažím používat sémantické verzování [23], tak se dle označení verze dá usuzovat, že jsem se soustředil na opravy chyb.

První zásadní chybou bylo špatné generování diff grafu, kde bylo třeba přepracovat netriviální SQL dotaz (upravit jej pro práci s více složkami skóre) a upravit postprocessing v podobě JS kódu.

Dále se začaly objevovat problémy s udržením spojení, nebo přesněji s jeho obnovením po ztrátě. Funkce pro udržování spojení jsem tedy přesunul do služby komunikace po BLE, která si je měla řešit vnitřně. Ostatním částem programu pak dává informace o stavu připojení pomocí AngularJS mechanismu broadcastů, což není nic jiného, než návrhový vzor publisher/subscriber. Z libovolného kontroleru pak stačí vznést požadavek na službu aby udržovala spojení a není třeba se o více starat.

3.8 Verze 0.4.0

Kromě zavedení používání linteru (JSLint) pro zajištění určité úrovně kódu a drobného refactoringu jsem se v této iteraci opět přesunul k implementaci funkčních požadavků.

3.8.1 Kamarádi z FB

Přidal jsem kontroler a připravil pohled pro kamarády, které jsem plánoval načítat z FB. Bylo nutné nastudovat dokumentaci ke Graph API od Facebooku [9] a doplnit do služby pracující s daty uživatele volání příslušného API endpointu. Potřebný přístupový token jsem již měl načtený a uložený v local storage od přihlášení uživatele. Načtená data jsem se rozhodl kešovat opět v local storage, aby komunikace nemusela probíhat při každém otevření aktivity.

3.8.2 Graf historie

Nahoru do pohledu historie lekcí jsem přidal sloupcový graf, který měl dle požadavků zobrazovat sumu skóre v jednotlivých dnech. SQL dotaz pro tyto účely jsem měl již připraven, stačilo tedy vytvořit pohled a doplnit předzpracování dat pro `Chart.js`.

3.9 Verze 0.4.1

Dále bylo třeba znovu se věnovat opravám chyb. Zadavatel při testování hlásil, že pokud lekce trvá déle a telefon se mezi začátkem a koncem automaticky uzamkne, tak se přestane načítat skóre. Vzhledem k tomu, že se tento problém projevoval pouze na iOS, bylo třeba hledat řešení v dokumentaci Apple. Problém způsobovala direktiva `UIBackgroundModes - bluetooth-central`, kterou bylo třeba doplnit do konfiguračního souboru pro build iOS. Tím bylo zaručeno, že aplikace bude mít možnost komunikace po Bluetooth i v případě zamčené obrazovky. Toto chování mi přišlo jako chyba v pluginu `cordova-plugin-ble-central`, proto jsem ji opravil a vytvořil žádost o začlenění opravy do oficiálního repozitáře [16], čímž jsem chtěl přispět open source komunitě.

Musel jsem také opravit kuriózní chybu v SQL dotazu, která zapříčiňovala nefunkčnost diff grafu za určitých okolností *pouze na některých implementacích SQLite*.

3.10 Verze 0.5.0

V této iteraci jsem se soustředil na vylepšení grafu historie, aby bylo možné přepínat rozsah mezi posledním týdnem (po dnech), měsícem (po dnech) nebo rokem (po měsících). Nad rozsah funkčních požadavků jsem také implementoval posouvání rozsahu, takže je možno otevřít kalendář a vybrat si specifické datum pro počátek rozsahu grafu nebo jednoduše šipkami vpravo/vlevo rozsah posouvat vpřed/vzad.

Kromě grafu jsem doplnil také číselné hodnoty průměru a součtu skóre a času za zobrazený rozsah a zajistil změnu rozsahu i v textovém seznamu lekcí.

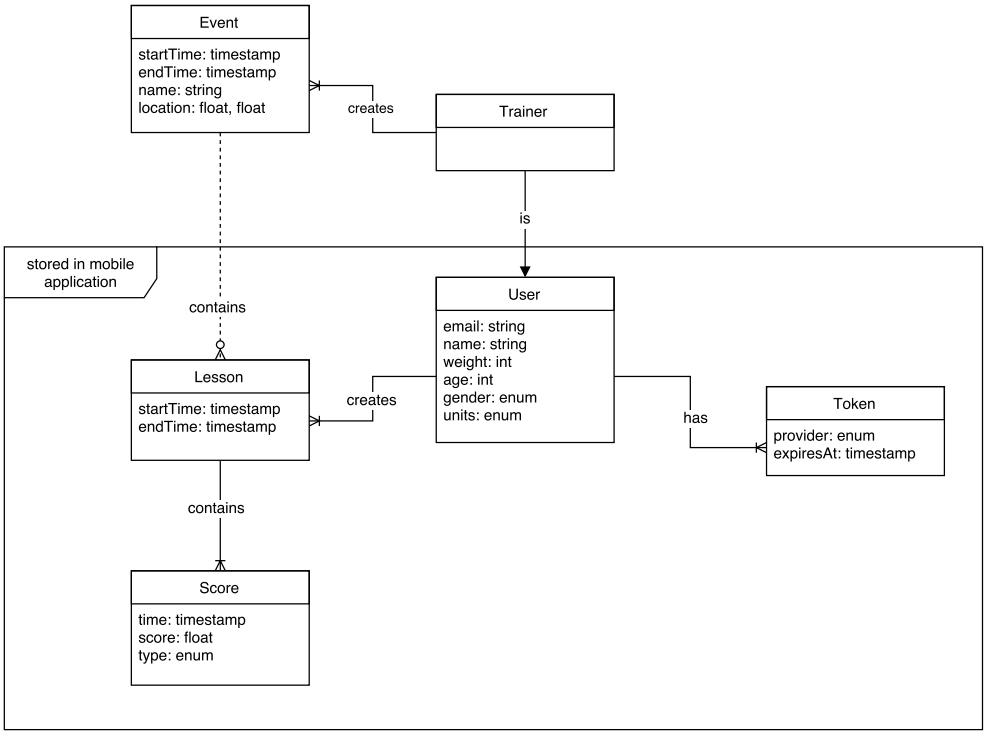
Udělal jsem také několik kosmetických CSS změn a trochu refactoringu.

3.11 API

Dále bylo nutné na základě integračních požadavků navrhnout a implementovat komunikaci s online službou (Jumping World Teamem). Programováním webového systému byla pověřena jiná firma, já dostal za úkol vymyslet rozhraní pro komunikaci. Vytvořil jsem tedy koncepční model pro lepší pochopení ze strany firmy a začal pracovat na návrhu API, na což jsem použil jazyk API Blueprint [4]. Při návrhu jsem se principiálně snažil, aby API mělo REST strukturu [24].

Jako obecné principy jsem stanovil:

- veškerá komunikace ve formátu JSON
- autorizace HTTP hlavičkou pomocí tokenů
- prefixování URL verzí API (např. `.../api/v1/...`)



Obrázek 3.3: Koncepční model problémové domény

- používání hlavičky `ETag` společně s HTTP kódem 304 pro kešování odpovědí
- omezení počtu dotazů za hodinu

Hlavní částí pak byla identifikace jednotlivých zdrojů (k čemuž mi také pomohl koncepční model) a na základě toho návrh jejich struktury (URL + HTTP metody). Dále tedy uvádím pouze stručný přehled, kompletní popis lze nalézt v souboru `api.apib`.

3.11.1 Zdroj uživatelé

- přihlášení přes různé „poskytovatele“ (FB, G+ nebo e-mail) – získání tokenu
- odhlášení – revokace tokenu
- vytvoření nového uživatelského účtu
- získání detailů o uživateli
- aktualizace detailů o uživateli
- obnovení zapomenutého hesla

3.11.2 Zdroj lekce

- získání všech lekcí s možností filtrování dle času startu
- nahrání nových lekcí
- smazání lekce

Všechny operace s lekcemi se týkají pouze aktuálně přihlášeného uživatele. Nedává smysl, aby si s nimi uživatelé mohli jakkoliv manipulovat navzájem.

3.11.3 Zdroj události

- získání událostí dle GPS polohy nebo názvu s možností filtrování dle času startu
- získání detailů o specifické události dle jejího ID

3.11.4 Zdroj kamarádi

- získání všech kamarádů aktuálního uživatele společně s jejich skóre z poslední lekce / týdne / měsíce

3.12 Verze 0.6.0

Hlavní funkcionalitou implementovanou v této verzi bylo upozornění na nízký stav baterie v zařízení. Použil jsem opět osvědčený návrhový vzor publisher/subscriber ve formě AngularJS broadcastů. Registraci notifikací na příslušnou BLE charakteristiku provede komunikační služba již při připojení a vše se vyhodnocuje vnitřně do doby, než je stav baterie 15%. Tehdy se začnou odesílat broadcasty zbytku aplikace, který má možnost s nimi naložit zcela libovolně – v mém případě pak zobrazením dialogu o nízkém stavu baterie.

Kromě několika dalších drobných oprav to bylo vše pro tuto verzi. Délka iterací a objem práce v nich se začal zkracovat/zmenšovat, což přisuzuji tomu, že hlavní funkční požadavky již byly implementovány. Jako poslední větší úkol zbývalo vyřešit integraci API...

3.13 Verze 0.7.0

Konečně jsem se pustil do samotné implementace API na straně aplikace. Pro tento účel jsem vytvořil službu, která obalovala HTTP komunikaci do JS metod a začal ji používat z různých dalších míst kódu – v této iteraci pak nejvíce ve službě pro práci s uživatelskými údaji. Bylo třeba zajistit, aby po přihlášení uživatele pomocí FB nebo G+ aplikace odeslala získané údaje na server, který uživateli založil Jumping účet a přidělil token pro komunikaci s API. Doplnil jsem také několik dalších kontrolerů a pohledů, např. pro reset

hesla, přihlášení existujícího uživatele pomocí e-mailu nebo aktualizaci údajů aktuálního uživatele.

Pro testování jsem používal službu Apiary.io, která dokáže na základě dodaného API Blueprintu vytvořit tzv. mockovací server, který odpovídá na HTTP požadavky přesně dle dokumentace (simuluje tedy chování reálného serveru v ostrém provozu).

3.14 Verze 0.8.0

V této verzi jsem implementoval přepínání rozsahu u kamarádů – poslední lekce / týden / měsíc – dle dat získaných z API. V pohledu jsem použil jsem stejný ovládací prvek, jako v záložce historie, abych uživatelům usnadnil orientaci a používání aplikace.

Přidal jsem také doposud chybějící možnost mazání lekcí, kde jsem použil gesto přejetí prstem do strany.

3.14.1 Synchronizace

Netriviálním problémem se stala implementace obousměrné synchronizace se serverem. Bylo nutné vymyslet způsob, který by dovoloval co nejvyšší paralelizaci procesu (kvůli rychlosti), nepřenášel by zbytečná data (kvůli možným omezením sítě ze strany uživatele) a byl odolný proti chybám (kvůli zachování konzistence) – přičemž bylo nutno zachovat určitou posloupnost kroků [1]. Naprogramoval jsem tedy službu, která obalovala synchronizaci dat uživatele, kamarádů a lekcí. Díky tomu je možno spustit kompletní synchronizaci z libovolného kontroleru pouze zavoláním jedné metody.

3.14.2 Přípravy na veletrh

Společně se zadavatelem jsme začali aplikaci přizpůsobovat pro prezentaci na veletrhu, který se měl zanedlouho konat. Proto jsme se rozhodli přidat možnost přeskočit přihlášení, aby si návštěvníci byli schopni vyzkoušet hlavní funkce i bez svazování aplikace se svým účtem.

Kromě toho bylo také třeba vytvořit demo režim, aby aplikace alespoň zdánlivě byla naplněna nějakými daty. Po dlouhém uvažování a dobré radě od kolegy Filipa jsem se rozhodl jít cestou mockování služby komunikace s API. Vytvořil jsem tedy kopii této služby se stejným JS rozhraním, kde jsem ale odstranil kód pro HTTP komunikaci a místo něj de-facto naprogramoval chování serveru s použitím několika JS objektů jako úložiště. Ve spolupráci se zadavatelem jsem pak pomocí krátkého skriptu v Pythonu nageneroval testovací data a vložil je do této služby. Z hlediska aplikace jako celku tedy přepnutí do demo režimu proběhne pouze výměnou jedné služby za její mockovací variantu (změna jednoho slova ve zdrojovém kódu).

3.14.3 Nová značka

Obrázek 3.4: Fyzické zařízení v novém designu připevněné na botě

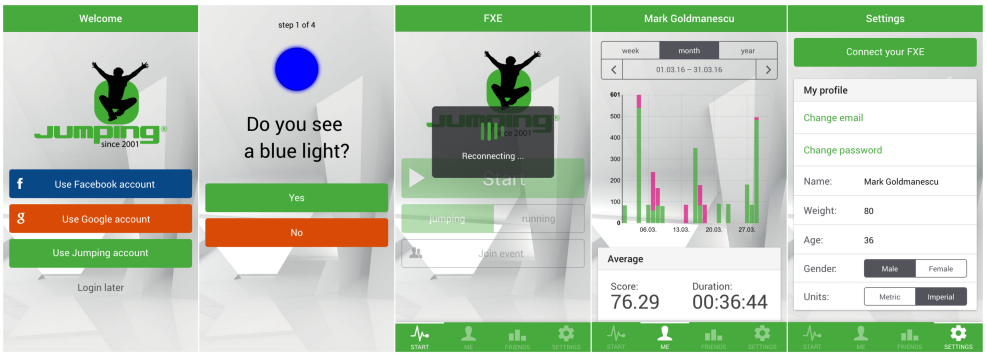
Před veletrhem se zadavatel také rozhodl přebrandovat produkt na nové jméno: FXE [2]. Bylo tak třeba vytvořit nové ikony a spouštěcí obrazovku v grafickém editoru, změnit pozadí a grafiku aplikace a přejmenovat mnoho různých názvů JS balíků apod. V rámci předělávání grafiky jsem udělal také mnoho drobných estetických úprav v přímé spolupráci se zadavatelem.

3.15 Verze 0.9.0

V zatím poslední verzi jsem se kvůli potřebě zvýšení výkonu rozhodl vyměnil knihovnu pro kreslení grafů. Odebral jsem z projektu `Chart.js` a místo toho jsem se soustředil technologii `D3.js`, což je knihovna pro manipulaci s DOM elementy velmi specifickým způsobem („Data-Driven Documents“). To implikovalo jednu věc: potřebu předělat grafy do formátu SVG, z původního HTML canvasu, který používal `Chart.js`. Vzhledem k tomu, že jsem nechtěl znovu ojevovat kolo, snažil jsem se najít vyšší knihovnu, která by využívala výkonu D3 a obsahovala některé běžné typy grafů již připravené. To se mi nakonec povedlo ve formě balíčku `angular-nvd3`, který bere kolekci předpřipravených grafů z `nvd3` (stavějícím nad D3) a to celé obaluje v AngularJS direktivě. Zvýšení výkonu pak bylo velmi znatelné, nejvíc v průběhu animace sloupcových grafů v záložce historie. Získal jsem tím také větší možnosti konfigurace a stylování, jelikož komponenty grafu jsou vlastně běžné DOM elementy.

Kvůli nadměrně se rozrůstající velikosti souboru bylo nutné provést refactoring služby pro komunikaci po BLE, kterou jsem rozdělil na dvě části.

[2] Bohužel, název této práce tímto krokem pozbyl na smyslu. Proto v práci stále opakuji slovní spojení „fyzické zařízení“, protože v současné době by použití starého názvu „Experience“ bylo nepřesné.



Obrázek 3.5: Některé z obrazovek aplikace ke konci 9. iterace

Jedna obaluje opravdu nízkoúrovňové metody pro komunikaci (a jejich chybové stavy) do AngularJS služby a systému JS promises + přidává několik metod užitečných pro spárování jednoho konkrétního zařízení. Druhá vytvořená služba obsahuje business logiku aplikace a fyzického zařízení jako například nutná sekvence příkazů pro spuštění měření nebo notifikace na nízký stav baterie.

Před vydáním jsem ještě přidal možnost měření běhu (kromě jumpingu), koncept pohledu pro výběr událostí a obrazovku s možnostmi pro vývojáře skryl pomocí známého Android triku – vícenásobného kliknutí na číslo verze aplikace.

Závěr

Podařilo se mi vytvořit aplikaci dle většiny zadaných požadavků a projít několika iteracemi vývoje a testování ze strany zadavatele. V těchto iteracích jsme postupně upravovali funkčnost na základě okamžité zpětné vazby. Také se vcelku dobře podařilo koordinovat činnost společně s kolegou vyvíjejícím HW a FW a vytvořit + schválit rozhraní pro komunikaci se serverovou částí.

Vývoj aplikace je aktuálně pozastaven kvůli jiným školním povinnostem. Přesto však už teď probíhá sběr požadavků do další verze a po skončení semestru budu v implementaci pokračovat. Technicky je vše připraveno pro uvolnění aplikace jako open source, stačilo by jediné kliknutí na Githubu. Přáním zadavatele však bylo aplikaci neuvolňovat do doby vydání v App Store a Google Play.

K dnešnímu dni jsem na projektu v GITu udělal *205 commitů* a napsal celkem *3433 čistých řádků kódu*.

Tabulka 3.1: Počet řádků kódu dle jazyka.

jazyk	soubory	prázdné	komentáře	kód
Javascript	26	586	201	2670
HTML	17	74	21	461
SASS	1	0	10	291
Bourne Shell	1	4	0	11
Celkem	**45**	**664**	**232**	**3433**

Největší nedostatky

Momentálně největším problémem aplikace je prakticky chybějící dokumentace a malé množství komentářů. Programátorskou dokumentaci jsem zanedbával hlavně proto, že struktura aplikace se v prvních iteracích výrazně mě-

nila a bylo by nutné stále upravovat i dokumentaci. Tím jsem se ale vytvořil dluh, který je třeba co nejdříve řešit – ideálně před uvolněním na Github.

Dále je v aplikaci více míst, které by si zasloužily rozdělit na drobnější samostatné jednotky, podobně jako jsem rozdělil službu pro BLE komunikaci. Dá se tedy říci, že by bylo dobré vyhradit jednu celou iteraci na refactoring a vyčištění programu. Díky tomu bych mohl navíc uvolnit části kódu jako znovupoužitelné moduly do open source JS komunity.

Velkou bolestí je také absence jakýchkoliv automatických testů, které se daly konzolově spouštět. Po zmiňovaném rozdělení na menší JS moduly tedy bude třeba zapracovat na jejich pokrytí testy. To bude určitě časově náročné, jelikož s testováním mobilní aplikace zatím nemám žádné zkušenosti – bude třeba nastudovat mnoho nových postupů a nástrojů.

Rád bych také zapracoval na variabilnějším systému skóre, který je momentálně pevně svázán s jednotlivými BLE charakteristikami. Toto pevné mapování bych chtěl odstranit a připravit se tak na budoucí rozšiřování skórovacího systému o další složky nebo sporty.

Výhled do budoucna

Ze strany zadavatele se jedná primárně o vydání aplikace pomocí oficiálních distribučních kanálů pro dané platformy. Pro to bude nejspíše třeba doladit několik technických detailů, které se dozvím až po podání návrhu na publikaci aplikace v App Store v rámci jejich schvalovacího procesu. Dále by zadavatel rád rozšiřoval možnosti aplikace o měření dalších sportů, konkrétně běhu a cyklistiky.

Já osobně vidím největší potenciál v ladění skórovacího algoritmu. Sice se nejedná o součást aplikace, ale myslím, že například použití neuronových sítí pro hodnocení fitness aktivity by mohlo přinést velmi zajímavé výsledky, namísto použití námi vymyšleného algoritmu.

Ze strany aplikace pak výzvou a zároveň nutností bude implementace BLE DFU profilu - tedy možnosti nahrání nového firmware do zařízení. Bude třeba na úrovni AngularJS služby implementovat netriviální protokol od firmy Nordic [2] a také vytvořit novou verzi API s možností stažení nejnovějšího firmware. Tím bude zajištěna možnost kompletní aktualizace obou částí projektu.

Rád bych také nastavil a zprovoznil continuous delivery server [21], abych v budoucnu byl chopen co nejrychleji dostat úpravy a opravy aplikace mezi uživatele. V rámci něj bych chtěl automaticky spouštět napsané testy, abych minimalizoval riziko vydání chybné aplikace.

Výhledově by se pak dalo uvažovat nad přepsáním aplikace do nové verze frameworku Ionic 2, která je založena na TypeScriptu. To je však obrovský úkol, který by vyžadoval vytvoření všeho znovu prakticky od nuly. Navíc Ionic 2 je aktuálně v beta verzi, takže bych raději počkal, než vývojáři vyladí počáteční problémy a API frameworku se ustálí.

Literatura

[1] *How best do you represent a bi-directional sync in a REST api?* [online]. Stack Exchange, Inc, 2012 . Dostupné z: http://programmers.stackexchange.com/q/135412

[2] *BLE DFU Profile* [online]. Oslo, Norsko: Nordic Semiconductor ASA, 2015 . Dostupné z: http://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk51.v10.0.0/bledfu_transport_bleprofile.html?cp=4_0_1_4_3_1_4_0

[3] *About Jumping* [online]. Nové Město, ČR: JSTB International s.r.o., 2016 . Dostupné z: http://www.jumping-fitness.com/en/about-jumping

[4] *API Blueprint* [online]. San Francisco, USA: Apiary Inc., 2016 . Dostupné z: https://github.com/apiaryio/api-blueprint

[5] *App Store* [online]. Cupertino, USA: Apple Inc., 2016 . Dostupné z: https://developer.apple.com/support/app-store/

[6] *Arduino Pro Mini 328* [online]. Niwot, USA: SparkFun Electronics, 2016 . Dostupné z: https://www.sparkfun.com/products/11113

[7] *Endomondo, Runkeeper, Runtastic, Nike+ Running* [online]. Mountain View, USA: Google Inc., 2016 . Dostupné z: https://play.google.com/store/apps

[8] *Frequently-asked questions* [online]. Mountain View, USA: Intel Corporation, 2016 . Dostupné z: https://crosswalk-project.org/documentation/about/faq.html

[9] *The Graph API* [online]. Menlo Park, USA: Facebook Inc., 2016 . Dostupné z: https://developers.facebook.com/docs/graph-api

[10] *MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass) MEMS MotionTracking^{TM} Device* [online]. San Jose, USA: InvenSense, Inc., 2016 . Dostupné z: http://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/

[11] *Nordic Semiconductor nRF51822-QFAC-R7* [online]. Mansfield, USA: Mouser Electronics, Inc., 2016 . Dostupné z: http://cz.mouser.com/Search/ProductDetail.aspx?qs=ts7L7qAQc8ciTsNcr7%2fvxQ%3d%3d

[12] *Platform Versions* [online]. Mountain View, USA: Google Inc., 2016 . Dostupné z: http://developer.android.com/about/dashboards/index.html#Platform

[13] *Trello* [online]. New York, USA: Trello, Inc., 2016 . Dostupné z: https://trello.com/

[14] Archibald, J.: *JavaScript Promises* [online]. Mountain View, USA: Google Inc., 2013 . Dostupné z: http://www.html5rocks.com/en/tutorials/es6/promises/

[15] Bawcombe, L.: *The Hamburger Menu-Icon Debate* [online]. Washington, USA: Atlantic Media, 2014 . Dostupné z: http://www.theatlantic.com/product/archive/2014/08/the-hamburger-menu-debate/379145/

[16] Bedřich, T.: *Fixed iOS notifications, Pull Request #176, don/cordova-plugin-ble-central* [online]. San Francisco, USA: GitHub, Inc., 2016 . Dostupné z: https://github.com/don/cordova-plugin-ble-central/pull/176

[17] Bedřich, T.: *Incompatibility with cordova-plugin-facebook4, Issue #213, EddyVerbruggen/cordova-plugin-googleplus* [online]. San Francisco, USA: GitHub, Inc., 2016 . Dostupné z: https://github.com/EddyVerbruggen/cordova-plugin-googleplus/issues/213

[18] Bedřich, T.: *Incompatibility with cordova-plugin-googleplus, Issue #193, jeduan/cordova-plugin-facebook4* [online]. San Francisco, USA: GitHub, Inc., 2016 . Dostupné z: https://github.com/jeduan/cordova-plugin-facebook4/issues/193

[19] Bergman, J.: *What WebKit version is in what Android version?* [online]. 2012 . Dostupné z: http://jimbergman.net/webkit-version-in-android-version/

[20] Boehm, B.: Anchoring the Software Process. *IEEE Software*, ročník 13, č. 4, 1996: s. 73–82, doi:10.1109/52.526834.

[21] Brown, S.: *Continuous Delivery with Ionic, iOS and TravisCI* [online]. 2015 . Dostupné z: http://blog.samuelbrown.io/2015/10/20/continuous-delivery-with-ionic-ios-and-travisci/

[22] Heikkinen, I.: *Typed Arrays* [online]. Mountain View, USA: Google Inc., 2012 . Dostupné z: http://www.html5rocks.com/en/tutorials/webgl/typed_arrays/

[23] Preston-Werner, T.: *Semantic Versioning 2.0.0* [online]. 2015 . Dostupné z: http://semver.org/spec/v2.0.0.html

[24] Sahni, V.: *Best Practices for Designing a Pragmatic RESTful API* [online]. 2015 . Dostupné z: http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api

[25] Shabir, N.: *Logging client side errors server-side in AngularJS* [online]. Talis, 2014 . Dostupné z: http://engineering.talis.com/articles/client-side-error-logging

[26] Stables, J.: *Best fitness trackers 2016: Jawbone, Misfit, Fitbit, Garmin and more* [online]. London, UK: Wareable, 2016 . Dostupné z: http://www.wareable.com/fitness-trackers/the-best-fitness-tracker

[27] Vidyarthi, A.: *Publish/Subscribe* [online]. 2012 . Dostupné z: http://learning-0mq-with-pyzmq.readthedocs.org/en/latest/pyzmq/patterns/pubsub.html

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface
BLE Bluetooth Low Energy
BT Bluetooth
CSS Cascading Style Sheets
DB Database
DFU Device Firmware Update
DOM Document Object Model
FB Facebook
FEL Fakulta elektrotechnická
FW Firmware
G+ Google+
GATT Generic Attribute Profile
GPS Global Positioning System
GUI Graphical User Interface
HR Heart Rate
HTML HyperText Markup Language
HTTP Hypertext Transfer Protocol
HW Hardware
IDE Integrated Development Environment
IMU Inertial Measurement Unit
JS JavaScript
JSON JavaScript Object Notation
LED Light-Emitting Diode
MAC Media Access Control
PDF Portable Document Format
REST Representational State Transfer
RGB Red-Green-Blue
SASS Syntactically Awesome Stylesheets
SDK Software Development Kit
SHA Secure Hash Algorithm

SPA Single-Page Application
SQL Structured Query Language
SVG Scalable Vector Graphics
SW Software
URL Uniform Resource Locator
USB Universal Serial Bus
ZMQ ZeroMQ
ČVUT České vysoké učení technické v Praze

Příloha **B**

Slovník pojmů

Pro přehlednost zde uvádím vysvětlení některých pojmů z problémové domény.

Uživatel používá aplikaci. Ve většině případů se do aplikace přihlásí a tím se také stává uživatelem online služby.

Trenér je speciální případ uživatele, který má na webovém portálu oprávnění mimo jiné na zakládání událostí. Z hlediska aplikace se však jedná o naprosto běžného uživatele.

Událost je časový rozsah vázaný na fyzické místo. Událost může založit pouze trenér mimo aplikaci.

Položka skóre je časová značka společně s absolutní hodnotu skóre ve formě nezáporného reálného čísla a typem (označením složky).

Lekce je časový rozsah spuštěný a ukončený uživatelem pomocí kliknutí na tlačítko v aplikaci. Lekce *mohou* být asociovány s událostmi a *mohou* obsahovat mnoho položek skóre (typicky řádově stovky). Výsledné skóre lekce je součtem maximálních položek skóre pro každou složku.

Kamarád je jiný uživatel aplikace, který zároveň má sociální spojení na FB nebo G+ s aktuálním uživatelem *včetně jeho samého*.

Diff graf je interpolovaný čárový graf, kde na ose X jsou dvouminutové časové bloky a na ose Y je součet maximálních položek skóre pro každou složku v těchto blocích.

Poskytovatel přihlášení je garant pravosti údajů uživatele. Může jím být buď online systém Jumping World Team, FB nebo G+.

Příloha C

Obsah přiloženého média

Médium obsahuje elektronickou verzi této práce a samotnou implementaci. Ta je zabalena v ZIP archivu `implementation.zip`, jehož obsah uvádím níže. Integritu tohoto archivu lze ověřit provedením kontrolního součtu SHA256 a porovnáním s následujícím výstupem:

```
$ shasum -a 256 -p implementation.zip | cut -d " " -f 1
be0aead169e47e30e0ab1ddd253b2c2007f2a4ffd1abe83b33879a2d6cf73f9d
```

- `implementation.zip/`
  - `app/` ..... hlavní zdrojové kódy v JS
  - `build/` ..... zkompilované balíčky pro iOS a Android
  - `scss/` ..... styly aplikace ve formátu SASS
  - `templates/` ..... šablony pohledů v HTML
  - `www/` ..... kořenový adresář výsledné aplikace
  - `api.apib` ..... specifikace API ve formátu API Blueprint
  - `README.md` ..... návod pro instalaci
- `thesis/`
  - `chapters/` ..... jednotlivé kapitoly ve formátu Pandoc Markdown
  - `img/` ..... obrázky v původní velikosti
  - `bibliography.bib` ..... literární zdroje
  - `BP_Bedrich_Tomas_2016.pdf` ..... výsledná práce ve formátu PDF
  - `BP_Bedrich_Tomas_2016.tex` ..... text práce ve formátu LaTeX