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

Diplomová práce

Mobilní platforma pro sběr dat z biosenzorů

Plzeň 2016 Bc. Luboš Müller

Prohlášení

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

V Plzni dne 8. května 2016

Bc. Luboš Müller

Poděkování

Děkuji Ing. Petru Ježkovi, Ph.D. za pomoc při vedení diplomové práce.
Mé poděkováni patři též přítelkyni a rodině za podporu v průběhu studia.

Abstract

Wireless sensors for measuring activities of the human body provide data that can be used for performing experiments or for subsequent analysis. Data must be retrieved and presented. Suitable platform for obtaining data from the sensors is a mobile phone application that allows communication with the subset of sensors using ANT + technology and Bluetooth.
Data retrieved by the application can be paired with experiments defined in EEGBase Portal. Because some sensors provide data online while performing the activity, the application will display this data either in the form of graph or text.

Abstrakt

Bezdrátové senzory pro měření aktivit lidského těla poskytují data, která lze použít k provádění experimentů nebo k následné analýze. Data je nutné určitým způsobem načíst a prezentovat. Vhodnou platformou pro získání dat ze senzorů je mobilní telefon s aplikací, která umožní komunikaci s danou podmnožinou senzorů pomocí technologií ANT+ a Bluetooth.
Data načtená aplikací bude možné spárovat s experimenty definovanými v portálu EEGBase. Protože některé senzory poskytují data online při vykonávání aktivity, bude aplikace tato data zobrazovat buď ve formě grafu nebo textově.

Obsah

1 Úvod

Výzkumná skupina neuroinformatiky na katedře informatiky a výpočetní techniky se specializuje na provádění experimentů a jejich následné zpracování a vyhodnocování experimentálních dat v oblasti biologických signálů, především EEG/ERP signálů.

Dostupnost bezdrátových technologií umožnila vytvoření nespočtu senzorů pro měření aktivit lidského těla. Jedná se například o měření srdečního tepu, tělesné hmotnosti, nebo od aktivity oddvíjející se veličiny jako je rychlost či počet ušlých kroků. Senzory mohou komunikovat například s aplikacemi v chytrém mobilním telefonu. Data poskytovaná senzory lze použít pro následnou analýzu nebo pro provádění experimentů.

Cílem práce je vytvořit mobilní aplikaci, která bude komunikovat s podmnožinou dostupných senzorů. Aplikace umožní načítání dat ze senzorů. Tato data bude možné spárovat s experimenty definovanými v portálu EEGBase. Protože některé senzory poskytují data online při vykonávání aktivity, bude aplikace data prezentovat přímo ve formě grafu nebo textově.

V kapitole 2 se zaměřuji na analýzu dostupných prvků navrhovaného systému. Čtenáře seznámím s bezdrátovými technologiemi implementovanými v senzorech, které jsou v rámci této práce použité. Dále je v této kapitole uvedena manipulace s jednotlivými senzory, jaké informace senzory poskytují a jak se s nimi pracuje. Následuje ukázka stávající mobilní aplikace pro sběr dat, na kterou nově navrhovaná aplikace navazuje. Naposled se tato kapitola zabývá částmi portálu EEGBase, které jsou ovlivněny touto prací. Kapitola 3 je věnována návrhu mobilní aplikace, seznámení s hybridními mobilními aplikacemi a popisu funkcí a ovládání aplikace. Je zde kladen důraz na přívětivost uživatelského rozhraní. Kapitola představuje frameworky pro vývoj hybridních aplikací a způsob komunikace se senzory v rámci mobilní aplikace. Kapitola 3 se soustředí na sestavení aplikace a její publikování pro koncového uživatele. V kapitole 4 je představena implementace úložiště získaných dat na straně portálu EEGBase. Je zde ukázána implementace webových služeb. Také jsou zde diskutovány problémy stávajícího systému vzhledem k vyvijené mobilní aplikaci. Kapitola 5 se zabývá analýzou naměřených dat v mobilní aplikaci. Kapitola 6 popisuje testování. Je zde ukázáno testování jak pomocí dostupného hardware tak za použití softwarových nástrojů a simulátorů konkrétních senzorů.

2 Seznámení s dostupnými prvky navrhovaného systému

2.1 Komunikační technologie ANT/ANT+

ANT je ultra-low power (ULP) bezdrátový protokol, který umožňuje propojení zařízení, které měří nebo zobrazují různá data (např. aktivity lidského těla). Ultra-low-power znamená, že protokol minimalizuje nároky na spotřebu elektrické energie. ANT zařízení jsou schopna fungovat několik let 24 hodin denně na jednu knoflíkovou baterii.

2.1.1 ISO/OSI model protokolu ANT

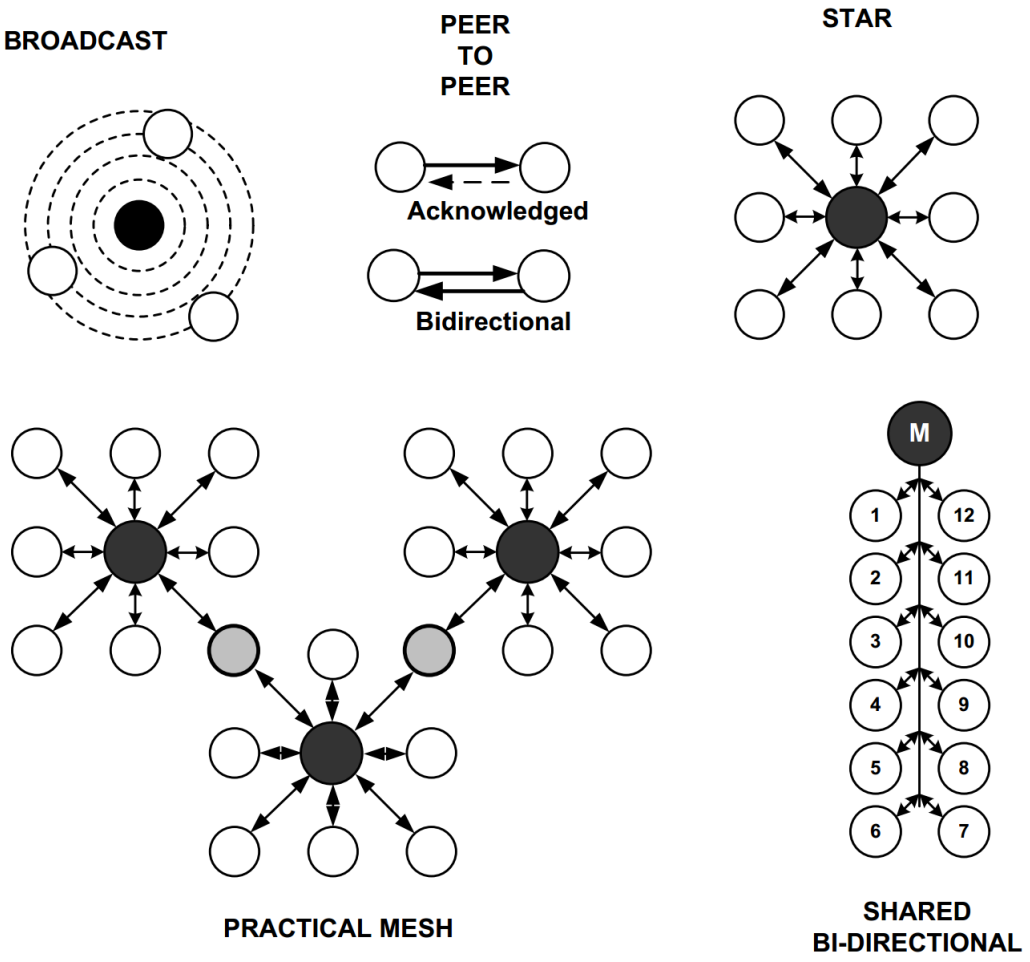
Z tabulky 2.1 je patrné, že ANT implementuje vrstvy 1-4 ISO/OSI modelu. Protokol ANT je tedy zodpovědný za přenos dat. Podporu aplikací zajišťují až jednotlivé ANT+ profily pracující na vrstvách ISO/OSI modelu 5-7.

2.1.2 Topologie sítě

Protokol ANT byl navržen tak, aby podporoval širokou škálu síťových topologií. Bezdrátové sítě ANT podporují topologie typu peer to peer (P2P), hvězda, mesh nebo strom. Naopak kruhové topologie nejsou podporovány. Na obrázku 2.1 jsou vybrané topologie ANT sítí podle [46].

<table>
<tr><td></td><td>ISO/OSI model</td><td></td><td></td></tr>
<tr><td>7.</td><td>Aplikační</td><td rowspan="3">ANT+ profil</td><td rowspan="7">ANT+ protokol</td></tr>
<tr><td>6.</td><td>Prezentační</td></tr>
<tr><td>5.</td><td>Relační</td></tr>
<tr><td>4.</td><td>Transportní</td><td rowspan="4">ANT protokol</td></tr>
<tr><td>3.</td><td>Síťová</td></tr>
<tr><td>2.</td><td>Linková</td></tr>
<tr><td>1.</td><td>Fyzická</td></tr>
</table>

Tabulka 2.1: ISO/OSI model protokolu ANT



Obrázek 2.1: Vybrané topologie protokolu ANT podle [46]

2.1.3 ANT+ profily

ANT+ profil je skupina pravidel pro komunikaci po síti. Profil se vztahuje vždy k určitému typu zařízení - například k senzoru srdečního tepu. V době psaní této práce existuje podle webu thisisant.com [45] celkem 611 registrovaných produktů, které implementují ANT+. Z toho je 68 mobilních aplikací. Každý profil v sobě zahrnuje nastavení parametrů komunikačního kanálu, formát přenášených dat a další specifické mechanismy potřebné pro komunikaci mezi dvěma zařízeními. Zda produkt podporuje určitý profil lze poznat poznat podle ikony profilu, která bude u produktu vyobrazená. V rámci ANT+ je implementováno téměř dvacet profilů.Některé z nich jsou následující:

- **Heart Rate Monitor** - měřič tepové frekvence je primárně určený k měření počtů tepů za minutu v reálném čase při provádění dané činnosti. Senzory pro měření tepové frekvence se zpravidla nosí v podobě hrudního pásu.

Obrázek 2.2: Ikona profilu HR

- **Foot Speed** - profil vyvinutý pro krokoměr nebo senzor pro měření rychlosti chůze. Podobně jako předchozí senzor umožňuje měření v reálném čase. Poskytuje např. údaje o ušlé vzdálenosti, rychlosti, počtu kroků, kadenci nebo spálených kaloriích.

Obrázek 2.3: Ikona profilu SPD

- **Weigth Scale** - profil váha se používá v zařízeních pro měření tělesné hmotnosti. V případě, že uživatel o sobě poskytne další údaje (výška, věk), lze na jejich základě výpočítat např. index tělesné hmotnosti. Některé váhy jsou schopné si pamatovat více uživatelských profilů, mezi kterými lze přepínat.

Obrázek 2.4: Ikona profilu WGT

- **Bike Speed and Cadence** - profil pro cyklo počítače. Podle typu a složitosti zařízení se tento profil dělí na tři podprofily a to samostatné profily pro měření rychlosti a kadence šlapání a společný profil dvou předchozích viz. obrázek 2.5. Opět jde o profily pro měření dat v reálném čase. Data jsou okamžitě posílána do dalších zařízení, která je mohou zpracovat (hodinky, mobilní telefon).

Obrázek 2.5: Ikony profilu SPD, CAD a S&C

- **Muscle Oxygen Monitor** - profil pro měření procenta kyslíkem nasyceného hemoglobinu ve svalech. Tato hodnota se zmenšuje s aktivitou svalu (třeba v případě zátěže při cvičení). Snižuje se, když se vlivem krevního oběhu sval více okysličí. Zařízení s tímto profilem jsou často využívána při tréninku atletů. Znovu se jedná o senzory, které poskytují data reálném čase.

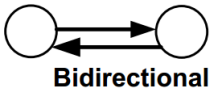
Obrázek 2.6: Ikona profilu MO_2

2.2 Komunikační technologie Bluetooth

Mezi protokoly pro bezdrátovou komunikaci se řadí také Bluetooth. Stejně jako ANT operuje ve frekvenčím pásmu 2400 MHz. Technologie Bluetooth je definována standardem IEEE 802.15.1 [17].

2.2.1 Topologie sítě

Technologie Bluetooth podporuje dva typy komunikace – dvoubodovou a vícebodovou. V případě dvoubodového přenosu se jedná o klasickou point to point komunikaci, tedy přímé spojení dvou zařízení [23]. V rámci této práce se využívá dvoubodová komunikace viz. obrázek 2.7.



Obrázek 2.7: Dvoubodová komunikace

2.2.2 Bluetooth Smart (Low Energy)

V současné době stále rozšířenější standard Bluetooth verze 4.0 a vyšší je především zaměřený na co nejnižší spotřebu energie. Další novinkou této verze je podpora tzv. Generic Attribute Profile(GATT), který vytváří struktorovaný systém pro přenost dat na základě daných profilů. Nízkou energetickou náročností a danými profily pro různé typy zařízení vytváří Bluetooth Smart značnou konkurenci pro ANT+.

2.2.3 Bluetooth profily

Opět v analogii s ANT/ANT+ pro každou činnost Bluetooth (Smart) používá jiný profil a každé zařízení obsahuje jinou sadu profilů podle toho, k čemu je využíváno.

Mezi Bluetooth profily patří:

- **Headset Profile (HSP)** - poskytuje podporu pro sluchátka ve spojení např. s mobilním telefonem nebo osobním počítačem.
- **Human Interface Device Profile (HID)** - profil pro ovládání zařízení jako jsou klávesnice, myši nebo joysticky.
- **Health Device Profile (HDP)** - profil pro přenos medicínkých dat. Do této kategorie spadají senzory pro měření srdečního tepu nebo teploměry. Bohužel kvůli absenci zařízení pro testování není HDP profil v rámci této práce dále prozkoumáván.

2.3 Biosenzory dostupné v laboratoři bioinformatiky na KIV

Laboratoř bioinformatiky na KIV využívá různé senzory a vybavení pro provádění experimentů. Kvůli vývoji mobilní aplikace jsem si vypůjčil hrudní pás a tonometr. K dispozici byla také osobní váha, ale tu jsem nakonec z větší části testoval pomocí simulátoru.

2.3.1 Hrudní pás

Hrudní pás se skládá ze snímače tepové frekvence Garmin Premium [30], který je připevněný na průžném pásu s elektrodami viz. obrázek 2.8. Snímač je krytý v plastovém pouzdře bez dalších ovládacích prvků nebo displaye. Snímač srdečního tepu je třeba nosit přímo na kůži těsně pod hrudní kostí.

Obrázek 2.8: Hrudní pás Garmin Premium

Musí být upevněn tak, aby zůstal při prováděných aktivitách na místě. Pro zlepšení vodivosti elektrod, je výrobce doporučuje lehce navlhčit. Po nasazení začne senzor snímat a vyhodnocovat data srdečního tepu a tato data okamžitě vysílá.

Senzor Garmin Premium implementuje ANT+ profil Heart Rate Monitor (HR). Díky tomuto profilu lze poměrně jednoduše aktivovat propojení mezi tímto senzorem a Android aplikacemi. Konkrétní řešení propojení pomocí pluginů ANT+ SDK [47] bude řešeno v následující kapitole, zde uvádím data poskytovaná senzorem [36]:

- čas posledního úderu srdce
- čas předchozího úderu srdce

- kumulativní počet úderů srdce
- srdeční tep
- R-R interval
- kumulativní operační čas senzoru
- ID výrobce a sériové číslo senzoru
- data specifická pro výrobce

2.3.2 Osobní váha

Druhý senzor z laboratoře KIV je osobní váha (viz. obrázek 2.9) od společnosti eVito. Tato osobní váha je založena na principu bioelektrické impedanční analýzy (BIA), kdy pomocí slabého a bezpečného elektrického signálu vysílaného do těla, a s ohledem na jednotlivé údaje jako věk, výška, pohlaví nebo stupně aktivity, je možné identifikovat sledovaná data.

Na osobní váze lze nastavit 8 uživatelských profilů. Zařízení nedisponuje vlastní pamětí. Prostřednictvím bezdrátové technologie ANT+ je možné naměřené hodnoty přenášet do mobilní aplikace, a to opět s využitím ANT+ SDK, protože váha implementuje profil Weight Scale (WGT) [26]. eVito inteligentní váha SL poskytuje tato data [36]:

- tělesná hmotnost (kg)
- procento hydratace těla
- procento tělesného tuku
- procento svalové hmoty
- hmotnost kostí (kg)
- klidová potřeba energie (kcal/den)
- aktivní potřeba energie (kcal/den)

Obrázek 2.9: eVito inteligentní váha SL

2.3.3 Tonometr

Tonometr FORA P30 Plus FC (viz. obrázek 2.10) je pažní tlakoměr, který umožňuje kontrolovat výši krevního tlaku a srdeční puls. Krevní tlak a srdeční frekvence jsou automaticky zaznamenávány pomocí integrovaného tlakového senzoru a oscilometrické měřicí metody. Tlakoměr má vnitřní pamět, naměřená data jsou automaticky uchovávána.

Tlakoměr fora komunikuje s ostatními zařízeními pomocí technologie Bluetooth. Je to zároveň jedinný Bluetooth senzor použitý v této práci. Bohužel tlakoměr neimplementuje Blood Pressure Profile - BLP určený pro tento typ zařízení. Po připojení komunikace probíha jako přes sériový port (Serial Port Profile - SPP). Aby mohla probíhat komunikace s mobilní aplikací, je nejprve nutné tlakoměr spárovat s mobilním telefonem.

Kvůli absenci profilu BLP bylo nutné analyzovat komunikační protokoly tohoto zařízení. Využil jsem znalostí převzatých z kapitoly 5.2.3 diplomové práce Mobilní aplikace pro monitoring pohybu a elektrofyziologických potenciálů [36].

Obrázek 2.10: Tonometr FORA P30 Plus

Tlakoměr poskytuje tato data:

- naměřený střední arteriální tlak
- vypočítaný systolický tlak
- vypočítaný diastolický tlak
- naměřenou tepovou frekvenci
- čas měření

2.4 Ostatní senzory

Některé senzory nebyly dostupné v neurolaboratoři KIV. Přesto jsem se rozhodl načítání dat z těchto senzorů implementovat. Analýza senzorů byla ztížená tím, že jsem neměl k dispozici konkrétní zařízení. Pracoval jsem pouze se specifikacemi senzorů a dokumentací k ANT+ profilům. Ověřování funkčnosti načítání dat probíhalo pomocí softwarových simulátorů (více v kapitole 6 Testování). Zaměřil jsem se na produkty společnosti Garmin. Důvodem byla velice dobrá zkušenost se zapůjčeným senzorem srdečního tepu.

2.4.1 Krokoměr

Prvním zpracovávaným zařízením bez hardware dostupného pro testování je krokoměr. Pro ilustraci použití jsem vybral produkt Garmin Foot Pod (viz. obrázek 2.11).

Obrázek 2.11: Krokoměr Garmin Foot Pod

Garmin Foot pod je malé a velmi lehké zařízení, které se připevňuje nejčastěji na nártovou stranu boty. Krokoměr se skládá ze dvou hlavních částí - z plastové spony a samotného senzoru, který je schovaný v plastovém pouzdře s knoflíkovou baterií. Baterie vydrží cca 1 rok. Podobně jako senzor srdečního tepu krokoměr neobsahuje žádné ovládací prvky nebo display. Cena nožního senzoru se pohybuje okolo 1800 Kč [2].

Garmin krokoměr imlementuje ANT+ profil Foot Speed (SPD). Po spárování a připojení krokoměru k mobilní aplikaci začne krokoměr vysílat data. Díky ANT+ Android SDK je možné získat tyto informace:

- změřená okamžitá rychlost včetně času měření
- změřená okamžitá kadence včetně času měření
- celková vzdálenost
- počet kroků
- kumulativní operační čas senzoru

- kumulativní počet spálených kalorií
- stav senzrou (umístění, stav baterie, zda je aktivní)
- informace o výrobci a produktu(čísla hardware, software a modelu zařízení)

2.4.2 Snímač rychlosti

Snímač rychlosti vybraný pro tuto práci je zařízení, které implementuje ANT+ profil Bike Speed (SPD). Je to senzor, který je určený pro cyklistiku. Snímač se zpravidla připevňuje na zadní náboj kola. Senzor je ukrytý v plastovém pouzdře bez ovládacích prvků. Je obalený ještě gumovým pouzdrem s pružnou přezkou. Ta zajišťuje jednouduchou instalaci k náboji. Cena senzoru je cca 1000 Kč [31].

Senzor se obvykle používá při indoor jízdě na cyklistickém trenažéru, nebo v případě, kdy je třeba měřit šikmou vzdálenost (GPS obvykle měří vodorovné vzdálenosti). Měření dat probíhá na základě počítání otáček kola.

Podle stránek výrobce probíhá kalibrace senzoru automaticky po spárování se sporttesterem [31]. V případě mobilní aplikace v této práci je třeba pro správnost výpočtu rychlosti a vzdálenosti zadat obvod kola.

Ve spojení s ANT+ Android SDK senzor poskytuje tyto informace:

- spočítaná okamžitá rychlost včetně času
- vypočítaná celková vzdálenost
- kumulativní otáčky
- kumulativní operační čas senzoru
- informace o výrobci a produktu(čísla hardware, software a modelu zařízení)

Obrázek 2.12: Snímač rychlosti Garmin

2.4.3 Snímač kadence

Druhým ze skupiny cyklistických senzorů je snímač kadence šlapání Garmin implementující ANT+ profil Bike Cadence (CAD) [29]. Podobně jako předchozí senzory i tento je ukrytý v odolném plastovém pouzdru (viz obrázek 2.13). Pomocí gumových pásků se připevňuje na vnitřní stranu kliky (na ní jsou nasazené pedály). Pro snímání není potřeba žádná další pomůcka (např. magnet), protože senzor má v sobě zabudovaný akcelerometr pro snímání otáček. Cena tohoto zaříze se pohybuje okolo 1000 Kč.

Obrázek 2.13: Snímač kadence Garmin

Ve spojení s ANT+ Android SDK poskytuje snímač rychlosti tyto informace:

- spočítaná okamžitá kadence šlapání
- informace o výrobci a produktu(čísla hardware, software a modelu zařízení)

2.4.4 Kombinovaný snímač rychlosti a kadence

Posledním z trojice cyklistickým senzorů je snímač, který kombinuje oba předchozí. Pro ukázku použití jsem vybral opět produkt společnosti Garmin, konkrétně typ GSC 10 (viz. obrázek 2.14).

Tento výrobek se skládá ze dvou magnetů a hlavního snímače. První magnet se montuje na drát kola a slouží pro snímání rychlosti. Druhý magnet se připevňuje na kliku, pomáhá snímat kadenci. Čidlo se musí umístit na trubku zadní vidlice kola tak, aby bylo v dosahu obou magnetů. Tento kombinovaný snímač se liší od předchozích tím, že má na sobě umístěné ovládací prvky - tlačítko pro resetování přístroje a LED kontrolku, která signalizuje správné zarovnání magnetů vůči čidlu [32].

Snímač rychlosti a kadence implementuje ANT+ profil Bike Speed and Cadence (S&C) a ve spojení s ANT+ Android SDK lze po spárování s mobilní aplikací načítat tato data:

- spočítaná okamžitá rychlost včetně času
- spočítaná okamžitá kadence šlapání
- vypočítaná celková vzdálenost
- kumulativní otáčky
- kumulativní operační čas senzoru
- informace o výrobci a produktu(čísla hardware, software a modelu zařízení)

Obrázek 2.14: Snímač rychlosti a kadence Garmin GSC 10

2.5 Současná aplikace pro sběr dat z biosenzorů

V rámci této práce navazuji na aplikaci Elfyz Data Mobile Logger (viz. obrázky 2.15a a 2.15b), zkráceně Mobile Logger, která byla vyvinuta v rámci diplomové práce Mobilní aplikace pro monitoring pohybu a elektrofyziologických potenciálů [36]. Mobile Logger je aplikace pro mobilní telefony s operačním systémem Android.

Aplikace umožnuje načítání dat z externích senzorů. Podporuje senzory dostupné v neurolaboratoři KIV:

- ANT+ senzor pro měření srdečního tepu
- ANT+ osobní váha
- Bluetooth tlakoměr
- Bluetooth glukoměr

Pro komunikaci s ANT+ zařízeními využívá tato aplikace ANT+ Android SDK. S načtenými daty aplikace dále pracuje ve formě lokálního ukládání dat,

přehledu statistik v grafech a nahrávání dat na server portálu EEGBase. Data jsou na serveru synchronizována ve formátu `GenericParameter`, který je ale omezený pouze pro použití v rámci portálu EEGBase.

Výhody této aplikace spočívají v lokálním úložišti dat a v možnostech uploadu dat do databáze portálu EEGBase nezávisle na ukončení experimentu. Aplikace rovněž hlídá typ datového připojení, což už není, dle mého názoru, v době dostupnosti mobilního připojení problém pro odesílání většího množství dat.

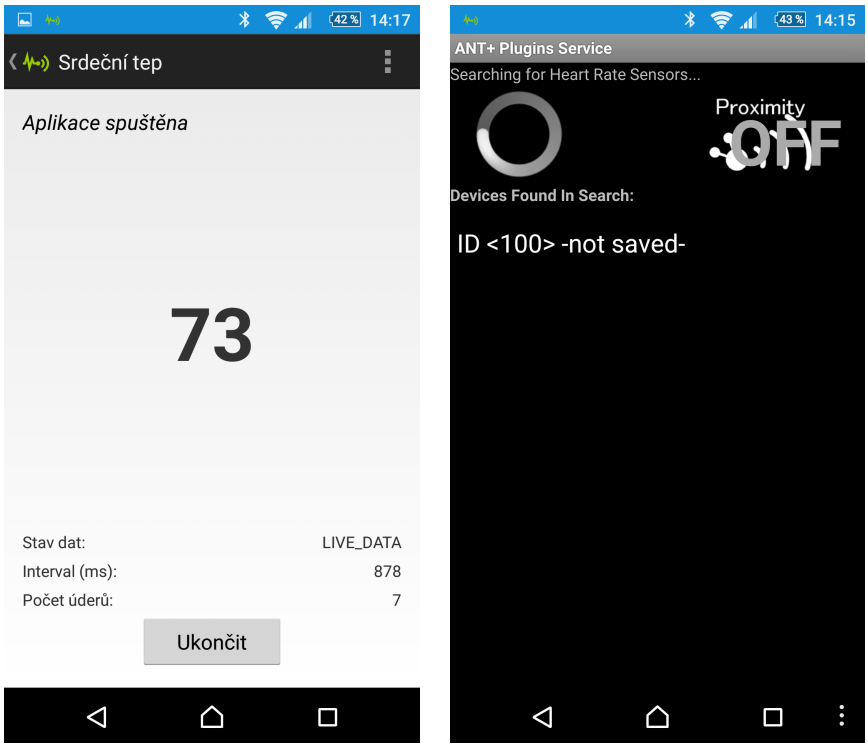
Nevýhod aplikace je také několik. Po nainstalování aplikace se menu telefonu objeví dvě stejné ikony (reprezentující Android aktivity) s různým názvem, neznalý uživatel může být kvůli tomu zmatený. Druhá nevýhoda také souvisí s uživatelskou přívětivostí. Uživatelské rozhraní aplikace je mírně spartánské a ne příliš intuitivní. Poslední nevýhodu shledávám v párování ANT+ senzorů. Pro vyhledávání a párování senzorů využívá Mobile Logger aktivitu externí aplikace ANT+ Plugins Service, do které se přepíná. Vyhledávání zařízení tedy neprobíhá přímo aplikaci Mobile Logger, což by se dalo zlepšit.

Původní myšlenka byla v rámci této diplomové práce upravit aplikaci Mobile Logger, přidat do ní další sadu senzorů a odstranit její nedostatky. Po analýze aplikace jsem se však rozhodl vytvořit aplikaci novou. Důvodem byl především příliš složitý refaktoring stávajícího kódu a nové možnosti pro tvorbu mobilních aplikací.

2.6 Portál EEGBase

Výzkumná skupina neuroinformatiky na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni vyvinula webový portál, který slouží pro uchováváni, správu, analýzu a sdílení elektrofyziologických dat. Účel tohoto portálu je poskytovat výzkumným skupinám či jiným subjektům platformu pro správu jejich dat.

Uživateli portálu EEGBase je po registraci a přihlášení umožněno vytvářet vlastní experimenty a náhravat k nim data. Ta mohou být binární (např. elektrické signály z elektrod) nebo ve formě metadat. Metadata obsahují např. informace, za jakých podmínek experiment probíhal. Mohou to být údaje o sledovaném subjektu, jeho věk, hmotnost, výška nebo pohlaví. Dal-



(a) Obrazovka Srdeční tep (b) Obrazovka párování senzoru

Obrázek 2.15: Aplikace Elfyz Data Mobile Logger

šími údaji jsou scénář měření nebo informace o hardware a software. Součástí metadat mohou být i doplňková data z dalších měření v rámci experimentu. Tato doplňková data bude poskytovat navrhovaná mobilní aplikace.

2.6.1 open metadata Markup Language

V rámci organizace International Neuroinformatics Coordinating Facility (INCF)[8], která sdružuje neuroinformatické skupiny po celém světě (včetně české), působí německý uzel G-Node [6]. Skupina G-Node vyvinula otevřený formát pro výměnu neuroinformatických metadat - open metadata Markup Language (odML) [35]. G-Node pro práci s odML poskytuje volně dostupná API pro programovací jazyky Python nebo Java. Portál EEGBase s využitím Java API umožňuje ukládání metadat ve formátu odML, proto jej bude implementovat i navrhovaná mobilní aplikace. Bude tak zaručena interoperabilita s jinými systémy.

Formát odML je založený na principu páru klíč-hodnota (např. rychlost = 15m/s). Model odML definuje 4 základní entity - Základní sekci (RootSection), Sekci (Section), Vlastnost (Property) a Hodnotu (Value). Sekce a jejich vlastnosti jsou jádrem formátu odML. Sekce obsahuje vlastnosti, ale také další podsekce.

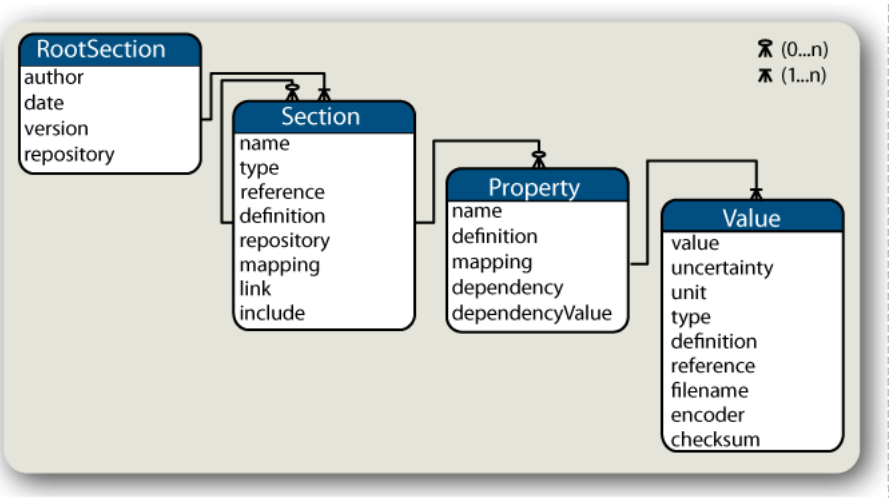
Tímto vzniká stromová struktura dat. Účelem tohoto modelu není definovat obsah, ale pouze strukturu, což zavádí určitou flexibilitu při definování vlastní struktury odML dat. Navrhovaná mobilní aplikace bude definovat několik struktur formátu odML pro sdílení dat z biosenzorů (více v kapitole 3). ER model na obrázku 2.16 ukazuje, jaké vazby musí být při implementaci odML dodrženy. Jsou zde také vidět základní parametry jednotlivých entit.

Nevýhodou formátu odML je velikost dat, která jsou v odML uchována. Pokud chceme dodržet čitelné pojmenování sekcí, vlastností a hodnot, tak narůstá objem těchto metadat a často převyšuje velikost klíčových informací. Část uživatelsky definované struktury pro uchování informací z měření tlaku ve formátu odML ukazuje výpis kódu 2.1.

```
{
  "odML": {
    "author": "mobio_app",
    "date": "1970-01-01T00:00:00.000+0000",
    "version": 1,
    "xmlns:gui": "http://www.g-node.org/guiml",
    "section": [
      {
        "name": "Blood Pressure Latest",
        "type": "bloodPressureLatest",
        "property": [
          {
            "name": "date",
            "value": {
              "type": "string",
              "content": ""
            }
          },
          {
            "name": "systolic",
            "value": {
              "type": "int",
              "content": 0
            }
          },
          {
```

```
27              "name": "diastolic",
28              "value": {
29                "type": "int",
30                "content": 0
31              }
32            }
33          ]
34        }
35      ]
36    }
37  }
```

Výpis kódu 2.1: Ukázka odML - aplikace ve formátu JSON



Obrázek 2.16: ER model formátu odML [28]

2.6.2 REST služby

REST (Representational State Transfer) je architektura rozhraní, navržená pro distribuované prostředí. REST je na rozdíl od známějších XML-RPC či SOAP, orientován datově, nikoli procedurálně. Všechny zdroje mají vlastní identifikátor URI a REST definuje čtyři základní metody pro přístup k nim (GET, POST, PUT, DELETE) [39]. Komunikace probíhá na základě protokolu HTTP a pro výměnu dat se nejčastěji používají formáty XML a JSON.

V portálu EEGBase jsou implementované REST služby pro práci s datovými soubory, uživateli, skupinami, scénáři, rezervacemi a experimenty.

REST služby jsou zabezpečené metodou Basic access authentication [48]. V každém požadavku musí být uvedena HTTP hlavička `Authorization: Basic jmeno:heslo`. Řetězec `jmeno:heslo` musí být navíc zakódován metodou Base64 kvůli znesnadnění čtení přihlašovacích informací člověkem. Base64 však nezávádí kryptografické zabezpečení jména a hesla.

Z již implementovaných REST služeb bude možné využít službu pro získání informací o uživateli a službu pro načtení dat o experimentech k danému uživateli. Naopak bude nutné přidat REST službu pro upload metadat ve formátu odML.

3 Aplikace MoBio

Tato kapitola se zaměřuje na vývoj mobilní aplikace pro sběr dat z biosenzorů. Jako název pro aplikaci jsem zvolil *MoBio*, což je spojení dvou stěžejních slov z názvu této práce - *mobilní* a *biosenzory*. Logo aplikace je na obrázku 3.1.

Při analýze platforem, které bude mobilní aplikace podporovat, bylo v úvaze několik variant. První varianta byla podporovat pouze Android a zároveň s tím rozšířit stávající mobilní aplikaci pro sběr dat ze senzorů Elfyz Data Mobile Logger [36]. Druhá varianta spočívala v tvorbě zcela nové čistě Android aplikace. Zvítězila ale možnost vyzkoušet vývoj hybridní mobilní aplikace, která bude schopná podporovat více platforem - kromě Android i iOS na mobilních telefonech iPhone nebo tabletech iPad.

Obrázek 3.1: Logo aplikace MoBio

3.1 Hybridní mobilní aplikace

Hybridní mobilní aplikace je fenomén posledních 2-3 let. Před touto dobou vývoj těchto aplikací také existoval, ale dostupné technologie neodpovídaly nárokum uživatelů, kteří byli zvyklí na klasické nativní aplikace.

Hybridní mobilní aplikace v kontextu této práce je aplikace, jejíž jádro je založené na technologiích JavaScript, HTML a CSS. Tato aplikace je vykreslena v komponentě WebView, jejíž primárním cílem je zobrazovat webový obsah v rámci nativní mobilní aplikace. Přes WebView aplikace komunikuje s operačním systémem mobilního telefonu, ale také s jednotlivými zásuvnými moduly (pluginy), které obstarávají komunikaci s hardwarem telefonu přes API programovacího jazyka konkrétní platformy.

Jako konkrétní řešení pro obstarání komunikace s funkcemi mobilního zařízení jsem vybral framework Apache Cordova (zkráceně Cordova) [20] a jeho zásuvné moduly. Za pomocí frameworku Cordova lze aplikaci postavit na jednom programovém kódu a podle potřeby sestavit aplikaci pro Android, iOS, Windows Phone a další. Mobilní aplikace MoBio bude plně podporovat operační systém Android, omezeně potom iOS.

Aby hybridní aplikace MoBio splňovala všechny nároky nativní aplikace, integroval jsem do ní následující pluginy [22]:

- **Cordova Device Plugin** - zajišťuje dostupnost informace o hardware a software telefonu [5]. Pomocí tohoto pluginu lze např. větvit zdrojový kód podle typu zařízení a operčního systému, na kterých je mobilní aplikace nasazená.
- **Cordova Whitelist Plugin** - implementuje pravidla pro povolování navigace pouze v rámci daných URL. Plugin je důležitý pro bezpečnost aplikace. [14]
- **Ionic Keyboard Plugin** - pomáhá při interakci s virtuální klávesnicí mobilního zařízení [11]
- **CrossWalk WebView** - různé verze operačního systému Android obsahují také různé verze komponenty WebView, které jinak vykreslují jednotlivé prvky HTML dokumentu. Plugin CrossWalk tyto nedostatky odstraňuje tím, že do aplikace integruje jednotné jádro webového prohlížeče založené na projektu Chromium [3]. Výhoda je v tom, že aplikace vypadá stejně ve starších i nových verzích Android. [4]
- **Cordova Splashscreen** - přidává a ovládá obrazovku při spuštění aplikace. [13]
- **Cordova InAppBrowser** - umožňuje v rámci mobilní aplikace spouštět odkazy na externí zdroje a v případě webových stránek je i zobrazovat. [7]
- **Bluetooth Serial Plugin for PhoneGap** - zajišťuje sériovou komunikace přes Bluetooth. [24]

Všechny pluginy jsou volně šiřitelné pod Apache Licencí [16].

3.2 Pluginy pro komunikaci s ANT+ sensory v prostředí Apache Cordova

Jeden záusvný modul ve výše uvedeném seznamu chybí. Jedná se o plugin pro komunikaci s ANT+ senzory (dále jen *plugin*). Nenalezl jsem žádnou hotovou a plně funkční implementaci takového pluginu - i přes to, že je komunita vývojářů pluginů pro Apache Cordova celkem rozsáhlá.

Chybějící plugin pro komunikaci s ANT+ zařízeními se stal příležitostí takový vytvořit. V plánu je hotový a otestovaný plugin publikovat v rámci oficiálního seznamu pluginů Apache Cordova a být tím prospěšný komunitě vývojářů této platformy.

Prerekvizitou pro vytvoření pluginu bylo nastudování a použití ANT+ SDK. To je prozatím dostupné pouze pro systém Android. Implementace pluginu je proto také omezená jen na Android. Druhou podmínkou bylo dodržet postup pro tvorbu Cordova pluginu podle dokumentace. Nejedná se tedy pouze o překopírování vzorové aplikace z balíku ANT+ Android SDK.

Plugin vytváří komunikační most mezi třídami a funkcemi nativního programovacího jazyka (v tomto případě Java) a programovacím jazykem hybridní aplikace (JavaScript).

```
1  module.exports = {
2      searchDevices: function (deviceType, successCallback,
           errorCallback) {
3          cordova.exec(successCallback, errorCallback, "Antplus",
               "searchDevices", [deviceType]);
4      },
5      stopSearchDevices: function (successCallback,
           errorCallback) {
6          cordova.exec(successCallback, errorCallback, "Antplus",
               "stopSearchDevices", []);
7      },
8      subscribeHR: function (antDeviceNumber, successCallback,
           errorCallback) {
9          cordova.exec(successCallback, errorCallback, "Antplus",
               "subscribeHR", [antDeviceNumber]);
10     }
```

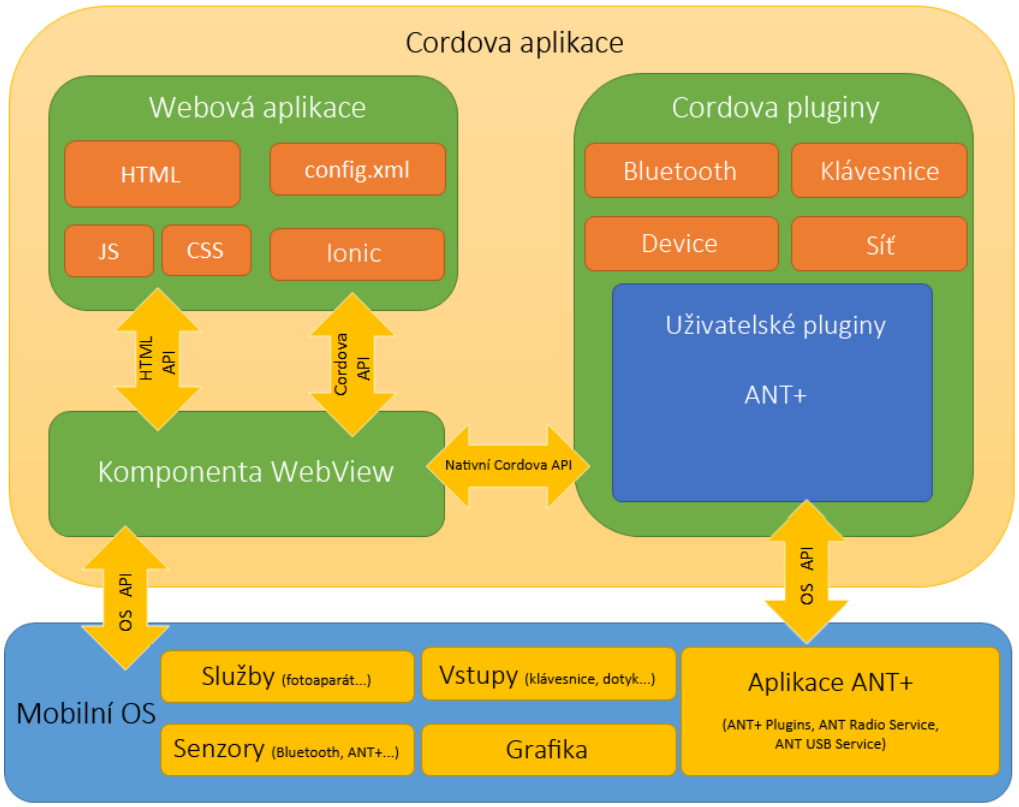
Výpis kódu 3.1: Ukázka části implementace JavaScriptové části ANT+ pluginu

Struktura implementovaného pluginu je následující:

- *plugin.xml* - obsahuje informace o názvu pluginu, zdrojových souborech a verzích a platformách, se kterými je plugin kompatibilní. Je zde také úveden název balíku (`cz.zcu.kiv.neuroinformatics.antplus`)
- *src/android/libs* - adresář s knihovnami. Obsahuje knihovny ANT+ Android SDK.
- *www/antplus.js* - přeneseně řečeno se jedná o první stranu komunikačního mostu. Zde jsou definované JavaScriptové funkce pro vyhledání ANT+ zařízení a aktivování a deaktivování komunikace. Parametry těchto funkcí jsou callbacky pro úspěšné a chybové volání, pak název Java třídy pro propojení. Dalším parametrem je název akce, která bude zavoláním funcke vyvolána na straně nativního kódu. Poslední parametr je libovolný a obsahuje pole s doplňkovými informacemi (např. data uživatelského profilu). Viz výpis kódu 3.1.
- *src/android* - adresář s třídami, ve kterých je implementována interakce s ANT+ Android SDK. Jedná se tedy o druhou stranu komunikačního mostu.
  - třída *Antplus* tvoří základ Cordova pluginu na straně nativního kódu. Inicializují se zde služby pro volání ANT+ senzorů a zároveň je v této třídě definováno propojení s JavaScriptovou částí kódu.
  - třída *AntplusMultiDeviceSearch* zajišťuje vyhledávání senzorů. Poskytuje funkce pro spuštění a ukončení vyhledávání. Poskytuje vyhledávání na základě typu senzoru, což umožňuje filtrovat nechtěná ANT+ zařízení.
  - třídy *AntplusBikeSpeedDistanceService, AntplusHeartRateService, AntplusStrideSDMService, AntplusWeightScaleService* poskytují funkcionalitu pro komunikaci s konkrétním typem ANT+ senzoru. Třídy pracují na základě notifikací (podobně jako u Bluetooth Smart). Pokaždé, když senzor vyvolá událost, je tato zpracována pomocí ANT+ Android SDK a propagována do tzv. receiveru (přijímače). Přijímač událost transformuje do JSON objektu a odešle buď data ze senzoru (úspěšné volání události) nebo data s chybou (neúspešné volání události) do JavaScriptové části pluginu.
  - plugin obsahuje i třídu *AntplusBloodPressureService*, která implementuje načítání dat z ANT+ zařízení pro měření krevního tlaku.

Bohužel nebylo možné nijak otestovat funkčnost implementace. Tonometr nebyl dostupný ani v laboratoři bioinformatiky na KIV, ani v rámci testovacího software. Proto jsem se rozhodl tento typ senzoru nezahrnout do výsledné aplikace.

Schéma architektury aplikace je znázorněno na obrázku 3.2.



Obrázek 3.2: Architektura aplikace MoBio. Původní obrázek architektury bez ANT+ je dostupný na webu projektu Cordova [21].

3.3 Uživatelské rozhraní

Několik prototypů obrazovek aplikace navrhované v této práci bylo vytvořeno v online nástroji NinjaMock [12]. V tomto software lze zvolit přímo druh prototypu mobilní aplikace. NinjaMock poskytuje skicy většiny ovládacích

prvků, které lze přesouváním vkládat na jednotlivé obrazovky. Výstup (nebo jinými slovy mockupy) jsem použil při navhrování celkového vzhledu aplikace. Výhodou je vygenerování prototypu jako interaktivní webové aplikace, která umožňuje přecházení mezi obrazovkami klikáním na ovládací prvky ve formě odkazů. Nutno podotknout, že některé výsledné obrazovky se liší od prototypu. Ale to je dáno postupným vývojem a rozhodně neznamená chybu nebo nedodržení postupu.

Obrázek 3.3 znázorňuje výstup programu NinjaMock. První obrazovka ukazuje rozložení prvků pro přepínání tabů, graf a tabulku. Na druhé obrazovce jsou znázorněné prvky pro editaci uživatelského profilu.

Obrázek 3.3: Prototypy aplikace

Prvním stupněm návrhu uživatelského rozhraní byly prototypy. Druhým stupněm byla volba frameworku pro tvorbu front-endu hybridní mobilní aplikace. Výběr technologií frameworku je daný z principu Apache Cordova - HTML, CSS a JavaScript. Při rozhodování mi pomohl článek The Top 7 Hybrid Mobile App Frameworks [37]. Framework by měl poskytovat rychlou odezvu prvků uživatelského rozhraní, protože vykreslování složitých HTML struktur v mobilní aplikaci stále nedosahuje takového výkonu jako na osobních počítačích. Hned zpočátku jsem tedy zavrhl jQuery a jeho deriváty jako

například jQuery Mobile [10] a na nich založené frameworky. Za úvahu stál framework Sencha Touch [44], u něj si mi ale nelíbilo zpracování základních prvků grafického rozhraní a také propojení s knihovnou Ext JS, ve které se příliš neorientuji.

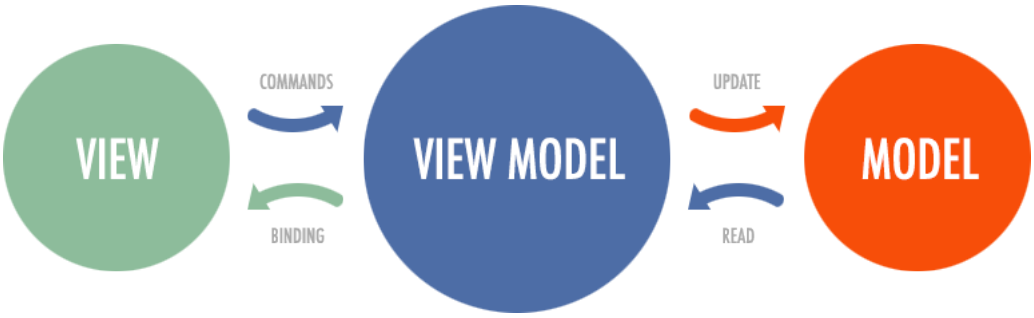
Všechny nedostatky předchozích zmíněných frameworků a knihoven odstraňuje Ionic [9]. Je to framework pro volné použití. Nabízí HTML5 prvky optimalizované pro mobilní zařízení. Optimalizuje chování gest v rámci webové-mobilní aplikace a nabízí vzhled uživatelského rozhraní jako v nativní aplikaci. Důležitým aspektem tohoto frameworku je také orientace na výkon aplikace a minimální manipulaci s objektovým modelem dokumentu (DOM). Ionic není pouze framework pro tvorbu uživatelského rozhraní. Jeho součástí je i podpora kompletního vývojového procesu hybridní aplikace a velmi nápomocné rozhraní přikazové řádky (CLI), které poskytuje příkazy pro vytvoření kostry projektu, ladění aplikace, správu platforem nebo sestavení aplikace. Odstraňuje tím nutnost používat další nástroje.

3.4 MVVM architektura a struktura aplikace

Ionic je postaven na úzké spolupráci s frameworkem AngularJS [33], který tvoří jádro Ionicu. Tedy to, jakým způsobem jsou zpracovány uživatelské vstupy, výměna dat v rámci aplikace nebo sdílení dat z externích zdrojů vychází právě AngularJS.

3.4.1 Model–view–viewmodel

AngularJS je založený na architektuře Model–view–viewmodel a na vkládání závislostí (DI - dependency injection). Do kontroleru (součást ViewModel), který řídí prezentační logiku, se vkládá objekt `$scope` (spojení mezi View a ViewModel). Parametry tohoto objektu jsou dostupné v šabloně obrazovky (View). View se aktualizuje automaticky ve chvíli, kdy se změní daný parametr objektu `$scope`. Vlastnost dvoucestného provázání dat (2-way data binding) umožňuje automatickou aktualizaci dat i opačným směrem - z view do kontroleru. Tato situace nastává například když uživatel aktualizuje textové pole. Jak funguje MVVM ilistruje obrázek 3.4. Data jsou sdílena mezi kontrolery pomocí tzv. služeb (service), což je implementace Modelu v prostředí AngularJS. Služby implementují návrhový vzor singleton, jejich lokální



Obrázek 3.4: Model–view–viewmodel (převzato z DevExtreme [34])

proměnné jsou zachovávány v průběhu navigace mezi obrazovkami. Stejně jako `$scope` se i služby injektují do kontrolerů. Služby lze vkládat ale i do jiných služeb.

Poslední důležitou částí jsou direktivy. Direktiva je způsob, jakým lze rozšířit funkcionalitu HTML. Pomocí direktiv se vytváří nové HTML elementy s vlastní logikou a vzhledem. V aplikaci MoBio je imlpementována direktiva, která zobrazuje animovaný koláčový graf.

Aplikace je členěna do adresářů (balíků/modulů), které sdružují zdrojové soubory podle toho, jakou funkcionalitu vykonávají (viz obrázek 3.5).

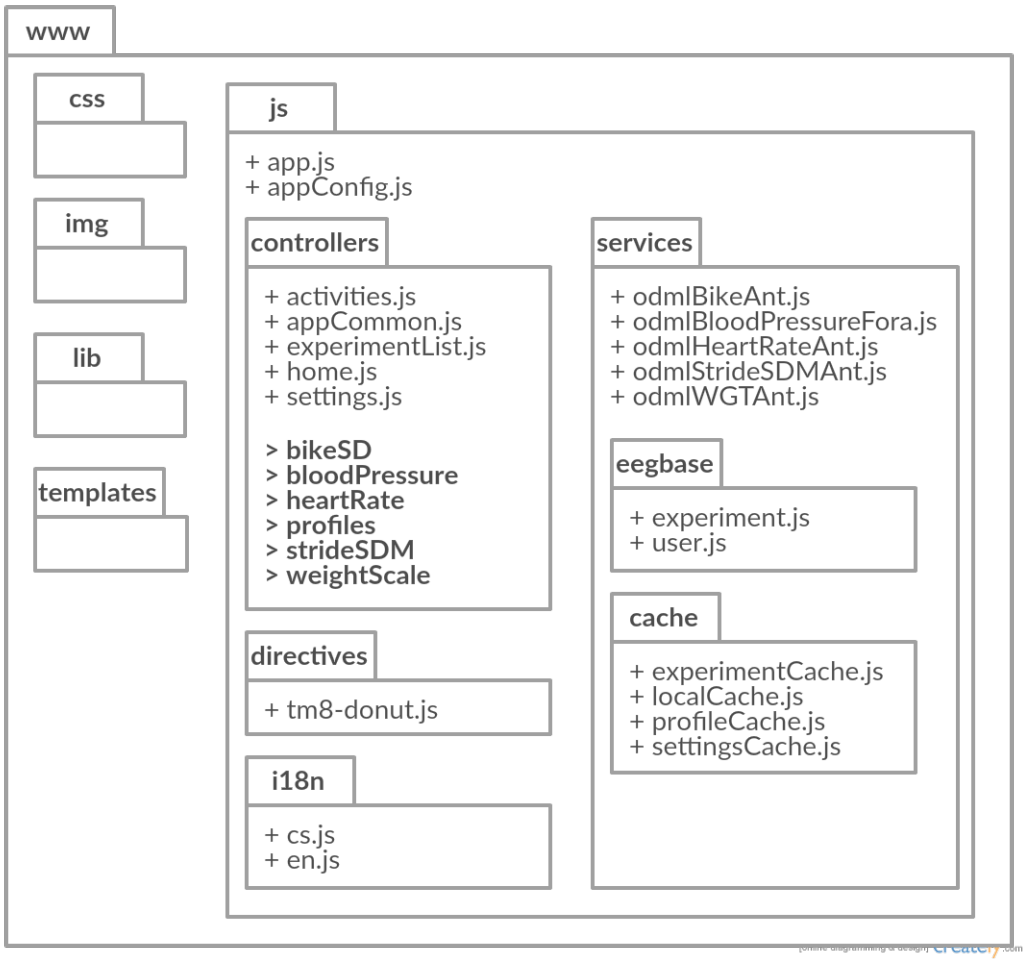
Hlavní kontrolery a služby jsou následující:

3.4.2 Vstupní bod aplikace - modul `mobio`

Modul `mobio` je definován v souboru `app.js`. Konfiguruje strukturu aplikace nastavením stavů - obrazovek, mezi kterými lze přecházet pomocí odkazů (jinými slovy definuje routy). Jsou zde zaregistrovány i ostatní moduly (`controllers, odML, cache, eegbase`).

3.4.3 Kontroler `AppCtrl` (appCommon.js)

Definuje objekty, funkce a události společné pro celou aplikaci.



Obrázek 3.5: Struktura aplikace

3.4.4 Kontroler `HomeCtrl` (home.js)

Obsahuje implementaci přihlašovacího dialogu a reakce na události přihlašovacího procesu. V případě úspešné odpovědi serveru, nastaví uživatele a v případě prvního přihlášení také přidá uživatelský profil do cache. Dále zpřístupní obrazovku pro vyběr profilu. Pokud přilhášení nebylo úspěšné, zobrazí chybovou hlášku a zakáže vstup do sekce profilů.

3.4.5 Kontroler `ExperimentListCtrl` (experimentList.js)

Ovládá stahování experimentů z portálu EEGBase.
Definuje funkci `$scope.syncExperiments()`, která je volána z view seznamu experimentů.

3.4.6 Kontroler `ActivitiesCtrl` (activities.js)

Nastavuje data pro view seznamu aktivit. Jsou zde uvedené popisy a názvy ikon senzorů.

3.4.7 Kontrolery pro práci s profily - v adresáři `profiles`

Tyto kontrolery definují funkce pro zvolení a nastavení aktivního profilu (`ProfilesListCtrl`), vytvoření nového profilu (`ProfilesNewCtrl`) a editaci zvoleného profilu (`ProfilesEditCtrl`). Funkce jsou volány ze souvisejících view.

3.4.8 Kontroler `BloodPressureCtrl` (bloodPressure.js)

První z kontrolerů pro práci se senzory. Tento řídí komunikaci s tlakoměrem FORA P30 Plus. Komunikace probíhá přes Bluetooth pomocí pluginu *Bluetooth Serial Plugin for PhoneGap* [24].

Po přechodu na obrazovku tlakoměru kontroler zjistí, zda je zapnuté Bluetooth a případně ho automaticky spustí. Následně zobrazí seznam spárovaných zařízení (tlakoměr už musí být předem spárovaný s mobilním telefonem, viz kapitola 6 Testování). Ke spárovaným zařízením zná aplikace jejich ID (MAC adresu). To je po vybrání zařízení ze seznamu nastavena do proměnné `$scope.data.selectedDevice.id` a dále použita pro připojení. Komunikace s tlakoměrem funguje na základě vyměňování osmi-bytových zpráv. Byty mají následující význam (citováno z [36]):

1. start byte
2. typ zprávy
3. data 1
4. data 2
5. data 3
6. data 4
7. end byte
8. checksum

Start a end byte jsou neměnné a mají hodnotu 81, resp. -93. Datové byty nesou jednotlivé významové bity, ze kterých se extrahují hodnoty měření. Checksum slouží pro kontrolu integrity. Jeho hodnotou je jednoduše součet všech předchozích bytů, přičemž se pro výpočet používá jeden byte a dochází k přetečení (resp. lze počítat s integery a použít LSB). Komunikaci začíná klient odesláním zprávy. Tlakoměr pak vrátí odpověď se stejným typem zprávy. Konec citace [36].

Tlakoměr používá následující identifikátory pro typ zprávy:

- MSG_START (43) pro navázázání komunikace
- MSG_A (37) pro zprávu obsahující čas měření tlaku
- MSG_B (38) pro zrpávu obsahující hodnoty tlaku
- MSG_END (80) pro ukončení komunikace

Iniciátor komunikace je mobilní aplikace. Po připojení ke spárovanému tlakoměru pomocí jeho ID pošle aplikace zprávu obsahující hodnotu MSG_START zapsanou v bytu číslo 1. Pokud tlakoměr odpoví kladně, zašle v bytu číslo dvě počet uložených měření. V tomto momentě se komunikace rozdělí do dvou větví.

```
var MSG_A = 37;
var startMsg = [81, MSG_START, 0, 0, 0, 0, -93, 0];
startMsg[7] = checkSum(startMsg);

bluetoothSerial.write(startMsg,
    function(success) {
        bluetoothSerial.subscribeRawData(
            function(result) {
                bluetoothSerial.unsubscribeRawData();
                var data = new Uint8Array(result);
                askForDateTime(0, data[2], onlyLatest);
            },
            function(failure) {
                setSubscribedFlag(false);
            }
        );
    },
    function(failure) {
        setSubscribedFlag(false);
    }
);
```

Výpis kódu 3.2: Ukázka započetí komunikace s tlakoměrem

První větev je zodpovědná za načítání časů měření tím, že posílá a přijímá zprávu s hodnotou MSG_A v bytu číslo 1. Aby aplikace dokázala synchronizovat čas měření s hodnotami tlaku, tak se vzápětí spustí druhá větev programu, které pracuje se zprávou MSG_B v bytu číslo 1. Obě větve se následně střídají podle toho, jak se inkrementuje čítač načtených měření.
Po přečtení posledního měření se komunikace ukončuje. V případě načítání pouze posledního měření se komunikace ukončí po přijetí první zprávy MSG_B.

Další funkcí kontroleru `BloodPressureCtrl` je transformování dat z osmibytových zpráv do formátu odML, to se děje vždy po načtení zprávy MSG_B, tedy v momentě kdy je známý pár čas a data měření. Data se transformují pomocí služby `odmlBloodPressureFora`.

```
bluetoothSerial.subscribeRawData(
    function(result) {
        bluetoothSerial.unsubscribeRawData();
        $scope.$apply(
            function() {
                var rawData = new Uint8Array(result);
                var dataToSet = {
```

```
8                       systolic: rawData[2],
9                       diastolic: rawData[4],
10                      mean: rawData[3],
11                      heartRate: rawData[5]
12                  };
13                  $scope.odMLData = odmlBloodPressureFora.
14                  addBloodPressureMeasurement(
15                      $scope.odMLData, lastDateIndex, dataToSet);
16              }
17          );
18          if (record < count - 1) {
19              record++;
20              askForDateTime(record, count, onlyLatest);
21          } else {
22              closeConnection();
23          }
24      },
25      function(failure) {
26          setSubscribedFlag(false);
27
28      }
29  );
```

Výpis kódu 3.3: Ukázka načtení hodnot měření tlaku a volání služby pro transformaci do odML (zdrojový kód je zkrácený pro ilustraci hlavní funkcionality)

Důležitou součástí kontroleru je funkce `$scope.uploadMeasurement()`, která spouští odeslání odML dat na server a v reakci na odpěď REST služby zobrazuje okno s informací o stavu odeslání dat.

3.4.9 Kontrolery `BikeSDCtrl` a `StrideSDMCtrl`

Svou implamentaci mají v souborech *bikeSD.js* a *strideSDM.js*. Funcke kontrolerů je v principech téměř stejná, liší se pouze typem ovládaného ANT+ senzoru. Z tohoto důvodu bude jejich funkcionalita popsána společně.

Stejně jako předchozí kontroler i tyto dva implementují funkci, která zajišťuje spuštění služby pro odeslání dat na server.
Zbylá funkcionalita se zaměřuje na ovládání komunikace se senzorem a čtení poskytovaných dat. Kontrolery používají funkce Cordova pluginu `cz.zcu.kiv.neuroinformatics.antplus`, implementovaného v rámci této práce.

`$scope.buttonStartSearchClick()` a `$scope.buttonStopSearchClick()` jsou funkce pro ovládání vyhledávání ANT+ zařízení na základě definovaného typu. Znamená to, že část aplikace pro práci otáčkoměrem není schopna vyhledat a přijímat data například ze senzoru pro měření srdečního tepu.

Funkce `$scope.buttonListenClick()` kontroluje, zda aplikace již se senzorem komunikuje. Pokud ano, tak komunikaci ukončí, jinak resetuje data kontroleru a zahájí komunikaci.

Funkce `$scope.subscribeSDM()` a `$scope.subscribeBike()` řeší přihlášení se aplikace k odběru dat událostmi antplus pluginu. Jakmile plugin vygeneruje událost, je propagována až do callbacku funkce `subscribe()`, kde jsou zpracována přijatá data do formátu odML pomocí předem definovaných služeb.

```
1  $scope.subscribeSDM = function(antDeviceNumber) {
2    try {
3      antplus.subscribeSDM(antDeviceNumber,
4        function(readResult) {
5          if (readResult.event == "instantaneousSpeedData") {
6            $scope.odMLData = odmlStrideSDMAnt.
7            addInstantaneousSpeedMeasurement($scope.odMLData,
                 readResult);
8            $scope.data.instantaneousSpeedData = readResult;
9          } else if (readResult.event ==
               "instantaneousCadenceData") {
10           $scope.odMLData = odmlStrideSDMAnt.
11           addInstantaneousCadenceMeasurement($scope.odMLData,
                 readResult);
12           $scope.data.instantaneousCadenceData = readResult;
13         } else if (readResult.event ==
               "cumulativeDistanceData") {
14           $scope.odMLData = odmlStrideSDMAnt.
15           setCumulativeDistance($scope.odMLData, readResult);
16           $scope.data.cumulativeDistanceData = readResult;
17         }
18       },
19       function(error) {
20         console.log(JSON.stringify(error));
21         $scope.$apply(
22           function() {
23             $scope.data.error = JSON.stringify(error);
24           });
25       });
26   } catch (e) {
27     console.log("antplus is not defined");
```

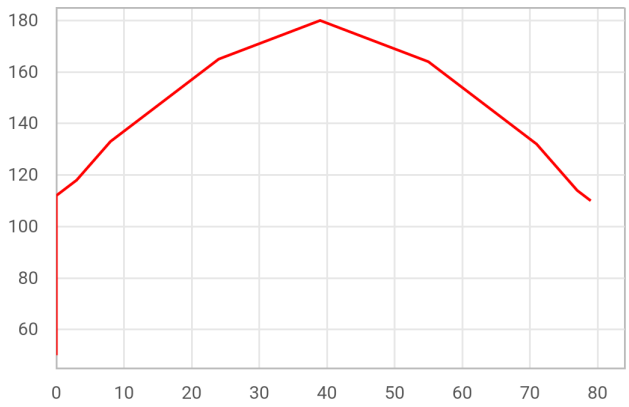
```
28     }
29 };
```

Výpis kódu 3.4: Ukázka načtení hodnot krokoměru a volání služby pro transformaci do odML (zdrojový kód je zkrácený pro ilustraci hlavní funkcionality)

3.4.10 Kontroler `HeartRateCtrl` (heartRate.js)

Tento kontroler řeší ovládání vyhledávání ANT+ senzoru pro měření srdečního tepu, připojení k tomuto senzoru a přihlášení se k odběru dat. Kontroler implementuje všechny výše popsané funkce.

Navíc kontroler obsahuje funkci `$scope.drawLineChart()`, která vytváří interaktivní čárový graf (obrázek 3.6). Ten v reálném čase zobrazuje naměřené hodnoty srdečního tepu. Implementace grafu je založena na knihovně D3.js (Data-Driven Documents) [19]. Graf umožňuje uživateli přibližování a posouvání pomocí gest. Protože se graf neustále aktualizuje, je doporučeno zastavit měření a až potom s grafem posouvat.



Obrázek 3.6: Čárový graf - osa X představuje čas v sekundách, osa Y zobrazuje hodnotu srdečního tepu v jednotce BPM

3.4.11 Kontroler `WeightScaleCtrl` (weightScale.js)

Kontroler pro obsluhu ANT+ váhy. Stejně jako již uvedené kontrolery implementuje funkce pro vyhledání vysílajícího senznoru a přihlášení k odběru kontinuálních dat, která senzor vysílá. Výjimkou tohoto kontroleru je implementace funkcí pro načítání měření na vyžádání (podobně jako u tlakoměru). Jde o funkci `$scope.requestBasicWGT()` pro žádost o základní data (čas a hmotnost) a o funkcii `$scope.requestAdvancedWGT()`, která žádá senzor o rozšířená data měření (čas, hmotnost či procento hydratece, tělesného tuku, svalové kostní hmoty). Pro vrácení rozšířených dat je potřeba senzoru předat data uživatelského profilu.

3.4.12 Služby odML (modul `mobio.odML`)

Služby pro práci s odML slouží pro uchování dat v rámci jednoho měření. Jejich úkolem je nastavit výchozí formát pro data každého senzoru. Dále obsahují funkce pro vkládání průběžných dat a aktualizace statických dat. Služby obsahují také gettery a settery pro ulehčení manipulace s daty.

3.4.13 Služby cache (modul `mobio.cache`)

Aplikace využívá cache kvůli uchovávání potřebných dat a udržení stavů i po vypnutí aplikace. Cache je implementována nad lokálním úložštěm HTML5 (HTML5 Local Storage). Dříve se pro tyto účely využívaly tzv. cookies. Local Storage je však bezpečnější a účinější pro větší objemy uložených dat. Nejdříve je nutné inicializovat prvek úložiště pod určitým jménem viz ukázka kódu 3.5.

```
1 window.localStorage.setItem('experimentCache',
    JSON.stringify({}));
```

Výpis kódu 3.5: Ukázka inicializace prvku lokálního úložiště. První parametr funkce setItem je jméno prvku. Druhý parametr je textový - v tomto případě JSON objekt převedený do stringu.

Poté lze z cache číst nebo do ní zapisovat. Pokud se do cache uloží JSON jako textový řetězec, lze data strukturovat a po načtení s nimi pracovat jako

s běžnými objekty (ukázka kódu 3.6).

```
.factory('experimentCache', function() {
  return {
    cacheAvailable: function() {
      return (typeof(Storage) === 'undefined' ? false : true);
    },
    getExperiments: function() {
      if (!this.cacheAvailable()) {
        return false;
      }
      return
          JSON.parse(window.localStorage.getItem("experimentCache"));
    },
    getExperimentById: function(id) {
      var experiments = this.getExperiments().experiments;
      for (var i = 0; i < experiments.length; i++) {
        if (experiments[i].experimentId == id) {
          return experiments[i];
        }
      }
      return null;
    }
  }
});
```

Výpis kódu 3.6: Ukázka použití lokálního úložiště

3.4.14 Služby EEGBase portálu (modul `mobio.eegbase`)

Zde je implementace HTTP metod GET a POST pro spojení se službami portálu EEGBase. Aplikace je využívá pro zjištění údajů o uživateli, ke stažení experimentů a kvůli nahrání odML metadat na server. Veškerá komunikae vyžaduje odeslání uživatelského jména a hesla zakódované pomocí Base64 v hlavičce `Authorization` (metoda jednoduchého ověření přístupu - Basic access authentication). Angular nabízí implementaci HTTP požadavků, které lze vykonávat asynchronně (AJAX). Po spuštění požadavku není aplikace blokována čekáním na odpověď serveru. Pouze se nastaví callbacky pro úspěch a chybu, ve kterých se vyřeší odpovědi serveru. Funkce, která spouští HTTP požadavek pouze nastaví odloženou odpověď a „slíbí“, že něco vráti (`return deferred.promise`)

```
uploadOdml: function(odML, experimentId) {
  var settings = settingsCache.getSettings();
  var deferred = $q.defer();
  $http.post(settings.restUrl +
      "/rest/experiments/addOdmlMobio/" + experimentId, odML, {
    headers: {
      'Authorization': "Basic " + btoa(settings.username + ":"
          + settings.password),
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    }
  }).then(function(response) {
    deferred.resolve(response);
  }, function(err) {
    deferred.reject(err);
  });

  return deferred.promise;
}
```

Výpis kódu 3.7: Ukázka použití lokálního úložiště

4 Úpravy Portálu EEGBase

Mobilní aplikace poskytuje funkce pro odeslání naměřených dat ve formátu odML na server. Taková možnost nebyla v portálu implementována a bylo nutné ji přidat.

Naměřená data patří vždy ke konkrétnímu experimentu. Experimenty umožňují definování metadat. Zpravidla se jedná o doplňkové údaje k experimentu.

Metadata se ukládají do NoSQL databáze, což platí i pro data naměřená mobilní aplikací. Experimenty mají vlastní POJO třídu `Experiment`. Parametr `ExperimentElastic` této třídy obsahuje údaje metadat ve formě sekcí a vlastností (`Section, Property`), tedy odML.

4.1 Třída `MobioMetadata`

Třída byla přidána do balíku `cz.zcu.kiv.eegdatabase.data.nosql`.

Veřejná metoda `createMetaData(JSONObject data)` má na vstupu JSON objekt ve formátu odML odeslaný aplikací MoBio. Z JSON tato metoda naprasuje sekce, podsekce a případně vlastnosti, ze kterých vytvoří pomocí odML API nový objekt Section. Tento objekt nakonec vrátí.

4.2 Metoda `addMobioMetadata` rozhraní `ExperimentService`

Metoda `addMobioMetadata` má za úkol danému experimentu (podle ID) buď přidat metadata, nebo vytvořit nová. K tomuto účelu využívá výše uvedenou funkci `createMetaData` třídy `MobioMetadata` (ukázka na přiloženém CD: `X:\portal\addMobioMetadata.txt`).

4.3 REST služba `addOdmlMobio`

Nový přístupový bod typu POST pro upload dat z mobilní aplikace. Metoda přijímá dva parametry. První je ID experimentu obsažené v URL. Druhý parametr je textového charakteru typu JSON. Metoda je namíru připravená pro přijímání dat z aplikace MoBio. Avšak při dodržení struktury JSON vstupu, není problém, aby ji využívaly i jiné aplikace.
V těle metody (ukázka na přiloženém CD: `X:\portal\addOdmlMobio.txt`) se textový JSON převede na objekt `JSONObject`, který je následně parametrem metody `addMobioMetadata` popsané výše.

4.4 CORS a úprava HTTP hlaviček

CORS (Cross-origin resource sharing, ve volném překladu sdílené zdroje odjinud) je mechanismus umožňující sdílení zdrojů (např. JavaScriptového kódu) webové stránky pro aplikaci na jiné doméně.[42]
REST API portálu EEGBase nebylo přizpůsobeno pro používání JavaScriptovou aplikací, která je spouštěná z jiné domény než API. Prohlížeč blokoval všechny dotazy na API z důvodu chybějících HTTP hlaviček v odpovědi serveru. Hybridní aplikace navíc neodesílá žádnou hlavičku `Origin`. Proto je jedinné akceptovatelné nastavení hlaviček následující:

- `Access-Control-Allow-Origin: *`
- `Access-Control-Allow-Methods: OPTIONS,GET,POST,PUT,DELETE`

Protože REST API podporuje Basic access authentication, byly do odpovědi serveru doplněny i hlavičky:

- `Access-Control-Allow-Credentials: true`
- `Access-Control-Allow-Headers: Authorization`

O doplnění hlaviček se postaral filtr `org.eclipse.jetty.servlets.CrossOriginFilter` nakonfigurovaný v souboru `web.xml`.

Další problém způsobovala hlavička `WWW-Authenticate` s hodnotou `Basic realm="Secure Area"` nastavovaná v HTTP odopvědi serveru po odeslání chybného uživatelského jména nebo hesla. Důsledkem bylo vyvolání okna operačního systému telefonu pro zadání přihlašovacích údajů. Je to standardní chování, ale v hybridní mobilní aplikaci nežádoucí.

Hlavičku nešlo upravit pomocí CORS filtru. Její nastavování totiž řídí Spring Security pomocí třídy `BasicAuthenticationEntryPoint` a její metody `commence`. Toto chování jsem musel upravit vlastním vstupním bodem pro Basic Authentication. Kód upravené metody commece je v ukázce 4.1. Jejím úkolem je změnit hodnotu hlavičky`WWW-Authenticate` na hodnotu, která nezpůsobí vyvolání nativního dialogu pro přihlášení na strané klienta.

```
1 public class RestAuthenticationEntryPoint extends
      BasicAuthenticationEntryPoint {
2     @Override
3     public void commence(HttpServletRequest request,
          HttpServletResponse response, AuthenticationException
          authException)
4             throws IOException, ServletException {
5         HttpServletResponse httpResponse =
              (HttpServletResponse) response;
6         httpResponse.setHeader("WWW-Authenticate", "FormBased");
7         httpResponse.sendError(HttpServletResponse.SC_UNAUTHORIZED,
              authException.getMessage());
8     }
9 }
```

Výpis kódu 4.1: Ukázka třídy RestAuthenticationEntryPoint

Konfigurace vstupního bodu autentizace spočívá v definici beanu a nastavení `basic-entry` v sekci `security` v souboru *applicationContext.xml*, viz. ukázka 4.2.

```
1 <security:http-basic
      entry-point-ref="restAuthenticationEntryPoint"/>
2
3 <bean id="restAuthenticationEntryPoint"
4 class="cz.zcu.kiv.eegdatabase.webservices.
5 rest.common.utils.RestAuthenticationEntryPoint" />
```

Výpis kódu 4.2: Ukázka konfigurace vstupního bodu autentizace

4.5 Zobrazení naměřených dat

O prezentační část portálu se stará framework Wicket [15]. Zdrojové soubory webové částí stránky, která zobrazuje metadata experimentu jsou součástí balíku `cz.zcu.kiv.eegdatabase.wui.ui.experiments.metadata`:

- Třída `ViewMetadataSectionPanel`
- Třída `ViewMetadataPropertyPanel`

Tyto třídy představují model části webové stránky a je zde implementována logika práce s prezentovanými daty. Ke každé třídě patří i stejně pojmenovaná HTML šablona, která určuje jak se budou data zobrazovat.
Analýzou těchto souborů bylo zjištěno, že jejich implementace je dostačující pro zobrazení dat z biosenzorů (viz. obrázek 4.1). Byly tedy ponechány beze změn.

Metadata

Blood Pressure Latest

Property	Value
date	2015-09-22T22:55:00+02:00
systolic	115
diastolic	84
mean	94
heartRate	80

Obrázek 4.1: Ukázka zobazení dat z mobilní aplikace v portálu EEGBase

5 Funkce pro analýzu dat

Mobilní aplikace umožňuje základní analýzu dat získavaných z některých senzorů. Data se analyzují automaticky po jejich načtení ze senzoru. Uživatel má tak okamžitou zpětnou vazbu o jeho aktuální zóně tepové frekvence, BMI, nebo upozornění na hypertenzi.

5.1 Analýza zón tepové frekvence

Tepová frekvence je jedním z důležitých aspektů při pohybových aktivitách. Zátěž způsobuje, že svaly spotřebovávají více kyslíku, který je nutné doplňovat. To zvětšuje frekvenci dýchání a i srdce musí rychleji tepat. [18]

Zjištění aktuální zóny tepové frekvence napomáhá rozpoznat stupeň zatížení organismu. Lze tak například přibližně určit, kdy tělo hubne, tím že spaluje tuky. Nebo už je námaha intenzivnější a místo spalování tuků dochází ke spalování cukrů.
Pro výpočet zóny tepové frekvence používá aplikace obecnější rovnici založenou na věku. Ta spočívá v odhadu maxiální tepové frekvence z věku (z nastavení uživatelského profilu) a konstanty 220 pro muže nebo 226 pro ženy:

$$maxHR = 220 - age$$

Aktuální zóna se spočítá podle tabulky 5.1, výpočet probíhá online při načítání dat ze senzoru srdečního tepu v hrudním pásu. Uživatel tak může v aplikaci sledovat okamžité změny zón. Aktivní zóna je zvýrazněna zeleně (viz. obrázek A.7a).

Key Target Zones	
(Zone 1) 60-70% of maximum heart rate	weight loss, building endurance
(Zone 2) 70-80% of maximum heart rate	weight management, improving cardio fitness
(Zone 3) 80%+ of maximum heart rate	interval workouts

Tabulka 5.1: Cílové zóny tepové frekvence podle [25]

5.2 Index tělesné hmotnosti

Index tělesné hmotnosti (Body Mass Index – BMI), taktéž nazývaný Queteletův index, je statistická metoda, která porovnává hmotnost a výšku osob. Používá se jako indikátor podváhy, běžné tělesné hmotnosti, nadváhy nebo obezity. BMI je vhodné spíše pro sledování statistik. Pro jednoho člověka je tato hodnota méně vypovídající, protože nejsou brány v potaz další aspekty jako typ postavy, procento tělesného tuku nebo množství svalstva. Některé z těchto faktorů posktytuje ANT+ váha, BMI v rámci aplikace působí jako doplňující informace (viz. obrázek A.7b).
Body Mass Index se vypočítá jako poměr hmotnosti v kilogramech vůči druhé mocnině výšky v metrech [40].

$$BMI = \frac{hmotnost[kg]}{(vyska[m])^2}$$

5.3 Analýza hypertenze

Arteriální hypertenze je pojem pro opakované nebo přetrvávající zvýšení krevního tlaku – podle definice na hodnoty 140/90 a vyšší. V dnešní době se arteriální hypertenze považuje za jedno z civilizačních onemocnění. V České Republice se odhaduje výskyt arteriální hypertenze u dospělých mezi 25-64 let až na 35%, přičemž její výskyt se zvyšuje s věkem. [43] [38]

Mobilní aplikace používá pro analýzu hypertenze data z posledního měření z tonometru FORA P30 Plus. Je důležité, aby měření tlaku probíhalo v naprostém klidu. Sledovaná osoba musí sedět. Měření je doporučeno několikrát opakovat. Přesto je výsledek udávaný aplikací pouze informativní. V aplikaci je zobrazena tabulka 5.2. Řádek odpovídající měření tlaku se po naměření hodnot barevně zvýrazní.

Is the blood pressure right or is it already a hypertension?		
	Systolic (mm Hg)	Diastolic (mm Hg)
Normal pressure	to 140	to 90
Lower hypertension	140 – 179	90 – 104
Medium heavy hypertension	180 – 199	105 – 114
Heavy hypertension	200 and more	115 and more

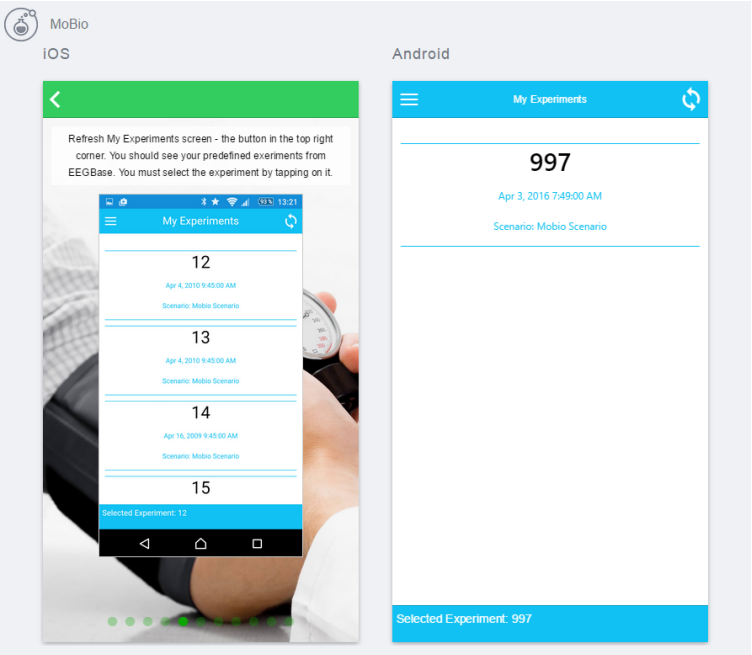
Tabulka 5.2: Přibližné určení hodnot krevního tlaku

6 Testování

Testování mobilní aplikace bylo důležitou a komplikovanou součástí tohoto projektu. Volba hybridní aplikace umožnila testovat cca 30% funckionality pomocí konzole webového prohlížeče Chrome. Komunikace se senory a chování aplikace v prostředí Android bylo nutné testovat buď v emulátoru nebo přímo v mobilním telefonu.

6.1 Testování uživatelského rozhraní

Testování uživatelského rozhraní je případ, kdy bylo možné si vystačit s webovým prohlížečem. Ionic framework integruje vlastní server pro testování aplikace. Server lze spustit příkazem `ionic serve`. Velice užitečný je přepínač `-l`, pomocí kterého Ionicu řekneme, že se má v prohlížeči spustit testovací laboratoř - porovnání toho, jak se různé komponenty UI zobrazují na mobilních zařízeních s operačním systémem Android nebo iOS (viz. obrázek 6.1). Ionic

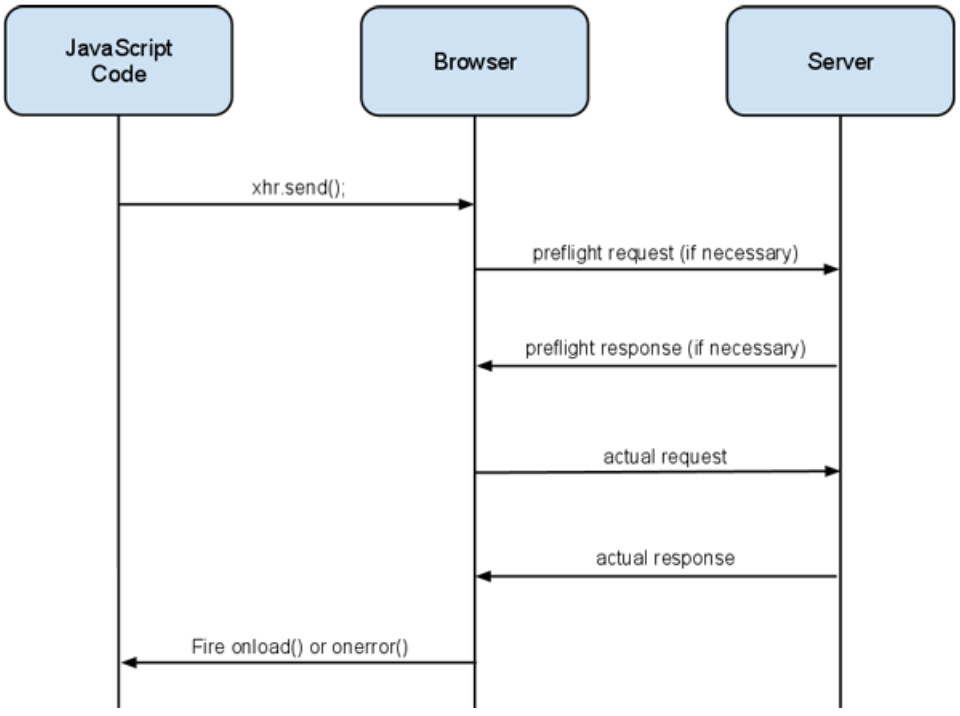


Obrázek 6.1: Ukázka Android a iOS aplikace pomocí Ionic Lab

Lab zároveň umožňuje dvě zobrazení aplikace procházet nezávisle, což může urychlit opakované testování v průběhu vývoje. Ionic server také poskytuje tzv. live reload, což je automatické obnovování webové stránky při uložení jakéhokoli zdrojového souboru - opět velmi vhodné pro pohodlný vývoj.

6.2 Testování komunikace s EEGBase

Ve webovém prohlížeči bylo také možné testovat komunikaci s REST službami portálu EEGBase. Pouze data ze senzorů bylo nutné simulovat. Odesílání a přijímání požadavků pomocí technologie AJAX mělo při testování z webového prohlížeče jednu odlišnost od mobilního telefonu. A to, že webový prohlížeč odesílá navíc *preflight* dotaz (diagram 6.2). Zjišťuje tak odesláním dotazu metodou OPTIONS, zda server umožňuje komunikaci z cizího zdroje (CORS). Až po kladné odpovědi prohlížeč odešle požadovaý dotaz (POST, GET...). Server bylo nutné nastavit, aby preflight dotazy podporoval (kapitola 4.4).



Obrázek 6.2: Jak probíhá preflight dotaz, převzato z [41]

6.3 Testování pomocí dostupných senzorů

Pro testování načítání dat ze senzorů bylo nutné aplikaci vždy zkompilovat a nainstalovat buď do Android simulatoru nebo do mobilního telefonu. Tento postup vývoj poměrně znesnadňoval. Kompilace trvá přibližně minutu, což při několikerém opakování prodlužovalo práci.

Naopak výhodu opět přinesla hybridní aplikace. Chyby bylo možné buď odchytávat příkazem `adb logcat` nebo opět pomocí konzole prohlížeče Chrome a jeho nástroje *Inspekce zařízení*. Aplikace má díky Crosswalk integrovaný prohlížeč na platformě Chromio. Desktopový Chrome proto dokáže hybridní aplikaci spuštěnou v telefonu rozpoznat a chová se k ní jako k webové stránce. JavaScript lze ladit a krokovat.

6.3.1 Tlakoměr FORA

Tlakoměr bylo možné testovat pouze v aplikaci nainstalované v mobilním telefonu. Tlakoměr je nutné nejprve s telefonem spárovat podle postupu uvedeném v návodu. Pokud je tlakoměr správně spárovaný, tak se objeví v seznamu Bluetooth zařízení v aplikaci. V mém případě měl název *TAIDOC TD329*.

Tlakoměr se přepne do módu vysílání dat automatiky po každém měření. Měření vyžaduje nasazení manžety na paži a čekání než se manžeta napumpuje. Pro opakované testování není tato varianta příliš vhodná, protože zdržuje. Fyzické měření tlaku jsem využil pouze jednorázově pro získání několika měření, které se následně uložily do paměti tlakoměru.

Alternativou je dlouze podržet tlačítko `M`. Tato možnost není uvedená v návodu k tlakoměru, ale na zadní straně přístroje na štítku. Tlakoměr se tímto opět přepne do módu komunikace přes Bluetooth. To je také signalizováno blikáním modré diody. V čase, kdy tlakoměr komunikuje s centrálním zařízením modrá kontrolka svítí. Po ukončení komunikace zhasne.

6.3.2 Hrudní pás

Hrudní pás se senzorem pro měření srdečního tepu bylo jedinné ANT+ zařízení, které jsem měl v průběhu práce na mobilní aplikaci k dispozici. Hrudní pás jsem využil zejména v první fázi práce na komunikaci s ANT+ senzory. Ihned po nasazení pás začne vysílat data. Blíže je hrudní pás popsán v kapitole 2.3.1.

Nejdříve jsem si pomocí mobilní aplikace *ANT+ Plugin Sampler* ověřil, zda a jaká data hrudní pás vysílá. Tuto aplikaci jsem poté použil jako referenční a vůči ní jsem následně porovnával i výsledky z aplikace MoBio.

6.3.3 ANT+ USB adaptéry a mobilní telefon

Nejdůležitější součástí testování byly ANT+ USB adaptéry, které zajišťují komunikaci mezi senzory a jiným hardware. Jsou to miniaturní antény, které implementují protokol ANT a ANT+ profily. Z laboratoře na KIV jsem měl k dispozici modul *ANTUSB-m* od společnosti Dynastream Innovations. Ten jsem využil zpočátku k testování komunikace s hrudním pásem. Další senzory mi pomohl testovat adaptér od společnosti Anself, který jsem zakoupil na eBay. Oba senzory mají prakticky stejnou funkcionalitu. Liší se pouze cenou a velikostí (viz. obrázek 6.3).

Obrázek 6.3: ANT+ USB adaptéry

Na webu *thisisant.com* je k dispozici seznam zařízení kompatibilních s ANT+ protokolem. Opatřil jsem si proto mobilní telefon SONY Xperia Z Ultra. Podle specifikace má mít tento telefon zabudovaný ANT+ čip. Bohužel se mi nepodařilo komunikaci přes interní ANT+ čip sprovoznit. Podle diskuzních fór není tato funkcionalita v jistých verzích Android na tomto telefonu podporována. Telefon však podporuje techlogii USB On-The-Go (zkráceně OTG). Ta umožňuje k telefonu pomocí adaptéru připojit různá USB zařizení - například flash disk nebo ANT+ modul.

Aby ANT+ modul přes OTG fungoval, je nutné do telefonu nainstalovat následující aplikace, které jsou dostupné zdarma na Google Play. Po nainstalování fungují na pozadí.

- ANT Radio Service
- ANT+ Plugins Service
- ANT USB Service

6.4 Testování pomocí simulátorů

Mobilní aplikace podporuje kromě ANT+ senzoru pro měření srdečního tepu i dalších pět typů ANT+ čidel. K těm nebyl k dispozici hardware. Pro účely testování je bylo nutné nějakým způsobem simulovat.

Pro potřeby vývojářů jsou na webu *thisisant.com* ke stáhnutí dva programy pro simulaci ANT+ senzorů nebo displayů.

6.4.1 ANT+ Sensor Simulator

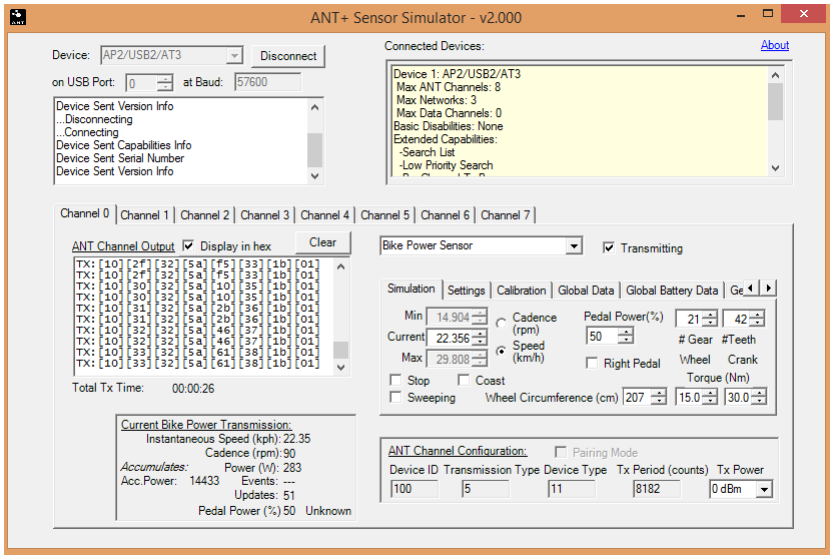
Pro fungování nástroje je nutné k počítači připojit ANT+ USB adaptér. Úvodní obrazovka programu obsahuje menu pro výběr USB zařízení a tlačítko *Connect* pro připojení k adaptéru. Pokud je připojení úspěšné, zobraz seí další volby. Software nabízí simulaci ANT+ senzorů až na osmi kanálech najednou, což mimo jiné vyplývá z možnosti ANT+ USB adaptérů. V pravé části okna se vybere typ senzoru a po zaškrtnutí volby *Transmitting* začne program přes USB adaptér vysílat data zvoleného ANT+ profilu. Pro každý typ senzoru nabízí Simulator různá nastavení. Lze například vysílat statická

data nebo nastavit jejich pravidelnou změnu.
V levé části okna lze při vysílání sledovat výpis datového proudu.

Pomocí tohoto software jsem simuloval a testoval především tyto typy senzorů:

- Osobní váha
- Snímač kadence a rychlosti (i oddělené senzory)
- Senzor pro měření srdečního tepu

ANT+ Sensor Simulator má v současné době již zastavený vývoj a je označený jako *deprecated* (zastaralý). To ale nebránilo jeho použití. Jeho výhodou je, že pro fungování mu postačí mít k počítači připojený pouze jeden ANT+ USB adaptér. Také možnost nastavení více kanálů pomohla v aplikaci otestovat vyhledávání několika aktivních senzorů (a to i v kombinaci s fyzickým hrudním pásem).

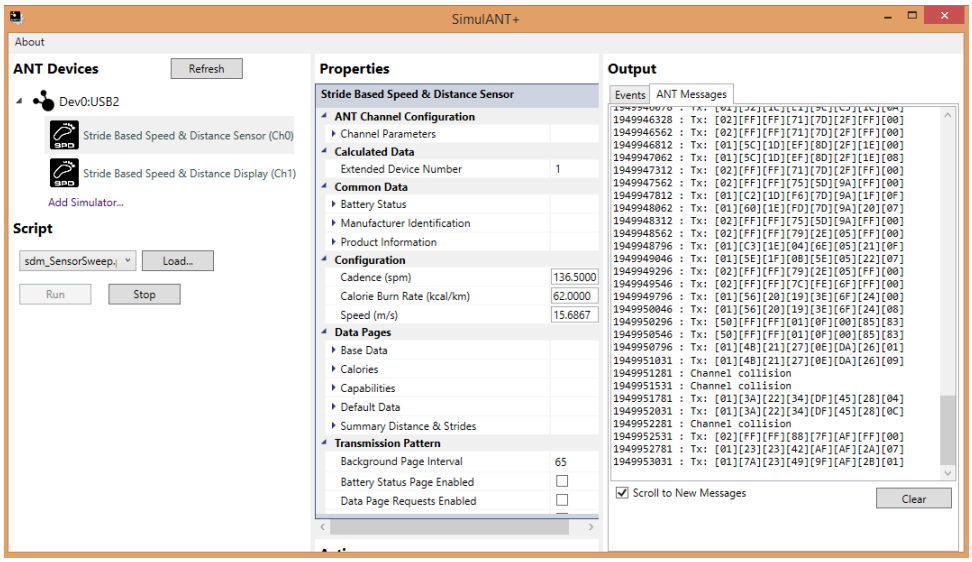


Obrázek 6.4: Ukázka programu ANT+ Sensor Simulator

6.4.2 SimulANT+

Nástroj SimulANT+ je nástupce programu ANT+ Sensor Simulator. Lze pomocí něj sledovat i podobnou množinu senzorů. Navíc obsahuje i volby pro testování i ANT+ displayů (například zobrazování na hodinkách a podobně).

Levá část okna slouží k přidání typu senzoru. Lze i nahrát vlastní Python skript, ketrý pozmění výchozí simulaci. Prostřední část okna představuje nastavitelné vlastnosti senzorů. V pravé části simulátoru je konzole s výpisem událostí nebo datového proudu.



Obrázek 6.5: Ukázka programu SimulANT+

SimulANT+ je vhodný především pro testování komunikace mezi senzory a displayi v rámci simulátoru. SimulANT+ umožňuje mimo jiné simulovat i senzor, který měří procento kyslíku ve svalech. Původně měl být tento senzor obsažený i v aplikaci MoBio. Bohužel jeho podpora není zahrnuta v Android ANT+ SDK.

6.4.3 Testování v Android emulátoru

Sočástí Android SDK je i emulátor. Pomocí něj lze vytvořit virtuální zařízení se Systémem Android na běžném počítači. Aplikace MoBio byla částečně tes-

tována i tímto způsobem (v mezičase, kdy nebyl k dispozici vhodný telefon).

Pro zprovoznění komunikace mezi Android emulátorem a ANT+ USB adaptérem je nutná instalace programu *ANT Android emulator Bridge*. Tím se připojuje adaptér podobně jako v případě simulátorů. Pro správné fungování se do emulátoru musí nainstalovat i všechny výše uvedené aplikace pro podporu ANT (kapitola 6.3.3) a navíc také aplikace *ANT Emulator Configuration*. Ta je součástí archivu *ANT Android emulator Bridge*, který lze stáhnout na webu *thisisant.com*.

7 Zhodnocení práce a závěr

Hlavním výstupem této práce je mobilní aplikace s názvem MoBio, která umožňuje sbírat data z biosenzorů. Je to pažní tlakoměr, který komunikuje pomocí technologie Bluetooth, a dále sada zařízení s technologií ANT+: senzor pro měření srdečních tepu, krokoměr, osobní váha, dále pak cykloměřič rychlosti, kadence šlapání a kombinovaný měřič rychlosti a kadence.
Tlakoměr a senzor pro měření srdečního tepu jsem měl po celou dobu práce na aplikaci k dispozici. Funkce ostatních senzorů bylo nutné simulovat pomocí dostupných softwarových nástrojů.

Pomocí REST služeb mobilní aplikace komunikuje s portálem EEGBase. Aplikace využívá služby pro přihlášení uživatele, načtení existujících experimentů a pro nahrání naměřených dat na server. Data jsou na serveru uložena ke zvolenému experimentu v NoSQL databázi a je možné je prohlížet na stránce s experimentem.

Nepsaným cílem bylo vyvinout hybridní aplikaci, která bude mít stejné možnosti jako aplikace nativní a navíc ji bude možné použít na více platformách. To se podařilo z části - aplikace je plně funkční pod platformou Android. Vývoj pro iOS vyžaduje především podporu ANT+ SDK, která momentálně neexistuje.

Při vývoji aplikace jsem dbal na přívětivost uživatelského rozhraní. Aplikace je členěná do logických celků, které jsou barevně rozlišené. Plusem je i průvodce funkcemi, který lze v aplikaci spustit. Průvodce má za úkol uživatele seznámit s fungováním aplikace. Lze namítnout, že intuitivní aplikace nemusí obsahovat žádného průvodce. Je to oprávněná výtka a proto by bylo vhodné se na tuto skutečnost zaměřit v dalším vývoji aplikace.

Od začátku práce na mobilní aplikaci jsem plánoval i její distribuci mezi uživatele. Nemělo by totiž smysl zabývat se aplikací, která by skončila hned s první testovací verzí. Z tohoto důvodu jsem vytvořil vývojářský účet na Google Play pro publikování aplikací a pro aplikaci MoBio jsem založil vlastní stránku. Výhodou je možnost testovat a jednoduše distribuovat aplikaci mezi uživatele. V době psaní této práce zatím není aplikace dostupná pro všechny, ale pouze pro pozvané testery.

V porovnání s aplikací Elfyz Data Mobile Logger, která vznikla na KIV také pro účely sběru dat ze senzorů poskytuje MoBio větší škálu senzorů,

ze kterých lze data načítat. Výhodou je i ukládání dat ve formátu odML, kerý používají neurolaboratoře po celém světě. Data tak lze jednoduššeji sdílet. Zde ale jako nevýhodu shledávám velikost režijních dat formátu odML, není příliš vhodný pro mobilní přenos dat. Jako alternativa se nabízí použití datového modelu *NIX* [27].

Dále pak oproti Elfyz Data Mobile Logger je aplikace MoBio koncipovaná spíše pro použití v laboratoři. Neumožňuje například uchovávat data mezi jednotlivými měřeními. Data by měla být vždy odeslána po ukončení měření na server EEGBase. S použitím aplikace mimo laboratoř souvisí i možnost přidat sledování pozice a nadmořské výšky měření, které lze poté zobrazit na mapě. Většina moderních mobilních telefonů má integrovaný gyroskop, ten lze využít například pro měření zrychlování.

A Uživatelská příručka

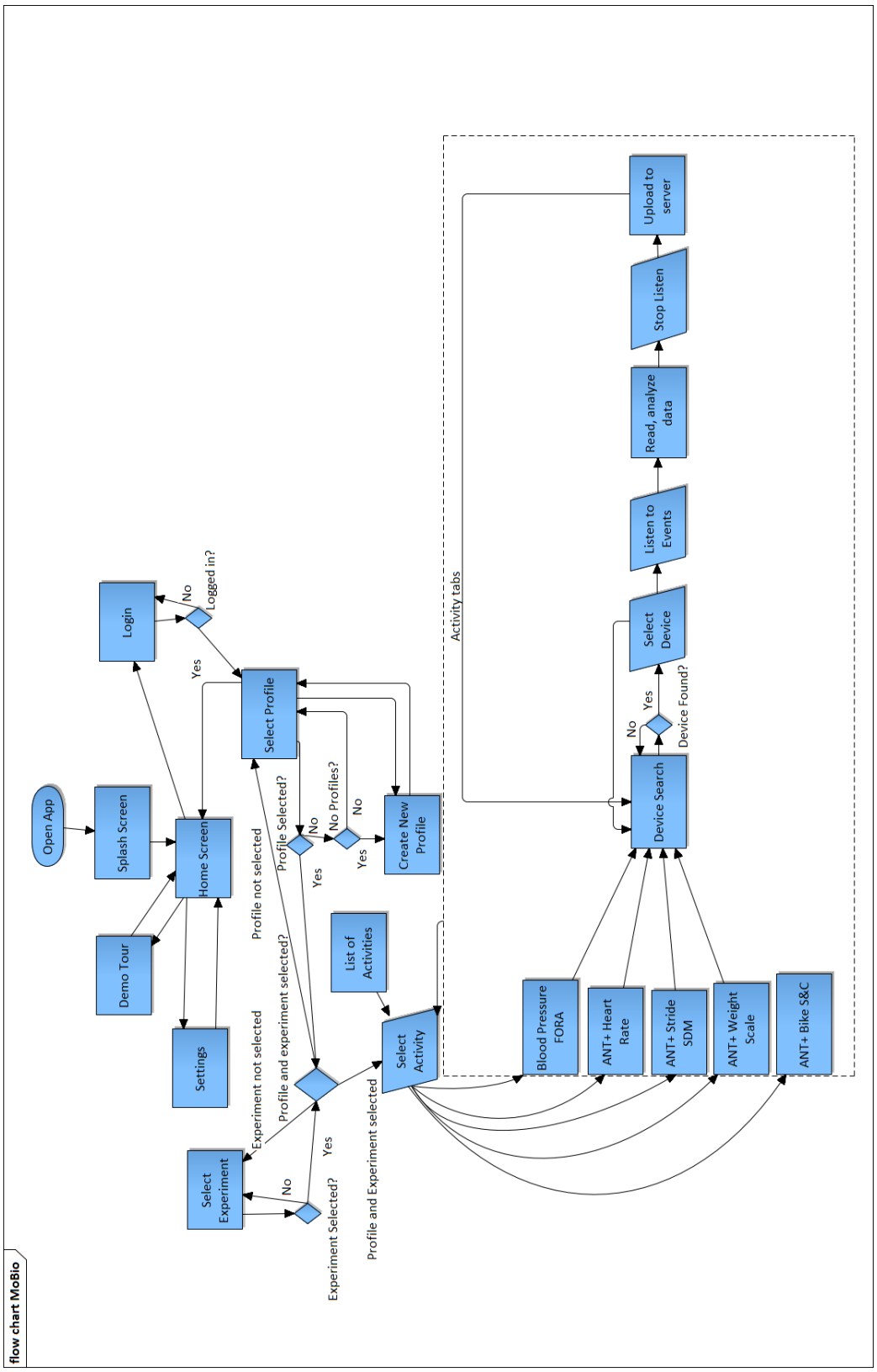
Diagram na obrázku A.1 naznačuje možné použití mobilní aplikace. Po spuštění aplikace ze zobrazí krátká úvodní obrazovka (splash screen), která plynule přejde na domovskou obrazovku (obrázek A.4a). Po prvním spuštění má uživatel možnost nastavit připojení k EEGBase, nebo zobrazit krátké demo aplikace s popisem jednotlivých obrazovek, nebo se může přihlásit (obrázek A.4b).

Po úspěšném přihlášení se aktivuje i poslední volba - vybrání profilu. Ověření uřivatelského jména a hesla probíhá pomocí REST služby portálu EEGBase.

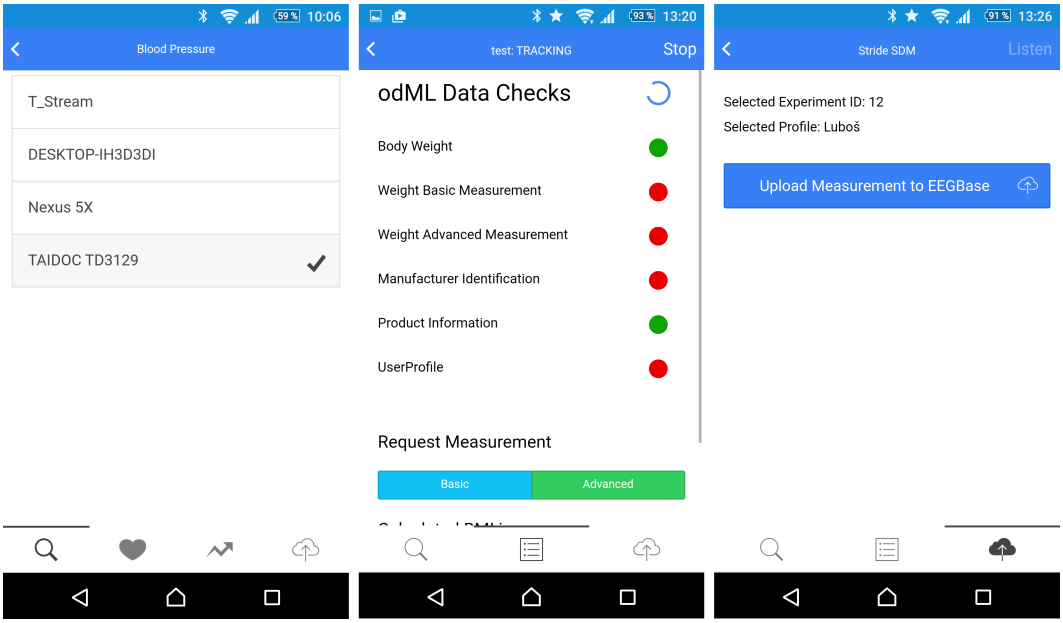
První verze aplikace však obsahovala ověření uživatelského jména a hesla pomocí služeb třetí strany - Auth0 [1]. To je služba, která nabízí integraci jednotného přihlášení. Důvodem použití Auth0 byla nemožnost serveru ověřit uživatelské jméno a heslo z požaddavku hybridní aplikace přes AJAX. Auth0 mělo velkou nevýhodu v tom, že byla použita oddělená databáze uživatelských účtů, která se nesynchronizovala s EEGBase. Po vyřešení restrikcí na straně serveru bylo od ověření službou Auth0 upuštěno.

Po přihlášení se přidá uživatelův profil, pokud ještě neexistuje. Údaje profilu lze změnit pouze v kontextu aplikace. Je zde také možnost nastavit vlasntí profil (obrázek A.5a). Data profilu slouží především pro komunikaci s ANT+ váhou.

Po zvolení profilu je nutné aktualizovat experimenty (záložka Experiments v levém menu). Položka experimentu obsahuje ID, datum vytvoření a název scénáře (obrázek A.5b). Uživatel dotykem zvolí experiment. V této chvíli se zaktivní položka Activities v menu (bez vybraného profilu a experimentu je neaktivní). Obrazovka Activities se skládá ze seznamu použitých typů senzorů s jejich krátkým popisem (obrázek A.6). Zvolením položky se uživatel přesune na obrazovku konkrétního typu senzoru. Každá obrazovka pro práci se senzorem se skládá ze tří až čtyř sekcí - tabů. Společné pro všechny jsou taby pro vyhledání senzoru, kontrolu načtených dat a nahrání dat na server (obrázek A.2). Kontrola načtených dat má v aplikaci název odML Data Checks. Jedná se o grafické znázornění toho, která data byla úspěšně přečtena a uložena do mezipaměti aplikace.



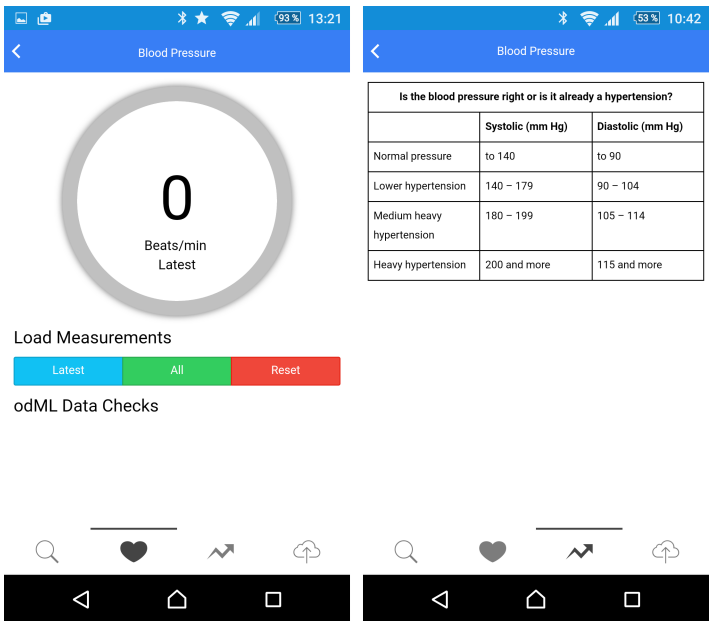
Obrázek A.1: Průchod aplikací



Obrázek A.2: Taby, které představují společné nebo podobné funkce v rámci obrazovek senzorů.

Prvky, ve kterých se obrazovky pro práci se senzory liší jsou následující:

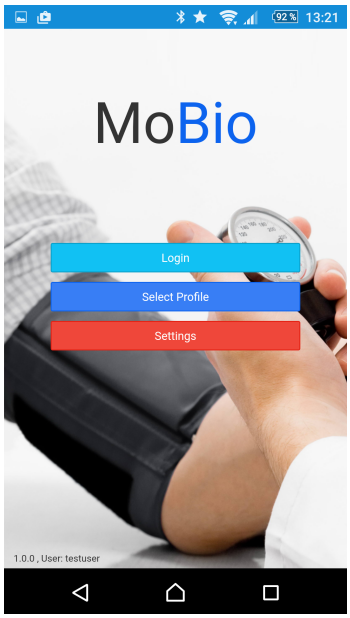
Tab odML Data Checks senzoru pro měření krevního tlaku obsahuje navíc animovaný koláčový graf. Ten zobrazuje počet tepů za minutu z posledního měření. Obrazovka tlakoměru má také extra tab s tabulkou pro analýzu hypertenze (obrázek A.3).



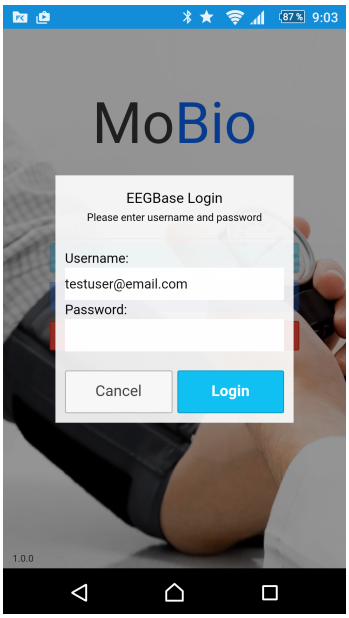
Obrázek A.3: Extra taby pro práci s tlakoměrem

Obrazovka senzoru pro měření srdeční tepu má navíc tab, kde se online zobrazují měřené hodnoty v liniovém grafu. Tab také obsahue tabulku analýzy zón srdečního tepu (obrázek A.7a).

Poslední výjimkou je tab odML Data Checks obrazovky senzoru ANT+ váha. Ten zobrazuje informaci o spočítaném BMI - obrázek A.7b.

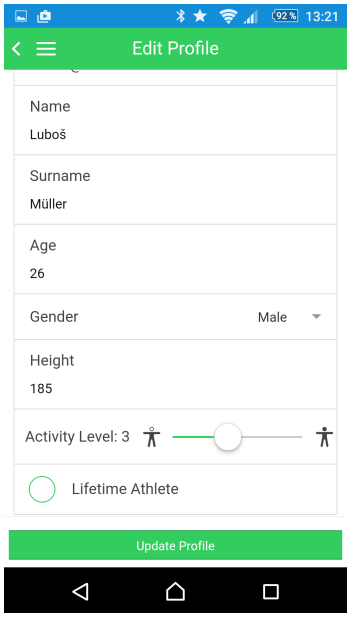


(a) Domovská obrazovka

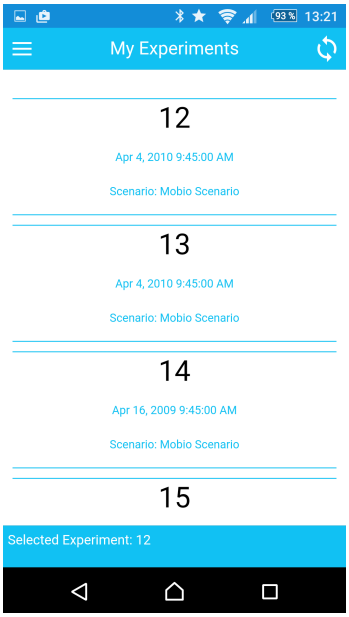


(b) Dialog přihlášení

Obrázek A.4: Obrazovky aplikace MoBio

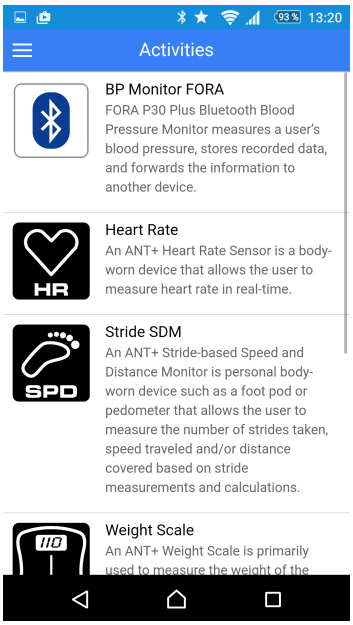


(a) Úprava profilu



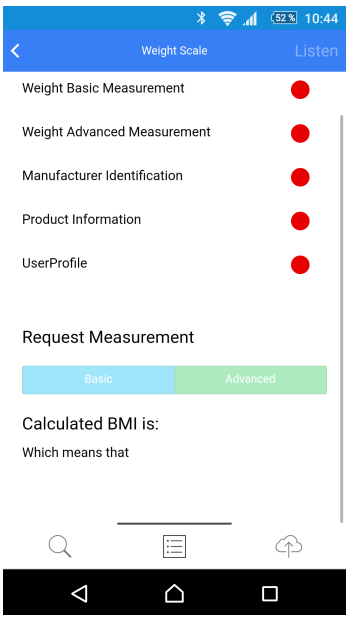
(b) Seznam experimentů

Obrázek A.5: Obrazovky aplikace MoBio



Obrázek A.6: Seznam aktivit, ke kterým lze načítat data

(a) Extra tab pro práci s HR senzorem



(b) Informace o spočítaném BMI

Obrázek A.7: Extra informace poskytnté aplikací

B Sestavení aplikace

Aplikaci MoBio pro Android lze sestavit pod operačním systémem Windows, Linux i MAC OS X. Návod na sestavení aplikace předpokládá systém Windows.

B.0.1 Android SDK

Jako první je potřeba nainstalovat Android SDK (pozn. vyžaduje alespoň JDK7). Není nutné instalovat kompletní Android studio, které je k vývoji aplikae zbytečné. Postačí pouze nutné knihovny Andorid API a nástroje pro ladění a testování.

- Stáhnout Android SDK z `http://dl.google.com/android/installer_r24.4.1-windows.exe`
- Nainstalovat Android SDK
- Vytvořit systémovou proměnnou `ANDROID_HOME`. Jako hodnotu bude mít cestu k nainstalovanému SDK: `[ANDROID_SDK_DIR\android-sdk`. Např. `c:\android\android-sdk`
- Do systémové proměnné `PATH` přidat cestu k `tools\` a `platform-tools\`. Např. `%ANDROID_HOME%\tools` a `%ANDROID_HOME%\platforom-tools`

B.0.2 Node.js

Vývoj a testování aplikace vyžaduje mít nainstalovaný Node.js verze 0.12.2. Nejnovější verze pravděpodobně nebude kompatibilní s nastavením prostředí pro vývoj. Společně s Node.js se nainstaluje i systém pro správu balíčků *npm*.

- Stáhnout Node.js: `https://nodejs.org/download/release/v0.12.2/`
- Nainstalovat Node.js

B.0.3 Cordova a Ionic framework

Pokud máme nainstalované *npm*, tak je třeba přidat globálně i prostředí Apache Cordova a Ionic framework.

- Spustit příkaz `npm install -g cordova ionic`

B.0.4 Příprava na sestavení aplikace

Všechny zdrojové soubory jsou dostupné na GitHub.

- Naklonovat aplikaci z větve master:
  `git clone https://github.com/NEUROINFORMATICS-GROUP-FAV-KIV-ZCU/MoBio.git`
- Nainstalovat Node balíčky. Zdrojové soubory obsahují přopravený *package.json* s informacemi o projektu.
  `npm install`
- Přidat platformu a nainstalovat pluginy. To lze provést příkazem

  `ionic platform add android@4.1.1`.

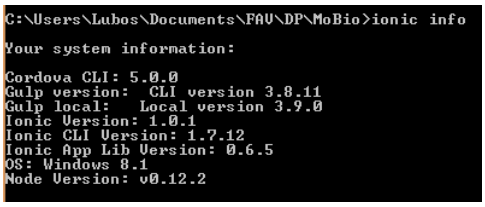
  Lepší je ale při prvním sestavení spustit příkaz

  `ionic state reset`

  Ten použije předkonfigurované nastavení.
- Nainstalovat prohlížeč Crosswalk (zajistí kompatibilitu mezi různými verzemi Android a WebView).

  `ionic browser add crosswalk`

Příkazem `ionic info` lze ověřit výsledek instalace. Výstup by měl vypadat přibližně jako na obrázku B.1.



Obrázek B.1: Výstup příkazu ionic info

B.0.5 Sestavení aplikace pro vývoj

Vývojový build je vhodný zejména pro testování komunikace se senzory.

- Aplikace se přeloží příkazem `ionic build android`
- přeložené APK lze nalézt v adresáři:
  `[PROJECT_DIR]\platforms\android\build\outputs\apk`
  Bude zde několik variant APK, pro instalaci na většinu mobilních zařízení (s architekturou ARM) je vhodné APK s názvem `android-armv7-debug.apk`.
- APK lze nainstalovat do telefonu, který má povolené instalování aplikací z neznámých zdrojů.

B.0.6 Sestavení aplikace pro publikování

Release build je vhodný především pro publikování aplikace na Google Play.

- Aplikace se přeloží příkazem `ionic build --release android`
- přeložené APK lze nalézt v adresáři:
  `[PROJECT_DIR]\platforms\android\build\outputs\apk`
  Bude zde několik variant nepodepsaných APK, pro instalaci na většinu mobilních zařízení (s architekturou ARM) je vhodné APK s názvem `android-armv7-release-unsigned.apk`, dále jen APK.

B.0.7 Podepsání APK

Aby bylo možné APK publikovat, je nutné ho podepsat privátním klíčem a tzv. optimalizovat. Klíč byl pro aplikaci vygenerován v průběhu vývoje a je součásti projektu, umístěný v kořenovém adresáři. Na Google Play může být aplikace publikována pouze, poud je podepsána tímto konkrétním klíčem: `mobio-release-key.keystore`. Podepsání vyžaduje heslo, které v textu této práce není z bezpečnostních důvodů uvedeno.
Postup je následující:

- Zkopírovat soubor s klíčem do `[PROJECT_DIR]\platforms\android\build\outputs\apk` a přesunout se do tohoto adresáře.
- Spustit příkaz
  `jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore mobio-release-key.keystore android-armv7-release-unsigned.apk mobio`
- Zadat heslo. Lze zjistit u autora nebo vedoucího této práce.
- Optimalizovat příkazem

  `zipalign -v 4 android-armv7-release-unsigned.apk MoBio.apk`

  Pokud systém, hlásí, že program *zipalign* neexistuje, lze jej najít v adresářové struktuře Android SDK: `\path\to\Android\sdk\build-tools\VERSION\zipalign`
- Výsledné APK je vytvořeno pod názvem `MoBio.apk`

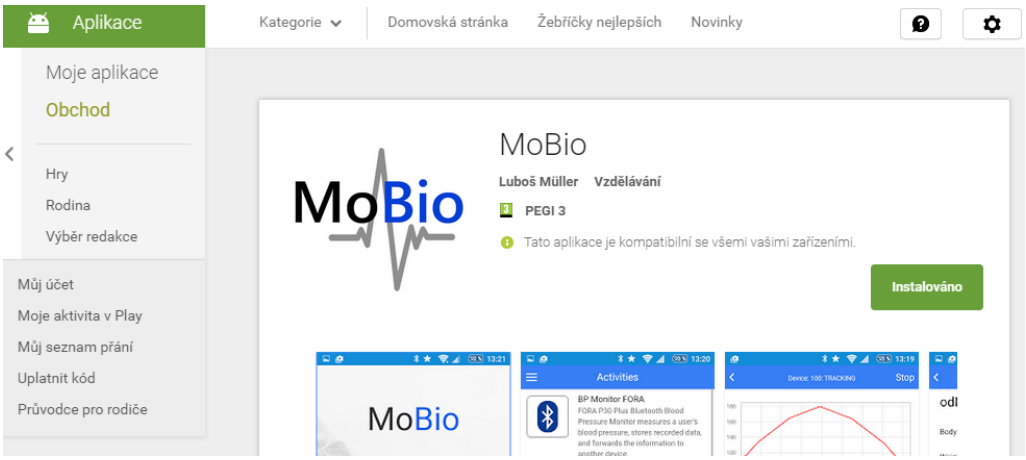
C Publikování na Google Play

Kvůli publikování aplikace MoBio na Google Play jsem založil vývojářský účet `mobiozcu@gmail.com`. Google založení vývojářského účtu zpoplatňuje $25. Ve vývojářské konzoli jsem přidal novou aplikaci a nastavil všechny potřebné údaje pro publikování (název, textový popis, screenshoty, loga, prohlášení o nezávadnosti obsahu...).

Aplikace je v době psaní tohoto textu dostupná ve formě otevřeného alfa testování. To znamená, že si ji lze stáhnout pouze pod následujícím odkazem:

`https://play.google.com/apps/testing/cz.zcu.kiv.neuroinformatics.mobio`

Po přejití na odkaz, bude uživatel dotázán, zde se chce stát testerem. Pokud potvrdí, že ano, tak bude přesměrován na stránku aplikace MoBio na Google Play (obrázek C.1).



Obrázek C.1: Stránka aplikace MoBio na Google Play

D Použité zkratky

1. ANT - protokol pro bezdrátovou komunikaci
2. APK - Android application package
3. BMI - Body Mass Index
4. CORS - Cross-origin resource sharing
5. EEG - Elektroencefalografie
6. HTTP - Hypertext Transfer Protocol
7. GATT - Generic Attribute Profile
8. ISO / OSI - International Standards Organization / Open Systen Interconnection
9. JSON - JavaScript Object Notation
10. KIV - Katedra informatiky a výpočetní techniky
11. MVVM - Model–view–viewmodel
12. P2P - Peer-to-peer
13. REST - Representational State Transfer
14. SDK - Software development kit
15. odML - open metadata Markup Language
16. USB - Universal Serial Bus

Seznam obrázků

Seznam tabulek

Literatura

[1] Auth0, 4 2016. Dostupné z: https://auth0.com/.

[2] Senzor nožní Garmin - krokoměr SDM4, 4 2015. Dostupné z: http://www.bike-eshop.cz/prislusenstvi-6/senzor-nozni-garmin-2-krokomer.

[3] Chromium, 4 2016. Dostupné z: https://www.chromium.org/Home.

[4] Changes the default WebView to CrossWalk, 4 2016. Dostupné z: https://www.npmjs.com/package/cordova-plugin-crosswalk-webview.

[5] Cordova Device Plugin, 4 2016. Dostupné z: https://www.npmjs.com/package/cordova-plugin-device.

[6] G-Node, 4 2016. Dostupné z: http://www.g-node.org/.

[7] Cordova InAppBrowser Plugin, 4 2016. Dostupné z: https://www.npmjs.com/package/cordova-plugin-inappbrowser.

[8] INCF, 4 2016. Dostupné z: http://www.incf.org/.

[9] Ionic, 4 2016. Dostupné z: http://ionicframework.com/.

[10] jQuery Mobile, 4 2016. Dostupné z: https://jquerymobile.com/.

[11] Ionic Keyboard Plugin, 4 2016. Dostupné z: https://www.npmjs.com/package/ionic-plugin-keyboard.

[12] NinjaMock, 4 2016. Dostupné z: http://ninjamock.com/.

[13] Cordova Splashscreen Plugin, 4 2016. Dostupné z: https://www.npmjs.com/package/cordova-plugin-splashscreen.

[14] Cordova Whitelist Plugin. Dostupné z: https://www.npmjs.com/package/cordova-plugin-whitelist.

[15] Apache Wicket, 4 2016. Dostupné z: http://wicket.apache.org/.

[16] APACHE. Apache License, 2004. Dostupné z: http://www.apache.org/licenses/LICENSE-2.0.

[17] ASSOCIATION, I. S. IEEE 802.15 - 2005. Dostupné z: https://standards.ieee.org/about/get/802/802.15.html.

[18] Benson Roy, Connolly Declan. *Trénink podle srdeční frekvence*. Grada. ISBN 978-80-247-4036-2.

[19] BOSTOCK, M. Data-Driven Documents, 4 2016. Dostupné z: https://d3js.org/.

[20] CORDOVA. Apache Cordova, 4 2016a. Dostupné z: https://cordova.apache.org/.

[21] CORDOVA. Cordova Architecture, 4 2016b. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/overview/index.html#architecture.

[22] CORDOVA. Apache Cordova, 4 2016c. Dostupné z: https://cordova.apache.org/plugins/.

[23] David Kovařík. Bluetooth – modrozub pod drobnohledem (vědecké okénko), 12 2011. Dostupné z: http://mobilizujeme.cz/clanky/bluetooth-modrozub-pod-drobnohledem-vedecke-okenko/.

[24] Don Coleman. Bluetooth Serial Plugin for PhoneGap, 4 2016. Dostupné z: https://github.com/don/BluetoothSerial.

[25] ELECTRO, P. How to calculate target heart rate zone, 4 2016. Dostupné z: http://support.polar.com/us-en/support/How_to_calculate_target_heart_rate_zone_.

[26] EVITO. eVito inteligentní váha SL, 4 2015. Dostupné z: https://www.evito.cz/eshop/product/3.

[27] G-NODE, 2016a. Dostupné z: https://github.com/G-Node/nix/wiki.

[28] G-NODE, 4 2016b. Dostupné z: http://www.g-node.org/projects/odml/erModel.png/view.

[29] GARMIN. Snímač kadence, 4 2016a. Dostupné z: http://www.garmin.cz/produkty/sport/prislusenstvi-pro-sport/bezdratova-cidla/snimac-kadence.html.

[30] GARMIN. Garmin Premium. Dostupné z: http://bit.ly/24q5jch.

[31] GARMIN. Snímač rychlosti, 4 2016b. Dostupné z: http://www.garmin.cz/produkty/sport/prislusenstvi-pro-sport/bezdratova-cidla/snimac-rychlosti.html.

[32] GARMIN. *GSC 10 návod k obsluze*, 1 2015. Dostupné z: http://www.garmin.cz/files/documents/garmin_gsc_10_om_cs.pdf.

[33] GOOGLE. AngularJS, 4 2016. Dostupné z: https://angularjs.org/.

[34] INC., D. E. MVVM, 4 2016. Dostupné z: http://js.devexpress.com/MobileDevelopment/Performance/.

[35] Jan Grewe, Thomas Wachtler, Jan Benda. A bottom-up approach to data annotation in neurophysiology. *frontiers in Neuroinformatics*. 8 2011. Dostupné z: http://journal.frontiersin.org/article/10.3389/fninf.2011.00016/full.

[36] Jan Krupička. Mobilní aplikace pro monitoring pohybu a elektrofyziologických potenciálů. Master's thesis, ZČU FAV, 2015.

[37] Jay Raj. The Top 7 Hybrid Mobile App Frameworks, 7 2014. Dostupné z: http://www.sitepoint.com/top-7-hybrid-mobile-app-frameworks/.

[38] Jiří Widimský a kol. *Arteriální hypertenze - současné klinické trendy*. Triton, 2007. ISBN 978-80-7254-962-7.

[39] Leonard Richardson, Sam Ruby. *RESTful Web service*. O'Reilly Media, 2007. ISBN 978-0-596-52926-0.

[40] Linda A. Ferrera. *Body Mass Index: New Research*. Nova Sciene Publishers Inc., 2005. ISBN 1-59454-282-1.

[41] Monsur Hossain. Using CORS, 10 2013. Dostupné z: http://www.html5rocks.com/en/tutorials/cors/.

[42] Monsur Hossain. *CORS in Action: Creating and consuming cross-origin APIs*. Manning Publications, 2014. ISBN 978-1617291821.

[43] prof. MUDr Josef Kautzner, CSc. Arteriální hypertenze, 4 2016. Dostupné z: http://www.ikem-kardiologie.cz/cs/pro-pacienty/co-u-nas-lecime/arterialni-hypertenze/.

[44] SENCHA. Sencha Touch, 4 2016. Dostupné z: https://www.sencha.com/products/touch/.

[45] THISISANT.COM. ANT+ Products, 04 2016a. Dostupné z: https://www.thisisant.com/directory.

[46] THISISANT.COM. *ANT Message Protocol and Usage*, 04 2016b. Dostupné z: https://www.thisisant.com/resources/ant-message-protocol-and-usage.

[47] THISISANT.COM. ANT+ Android SDK. Dostupné z: http://www.thisisant.com/developer/resources/downloads/.

[48] Vivek Chopra, Sing Li, Rupert Jones, Jon Eaves, John T. Bell. *Beginning JavaServer Pages*. Wiley Publishing, Inc., 2005. ISBN 0-7645-7485-X.