Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Projektová dokumentácia – Move2Play

Pavol Bielik, Peter Krátky, Štefan Mitrík, Michal Tomlein

Tím č. 1:	iDroids on Rails
Vedúci tímu:	Ing. Michal Barla, PhD.
Predmet:	Tímový projekt I
Študijný program:	Softvérové inžinierstvo, Ročník: 1.
Akademický rok:	2011/2012, zimný semester
Kontakt:	move2play@gmail.com

Obsah

1 Stav pred prvým šprintom

- Návrh a implementácia triedy EnergyManager — Pavol Bielik
- Rozšírenie EnergyManager o stavový automat — Pavol Bielik
- Návrh a implementácia triedy TrackingManager — Peter Krátky
- Analýza knižníc pre vizualizáciu grafov pre platformu Android — Štefan Mitrík
- Analýza grafov v existujúcich aplikáciách — Štefan Mitrík

1.1 Návrh a implementácia triedy EnergyManager

#732, Zodpovedná osoba: Pavol Bielik

1.1.1 Úloha

Návrh a implementacia riadiaceho prvku, ktorého cieľom je zapínanie a vypínanie rôznych spôsobov merania aktivity. Rozhodovanie má byť vykonávané na základe viacerých vstupov s možnosťou ľahkého rozšírenia v budúcnosti.

1.1.2 Návrh

Základnou triedou je EnergyManager. Jej hlavnou úlohou je určiť stupeň presnosti merania. Presnosť sa určuje pomocou troj-stupňovej škály low, medium, high. EnergyManager následne pošle požiadavku triede TrackingManager, ktorá podla stupňa presnosti určí a spustí vhodný spôsob merania (teda či chceme zapnúť GPS, Wifi alebo GSM). Presnosť merania taktiež ovplyvňujú externé udalosti ako napríklad stav baterky. EnergyManager ďalej určuje frekvenciu merania. Napríklad ak zistíme, že sa človek nehýbe, tak znížime frekvenciu merania za účelom úspory baterky.

EnergyManager vykonáva prepočet dvoma spôsobmi:

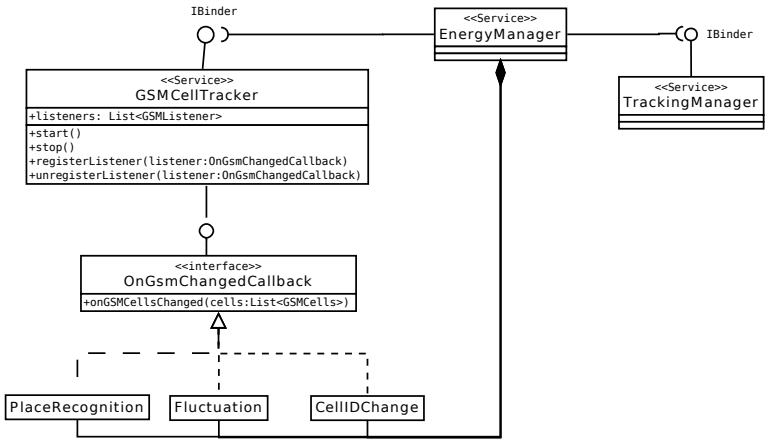
- Periodicky. Každých x minút.
- Po zmene stavu niektorého merača, ktorý o tejto zmene notifikuje EnergyManager.

EnergyManager na začiatku vytvorí listenery, ktoré periodicky spracúvajú dáta, ktoré im posiela GSMTracker a analyzujú ich. Momentálne všetky navrhnuté listenery spracúvajú dáta s GSMTrackera, čo vôbec nie je podmienkou. Je to ale spôsobené tým, že listenery ako aj zdroj ich dát musia byť energeticky nenáročné. Diagram tried riadiaceho prvku je zobrazený na obrázku. Medzi listenermi, ktoré sa dajú implementovať sú:

- PlaceRecognition – sledovanie pohybu človeka za účelom určenia súradníc, na ktorých je človek statický. Ak zistíme, že sa človek na týchto súradniciach nachádza, jeho pravdepodobnosť jeho pohybu je nižšia.
- Fluctuation –zisťovanie, či sa človek hýbe na základe GSM fluktuácie
- CellIDChange – zisťovanie pohybu človeka na základe zmeny pripojených GSM veží

1.1.3 Implementácia a testovanie

EnergyManager, GSMTracker a TrackingManager sú realizovane ako služby operačného systému android. GSMCellTracker a TrackingManager implementujú rozhranie IBinder, ktoré umožňuje posielanie správ ako aj priame volanie metód implementujúcej Service. Notifikácia o zmene stavu jednotlivých listenerov je realizovaná využitím mechanizmu broadcast správ, ktoré poskytuje android systém. Všetky výpočty a získavanie údajov o polohe používateľa by mali byt v realizovane v samostatnom vlákne.



Obr. 1: EnergyManager - Diagram tried

1.2 Rozšírenie EnergyManager o stavový automat

#749, Zodpovedná osoba: Pavol Bielik

1.2.1 Úloha

V rámci meracieho modulu je potrebné vytvoriť logiku riadenia jednotlivých senzorov na meranie pohybu, resp. zisťovanie či sa človek hýbe. Riadenie senzora spočíva v rozhodovaní, či daný senzor zapnúť, vypnúť, pripadne zmeniť nastavenie niektorých jeho parametrov.

1.2.2 Analýza

Návrh musí zabezpečiť nasledovné požiadavky:

- Ľahkú rozšíriteľnosť o ďalší typ senzora
- Možnosť podmienenia spustenia senzora určením podmienok.
- Zistenie aktuálneho stavu

1.2.3 Návrh

Vzhľadom na charakter problému je vhodné ho riešiť pomocou stavového automatu. Možné konfigurácie nastavenia meracích senzorov budú reprezentované ako stavy. Rozšíriteľnosť o nové typy senzorov je možné pridaním nových stavov a definovania možných prechodov. Pri použití senzorov GSM, Wi-Fi a GPS definujeme stavy:

- OFF – začiatočný a koncový stav, všetky senzoru sú vypnute
- MONITORING_WIFI – zapnuté je monitorovanie pomocou senzoru Wi-Fi
- MONITORING_GSM - zapnuté je monitorovanie pomocou senzoru GSM
- MONITORING_FULL – zapnuté je monitorovanie Wi-Fi aj GSM súčastne
- REQUEST_TRACKING_NETWORK - žiadosť o meranie pohybu používateľa s presnostou *NETWORK*.
- REQUEST_TRACKING_GPS - žiadosť o meranie pohybu používateľa s presnostou *GPS*
- TRACKING_NETWORK - žiadosť o meranie bolá úspešná
- TRACKING_GPS - žiadosť o meranie bolá úspešná

Pre každý stav je možné zadefinovať

- Akciu, ktorá sa vykoná pri vstupe do stavu
- Akciu, ktorá sa vykoná pri opustení stavu
- Podmienky, ktoré musia byt splnené pre vstup do stavu
- Definovanie možných prechodov do iných stavov

1.2.4 Implementácia a testovanie

Jednotlivé stavy sú implementované pomocou konštrukcie enum v jazyku Java, ktorá pre každý vymenovaný prvok vytvorí statickú triedu. Trieda EnergyManager má obsahuje premennú mState, ktorá reprezentuje aktuálny stav. Prepínanie stavov je realizované pomocou metódy changeState(State, EnergyManager), ktorá vykonáva kontrolu či je do daného vstupu možné prejsť a následne volanie metód opustenia stavu a vstupu do nového stavu. Príklad prázdneho stavu je nasledovný:

```
OFF (0) {

@Override
protected State onTurnOn(EnergyManager manager, State lastState) {
    ...
    return OFF ;
}

@Override
protected State onTurnOff(EnergyManager manager, State newState) {
    ...
    return OFF;
}

@Override
protected boolean allowedStates(State state) {
    switch (state) {
    case MONITORING_WIFI:
    case MONITORING_GSM:
        return true;
    default:
        return false;
    }
}
};
```

1.3 Návrh a implementácia triedy TrackingManager

#731, Zodpovedná osoba: Peter Krátky

1.3.1 Úloha

Návrh manažéra merania, ktorého úlohou je získavať súradnice pomocou GPS a Wifi.

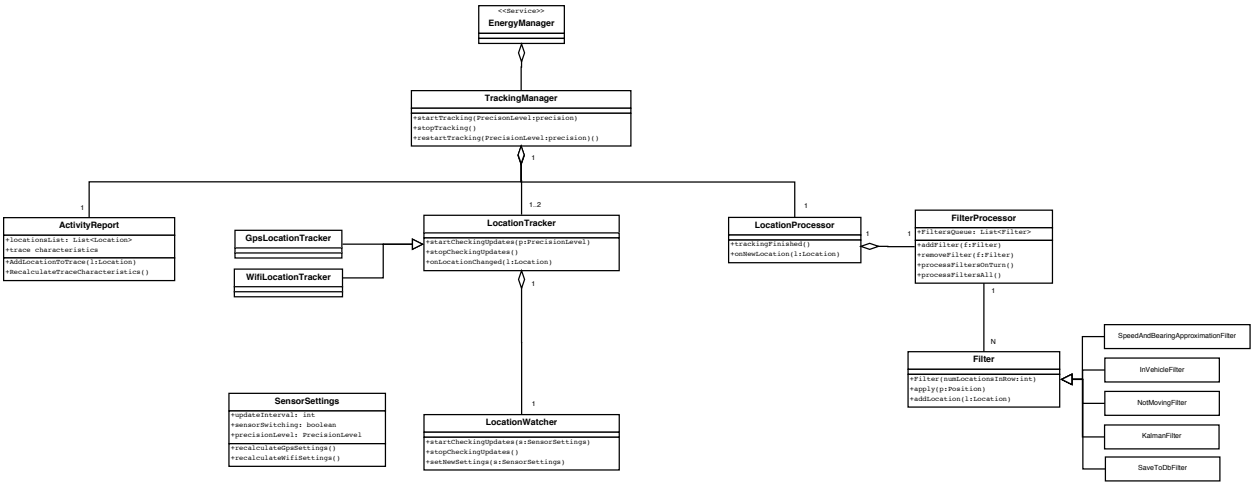
Pozn. súradnica = bod = lokácia na mape

Požiadavky na triedu TrackingManager sú nasledovné:

- Súradnice získavať dvoma spôsobmi – senzor je zapnutý stále alebo sa zapína len keď sa zisťuje pozícia
- Získané súradnice sa majú spresňovať aplikovaním rôznych filtrov
- Aplikácia filtrov sa má konať po dávkach, čiže naraz sa spracuje dané množstvo bodov
- Zaznamenať údaje do databázy

1.3.2 Návrh

Objektový návrh modulu je zobrazený v diagrame tried na obrázku.



Obr. 2: Diagram tried pre modul TrackingManager

Hlavnou triedou je *TrackingManager*, ktorý podľa zvolenej úrovne presnosti merania spustí *LocationTracker* (buď *GpsLocationTracker*, *WifiLocationTracker* alebo oba). Je teda spúšťač celého mechanizmu sledovania pohybu. Vykonávajú sa tu operácie nezávislé od druhu trackera ako napríklad vytvorenie reportu o aktivite, spojenie aktivity s predošlou, atď.

LocationTracker tvorí vrstvu pod *TrackingManager*-om, ktorý obsahuje špecifické operácie pre druh trackera. Nachádza sa tu stavový automat, ktorý mení stavy v závislosti od podmienok signálu na určenie polohy (*GpsTracker* má stavy OFF, SEARCHING, RECEIVED_LOCK). Spína *LocationWatcher* a poskytuje mu nastavenia pre neho zabalené ako *SensorSettings* a umožňuje meniť ich.

Na získavanie súradníc slúži *LocationWatcher*. Získava súradnice v intervaloch, ktoré sú určené poskytnutými nastaveniami. A to buď tak, že senzor necháva stále zapnutý alebo ho zapína a vypína.

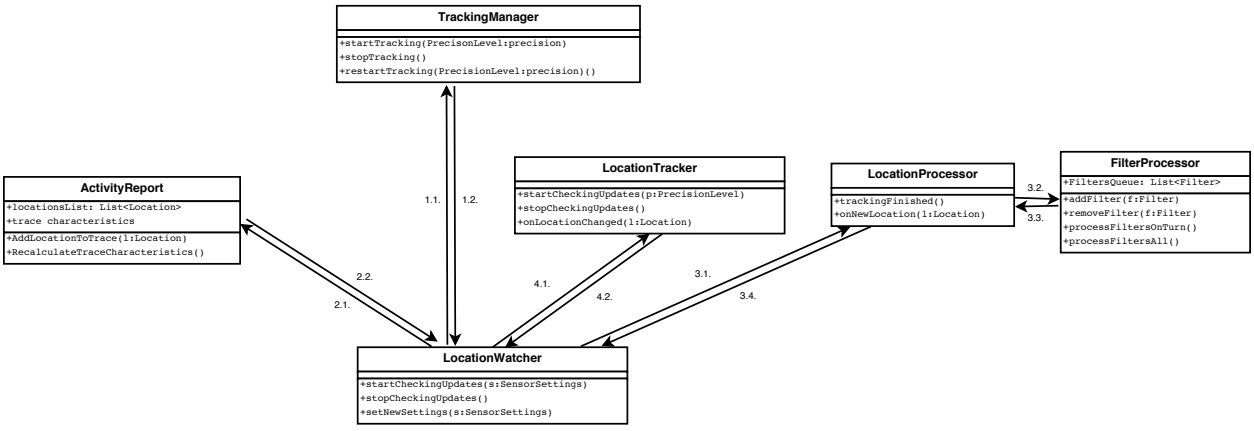
Získané súradnice uchováva trieda *ActivityReport*. Uchováva charakteristiky (dĺžka, priemerná rýchlosť) vykonávanej trasy a poskytuje metódu na ich prepočet.

Súradnice spracováva trieda *LocationProcessor*. Surové súradnice sa spracúvajú pomocou filtrov. Tie som zatiaľ identifikoval tieto:

- *SpeedAndBearingApproximationFilter* – filter upravuje veľké odchýlky v rýchlosti a smeru, prípadne dopočítava chýbajúce údaje z predošlých súradníc.
- *InVehicleFilter* – zisťuje, či sú získané súradnice z pohybu vo vozidle. Ak áno, označí ich tak, aby boli odfiltrované.
- *NotMovingFilter* – zisťuje pravdepodobnosť, či sa človek prestáva hýbať.
- *KalmanFilter* – vyhladzuje trasu podľa Kalmanovho algoritmu.
- Špeciálnym druhom filtra je *SaveToDbFilter*, ktorý dáva podnet, ktoré súradnice sa budú ukladať do databázy.

Naplánovanie a spúšťanie filtrov riadi *FilterProcessor*, ktorý si udržiava prioritný rad filtrov podľa toho, kedy sú naplánované. Filtre sa plánujú tak, aby sa spracovalo naraz viacero súradníc.

Na obrázku je v diagrame kolaborácií zobrazený tok správ. *LocationWatcher* generuje nové súradnice *ActivityLocation*.



Obr. 3: Diagram kolaborácií pre modul TrackingManager

1. Objekt je najprv zaslaný *TrackingManager*-u, ktorý ak ide o prvú súradnicu zisťuje, či ide o súradnicu, ktorú možno napojiť na predchádzajúcu aktivitu.
2. Nasledovne je objekt lokácie poslaný do *ActivityReport*, kde sa pridá do zoznamu súradníc v lokálnej pamäti.
3. Lokácia je zaslaná *LocationProcessor*-u, kde je prúdovo spracovaná všetkými filtrami. Posledný *SaveToDbFilter* dáva podnet na uloženie do databázy.

4. Nakoniec sa na základe spracovanej lokácie rozhodne *LocationTracker*, do akého stavu prejde (či zastaví meranie kvôli tomu, že človek stojí, alebo či prestane zaznamenávať lokácie, kvôli tomu, že sa cestuje v dopravnom prostriedku)

1.3.3 Implementácia a testovanie

Na implementáciu získavania súradníc bolo použité Google API. Bolo vykonané porovnanie so Skyhook API, ktoré je konkurenčným nástrojom, ale nebolo zistené zlepšenie.

Testovanie bolo vykonávané na viacerých telefónoch a a boli zistené obrovské rozdiely v kvalite získaných súradníc a taktiež v čase, ktorý bol potrebný na získanie prvej súradnice.

Pre pridanie nového filtra je potrebné zdediť od triedy *Filter*, doplniť metódu *ProcessFilter()*. Ďalej v konštruktore triedy *LocationProcessor* vytvoriť objekt tohto filtra a pridať do *FilterProcessor* s parametrom koľko súradníc naraz bude spracovávať.

1.4 Analýza knižníc pre vizualizáciu grafov pre platformu Android

#694, Zodpovedná osoba: Štefan Mitrík

1.4.1 Úloha

Analyzovať a vybrať vhodnú knižnicu pre vizualizáciu grafov pre platformu Andoird.

1.4.2 Analýza

Táto analýza vznikla v júli

Android Charts

- http://www.artfulbits.com/products/android/aiCharts.aspx#Overview_1

Komerčná knižnica pre platformu Android, ponúka celú škálu rôznych grafov. Knižnica ponúka viaceré typy grafov. Dokumentácia obsahuje ukážky a na stránke sa nachádza relatívne aktívne fórum. Hlavnou nevýhodou je to, že táto aplikácia je komerčná a cena jednej licencie pre vývojára sa pohybuje na úrovni 300 dolárov.

kiChart

- http://www.kidroid.com/kichart/

kiChart sa aktuálne (júl 2011) nachádza stále vo verzii 0.2. Podporuje len tri druhy grafov a minimálne možnosti štýlovania. kiChart je aj napriek svojej jednoduchej povahe komerčnou aplikáciou a cena jednej licencie pre vývojára je približne 20 dolárov.

KeepEdge – Android Chart

- http://www.keepedge.com/products/android_charting/#download

Komerčné riešenie bez akejkoľvek ukážky grafov na stránke. Chýba im aj akýkoľvek tutoriál. Cena vývojárskej licencie je okolo 300 dolárov.

GraphView

- https://github.com/jjoe64/GraphView-Demos

GraphView je minimalistická knižnica pre tvorbu grafov. Je poskytovaná zadarmo. Podobne ako u ostatných knižníc jej chýba hlbšia dokumentácia. Táto knižnica poskytuje API len na vytváranie základných čiarových grafov.

Andoridplot

- http://androidplot.com/wiki/Home

AndoridPlot sa prezentuje ako OpenSource knižnica a v súčasnosti ponúka štyri druhy grafov. Napriek tomu že sa knižnica prezentuje ako ako OpenSource je v súčasnosti dostupná len ako binárna JAR knižnica. Autori nevedia kedy sa im podarí uvoľniť OpenSource verziu.

ChardDroid

- http://code.google.com/p/chartdroid/

ChartDroid je knižnicou postavenou na Intent-och prostredníctvom ktorých sa dajú staticky generovať grafy. Pri tomto riešení nie je možné grafy jednoduchým spôsobom modifikovať. Knižnica poskytuje viacero druhov grafov. Aktivita na projekte je v rámci Google Code označená za nízku.

aChartEngine

- http://code.google.com/p/achartengine/

Achartengine je zrejme najpopulárnejšou open source android knižnicou pre grafy. Ponúka viac ako 10 druhov grafov. Aktivita je v rámci Google Code označená ako vyskoká a autor knižnice pravidelne opravuje nájdené chybu. Autor vytvoril demo aplikáciu kde ukazuje všetky druhy grafov a ich možnosti.

1.4.3 Zhrnutie analýzy

Pri výbere knižnice sme dbali nato, aby bola jednoduchá na ovládanie a aby bolo možné čo najjednoduchšie grafy prispôsobovať hlavne po vizuálnej stránke. Komerčné riešenia poskytovali relatívne drahé licencie a neumožňujú zasahovať do kódu, preto sme sa rozhodli využiť niektoré z OpenSource riešení.

Pri OpenSource riešeniach bola vo vačšine prípadov poskytnutá dokumentácia, ktorá nebola veľmi rozsiahla. Viaceré knižnice už dlhší čas nepreukazovali aktivitu v ich vývoji a poskytovali len základné druhy grafov.

Pre aChartEngine knižnicu sme sa rozhodli na základe nasledovných výhod:

- Open source riešenie
- Aktívny vývoj
- Široké spektrum grafov
- Relatívne vysoký počet príkladov

1.5 Analýza grafov v existujúcich aplikáciách

#736, Zodpovedná osoba: Štefan Mitrík

1.5.1 Úloha

Analyzovať grafy v súčastných aplikáciách venujúcich sa meraniu pohybu.

1.5.2 Analýza

Pre analýzu som vybral najpopulárnejšie aplikácie pre meranie a vyhodnocovanie pohybu pre platformu Android. Táto analýza vznikla v auguste 2011.

Endomondo

Endomondo je zrejme najpopulárnejšou aplikáciou na meranie pohybu pre platform Android. Používateľovi však neposkytuje v mobilnom telefóne žiadne grafy. Všetky grafy sú dostupné prostredníctvom webovej aplikácie.

Sports Tracker

Sports Tracker je aplikácia s prepracovaným používateľským rozhraním. Poskytuje pekný graf kde môže používateľ sledovať svoju rýchlosť a nadmorskú výšku.

Obr. 4: Sports tracker - graf

Interakcia v grafe Používateľ si môže vybrať či chce sledovať rýchlosť, nadmorskú výšku alebo oboje. Okrem toho si môže graf približovať, teda zmenšovať alebo zmenšovať zobrazovaný rozsah kilometrov na x-osi. Graf zväčšuje resp. zmenšuje prostredníctvom tlačidiel plus a mínus. Po grafe sa následne môže posúvať pohybom oranžového posuvníka medzi tlačidlami.

Okrem tohto grafu aplikácia zobrazuje graf srdcovej frekvencie, nakoľko som však nepoužil merač pulzu, nemal som namerané žiadne údaje.

RunKeeper

Aplikácia RunKeeper zobrazuje len jeden graf, ktorý zobrazuje relatívnu rýchlosť v určitom časovom úseku.

Obr. 5: RunKeeper - graf

Tento graf však nemá zobrazené hodnoty na x a y osi, preto nemá takmer žiadnu výpovednú hodnotu.

Interakcia v grafe Interakcia v grafe je umožnená posúvaním grafu prstom doľava alebo doprava.

RunStar

RunStar je aplikácia s jednoduchým a graficky príťažlivým používateľským rozhraním. Používateľovi poskytuje dva druhy grafov. Prvým sú sumárne grafy za mesiac, rok alebo za celý čas. Tu si používateľ môže pozrieť celkovú dĺžku ktorú prešiel, priemernú rýchlosť a celkový čas ktorý strávil.

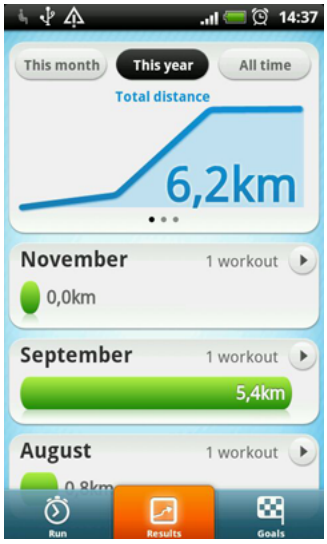
Druhým typom grafu, je graf pre jednotlivé behy. Tento graf zobrazuje rýchlosť a nadmorskú výšku v čase ale keďže nemá x-os a jeho čiary sú hrubé nie je veľmi prehľadný.

Interkacia v grafoch Grafy v tejto aplikácii neumožňujú žiadnu interkaciu.

Sports Tracker

Aplikácia s rovnakým názvom ako predošlý Sports Tracker. Poskytuje používateľovi celú škálu grafov pre jednotlivé aktivity ktoré vykonal. Patria tu:

- Rýchlosť v závislosti od vzdialenosti v km/h
- Rýchlosť v závislosti od vzdialenosti vyjadrená ako km/min (prevrátený predošlý graf)
- Nadmorská výška
- Rytmus



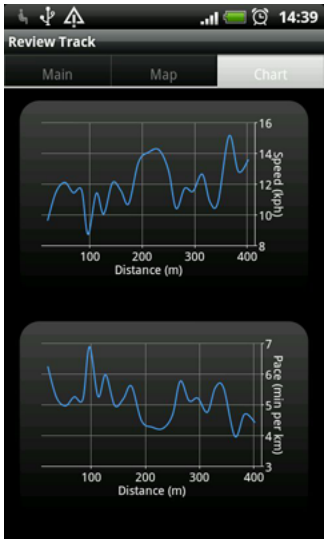
Obr. 6: RunStar - graf

Obr. 7: RunStar - graf

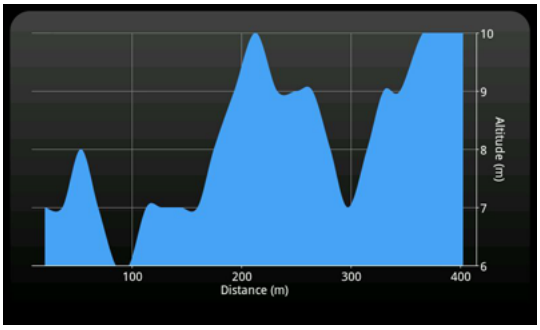
- Srdcová frekvencia
- Dychová frekvencia
- Teplota pokožky

Všetky grafy využívajú rovnaký jednoduchý čiarový graf, jedine graf nadmorskej výšky sa líši tým že má vyfarbenú plochu.

Interakcia v grafoch Po klepnutí na graf sa graf zobrazí na celej obrazovke a používateľ si ho môže približovať, vzďaľovať a pohybovať sa doprava resp. doľava. Interkacia je však veľmi neintuitívna. Dvojitým klepnutím sa približuje resp. vzďaľuje, neprišiel som však nato, kedy sa približuje a kedy vzďaľuje. Navyše sa zväčšuje/zmenšuje rozsah osi x aj y súčasne, čo nie je veľmi



Obr. 8: Sports Tracker - graf



Obr. 9: Sports Tracker - graf

praktické. Pohyb sa uskutočňuje klasicky, ťahaním prsta doprava, doľava resp. hore a dole. Preskrolovanie však nie je hladké ale skokové.

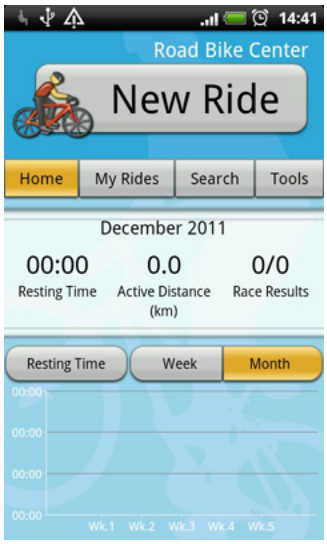
AllSports GPS

AllSport GPS je aplikácia na meranie rôznych druhov aktivít s nie príliš príťažlivým používateľským rozhraním.

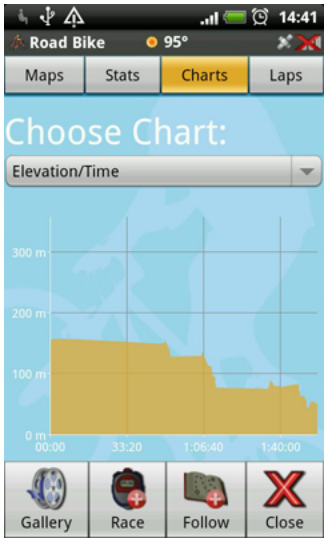
Po výbere aktivity napr. bicyklovanie sa používateľovi zobrazí stĺpcový graf pre danú aktivitu. Jednotlivé stĺpce predstavujú týždne, po kliknutí na týždeň dni. Používateľ si môže vybrať ktorý z približne 15 parametrov chce sledovať.

Ak si napríklad vyberie vzdialenosť, tak stĺpce vyjadrujú prejdenú vzdialenosť za daný týždeň resp. deň. Po klepnutí na stĺpec s dňom, sa zobrazí list aktivít, ktoré v daný deň vykonal s grafmi rýchlosti a nadmorskej výšky v čase, resp. rýchlosti.

Interakcia v grafoch Na stĺpcové grafy sa dá klepnúť, a po klepnutí sa zobrazí deň, resp. list aktivít v danom dni. Grafy pre konkrétnu aktivitu neponúkajú žiadnu možnosť interakcie čo ich robí v prípade dlhšie trvajúcich aktivít neprehľadnými.



Obr. 10: All Sports Gps - graf



Obr. 11: All Sports Gps - graf

2 Spišská Nová Ves

- Pridanie nenameranej aktivity — Michal Tomlein
- Poslanie debug dumpu aktivity e-mailom — Michal Tomlein
- Upozornenie na zakázané GPS pri zapnutí merania — Peter Krátky
- Upozornenie pri zakázaní GPS počas merania — Peter Krátky
- Šetriaci mód pri nízkej hladine baterky — Peter Krátky
- Odporúčanie denného plánu — Pavol Bielik
- Avatar — Štefan Mitrík
- Vibrovanie telefónu pri zmene stavu — Štefan Mitrík

2.1 Pridanie nenameranej aktivity

#944, Zodpovedná osoba: Michal Tomlein

2.1.1 Úloha

Keďže naším cieľom je poskytnúť používateľovi prehľad o všetkej vykonanej aktivite a chceme zároveň poskytnúť čo najlepšie odporúčanie, je potrebný spôsob, ako zadať do aplikácie aktivitu, ktorú nie je možné namerať senzormi telefónu (napr. plávanie).

2.1.2 Návrh

Na vyhodnotenie aktivity potrebujeme od používateľa minimálne nasledovné údaje:

- Typ aktivity
- Dĺžku trvania
- Dátum a čas

Nie je rozumné pýtať od používateľa viac, pretože s počtom údajov sa znižuje pravdepodobnosť, že používateľ bude ochotný údaje vyplniť. Zároveň je vhodné, aby zadávanie údajov predstavovalo čo najmenej kliknutí.

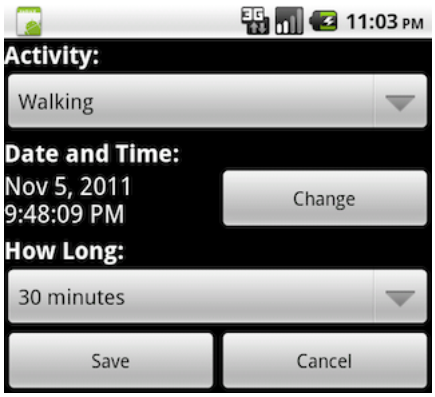
Dôležité je preto zvoliť vhodné východzie hodnoty. Pre začiatok môžeme predpokladať, že najčastejšie trvanie aktivity bude polhodina, a preto bude tento údaj predvolený. Neskôr, keď získame od väčšieho množstva používateľov údaje, sa toto nastavenie môže zmeniť. Ďalej môžeme predpokladať, že vo veľkej časti prípadov používateľ bude chcieť zadať práve vykonanú aktivitu, teda jej dátum a čas vieme prednastaviť na aktuálny dátum a čas mínus polhodina.

Okrem toho vieme ušetriť používateľovi prácu aj tak, že pri zvolení väčšej dĺžky trvania ako je rozdiel medzi aktuálnym časom a časom začiatku aktivity, čas začiatku aktivity automaticky posunieme dozadu. Pokiaľ však už používateľ čas začiatku zmenil, je nutné rešpektovať jeho voľbu a nemeniť ju.

2.1.3 Implementácia a testovanie

Do aplikácie sme pridali tlačidlo *Report activity*, po kliknutí na ktoré sa zobrazí rozhranie ako na nasledujúcom obrázku.

Keďže Android neposkytuje jednoducho použiteľný widget na zadávanie času a dátumu zároveň s presnosťou na 15 minút a sliderom na posúvanie sa medzi hodnotami, použili sme voľne dostupné riešenie DateSlider.



Obr. 12: Obrazovka pridania nenameranej aktivity

2.2 Poslanie debug dumpu aktivity e-mailom

#1019, Zodpovedná osoba: Michal Tomlein

2.2.1 Úloha

Pre účely vývoja a zlepšovania modulu sledovania aktivity je potrebný jednoduchý prístup k údajom, ktoré sa generujú počas jeho činnosti. Cieľom tejto úlohy je vytvoriť používateľské rozhranie určené pre testerov, ktoré umožní odoslať e-mailom dump týchto údajov na analýzu.

2.2.2 Návrh

Je potrebné rozhranie, v ktorom používateľ jedným kliknutím vyberie aktivitu (napr. podľa dátumu a času), údaje o ktorej chce odoslať na analýzu. Následne by malo byť možné do automaticky vytvoreného e-mailu dopísať komentár.

Keďže údaje majú byť odoslané v jednoducho čitateľnom a spracovateľnom formáte, bol vybraný formát CSV.

2.2.3 Implementácia a testovanie

Do aplikácie bolo pridané tlačidlo *Send dump*, po kliknutí na ktoré sa zobrazí zoznam aktivít, ktoré je možné odoslať. Po kliknutí na jednu z aktivít kontrolu prevezme OS, ktorý otvorí dialóg s možnými formami odoslania správy, kde je potrebné vybrať Gmail. Tým sa vytvorí e-mail s prílohami, pričom je možné dopísať komentár a e-mail odoslať.

Bola vytvorená trieda CsvFormatter, ktorá skonvertuje obsah Cursora na CSV reprezentaciu.

O vytvorenie dumpu aktivity pomocou CsvFormattera, vytvorenie e-mailu a Intentu na jeho odoslanie sa stará trieda ActivityMailDump. Umožňuje jednoducho pridať ďalšie tabuľky do dumpu.

Boli vytvorené JUnit testy CsvFormatterTest a ActivityMailDumpTest.

2.3 Upozornenie na zakázané GPS pri zapnutí merania

#945, Zodpovedná osoba: Peter Krátky

2.3.1 Úloha

Cieľom je upozorniť používateľa ak zapne meranie v aplikácií, že nemá povolené získavanie údajov o polohe cez GPS. V opačnom prípade by používateľ nemusel zbadať, že GPS má zakázané a nenameralo by mu nič.

2.3.2 Analýza

GPS poskytovateľ môže nadobúdať v telefóne dva základné stavy:

- *povolený* – umožňuje zaregistrovanému listeneru hľadať satelity a získavať údaje. Vtedy môže GPS senzor nadobúdať tri stavy, a to úsporný režim, hľadanie satelitov, získavanie súradníc pomocou satelitov
- *zakázaný* – GPS senzor je stále v úspornom režime a nedokáže prejsť do iného.

2.3.3 Návrh

Pred začatím merania je používateľ upozornený na nedostupnosť GPS dialógovým oknom, ktoré ho presmeruje na obrazovku s nastaveniami telefónu. Keď sa používateľ vráti do aplikácie, meranie sa začne automaticky. Toto je odporúčané riešenie ako povoliť GPS. Priame povolenie ani nie je možné. Napriek tomu existujú spôsoby, ako obísť toto zabezpečenie, avšak tie sú špecifické pre výrobcov a aj závislé od verzie operačného systému, a teda nemusia fungovať v budúcnosti.

2.3.4 Implementácia

Cez triedu *LocationManager* je jednoduché zistiť, či je GPS povolené. Ak nie, zobrazí sa štandardný *AlertDialog* s upozornením a ponukou presmerovania do obrazovky s nastaveniami. Meranie sa spustí až po návrate naspäť do aplikácie – použitím metódy *onResume()*.

2.4 Upozornenie pri zakázaní GPS počas merania

#946, Zodpovedná osoba: Peter Krátky

2.4.1 Úloha

Keď používateľ zakáže používanie GPS na určovanie polohy v nastaveniach telefónu, aplikácia na to upozorní vhodným spôsobom a ponúkne používateľovi akciu na riešenie vzniknutého problému. A to aj ak nie je aplikácia priamo spustená, ale beží len služba na meranie aktivity.

2.4.2 Analýza

Aplikácia Endomondo na svojej hlavnej obrazovke zobrazuje stav GPS. Ak je GPS zakázané v nastaveniach Androidu, zobrazuje sa upozornenie, že treba skontrolovať nastavenia GPS. Aplikácia SportsTracker upozorní aj dialógom, ale tiež len v rámci aplikácie, nie v rámci služby bežiacej na pozadí.

2.4.3 Návrh

Pre túto úlohu bola identifikovaná nasledujúca postupnosť akcií:

1. Odchytenie zakázania GPS používateľom
2. Zobrazenie notifikácie
3. Po otvorení notifikácie zobrazenie ponuky povolenia GPS
4. Po povolení GPS sa aplikácia vráti do pôvodného stavu

Podľa odporúčaní pre tvorbu aplikácií na Android by sa by proces, ktorý beží v pozadí (služba) nemal upozorňovať formou dialógu, ale notifikáciou v stavovom paneli alebo toastom.

Android nepodporuje priame povolenie senzoru GPS inak ako cez nastavenia telefónu kvôli bezpečnostným dôvodom., takže je potrebné presmerovať používateľa na obrazovku s nastaveniami.

2.4.4 Implementácia

Odchytenie zakázania GPS používateľom

Navrhované implementačné riešenia, ktoré na internetových diskusiách riešili tento problém boli nasledovné:

- Počúvanie systémových správ, ktoré hovoria o zmene nastavení pre poskytovateľov polohy problém: funkčné až od API 9
- Použitie rozhranie GpsStatus.Listener. Ten upozorňuje zmeny v stavoch GPS (úsporný režim, hľadanie satelitov, získavanie súradníc pomocou satelitov) problém: Neupozorňuje však na to, či bol používateľom zakázaný alebo povolený
- Sledovať v pravidelných intervaloch (3 sekundy), či je GPS dostupné. problém: Môže mať dopad na baterku

Riešenie, ktoré by bolo verziovo nezávislé a efektívnejšie sa podarilo nájsť. Zaregistrujeme listener pre zmeny v systémovej tabuľke, ktorá uchováva nastavenia. Aplikácia je síce upozornená na všetky zmeny v nastavenia, tie však sú zriedkavé. Kontrolujeme dostupnosť GPS a v prípade zmeny, notifikujeme.

Zobrazenie notifikácie

Notifikácia pri zakázaní GPS je implementovaná ako notifikácia v stavovom paneli. Pri povolení GPS notifikácia zmizne.

Po otvorení notifikácie zobrazenie ponuky povolenia GPS

Po kliknutí na notifikáciu sa zobrazí systémová obrazovka s nastaveniami poskytovateľov polohy.

Po povolení GPS sa aplikácia vráti do pôvodného stavu

Keď používateľ povolí GPS v nastaveniach, aplikácie pokračuje so sledovaním pohybu a notifikácia zmizne.

2.5 Šetriaci mód pri nízkej hladine baterky

#996, Zodpovedná osoba: Peter Krátky

2.5.1 Úloha

Môže vzniknúť situácia, keď používateľ je mimo domu, dochádza mu baterka a zabudne si vynúť meranie aktivity, tak zostane s vybitým telefónom. V tom prípade je prioritou nie meranie, ale používateľova baterka. Preto je dobré, ak sa pri nízkej hladine aplikácie prepne do šetriaceho módu.

2.5.2 Implementácia

Aplikácia prijíma systémové správy o hladine baterky. Stav batérie sleduje objekt triedy *BatteryManager*, ktorý upozorňuje zaregistrovaný listener o tom, že sa zmenila úroveň baterky.

Stavový automat v triede *EnergyManager* je obohatený o stav BATTERY_SAVING. *EnergyManager* spracuje informáciu o stave baterky tak, že ak je pod 20%, prejde automat prejde okamžite do stavu BATTERY_SAVING. Vtedy sa prestane vykonávať aktivita služby. Ak sa vráti nad danú hranicu, tak prejde do stavu MONITORING_WIFI.

2.6 Odporúčanie denného plánu

#952, Zodpovedná osoba: Pavol Bielik

2.6.1 Úloha

Táto úloha sa zaoberá návrhom a realizáciou základného spôsobu odporúčania aktivity pre používateľov. Cieľom odporúčania je poskytnúť používateľovi informáciu o vhodnom a odporúčanom množstve aktivity, ktoré je pre neho vhodné.

2.6.2 Analýza

Analýzu tvorí prehľad o existujúcich knižniciach pre pravidlové systémy v jazyky ruby, z ktorých sme si vybrali knižnicu Ruleby.

Dostupné pravidlové systémy pre jazyk Ruby

Ruleby

url https://github.com/codalytics/ruleby

aktivita July 26, 2011

verzia 0.9.b5

stav 171 watchers, 11 forks

popis Jediný schopný odvodzovací engine v ruby. Ostatné sú v podstate nepoužiteľné. Ak tento nebude stačiť bude treba asi hľadať niekde inde.

RDFS

url https://github.com/bendiken/rdfs

aktivita June 14, 2010

verzia 0.1.0

stav 10 watchers, 2 forks

popis Dopredný odvodzovací engine ktorý implementuje RDFS pravidlá.

Unruly

url http://rubygems.org/gems/unruly

aktivita May 13, 2011

verzia 0.0.1

stav nový projekt

popis Celkom čerstvý projekt ale verzia 0.0.1 bez žiadnej dokumentácie.

Ruby-Rules

url http://xircles.codehaus.org/projects/ruby-rules

aktivita March 30, 2007

verzia ?

stav mŕtve

popis Nepodarilo sa nájsť zdrojové kódy ani dokumentáciu.

Ruby Rools

url http://rools.rubyforge.org/

aktivita November 5, 2007

verzia 0.4

stav mŕtve

popis Dostupná základná dokumentácia.

2.6.3 Návrh

Odporúčanie vychádza z existujúcej verzie projektu move2play, ktorá bola implementovaná v prostredí .NET. Samotné odporúčanie pozostáva z viacerých častí:

- Tréningové plány
- Plán odporúčania
- Odporúčanie aktivity

Výsledkom odporúčania je odporučené množstvo aktivity pre zvolený deň a používateľa.

Tréningové plány

Pre účely základného spôsobu odporúčania aktivity sme navrhli plán chôdze, ktorý pozostáva z 12 úrovní od 5000 krokov až po 10500. Tréningový plán sa časom nemení pričom je možné doplnenia ďalších tréningových plánov.

Plán odporúčania

Plán odporúčania predstavuje základ pre odporúčanie aktivity. Definuje tréningový plán a aktuálnu úroveň používateľa. Každý plán má určené časové obdobie, kedy je platný, pričom v rovnakom časovom období môže byť platný plán iba jeden. Plán odporúčania sa periodicky prepočítava a upravuje úroveň používateľa na základe jeho vykonanej aktivity.

Odporúčanie aktivity

Pri odporúčaní množstva aktivity vychádzame s plánu odporúčania, ktorý je následne upravený o doménové znalosti. Doménové znalosti v oblasti fyzickej aktivity predstavujú znalosti o tom, ako sa ľudia vo všeobecnosti hýbu a faktory ovplyvňujúce vhodné množstvo fyzickej aktivity. V základnom návrhu odporúčania budeme uvažovať faktory veku, pohlavia a dňa v týždni. Pre účely odporúčania použijeme pravidlový systém, ktorý bude obsahovať fakty a pravidlá opisujúce faktory vhodného množstva fyzickej aktivity. Vďaka použitiu pravidlového systému bude systém ľahko rozšíriteľný do budúcnosti a umožní využiť výsledky predchádzajúcej práce.

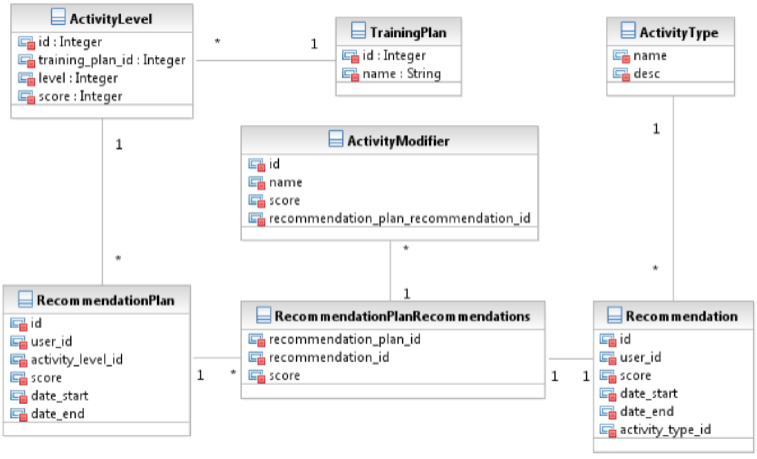
Mechanizmus odporúčania

Aktivita sa odporúča na nasledovný týždeň. Ak už bola odporučená tak sa znova neodporúča. Odporúčanie pre daný deň a používateľa má nasledovné kroky:

1. Aktualizácia plánu odporúčania a stanovenie základnej hodnoty vhodného množstva fyzickej aktivity.
2. inicializácia engine pravidlového systému a načítanie vhodných faktov a pravidiel
3. vykonanie odvodzovanie s dopredným reťazením, ktorého výsledkom je sada modifikácii množstva aktivity, ktoré sa následne aplikujú.
4. Po aplikovaní modifikácií sa uloží výsledok do databázy.

2.6.4 Implementácia a testovanie

Odporúčanie je implementované na strane servera v jazyku ruby. Pre pravidlový systém sme sa na základe existujúcich riešení rozhodli použiť gem ruleby. Pre účely odporúčania sme vytvorili cron job, ktorý každý deň odporučí vhodnú aktivity všetkým používateľom systému. Dátový model odporúčania je zobrazený na obrázku 1. Pre tabuľky sú vytvorené RESTful rozhrania a tabuľky recommendations a recommendation_plans sú synchronizované s ich ekvivalentmi v mobilnom telefóne.



Obr. 13: Dátový model odporúčania aktivity

Pri testovaní sme sa zamerali najmä na zaručenie konzistencie údajov. To spočívalo v zabezpečení, že napríklad nebude viac odporúčaní v jeden deň, viac plánov v odporúčaní s rovnakým dátumom, správne prepočítavanie plánu odporúčaní a ďalšie.

2.7 Avatar

#1020, Zodpovedná osoba: Štefan Mitrík

2.7.1 Úloha

Vytvoriť komponent avatara a jeho dátový model a zobraziť avatara z údajov z databázy. Pri vytvorení avatara treba vychádzať z prototypu, ktorý bol vytvorený na platformu WP7.

2.7.2 Návrh

Avatar sa vytvára vrstvením obrázkov. Android na rozdiel od platformy Windows Phone 7 nepodporuje adresárovú štruktúru v priečinku, kde sa ukladajú obrázky. Preto je nutné všetky obrázky premenovať a z *avatar/female/clothes/1_1_1.png* vzniklo *avatar_female_clothes_1_1_1.png*.

Pričom *avatar_female_clothes_1_1_1.png* reprezentuje obrázok oblečenia pre avatara ženského pohlavia a tri čísla 1_1_1 reprezentujú:

- Sadu
- Typ
- Farbu

V databáze by však bolo zbytočné vytvárať tri stĺpce pre každú grafickú časť avatara, preto ich reprezentujeme v jednom čísle. Napríklad 121314 sa transformuje na 2_3_4 alebo 213515 sa transformuje na 11_25_5. Číslo je rozdelené na tri dvojice cifier napr.: 21 — 35 — 15 a od každej dvojice sa odpočíta 10, čím nám vznikne výsledné číslo.

2.7.3 Implementácia

Android poskytuje na tvorbu komponentov `Fragment`, ktorý je však dostupný až od verzie 3.0, preto je do aplikácií, ktoré podporujú nižšiu verziu API potrebné importovať knižnice jeho podpory.

Fragment avatara pri inicializácii vyžaduje `ImageDataProvider`, ktorý poskytuje obrázky pre jednotlivé atribúty ako pozadie alebo oblečenie. `ImageDataProvider` poskytuje obrázky na základe používateľových údajov v databáze.

2.8 Vibrovanie telefónu pri zmene stavu

#1018, Zodpovedná osoba: Štefan Mitrík

2.8.1 Úloha

Vytvoriť vibračného manažéra s jednoduchým API, ktorý bude zabezpečovať vibrovanie mobilu s rôznymi dĺžkami a počtom opakovaní vibrovania. Implementovať volania do `EnergyManager`, aby telefón vibroval pri zmene stavu.

2.8.2 Návrh

Na vibrovanie sa v Androide využíva trieda `android.os.Vibrator`, ktorej inštancia sa dá získať z kontextu aplikácie ako systémová služba. `VibrationManager` vytvára wrapper nad touto triedou a zjednodušuje volanie jej metód.

2.8.3 Implementácia

`VibrationManager` poskytuje nasledovné metódy na vibrovanie:

- vibrate – jednorázové vibrovanie
- repetitiveVibrate – vibrovanie na základe vzoru

Pri oboch metódach sa dá nastaviť, koľkokrát sa má vibrovanie opakovať a aké dlhé pauzy majú byť medzi opakovaniami.

3 Poprad-Tatry

- Zabezpečenie servera — Michal Tomlein
- Jenkins pre Android aplikáciu — Michal Tomlein
- Spustenie synchronizácie so serverom — Michal Tomlein
- Miesta, kde sa nevykonáva meranie — Peter Krátky
- Filtrovanie pohybu v dopravnom prostriedku — Peter Krátky
- Android kompabilita obrazoviek na telefónoch s rôznym rozlíšením — Pavol Bielik
- Automatický build a notifikácia o novej verzii android aplikácie — Pavol Bielik
- Úvodná obrazovka — Štefan Mitrík

3.1 Zabezpečenie servera

#1404, Zodpovedná osoba: Michal Tomlein

3.1.1 Úloha

Cieľom tejto úlohy je zabezpečiť prístup a komunikáciu so serverom:

- inštaláciou firewallu (Shorewall),
- zakázaním prihlásenia sa ako root cez SSH,
- prihlasovaním pomocou SSH kľúčov,
- umožnením prístupu k webovej stránke Move2Play a production deployu Rails aplikácie cez HTTPS.

3.1.2 Implementácia a testovanie

Shorewall

Na server bol nainštalovaný Shorewall. Shorewall bol nastavený tak, že povoľuje komunikáciu na portoch:

- 22 (SSH)
- 80 (HTTP)
- 443 (HTTPS)
- 8080 (Jenkins)

Tiež je povolený Ping.

Pri nastavení boli zmenené konfiguračné súbory Shorewallu:

- `/etc/shorewall/interfaces`
- `/etc/shorewall/zones`
- `/etc/shorewall/policy`
- `/etc/shorewall/rules`
- `/etc/shorewall/shorewall.conf`

Root login

Root login bol vypnutý v súbore `/etc/ssh/sshd_config` a prihlásenie cez SSH je dovolené len členom skupiny sshusers:

```
PermitRootLogin no
AllowGroups sshusers
```

Bol vytvorený používateľ deployer, ktorý je členom sshusers.

Prihlasovanie pomocou SSH kľúčov

Prihlasovanie pomocou kľúčov bolo zapnuté v súbore `/etc/ssh/sshd_config` a prihlásenie pomocou hesla bolo zakázané:

```
RSAAuthentication yes
PubkeyAuthentication yes
PasswordAuthentication no
```

HTTPS

Pre HTTPS sa použilo východzie nastavenie Apache a automaticky vygenerovaný root certifikát.

Production a development (staging) deploy Rails aplikácií boli sprístupnené cez HTTPS, pričom production deploy si automaticky vynucuje pripojenie cez HTTPS.

Zmenené súbory:

- `/etc/httpd/conf/httpd.conf`
- `/etc/httpd/conf.d/ssl.conf`

3.2 Jenkins pre Android aplikáciu

#1410, Zodpovedná osoba: Michal Tomlein

3.2.1 Úloha

V rámci podpory vývoja je potrebné zaviesť systém, ktorý bude automaticky:

- kompilovať Android aplikáciu po každej zmene,
- generovať dokumentáciu Javadoc,
- spúšťať a vyhodnocovať JUnit testy,
- informovať o zlyhaní tých, ktorí ho spôsobili.

3.2.2 Návrh

Na realizáciu tejto úlohy bol vybraný systém Jenkins.

3.2.3 Implementácia a testovanie

Jenkins bol nainštalovaný na server a je dostupný na adrese http://147.175.159.167:8080. Boli vytvorené dve úlohy:

- Android-ActivityTracking
- Android-Move2Play

Tieto úlohy sú nastavené tak, že priebežne kontrolujú vetvy *staging* repozitárov na gitbus.fiit.stuba.sk. Na kompilovanie, generovanie dokumentácie a spúšťanie testov sa používa Maven, ktorého nastavenie bolo predmetom inej úlohy. Jenkins je nastavený tak, aby rozposielal v prípade zlyhania e-maily osobám, ktoré mohli svojimi commitmi v repozitári problém spôsobiť.

3.3 Spustenie synchronizácie so serverom

#1409, Zodpovedná osoba: Michal Tomlein

3.3.1 Úloha

Android aplikácia obsahuje implementáciu triedy SyncAdapter, ktorá synchronizuje obsah databázy v projekte Move2PlayProvider. Synchronizácia však nie je nastavená tak, aby používala server projektu, na ktorom bol spustený production deploy Rails aplikácie. Synchronizácia tiež nebola testovaná po aktualizácii na Rails 3.1.

3.3.2 Implementácia a testovanie

Synchronizáciu sa podarilo spustiť a nastaviť na production deploy Rails aplikácie, pričom bolo potrebné:

- Opraviť kompatibilitu s Rails 3.1:
  - Už sa nepoužívajú JSON objekty vnorené pod kľúčom podľa mena modelu.
- Vytvoriť špeciálneho HTTP klienta, ktorý akceptuje root certifikát nášho servera.

HttpClient

Trieda HttpClient rozširuje triedu DefaultHttpClient tak, že pre schému `"https"` vytvára SSLSocketFactory s KeyStore, ktorý sa inicializuje načítaním zo súboru Move2PlayProvider/res/raw/keystore.bks. Tento súbor obsahuje root certifikát servera Move2Play.

Root certifikát servera Move2Play je vytvorený s URL `move2play.fiit.stuba.sk`, ktorá zatiaľ neexistuje. Aby HttpClient akceptoval aj IP adresu servera, implementovali sme triedu HttpClient.Verifier, ktorá rozširuje AbstractVerifier. HttpClient vytvára SSLSocketFactory tak, že použije triedu Verifier.

3.4 Miesta, kde sa nevykonáva meranie

#1419, Zodpovedná osoba: Peter Krátky

3.4.1 Úloha

Aplikácia by nemala vykonávať meranie v dome alebo v škole, kde je veľká pravdepodobnosť, že sa človek nehýbe. Ak by sa spustilo meranie napríklad v noci, keď si to používateľ nevšimne, môže prísť k tomu, že sa mu vybije celkom baterka. Preto je dobré ak si používateľ bude vedieť zvoliť miesta, kde sa nehýbe a teda sa nebude vykonávať žiadne meranie.

3.4.2 Návrh

Používateľ zadá, že miesto kde sa práve nachádza je miesto, kde sa nemá merať. Zobrazí sa mapa so zvýraznenou oblasťou, kde sa nemá merať, blízko jeho pozície. Túto oblasť môže upraviť. Vytvorí sa nový záznam v tabuľke, ktorá obsahuje zemepisnú šírku, dĺžku, polomer oblasti, a najbližšiu WiFi MAC adresu. Keď používateľ je mimo oblasti a vstúpi do nej a zároveň je v dosahu daná wifi, nie je potrebné vykonávať meranie. Akonáhle sa odpojí od wifi, meranie je opäť zapnuté.

3.4.3 Implementácia

Používateľ bude zrejme vnútri keď sa bude pokúšať zistiť pozíciu, preto sa pozícia zisťuje využitím GSM a Wifi – *NetworkProvider*. Zároveň sa aj preskenujú WiFi v oblasti a vyberie sa MAC adresa tej s najvyšším signálom. Po potvrdení sa uloží do databázy.

Na sledovanie toho, či sa nachádza používateľ v oblasti kde sa nemá merať slúži *RestrictedAreaManager*, ktorý je používaný v triede *TrackingManager*. Ide o stavový automat, ktorý má tri stavy – LEFT (mimo oblasti), ENTERED (vošiel do oblasti), WIFI (našla sa aj WiFi). Ak je v stave LEFT, pomocou GPS sa sleduje, či vstúpil do oblasti. Ak áno, prejde do stavu ENTERED a tam sa pokúša nájsť WiFi sieť zhodnú s tou v databáze. Ak sa po vypršaní timeout-u nenájde, prejde sa opäť do stavu LEFT. Ak sa nájde, prejde do stavu WIFI, kde bude stále sledovať, či sa neodpojil od WiFi. V prípade, že sa odpojil, prejde sa do stavu ENTERED.

3.5 Filtrovanie pohybu v dopravnom prostriedku

#1424, Zodpovedná osoba: Peter Krátky

3.5.1 Analýza

Aplikácia má zmerať len prejdenú vzdialenosť, ak človek išiel pešo. Preto treba odfiltrovať pohyb v dopravnom prostriedku. Bežná rýchlosť chôdze je o niečo viac ako $1ms^{-1}$. Sem tam človek prebehne na autobus, ale pokiaľ nevykonáva cielene aktivitu ako beh, tak môžeme predpokladať, ak je nad $3ms^{-1}$ tak asi už nepôjde o chôdzu. Pre porovnanie, najrýchlejší šprintéri bežia $10ms^{-1}$.

3.5.2 Implementácia

Na filtrovanie slúži nový filter `InVehicleFilter`. Jeho úlohou je sledovať, keď rýchlosť stúpne nad hranicu ($3ms^{-1}$). Obsahuje tiež premennú, ktorá uchováva pravdepodobnosť toho, že je vo vozidle. Ak je človek nad hranicou, pravdepodobnosť stúpa, ak pod hranicou, klesá. V našom prípade po prekročení hranice stúpa pravdepodobnosť takto: rýchlosť$/10 \cdot 2$. Pravdepodobnosť pod 0.5 považujeme za chôdzu, nad 0.5, že je v aute. *Ak sú teda prijaté 3 súradnice s rýchlosťou* $4ms^{-1}$*, dostaneme pravdepodobnosť* $0.2 + 0.2 + 0.2 = 0.6$.

Trieda `ActivityLocation` obsahuje atribút `inVehicle`, ktorý určuje, či počas získania súradnice bol používateľ vo vozidle. Ak sa objaví takáto súradnica s `inVehicle␣==␣true`, ukončí sa aktuálna aktivita a prestanú sa ukladať súradnice do databázy. Ak sa potom objaví súradnica s premennou `inVehicle␣==␣false`, začne sa nová aktivita.

3.6 Android kompabilita obrazoviek na telefónoch s rôznym rozlíšením

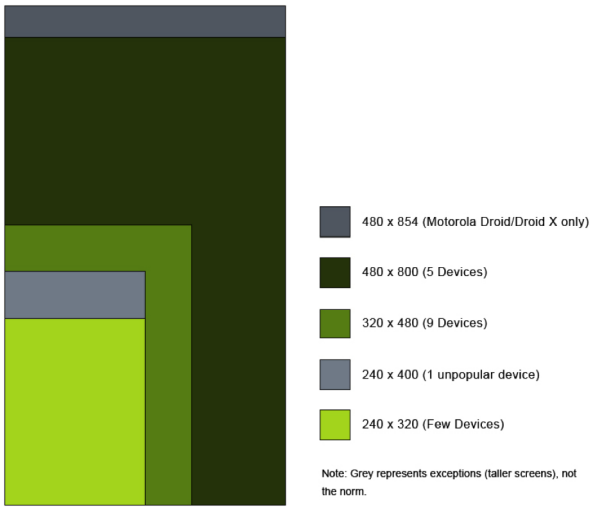
#1346, Zodpovedná osoba: Pavol Bielik

3.6.1 Analýza

Fragmentácia android zariadení prináša okrem iného dva problémy týkajúce sa dizajnu aplikácie:

- rôzne rozlíšenia a
- hustota pixelov

Hlavným problémom je nemožnosť softvérového škálovania obrázkov a iných zdrojov počas behu aplikácie kvôli rôznym pomerom strán. Rôzne konfigurácie rozlíšenia a hustoty pixelov android zariadení sú zobrazené nasledovnom obrázku[1] .



Obr. 14: Porovnanie rôznych rozlíšení a veľkostí obrazoviek

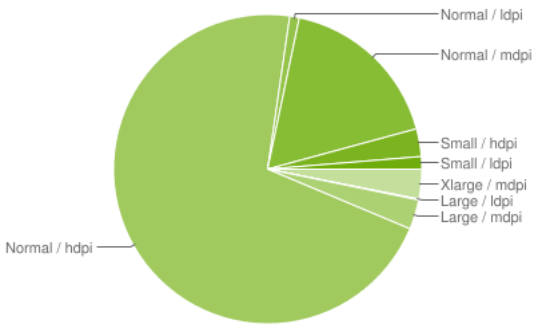
Existuje naozaj mnoho rôznych modelov. Pri rozhodovaní, ktoré formáty podporovať, je veľmi podstatný ich podiel na trhu zobrazený v nasledovnom grafe[2] . Najväčší podiel majú novšie modeli prispôsobené na širokouhlé filmy s rozlíšením 16:9 a staršie modely s pomerom strán 4:3.

3.6.2 Návrh

Rôzne rozlíšenia

Na základe rozšírenia jednotlivým model sme sa rozhodli podporovať dva pomery strán 4:3 reprezentované rozlíšením 320x480 a pomer strán 16:9 reprezentované rozlíšením 16:9. Pomer strán, pre ktorý sa bude navrhovať hlavné rozhranie je 16:9, keďže je najrozšírenejšie a obsahujú ho v podstate všetky high-end telefóny. Pre každé rozlíšenie je nutné vytvoriť vlastný layout, ktorý sa dynamický načíta na základe pomeru strán daného telefónu. Každé rozlíšenie má určený vlastný priečinok[3] v rámci priečinka res, na základe ktorého android systém vyhľadáva súbory. Okrem priečinkom

[1] zdroj: Designer's Guide to Supporting Multiple Android Device Screens
[2] zdroj: Android Developers: Screen Sizes and Densities
[3] Android Developers: Configuration Qualifiers



Obr. 15: Podiel na trhu podľa typu obrazovky

určených pre konkrétne rozlíšenie existuje aj priečinok default. Zdroje použité z tohto priečinku sú následne automaticky prispôsobené použitému rozlíšeniu, pričom sa dá zmena ich veľkosti explicitne zakázať.

Rôzna hustota pixelov

Pre manažovanie rôznych hustôt pixelov je vytvorený rovnaký spôsob ako ten uvedený vyššie. Obrázky pre každú hustotu sú uložené v vopred určených adresároch, pričom počas behu programu sa automaticky vyhľadávajú tie správne. Ak sa nenájde adresách pre danú hustotu, vyhľadáva sa najpodobnejší[4] . Pre zachovanie najlepšej kvality je teda potrebné vytvoriť viacero verzií rovnakého obrázku, pričom výber toho vhodného zabezpečí systém.

[4]Android Developers: Best Match

3.7 Automatický build a notifikácia o novej verzii android aplikácie

#1347, Zodpovedná osoba: Pavol Bielik

3.7.1 Úloha

Cieľom tejto úlohy je vytvorenie automatizovaného build procesu Android aplikácie na centrálnom mieste (servery). Build proces by mal vedieť pracovať s rôznymi verziami aplikácie ako aj číslovať jednotlivé buildy. V prípade novej verzie by sa mal poslať všetkým používateľom mail s novou verziou. Vybraný nástroj musí podporovať android projekt ako aj knižničné android projekty (apklib).

3.7.2 Návrh

Pre účely automatického buildu aplikácie sme zvolili nástroj Maven. Automaticky build sa vytvára na servery, ktorý v určených časových intervaloch kontroluje či je k dispozícii nová verzia aplikácie. Na servery sú vytvorené dva priečinky master a staging, ktoré obsahuju stabilnú verziu a verziu vo vývoji. Aplikáciu move2play sme rozdelili na tri samostatné maven moduly – libraries, activity tracking a move2play – ktoré sú spojené v jednom rodičovskom projekte. Libraries obsahuje knižničné projekty ako napríklad charts, ktoré sú využívané v ostatných projektoch. Activity Tracking a Move2Play sú dve samostatné android aplikácie, ktoré každá vygeneruje apk súbor. V nástroji maven sú vytvorené dve konfigurácie – debug a release. Pri release konfigurácii je nutné vykonať doplnujúce úpravy výsledného projektu:

- Optimalizácia kódu pomocou nástroja Proguard
- Optimalizácia nekomprimovaných súborov Android aplikácie akými sú napríklad obrázky alebo surové (raw) súbory.
- Podpísanie aplikácie certifikátom

Posielanie mailu je realizované pomocou rake task v rails aplikácii, kedy sa každému používateľovy pošle mail s novou verziou. Mail sa posiela iba za podmienky, že existuje nová verzia aplikácie a nie po každom builde.

3.7.3 Implementácia

Inštalácia a konfigurácia prostredia

Pri realizácii sme použili Maven verziu 3.0.3. Pre možnosť buildovania android projektoch sme využili android-maven-plugin verziu 3.0.0-alpha-13. Kedže maven pracuje na repozitarmi, postupne sme importovali všetky externé knižnice do lokálneho repozitára.

```
mvn install:install-file -DgroupId=com.skyhookwireless.wps
-DartifactId=skyhook -Dversion=1.0 -Dpackaging=jar
-DgeneratePom=true -Dfile=ActivityTracking/wpsapi.jar
```

Pre buildovanie android aplikácie je ďalej potrebné android sdk, pričom sme nainštalovali všetky dostupné verzie ako aj zastarané verzie.

```
android update sdk --no-ui {obsolete
```

Následne sme vytvorili emulátor, ktorý je potrebný pre spúšťanie unit testov.

```
android create avd -n 16 -t 16
```

Keďže štruktúra android projektu je rozdielna, ako štruktúra štandardného java projektu, resp. základná štruktúra s ktorou pracuje maven, je potrebné nastaviť cesty k zdrojovým súborom ako aj k testom explicitné pomocou:

```
<build>
...
     <sourceDirectory>src</sourceDirectory>
     <testSourceDirectory>test</testSourceDirectory>
...
</build>
```

Pre release verziu sme použili nasledovné pluginy pre zipalign, podpisovanie a proguard v tomto poradí.

```
<groupId>com.jayway.maven.plugins.android.generation2</groupId>
<artifactId>android-maven-plugin</artifactId>

<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-jarsigner-plugin</artifactId>

<groupId>com.pyx4me</groupId>
<artifactId>proguard-maven-plugin</artifactId>
```

Pre podpisovanie je nutné určiť lokáciu certifikačného kľúča. To je riešené vytvoreným profilu v súbore $M2_HOME/conf/setting.xml, ktorý vieme následne referencovať pomocou properties z konfiguračných súborov pre jednotlivé android projekty.

```
<profile>
  <activation>
<activeByDefault>true</activeByDefault>
  </activation>
  <properties>
<sign.keystore>~/.android/debug.keystore</sign.keystore>
<sign.alias>androiddebugkey</sign.alias>
<sign.keypass>android</sign.keypass>
<sign.storepass>android</sign.storepass>
  </properties>
</profile>
```

Pre potreby android aplikácie vie maven pracovať s dvoma typmi aplikácii - apk a apklib, pričom typ aplikácie je určený tagom

Skripty pre automatické buildovanie

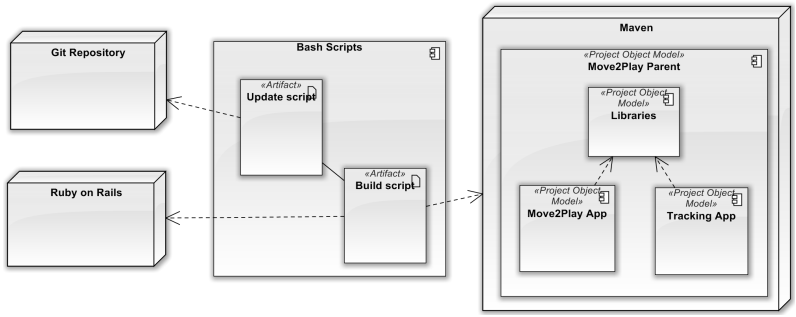
Aby sme zabezpečili automatické buildovanie, vytvorili sme skript, ktorý každých 5 minút vykonáva príkaz git pull, a v prípade že nastali zmeny, automaticky vytvorí nový build aplikácie. Tento skript okrem iného zabezpečuje zvyšovanie verzie aplikácie ako aj nastavenie čísla buildu.

Po tom čo je vytvorený nový build aplikácie sa pošle mail všetkým používateľom pomocou rails rake úlohy. Pri posielaní mailu sa využíva ActionMailer. Keďže nechceme posielať mail po každom builde, skript kontroluje verzie aktuálne buildu, a pošle mail iba v prípade, že sa jedná o novšiu verziu. Nová verzia sa dá vytvoriť dvoma spôsobmi.

1. Ako argument buildovacieho skriptu
2. Zmenením verzie projektu priamo v AndroidManifest.xml, kedy skript porovnáva hodnotu verzie z tohto súboru s poslednom vytvorenou verziou.

Proces automatického buildovania aplikácie

Diagram nasadenia je zobrazený na obr. 1. Na servery je nastavený cron job, ktorý každých 5 minút spúšťa Update script. Ten vytvorí dopyt na git repozitár a v prípade novej verziu ju stiahne a spustí Build script. Build script vykonáva kontrolu čísla verzie a buildu a spustí samotný build pomocou nástroja maven. Ak sa jedná o novú verziu, zavolá sa rake task ktorý posiela maily všetkým zaregistrovaným používateľom.



Obr. 16: Diagram nasadenia automatického buildu android aplikácie

3.7.4 Testovanie

Maven umožnuje automatické testovanie android projektov. Testovací projekt je typu apk, pričom musí byť definovaná závislosť na testovanej aplikácii jednak jej priamim refenrencovanim pomocou apk, ale aj pomocou jar. Pre testovanie je taktiez potrebné zadefinovať spustenie emulátora a referencovanie junit knižnice. Emulátor musí byť vytvorení pomocou konfigurácie

```
<plugin>
<groupId>com.jayway.maven.plugins.android.generation2</groupId>
   <artifactId>android-maven-plugin</artifactId>
   <version>3.0.0-alpha-13</version>
   <configuration>
      <sdk>
         <platform>9</platform>
      </sdk>
      <emulator>
         <avd>9</avd>
         <wait>10000</wait>
         <options>-no-window -noskin</options>
```

```
      </emulator>
      <undeployBeforeDeploy>true</undeployBeforeDeploy>
  </configuration>
</plugin>
```

3.8 Úvodná obrazovka

#1389, Zodpovedná osoba: Štefan Mitrík

3.8.1 Úloha

Navrhnúť a implementovať hlavnú obrazovku aplikácie. Identifikovať podstatné informácie, ktoré chceme používateľovi zobrazovať a zobraziť ich na hlavnej obrazovke. Medzi nimi by nemal určite chýbať prehľad o aktuálnom stave plnenia denného plánu.

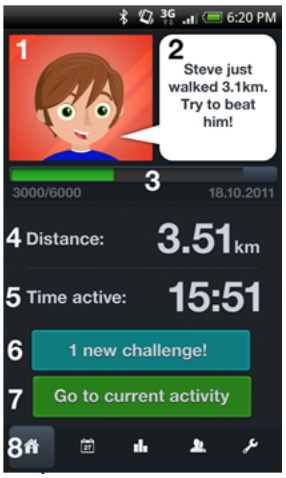
3.8.2 Analýza

Existuje viacero mobilných aplikácií ktoré sa venujú meraniu aktivity používateľa. Pri návrhu nášho používateľského rozhrania sme vychádzali z existujúcich aplikácií a snažili sme sa identifikovať ich dobré súčasti.

Častou chybou rozhraní mobilných aplikácií je snaha zobraziť čo najviac informácií, čím sa stáva rozhranie menej prehľadné a v konečnom dôsledku menej použiteľné. Druhou častou chybou je zložitá navigácia, kvôli ktorej používateľ často krát nevie kde sa práve nachádza.

3.8.3 Návrh

Na obrázku je zobrazený návrh používateľského rozhrania pre úvodnú obrazovku.



Obr. 17: Návrh úvodná obrazovka

Návrh sa skladá z nasledujúcich komponentov:

1. Avatar – postavička ktorá reprezentuje používateľa v našej aplikácii a môže si ju naštýlovať podľa svojich preferencií
2. News Feed – aktuality v rámci aplikácie, príkladom takýchto aktualít môže byť aktivita vykonaná priateľom prípade novinka alebo výzva
3. Daily plan progress – zobrazenie aktuálneho stav plnenia denného plánu v aplikácii, vyjadrené číselne aj graficky prostredníctvom progress baru
4. Vzdialenosť ktorú prekonal používateľ v rámci dňa

5. Čas ktorý bol používateľ aktívny v rámci dňa
6. Tlačidlo na prejdenie na obrazovku s výzvami
7. Tlačidlo na zobrazenie aktuálne vykonávanej aktivity, príp. informácia o tom že sa používateľ nehýbe
8. Menu v rámci aplikácie

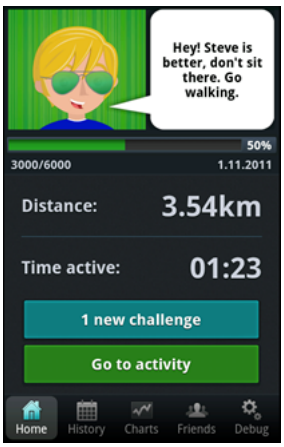
3.8.4 Implementácia

Android ponúka vytváranie grafiky komponentov dvomi spôsobmi:

- Obrázkami
- XML štýlovaním

Zatiaľ čo XML štýlovanie je zložitejšie a nedajú sa ním vytvoriť niektoré grafické prvky, prináša výhody v tom, že nie je závislé na rozlíšení obrazovky a jeho veľkosť je v porovnaní s obrázkami minimálna. Preto sme vytvorili väčšinu grafických komponentov práve týmto spôsobom a obrázky sme použili len na zložitejšie farebné prechody s tieňmi aké majú napr. tlačidlá.

Na nasledujúcom obrázku je zobrazeá implementácia úvodnej obrazovky:



Obr. 18: Návrh úvodná obrazovka

4 Štrba

- Vyžadovať potvrdenie na zapnutie Wi-Fi — Michal Tomlein
- Sprístupňovanie dát z ActivityTracking do Move2Play — Michal Tomlein
- Vyhodnotenie anotovania pohybu — Pavol Bielik
- Navrhúť obrazovky pre aktuálnu aktivitu — Štefan Mitrík
- Prepínateľné zobrazenie odfiltrovaných súradníc — Peter Krátky

4.1 Vyžadovať potvrdenie na zapnutie Wi-Fi

#1715, Zodpovedná osoba: Michal Tomlein

4.1.1 Úloha

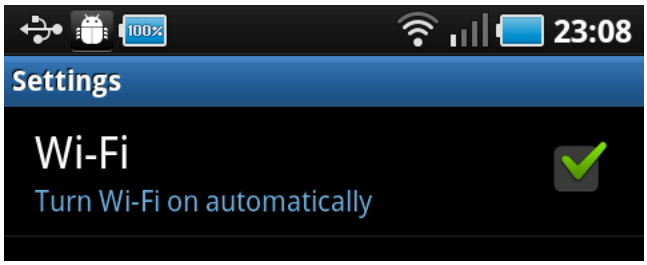
ActivityTracking zapína Wi-Fi automaticky, keď to potrebuje. Deje sa to však bez súhlasu používateľa a spôsobuje to problém pri zapínaní Wi-Fi hotspotu, kedy Android potrebuje vypnúť Wi-Fi. Hneď po vypnutí Wi-Fi ho ActivityTracking znovu zapne, vďaka čomu zapnutie hotspotu zlyhá. Je potrebné riešenie, ktoré bude požadovať od používateľa súhlas pred prvým zapnutím Wi-Fi a umožní automatické zapínanie Wi-Fi neskôr vypnúť.

4.1.2 Návrh

- Ak Wi-Fi nie je zapnuté a ešte nebolo povolené/zakazané, mala by sa poslať notifikácia.
- Ak Wi-Fi nie je zapnuté a bolo povolené, malo by sa spustiť, pričom by sa mala vytvoriť notifikácia o zapnutí Wi-Fi.
- Ak Wi-Fi nie je zapnuté a bolo zakázané, nemalo by sa stať nič (používateľ dal explicitne najavo, že si Wi-Fi neželá, takže rešpektujeme jeho rozhodnutie).
- Ak používateľ Wi-Fi vypne, malo by sa zapnúť alebo nezapnúť podľa predchádzajúcich pravidiel.

4.1.3 Implementácia a testovanie

Bola vytvorená obrazovka s nastaveniami v aplikácii ActivityTracking, do ktorej je možné sa dostať voľbou *Preferences* v menu. Ak ešte nebolo povolené/zakázané Wi-Fi a ActivityTracking ho potrebuje zapnúť, pošle sa notifikácia, ktorá to oznamuje. Na túto notifikáciu je možné kliknúť, čím sa otvorí obrazovka s nastaveniami.



Obr. 19: Obrazovka nastavení v aplikácii ActivityTracking

4.2 Sprístupňovanie dát z ActivityTracking do Move2Play

#1711, Zodpovedná osoba: Michal Tomlein

4.2.1 Úloha

Aplikácie ActivityTracking a Move2Play obsahujú vlastné databázy, ktoré majú niektoré tabuľky rovnaké, pričom ich obsah nie je zosynchronizovaný. V ideálnom prípade by projekt ActivityTracking nemusel mať databázu vôbec, pričom uchovávanie nameraných údajov by sa dialo len v Move2Play. ActivityTracking však potrebuje pre svoju činnosť prístup k starším záznamom, ktoré v niektorých prípadoch aj mení. Je preto potrebné zabezpečiť, aby sa obsah databázy ActivityTracking dostal aj do Move2Play a naopak, t.j. nejakú formu synchronizácie.

4.2.2 Návrh

Do úvahy prichádzajú dve riešenia:

1. ActivityTracking bude poskytovať listener rozhranie, ktoré bude Move2Play používať na to, aby sa dostalo k nameraným údajom.
   - Toto rozhranie by muselo notifikovať o pridaných ale aj zmenených záznamoch, keďže ActivityTracking sa niekedy vracia k starším záznamom a upravuje ich. To by si vyžadovalo použitie spoločných unikátnych identifikátorov.
   - Nevýhodami tohto riešenia sú duplikácia údajov, pridanie zbytočnej réžie, ktorá sa môže prejaviť na skrátení životnosti batérie, a pridanie možného bodu zlyhania.
2. Zrušia sa duplicitné tabuľky, t.j. každá tabuľka sa bude nachádzať buď len v ActivityTracking alebo len v Move2Play, čím sa odstráni nutnosť vytvoriť listener rozhranie a zabezpečiť tak synchronizáciu.
   - Nevýhodou tohto riešenia je nutnosť pridať do tabuliek v ActivityTracking stĺpce potrebné na synchronizáciu (`sync_id` , `sync_state`).

Po dohode tímu bolo rozhodnuté realizovať druhú možnosť.

4.2.3 Implementácia a testovanie

Z projektu Move2Play boli odstránené tabuľky:

- `reports`
- `gps_logs`
- `activity_types`

Modely `Report` , `GpsLog` a `ActivityType` boli prepísané tak, aby používali tabuľky z databázy ActivityTracking.

Tabuľky v projekte ActivityTracking boli prispôsobené na synchronizáciu pridaním stĺpcov `sync_id` a `sync_state` a vytvorením potrebných triggerov.

4.3 Vyhodnotenie anotovania pohybu

#1696, #1695, Zodpovedná osoba: Pavol Bielik

4.3.1 Analýza

Pre vyhodnotenie a zlepšovanie merania pomocou Wi-Fi je potrebné získať veľké množstvo meraní. Tieto merania musia byť anotované a musia byť zozbierané z rôznych zariadení, ktoré používajú jednak vývojári ale aj používatelia. Je preto nutné vytvoriť podporný systém, ktorý bude umožňovať anotovanie pohybu, jeho poslanie na server a následné spracovanie a analýzu na servery. Pri samotnej implementácii je vhodné využiť už existujúce časti, akými sú napríklad exportovanie záznamov do csv súboru a poslanie mailom.

4.3.2 Návrh

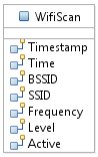
Proces anotovanie a vyhodnotenia merania bude pozostávať s nasledovných krokov:

1. **Anotovanie** údajov na zariadení.
2. **Upload na Server** anotovaných údajov spolu s ich stručným popisom.
3. **Spracovanie** údajov na servery
4. **Vyhodnotenie** spracovaných údajov. Bude prebiehať ručne.

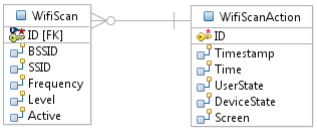
Anotovanie

Pri anotovaní údajov sa využije existujúci spôsob, ktorý sa rozšíri o ďalšie typy pohyby. Tu je dôležité aby mal používateľ možnosť zmeniť anotáciu aj dodatočne, keďže je veľmi pravdepodobné, že zabudne zmeniť stav pri meraní.

Pre ucely testovania a vyhodnotenia je potrebna zmena a rozsirenie datoveho modelu, ktory zachytava wifi scany.



Obr. 20: Pôvodný dátový model



Obr. 21: Pridanie tabuľky WifiScanAction a doplnujúcih atribútov

Upload

Pri uploade existujú dve možné riešenia a to využitie synchronizácie tabuliek alebo posielanie záznamov mailom. Posielanie mailom sa ukázalo ako nevhodné, keďže bolo potrebné pripojenie k internetu v dobre vytvárania záznamov ako aj následné ručne spracovanie a prekopírovanie súborov. Ako vhodnejší spôsob bude teda vo finálnej verzii implementovaná synchronizácia tabuliek.

Spracovanie

Spracovanie pozostáva z dvoch častí. Je potrebné jednak rozšíriť existujúci testovací modul, tak aby sme vedeli testovať WifiWatcher ako celok (metódou black box). Tieto dáta je následne potrebne exportovať do súboru tak, aby sme s nimy vedeli pracovať v rails aplikácii. Druhou častou je samotné spustenie testov na servery. Je teda potrebne nakopirovať uploadnuté záznamy do testovacieho programu, spraviť build, uploadnut na emulator, pustit testy, vytiahnut z emulatora zaznam o priebehu testov a načítať ho v rails aplikacii. Tento proces bude pozostávať jednak s vytvorenie bash skriptov ako aj konfigurácie mavenu pre prácu s emulatorom a android projektom.

Vyhodnotenie

V poslednej casti je potrebne vyhodnotit získané údaje. Tu je vhodne ich vizualizovať. Pre tento účel využijeme knižnicu HighCharts, ktorá je volne dostupná. Vizualizácia bude formou čiarového grafu, kde os x bude čas a os y bude zobrazovať namerané hodnoty jednotlivých listenerov, prípadné iné údaje.

4.3.3 Implementácia a testovanie

Upload

Pre účely uploadu sme doplnili a vyskúšali existujúcu synchronizáciu tabuliek so serverom. Synchronizácia ale prebiehala značne pomaly (3s na záznam). Keďže sme chceli synchronizovať radovo tisice záznamov denne takéto riešenie nebolo možné nasadiť. Rozhodli sme sa teda použiť export databázy do csv súboru ktorý následne uploadujeme.

Pre účely synchronizácie je v androide vytvorená tabuľka dump_sync, ktorá obsahuje časové intervaly spolu s popisom záznamu. Pri synchronizácii sa pre každý časový intervali vytvorý csv súbor. Synchronizácia je spúštaná automaticky integráciou do sync_adaptera v projekte move2play alebo ručne cez menu.

Na servery sme pre uplad súborov použili gem carrierwave. Z androidu sa posiela multipart POST request. Na jeho vyskadanie sme použili kniznicu apache httpmime-4.1.2. Tu sa bohužial podarilo použít iba čiastočne, kedže jej formátovanie requestu nie je kompatibilné s rack gemom, ktorý prichádzajúci request parsuje na strane servera. Problémom bolo znak zakončenia riadku.

Spracovanie

Spracovanie uploadovaných súborov prebieha na servery. Pre spracovanie súborov sme využili gem delayed_job. Pre účely spracovania bolo potrebné vytvoriť dátový model, ktorý obsahuje jednak tabulky z android ako aj tabulky pre spracované údaje.

Spracovania začína načítaným údajov z csv súboru do databázy. Kedže importujeme značné množstvo záznamov, pre zrýchlenie importu sme použili gem activerecord_import. Po importe do databázy sa prekopírujú csv súboru do priečinka assets v ActivityTrackingTest projekte a spustí sa testovací skript.

Unit Testy

V unit testoch sa testujú jednotlivé listenery ako aj samotný wifiwatcher. Je vytvorený MockTelephonyManagerProviderStreaming, ktorý streamovane číta dáta z csv súborov a poskytuje ich na spracovanie. Pri testovaní sa všetky výstupné hodnoty listenerov, wifiwatchera ako aj stav pri každom Wi-Fi scane zapisuje do csv súborov. Tie sú po skončený testov skopírované s SD karty simulátora a importované do rails aplikácie pre analýzu.

Na spustenie testov bolo potrebné upravenie konfigurácie maven-u. Bolo potrebné automaticky zapnúť android emulátor, pričom sme potrebovali spustiť iba vybrané testy (testy su už aj tak pomalé, takže nechceme spúštať všetky testy), bez toho aby sme museli vytvoriť nový testovací android projekt. Vybrané testy vieme spustiť nasledovným spôsobom:

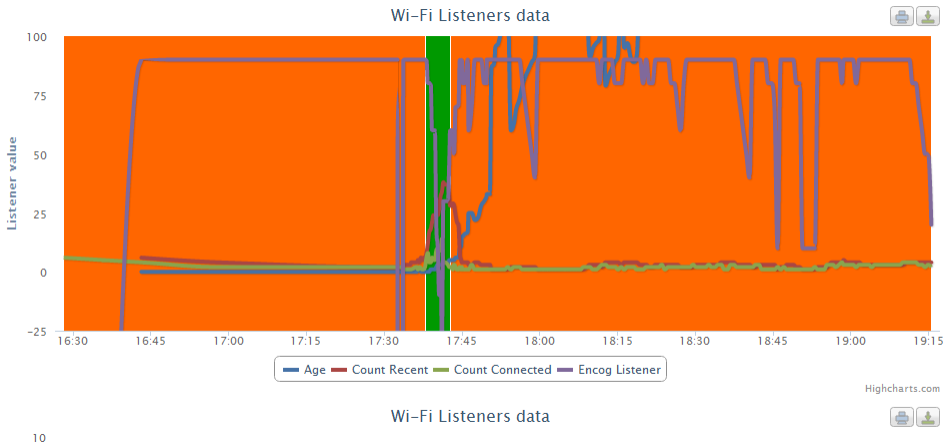
```
<code>
<plugin>
<groupId>com.jayway.maven.plugins.android.generation2</groupId>
    <artifactId>android-maven-plugin</artifactId>
    <configuration>
       <test>
         <skip>false</skip>
         <!--<classes>
         <class>sk.stuba.fiit.move2play.test.wifi.WifiWatcherTest</class>
         </classes> -->
         <packages>
         <package>sk.stuba.fiit.move2play.test.wifi</package>
         </packages>
       </test>
    </configuration>
  <extensions>true</extensions>
</plugin>
</code>
```

Problém bol ale v konfigurácii maven-u, ktorý vie štandardne pracovať len s JUnit testami a nie androidom. Aby sme tento problém vyriešili bolo potrebné nasledovné zmeny v konfigurácii:

- testy musia byť v adresári *sourceDirectory* nie v *testSourceDirectory*
- pri referencovaní projektov je potrebné nastaviť
- pre plugin org.apache.maven.plugins treba dať tag *jar-nofork* namiesto *testjar-nofork*

4.3.4 Vyhodnotenie

Pri vyhodnotený sa csv súbory vytvorené spustením testov importujú do databázy a údaje sa vykreslia v grafe.



Obr. 22: Vizualizácia analyzovaných wifi údajov

4.4 Navrhúť obrazovky pre aktuálnu aktivitu

#1705, Zodpovedná osoba: Štefan Mitrík

4.4.1 Úloha

Navrhnúť obrazovky pre aktuálne aktívnu aktivitu. Aktuálne aktívna aktivita ja aktivita, ktorú používateľ práve vykonáva.

4.4.2 Analýza

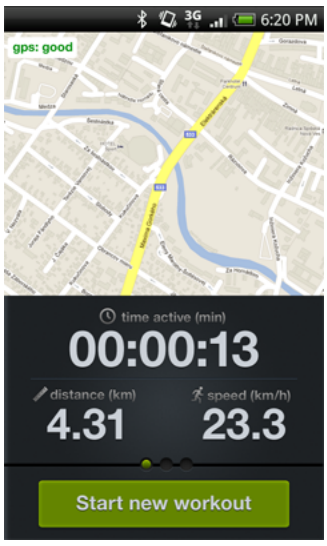
Analyzovaním existujúcich aplikácií som dospel k záveru, že používatelia sú zvyknutý minimálne na tieto informácie:

- Mapa
- Údaje o rýchlosti a vzdialenosti
- Údaje o nadmorskej výške
- Údaje o spálených kalóriách

Okrem toho je vhodné vizualizovať nadmorskú výšku a rýchlosť v grafe.

4.4.3 Návrh

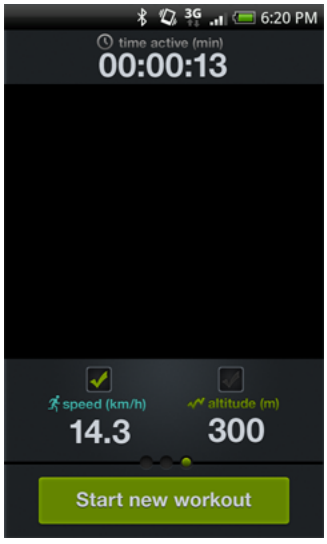
Na základe analýzy som pre aktívnu aktivitu navrhol nasledovné obrazovky:
Prvá obrazovka s mapou a základnými údajmi:



Obr. 23: Úvodná obrazovka aktívnej aktivity

Druhá obrazovka s detailnými informáciami o aktivite:
Tretia obrazovka s grafom rýchlosti a nadmorskej výšky:

Obr. 24: Druhá obrazovka aktívnej aktivity



Obr. 25: Tretia obrazovka aktívnej aktivity

4.5 Prepínateľné zobrazenie odfiltrovaných súradníc

#2087, Zodpovedná osoba: Peter Krátky

4.5.1 Úloha

V zozname vykonaných aktivít by malo byť umožnené pozrieť si, ktoré aktivity boli odfiltrované v dôsledku jazdy v dopravnom prostriedku. Taktiež by malo byť zobrazené priamo na mape, ktoré súradnice už boli odfiltrované.

4.5.2 Návrh

Pre testovacie účely zoznam vykonaných aktivít zobrazuje pre každú položku celkovú dĺžku aktivity, a okrem toho aj koľko z nej bola chôdza. Aktivity, ktoré boli celé zmazané nie sú zobrazené v zozname, ale dajú sa zobraziť. Tie sú odlíšené farebne.

Na mape sú zobrazené súradnice namerané pri chôdzi. Je možné zobraziť aj odfiltrované (pri pohybe vo vozidle). Okrem toho, pre názornejšie zobrazenie sú na mape farebne odlíšené úseky s rôznymi rýchlosťami. Sú použité 3 farby:

- modrá - rýchlosť do $1ms^{-1}$
- zelená - rýchlosť $1ms^{-1}$ až $3ms^{-1}$
- oranžová - rýchlosť nad $3ms^{-1}$

4.5.3 Implementácia

Aby bolo možné zobrazovanie odfiltrovaných súradníc, nie sú zmazané, ale označené flagom v databáze ako “hidden”.

Prepínateľné zobrazenie aktivít alebo súradníc v zozname je umožnené tlačidlom v menu.

5 Liptovský Mikuláš

- Aplikovanie metód umelej inteligencie pri Wi-Fi meraní — Pavol Bielik
- Ladenie Wi-Fi merania — Pavol Bielik
- Pokročilé spracovanie GPS súradníc (Kalmanov filter) — Peter Krátky
- Pokročilá synchronizácia — Michal Tomlein
- Implementácia prvej obrazovky pre aktívnu aktivitu — Štefan Mitrík

5.1 Aplikovanie metód umelej inteligencie pri Wi-Fi meraní

#2153, Zodpovedná osoba: Pavol Bielik

5.1.1 Úloha

Cieľom tejto úlohy je natrénovať neurónovú sieť tak, aby bola schopná klasifikovať či sa človek hýbe alebo je v pokoji. Natrénovanú neurónovú sieť je potrebné následne integrovať do ActivityTracking modulu a overiť na dostupnej dátovej vzorke.

5.1.2 Analýza

Pôvodným zámerom bolo natrénovať najskôr neurónovú sieť na fluktuáciu Wi-Fi signálu (vychádzame s metódy pre GSM fluktuáciu) a následné aj pomocou iných vstupných údajov. Na základe analýzy nameraných dát sme ale dospeli k záveru, že Wi-Fi fluktuácia nie je vhodný parameter vzhľadom na relatívne krátky dosah Wi-Fi (oproti GSM). Ďalej sa teda týmto spôsobom nebudeme zaoberať.

Pri trénovaní je potrebné určiť dve základné veci, a to vstupné dáta a model trénovania.

Dostupné dáta

Pre trénovanie neurónovej siete máme dostupne nasledovné dáta, ktoré získavame s ActivityTracking modulu periodicky každých 10 až 40 sekúnd.

- timestamp merania
- aktuálne dostupné veže - vek, sila signálu, frekvencia, kanál, ssid, bssid
- nedávno dostupné veže - vek, timeout, priemerná sila signálu
- aktuálne pripojená veža (ak taká existuje)
- stav zariadenia
  - zapnutá obrazovka
  - výrobca

Z týchto údajov sme určili nasledovné črty, ktoré vieme použiť pri trénovaní:

- *cur_count* počet aktuálne pripojených veží
- *rec_count* počet nedávno pripojených veží
- *new_count* počet nových veží
- *active* 1 ak je nejaká veža aktívna, inak 0
- *max_age* vek najstaršej veže
- *avg_age* priemerný vek nedávno pripojených veží
- *min_timeout* najmenší timeout
- *avg_timeout* priemerný timeout nedávno pripojených veží

- *delta_age* súčet zmeny vekov za posledných 5 meraní
- *delta_cell* absolútna hodnota rozdielu *cur_count* a *rec_count*
- *delta_new_cell* maximálna hodnota *new_count* za posledných 5 meraní

Ďalej vieme využiť už predspracované hodnoty, ktoré nám poskytujú listenery *contex*, *place* a *change*.

Attributes	context	place	cur_count	rec_count	avg_age	avg_timeout	new_cell_c...	delta_age	delta_cell	delta_new_...	min_age	max_age	min_timeout	max_timeout
context	1	0.432	0.185	0.320	-0.010	0.432	0.186	-0.101	0.302	-0.020	-0.005	0.009	0.146	0.701
place	0.432	1	0.398	0.243	0.094	0.531	0.145	-0.008	0.153	0.070	0.029	0.110	0.191	0.736
cur_count	0.185	0.398	1	0.413	-0.083	0.245	0.400	0.004	0.191	0.031	-0.193	0.016	-0.097	0.314
rec_count	0.320	0.243	0.413	1	0.266	0.206	0.378	0.280	0.970	0.137	0.106	0.176	0.424	0.179
avg_age	-0.010	0.094	-0.083	-0.266	1	0.343	-0.133	0.095	-0.267	0.055	0.580	0.881	0.333	0.205
avg_timeout	0.432	0.531	0.245	-0.205	0.343	1	-0.109	0.167	-0.294	0.043	0.243	0.277	0.769	0.772
new_cell_cou	0.186	0.145	0.400	0.378	-0.133	-0.109	1	-0.186	0.335	-0.047	-0.077	-0.106	-0.172	0.010
delta_age	-0.101	-0.008	0.004	-0.280	0.095	0.167	-0.186	1	-0.305	-0.051	0.042	0.075	0.181	0.016
delta_cell	0.302	0.153	0.191	0.970	-0.267	-0.294	0.335	-0.305	1	0.115	-0.160	-0.197	-0.435	0.099
delta_new_ce	-0.020	0.070	0.031	0.137	0.055	0.043	-0.047	-0.051	0.115	1	0.016	0.069	-0.069	0.144
min_age	-0.005	0.029	-0.193	-0.106	0.580	0.243	-0.077	0.042	-0.160	0.016	1	0.252	0.364	0.084
max_age	0.009	0.110	0.016	-0.175	0.881	0.277	-0.106	0.075	-0.197	0.069	0.252	1	0.185	0.215
min_timeout	0.146	0.191	-0.097	-0.424	0.333	0.769	-0.172	0.181	-0.435	-0.069	0.364	0.185	1	0.346
max_timeout	0.701	0.736	0.314	0.179	0.205	0.772	0.010	0.016	0.099	0.144	0.084	0.215	0.346	1

Obr. 26: tabuľka korelácie atribútov

Model trénovania

Najjednoduchšia metóda je klasická dopredná neurónová sieť so spätným šírením signálu. Vzhľadom na povahu vstupných dát, kde je zrejmá časová závislosť medzi jednotlivými meraniami, sú vhodnejšie modely rekurentná prípade time-delayed neurónová sieť.

5.1.3 Návrh

Vhodný výber a predspracovanie vstupov predstavuje ťažisko tejto úlohy. V tejto časti opíšeme metódy predspracovania dát, ktoré môžeme využiť pred samotným trénovaním neurónovej siete.

Žiadne

Teda dáta získané z ActivityTracking modulu idú priamo na vstup neurónovej sieti. Tejto prístup je pri väčšine prípadov nevhodný.

Mean Normalization

Normalizácia hodnôt do intervalu <-1,1> na základe vzorca (1):

$$\frac{x_i - \mu}{max(x) - min(x)} \tag{1}$$

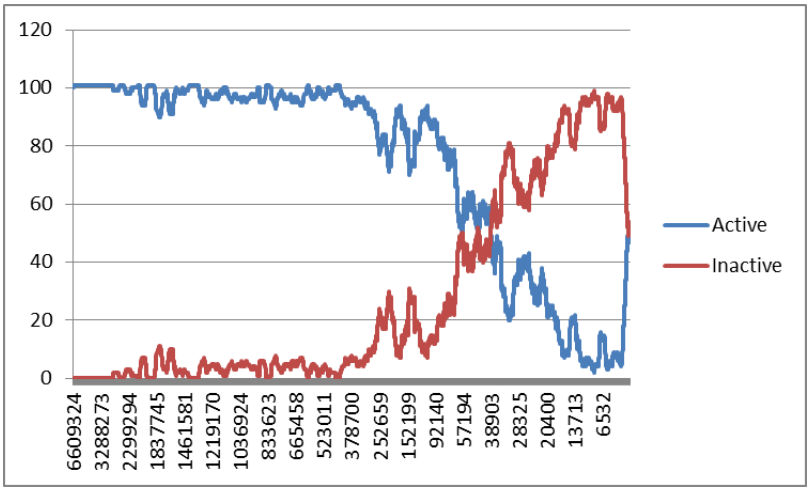
Aj keď sú hodnoty normalizované, pre neurónovú sieť môže byť problém veľký počet hodnôt, ktoré vstupy nadobúdajú.

Stanovenie tried (košov)

Vstupné dáta sa na základe stanovených rozsahov rozdelia do tried resp. košov, pričom v rámci jedného koša sú všetky hodnoty považované za rovnocenné.

Rozsahy vieme určiť manuálne alebo pomocou klastrovania. V našom prípade sme rozsahy určili ručne keďže vieme lepšie využiť znalosť problémovej domény ako aj využiť anotáciu dát. Pri ručnom určovaní postupujeme nasledovne

1. zotriedime vstupné hodnoty črty, ktorej chceme stanoviť triedy.
2. pre intervaly (n=100) si vypočítame počet vstupných hodnôt anotovaných stavom aktívny a počet hodnôt neaktívny.
3. vykreslíme graf vypočítaných hodnôt na základe ktorého určíme vhodné triedy (príklad grafu pre *avg_age* je na obrázku).



Obr. 27: Analýza črty priemerného veku veže

V rámci triedy následne ešte existujú dva spôsoby akým poskytnúť hodnotu na vstup neurónovej siete.

- Ako jeden neurón s viacerými možnými hodnotami aktivity
- Ako viacero neurónov, kedy vždy iba jeden je aktívny

5.1.4 Implementácia a testovanie

Implementačné prostredie

Pri trénovaní sme využili nasledovné knižnice *Encog*, *FANN*, *Neuroph*, *NeuroSolutions*, *RapidMiner*. V Android projekte využívame knižnicu Encog, ktorá nemá žiadne externé závislosti a poskytuje všetku potrebnú funkcionalitu.

Trénovanie neurónovej siete

1. Verzia

Vstupy avg_age, avg_timeout, [active_i, age_i, timeout_i]^10

Predspracovanie Mean Normalization

Popis Trénovacie vstupy pozostávajú z všeobecnej charakteristiky veží, ale najmä s popisu jednotlivých veží. Problémom boli trénovacie vstupy, ktoré mali menej veží ako bol ich stanovený počet (10). V takom prípade sa vstupy nahradili hodnotami práve pripojených veží. Takýto prístup sa ukázal ako nevhodný.

2. Verzia

Vstupy recent_count, current_count, active, [active_i, age_i, timeout_i]^10

Predspracovanie Mean Normalization

Popis Podobné ako prvá metóda ale použili sme rekurentnú sieť.

3. Verzia

Vstupy delta_rec_count_i, delta_cur_count_i, context_i, delta_avg_age_i, delta_avg_timeout_i^5

Predspracovanie Stanovenie tried - 2. triedy

Popis Delty v tomto prípade predstavujú rozdiel medzi dvoma nasledovnými meraniami. Keďže sa v predchádzajúcich pokusoch často stávalo, že sme museli dopĺňať vstup prázdnymi hodnotami, znížili sme počet veží. Stanovenie tried v tomto prípade až moc zúžilo priestor vstupov a tak sme nedosiahli použiteľné výsledky.

4. Verzia

Vstupy age, active, [delta_rec_count_i, delta_cur_count_i, delta_avg_age_i, delta_avg_timeout_i]^5

Predspracovanie Stanovenie tried - 2. triedy

Popis Veľmi podobné verzii 3 s menšou obmenou vstupov.

5. Verzia

Vstupy active, context, place, delta_age, delta_cell, delta_new_cell

Predspracovanie Žiadne

Popis Vyskúšanie trénovania na nových črtách.

6. Verzia

Vstupy context, place, delta_cell, delta_age, delta_new_cell, avg_age

Predspracovanie Stanovenie tried - aktívne neuróny

Popis Vylepšenie 5 verzie s miernou zmenou čŕt. Vylepšenie spočívalo hlavne v predspracovaní údajov. V tejto verzii sa nám konečne podarilo natrénovať neurónovú sieť tak, aby dávala stabilné výsledky (znamená to, že sme nemali veliký rozptyl hodnôt v krátkych časových úsekoch).

5.1.5 Vyhodnotenie

Pri trénovaní sme použili vzorku 64 meraní, ktoré spolu obsahujú 17328 záznamov. Tie sme ešte dodatočne vyfiltrovali na počet 13530. Pri trénovaní sme vybrali náhodným spôsobom 60% z celkovej vzorky a po 20% tvorili testovacia a cross-validation množina.

Vzhľadom na charakter dát sme mali často problémy s ich konzistentnosťou, keďže viaceré stavy boli často krát anotované rôzne. Taktiež dátová vzorka obsahovala anotovaný neaktívny stav približne 4-krát viac ako stav aktívny. Pri trénovaní potom prevážila početnosť dát v prípadoch, keď boli vzorky nekonzistentné.

Neurónovú sieť sa nám podarilo nakoniec celkom dobré natrénovať. Myslím si, že lepšou analýzou a výberom vstupných dát sme schopný dosiahnuť ešte lepšie výsledky. Treba si však uvedomiť, že dáta samotné obsahujú určité chyby ako aj spomenutú ne-konzistentnosť čo znamená, že 100% úspešnosť sa nám dosiahnuť určite nepodarí.

	-2	-1	0	1	2	-20	#
UNKNOWN	0.00%	0.00%	2.04%	93.88%	0.00%	4.08%	49
STATIONARY_INDOORS	0.31%	0.22%	4.60%	94.47%	0.00%	0.39%	9665
STATIONARY_OUTDOORS	0.00%	0.62%	2.87%	95.89%	0.00%	0.62%	487
WALKING_INDOORS	0.14%	0.97%	3.87%	94.20%	0.00%	0.83%	724
WALKING_OUTDOORS	3.42%	3.05%	8.33%	84.15%	0.00%	1.06%	1609
DRIVING	5.52%	5.02%	6.22%	81.22%	0.00%	2.01%	996
							13530

5.2 Ladenie Wi-Fi merania

#2154, Zodpovedná osoba: Pavol Bielik

5.2.1 Analýza

Táto úloha pozostáva s nasledujúcich častí:

- spracovanie tréningovej sady anotovaných údajov Wi-Fi merania
- vyhodnotenie úspešnosti existujúcich listenerov nad zozbieranými dátami
- vylepšenie listenerov na základe analýzy ich nedostatkov
- overenie výsledného návrhu listenerov

Prvé dve úlohy budú realizované v ruby na servery, druhé dve budú implementované v android aplikácii ActivityTracking. Pri spracovaní údajov a vyhodnotení úspešnosti budeme vychádzať z výsledkov úloh (#1695) a (#1696).

5.2.2 Návrh

Spracovanie tréningovej sady je potrebné vytvoriť ručne na základe anotovaných logov z android zariadení. Pri spracovaný sa zameriavame najmä na identifikovanie zlých logov, ktoré sú:

- zle anotované
- nekonzistentné
- nastala chyba pri posielaní logov zo zariadenia
- odstránenie duplicity
  - priamej tj. presne rovnaké logy
  - nepriama tj. veľa dumpov z rovnakej lokácie

Okrem manuálneho vyhodnotenia (), je potrebné vytvoriť vyhodnotenie automatické. Vyhodnotenie spočíva vo vypočítaní úspešnosti pre každý listener v každom dostupnom stave zobrazené v nasledovnej tabuľke.

	Stationary	Walking
HIGH_ACTIVE	55	4
LOW_ACTIVE	34	8
NEUTRAL	13	10
LOW_INACTIVE	12	24
HIGH_INACTIVE	3	44
UNKNOWN	3	24

Na základe týchto údajov bude možné následne porovnávať a vyhodnotiť zmeny v listenerov a ich účinok na celkovú úspešnosť.

5.2.3 Implementácia a testovanie

Pri spracovaní testovacích údajov sme k 7.12.2011 získali 78 jedinečných logoch, z ktorých sme ako platné vyhodnotili a zaradili do testovacej množiny 60.

Pri vyhodnotení úspešnosti sme vytvorili export do súboru csv, ktorý je následne importovaný do excel-u, kde sú dáta prehľadne vizualizované. (obrázok).

Vylepšenie listenerov

Context Listener Na základe analýzy údajov sme implementovali nasledovné vylepšenia:

- kontext sa počíta s aktuálne pripojených veží ale aj s nedávno pripojených veží.
- 2 minútové okno po zmene kontextu.

Place Listener Pre Place Listener sme identifikovali nasledovné problémy:

- učenie je príliš rýchle
- nezabúdanie lokácii spôsobuje problémy
- vyhodnocovanie všetkých aktuálne pripojených veží spôsobuje, ze výsledná hodnota sa drží v strede povoleného rozsahu

na základe identifikovaných problémov sme implementovali nasledovné vylepšenia:

- Place listener nepristupuje k databáze ale k lokálnej pamäti, kde si uchováva nedávne Wi-Fi veže.
- Z pripojených veží sa vyberá tá s najväčším vekom.

Testovanie pozostávalo zo overenia účinku zmien porovnaním dosiahnutých výsledkov ako aj analýzov hraničných prípadov. Za účelom analýzy zle klasifikovaných dát sme využili query, ktorá nám vráti počet zle klasifikovaných záznamov pre každý log.

```
SELECT wifi_scan_actions.tracking_dump_id, count(*) FROM tracking_dump_reports inner join wifi
```

Dosiahnuté výsledky pre Context listener a Place listener sú uvedené v nasledovných tabuľkách.

Context Listener

	-2	-1	0	1	2	-20	#
UNKNOWN	44.90%	0.00%	0.00%	0.00%	55.10%	0.00%	49
STATIONARY_INDOORS	88.68%	0.00%	0.00%	0.00%	11.32%	0.00%	9665
STATIONARY_OUTDOORS	33.47%	0.00%	0.00%	0.00%	66.53%	0.00%	487
WALKING_INDOORS	51.66%	0.00%	0.00%	0.00%	48.34%	0.00%	724
WALKING_OUTDOORS	16.28%	0.00%	0.00%	0.00%	83.72%	0.00%	1609
DRIVING	3.82%	0.00%	0.00%	0.00%	96.18%	0.00%	996
							13530

Place Listener

	-2	-1	0	1	2	-20	#
UNKNOWN	44.90%	0.00%	4.08%	0.00%	44.90%	6.12%	49
STATIONARY_INDOORS	82.92%	2.56%	3.69%	2.94%	3.76%	4.14%	9665
STATIONARY_OUTDOORS	30.39%	6.37%	7.80%	4.93%	47.43%	3.08%	487
WALKING_INDOORS	49.59%	3.73%	4.70%	5.11%	30.11%	6.77%	724
WALKING_OUTDOORS	13.55%	3.73%	6.28%	6.96%	62.77%	6.71%	1609
DRIVING	6.43%	0.00%	2.01%	1.71%	68.27%	21.59%	996
							13530

5.2.4 Vyhodnotenie

Ako vidno z vyhodnotenia, pre context listener sa nám podarilo dosiahnuť úspešnosť klasifikácie 88% pre stacionárny stav v budove a 84% pre aktívny stav mimo budovy čo predstavuje výrazné zlepšenie oproti pôvodnej metóde. Ak keď sú výsledky veľmi dobré, ešte stále je priestor na zlepšenie. Treba si ale uvedomiť, že 100% sa nám dosiahnuť nepodarí, keďže musíme jednak počítať s určitou nepresnosťou v anotovaných údajoch, obmedzeniami samotnej metódy (pridanie intervalov, kedy je človek aktívny po zmene kontextu) ako aj obmedzeniami senzorov (napr. výpadky Wi-Fi signálu, dosah).

Na základe testovania sme ešte navrhli ďalšie vylepšenia, ktoré by mohli zlepšiť klasifikáciu a to:

- Periodické obnovovania kontextu namiesto jeho jednorázového určenia pri zmene.
- Filtrovanie veží, ktoré sa započítajú do z nedávno pripojených veží na základe veku, sily signálu.
- Začlenenie place listenera pri zmene kontextu, za účelom zníženia intervalu po zmene kontextu).

5.3 Pokročilé spracovanie GPS súradníc (Kalmanov filter)

#2161, Zodpovedná osoba: Peter Krátky

Úloha

Výstupom niektorých telefónov pri meraní GPS je nameraná dráha s odskokmi od skutočnej. Je tak zubatá a nepodobá sa skutočnosti. Požiadavkou na aplikáciu je vyhladiť nerovnosti dráhy, aby sa viacej podobala skutočnosti.

Analýza

V praxi sa veľmi často používa na vyhladenie zašumeného signálu štatistická metóda s názvom Kalmanov filter. Použitie je vhodné vtedy, ak sú k dispozícii namerané údaje, a okrem toho sa dajú tieto údaje nejakým spôsobom približne vypočítať.

Model pozostáva z dvoch fáz:

1. Predikcia stavu
   $x_{k|k-1} = F_k * x_{k-1|k-1} + B_k * u_k$ - výpočet nového stavu (odhad) $P_{k|k-1} = F_k * P_{k-1|k-1} * F_k^T + Q_k$ - výpočet kovariančnej matice (odhad)

2. Aktualizácia
   $K_k = P_{k|k-1} * H_k^T + (H_k * P_{k|k-1} * H_k^T + R_k)^{-1}$ - výpočet kalmanovej matice prírastkov
   $x_{k|k} = x_{k|k-1} + K_k * (z_k - H_k * x_{k|k-1})$ - výpočet nového stavu na základe odhadu stavu $x_{k|k-1}$ a odmeraného stavu z_k
   $P_{k|k} = (I - K_k * H_k) * P_{k|k-1}$ - úprava matice kovariancií na základe predchádzajúcej úpravy nového stavu
   Jednotlivé významy symbolov sú nasledujúce: $x_{k|k-1}$ je odhad stavu, z_k je nameraný stav, $x_{k|k}$ je stav po kombinácií odhadu s meraním, P je kovariančná matica, ktorá prenáša chybu z predchádzajúcich meraní, K je kalmanova matica prírastkov, ktorá určuje, či bude mať väčšiu váhu meranie alebo odhad, Q_k je matica chýb merania, R_k je matica chýb odhadu

Návrh

Súradnice, ktoré vracia GPS sú v Kalmanovom filtri reprezentované ako namerané hodnoty. Výpočet odhadovaných hodnôt je uskutočňujeme na základe rýchlosti a uhlu nasmerovania voči severu, ktoré vracia GPS a považujeme ich za pomerne dobré.

Pre reálne súradnice získané z GPS je potrebné najprv konvertovať súradnice so zemepisnou šírkou a výškou na vzdialenosť v metroch od prvého bodu. Je to preto, aby sa jednoducho dali vypočítať nové súradnice na základe rýchlosti a uhlu.

Pre čo najlepší výsledok filtra je potrebné vhodne reprezentovať problém pomocou matíc v matematickom modeli.

Pre problém týkajúci sa pohybu, sú odporúčané tieto tvary matíc:

$$x = \begin{matrix} x_pos \\ y_pos \\ dx_pos \\ dy_pos \end{matrix}$$

kde x_pos, y_pos sú súradnice pozície od prvej súradnice v metroch a dx_pos, dy_pos sú zmeny polohy. Tie vypočítame ako
$dx_pos = sin(uhol) * rychlost * dt$

$dy_pos = cos(uhol) * rychlost * dt$
kde dt je časový rozdiel medzi súradnicami.

$$F = \begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$H = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

Q_k a R_k sa menia v čase

$$R_k = \begin{matrix} \sigma_m^2 & 0 \\ 0 & \sigma_m^2 \end{matrix}$$

kde σ_m je Odchýlka merania, a tú vracia GPS pre každú súradnicu.

$$Q_k = \sigma_o^2 * \begin{matrix} dt^4/4 & 0 & dt^3/2 & 0 \\ 0 & dt^4/4 & 0 & dt^3/2 \\ dt^3/2 & 0 & dt^2 & 0 \\ 0 & dt^3/2 & 0 & dt^2 \end{matrix}$$

kde σ_o je odchýlka odhadu. V našom prípade je inicializovaná na hodnotu 0.1

Matica kovariancií P sa inicializuje na hodnotu

$$P = 10 * \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

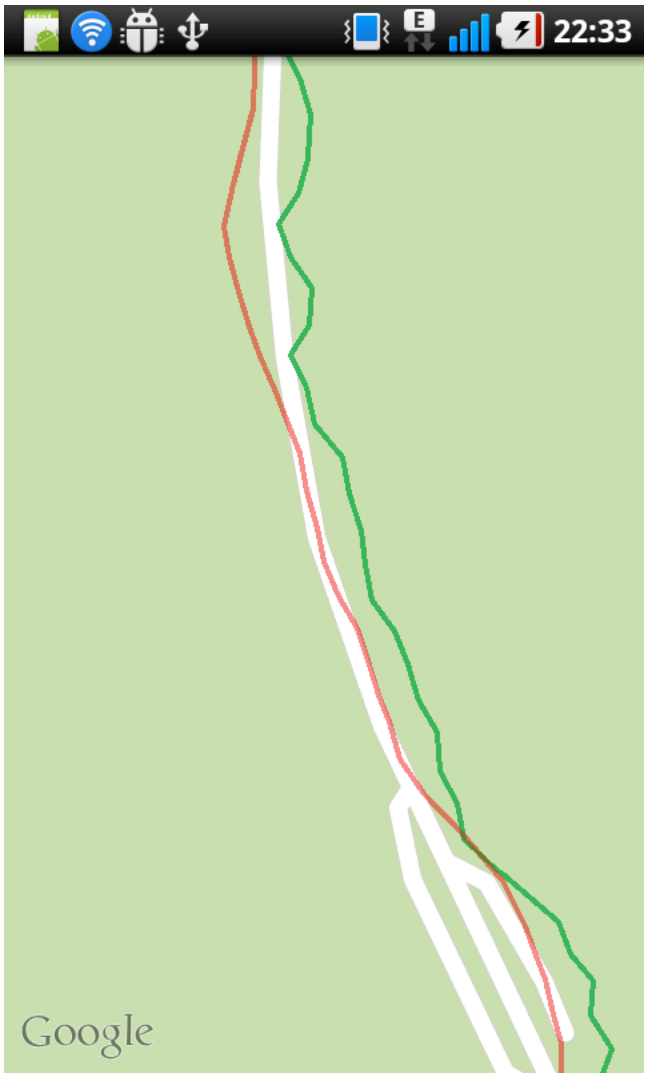
Overenie návrhu bolo vykonané v programe Scilab, ktorý umožňuje jednoduchú prácu s maticami a grafmi.

Implementácia

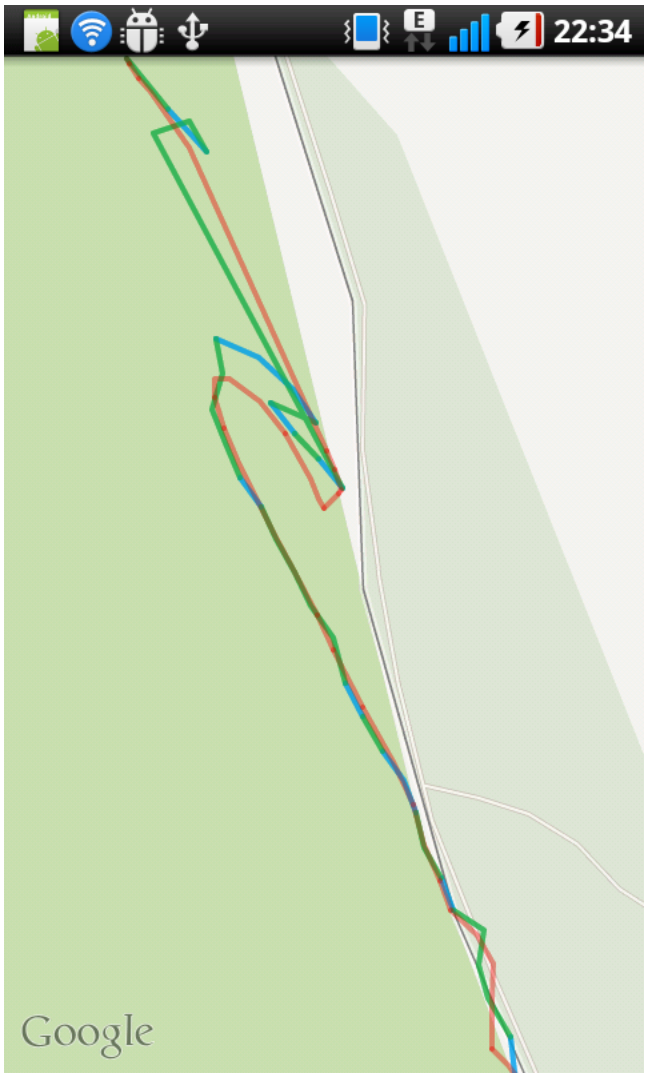
Pre implementáciu Kalmanovho filtra na Andorid je potrebné implementovať tiež operácie s maticami v novej triede *Matrix*. Algoritmus Kalmanovho filtra v aplikácii je v triede *KalmanFilter*. Po vykonaní filtra však musíme relatívne súradnice udané v metroch od prvej súradnice, s ktorými sme počítali, nahradiť absolútnymi v tvare zemepisná šírka a výška. Keďže v našom prípade pôjde o krátke vzdialenosti medzi súradnicami (10m), tak môžme zanedbať rôzny polomer Zeme pri rôznych zemepisných šírkach.

Problém nameraných údajov z GPS je ten, že uhol, ktorý vracia nezodpovedá celkom realite a dosť kolíše. Vzhľadom na to, berieme do úvahy uhol, ktorý vytvára posledná nameraná súradnica s predchádzajúcou. A to tak, že tento uhol spriemerujeme s uhlom, ktorý vracia GPS. Tak docielime vyhladenie nerovnomerných uhlov.

Výsledok aplikácie filtra je možné vidieť na obrázku 1, kde vyhladzuje súradnice s výchylkami a na obrázku 2, kde sú ťažko čitateľné zhluky čiastočne vyrovnané a skrátená dĺžka dráhy. Namerané súradnice sú zobrazené zelenou farbou, po aplikovaní filtra červenou.



Obr. 28: Kalmanov filter aplikovaný na “zubaté” súradnice



Obr. 29: Kalmanov filter aplikovaný na zhluky spôsobené veľkou nepresnosťou

5.4 Pokročilá synchronizácia

#1077, Zodpovedná osoba: Michal Tomlein

5.4.1 Úloha

Ciele tejto úlohy sú:

- Zrýchliť synchronizáciu: keďže každý pridaný a zmenený záznam sa posiela v samostatnom requeste na server, synchronizácia je veľmi pomalá.
- Implementovať operáciu DELETE, ktorá zatiaľ nie je podporovaná.
- Nastaviť synchronizáciu tabuliek `physical_attributes`, `avatars` a `users`.

5.4.2 Návrh

Zrýchlenie synchronizácie

Synchronizáciu je potrebné zrýchliť tak, že sa budú viaceré nové a zmenené záznamy posielať v jednom requeste. Aktuálne sa údaje posielajú vo formáte JSON. Do úvahy prichádzajú dva spôsoby realizácie:

1. Použiť multipart requesty.
2. Posielať v jednom request JSON array (pole) objektov.

Rozhodli sme sa pre druhú možnosť, pretože sa pri nej bude posielať menej dát (multipart requesty so sebou prinášajú určitú réžiu) a jej spracovanie na strane servera je rýchlejšie.

Operácia DELETE

Operáciu DELETE je možné realizovať dvomi spôsobmi:

1. Odstránené záznamy ponechať v databáze, nastaviť im len flag „odstránený“.
2. Použiť žurnál, do ktorého sa budú automaticky vkladať informácie o odstránených záznamoch (tabuľka, ID záznamu).

Vybrali sme druhú možnosť, pretože použitie prvej možnosti by pre našu aplikáciu znamenalo značné skomplikovanie všetkých požiadaviek na databázu. Každá požiadavka by musela obsahovať podmienku, ktorou by sa vylúčili záznamy s flagom „odstránený“.

5.4.3 Implementácia a testovanie

Implementácia zrýchlenia synchronizácie

Na strane servera bolo vytvorené rozšírenie triedy ActiveRecord::Base s názvom ActiveRecordExtensions. Tento modul obsahuje metódu `update_or_initialize_with_params`, ktorá sa používa v metóde `create` napr. takto:

```
@report = Report.update_or_initialize_with_params(params)
# namiesto: @report = Report.new(params[:report])
```

Kde `params` môže obsahovať jeden objekt pod kľúčom podľa názvu modelu (napr. `params[:report]`), ale aj pole JSON objektov, ktoré Rails 3.1 automaticky vloží pod kľúč `'_json'`.

Na strane klienta bola vytvorená metóda `NetworkUtilities.postJSONArray()`, ktorá umožňuje odoslať pole JSON objektov na danú URL. Zároveň bol upravený SyncAdapter tak, aby agregoval nové a zmenené záznamy do poľa JSON objektov a odoslal ich pomocou tejto metódy.

Implementácia operácie DELETE

V Rails aplikácii bola vytvorená tabuľka `journal`, model JournalEntry a controller JournalController. Do tabuľky `journal` sa ukladajú záznamy o odstránení z tzv. žurnálovaných modelov. Za týmto účelom bolo vytvorené rozšírenie triedy ActiveRecord::Base s názvom ActiveRecordJournaling, ktoré obsahuje metódu `journaled`. Táto metóda sa volá v modeli, ktorý má byť žurnálovaný. Príklad:

```
class Report < ActiveRecord::Base
  journaled
end
```

Modely, ktoré neobsahujú stĺpec `user_id`, ale patria pod model User priradením `belongs_to␣User, :through␣=>␣X`, musia metódu `journaled` volať s parametrom – lambda funkciou, ktorá vráti `user_id`:

```
class GpsLog < ActiveRecord::Base
  journaled :user_id => lambda { |gps_log| gps_log.report.user_id }
end
```

Modely, ktoré obsahujú zdieľané údaje, a preto nie sú priradené pod model User, musia metódu `journaled` volať s parametrom `:public␣=>␣true`:

```
class ActivityType < ActiveRecord::Base
  journaled :public => true
end
```

V Android aplikáciách boli tiež vytvorené tabuľky `journal` a modely Journal. Do týchto tabuliek sa automaticky ukladajú záznamy o odstránení z tých modelov, pre ktoré boli vytvorené databázové triggery pomocou metódy `SyncSchema.createUpdateTriggers()`.

Bola vytvorená metóda `NetworkUtilities.delete()`, ktorá slúži na posielanie DELETE requestov. SyncAdapter bol rozšírený o metódu `propagateLocalDeletions()`, ktorá postupne prechádza lokálny žurnál a pre každý riadok pošle DELETE request na server a po kladnej odpovedi daný riadok zmaže. Okrem toho bola do triedy SyncAdapter pridaná metóda `syncDeleted()`, ktorá stiahne žurnál zo servera, prejde zoznamom odstránených záznamov, pričom ich odstraňuje z lokálnej databázy. Táto metóda si pamätá dátum posledného dopytu na žurnál, ktorý vždy posiela ako parameter requestu na žurnál, čím sa predíde prenosu už spracovaných záznamov.

5.5 Implementácia prvej obrazovky pre aktívnu aktivitu

#2167, Zodpovedná osoba: Štefan Mitrík

5.5.1 Úloha

Implementovať prvú obrazovku podľa návrhu.

5.5.2 Návrh

Jednotlivé obrazovky sú navrhnuté ako sled obrazoviek medzi ktorými sa dá posúvať prstom. Obrazovky sú tak komponenty typu `Fragment`. Na posúvanie medzi obrazovkami existuje komponent `ViewPager`. Vizualizácia podľa návrhu využíva na prepínanie medzi obrazovkami *bullety*. Za týmto účelom sa dá využiť knižnica `ViewPagerIndicator`.

5.5.3 Implementácia

ViewPagerIndicator

Pri implementovaní prepínania medzi obrazovkami s využitím `ViewPagerIndicator`, nastal problém v tom, že `ViewPagerIndicator` nepodporuje obrázkové komponenty. Teda *bulletom* sa dá nastaviť farba a okraj, ale na tento účel sa nedá využiť obrázok, čo je v našom prípade kvôli zložitému tieňovaniu potrebné. Ako riešenie sa ukázalo upraviť knižnicu `ViewPagerIndicator` konkrétne triedu `CirclePageIndicator` a pridať možnosť využiť obrázok pre *bullet* aj pozadie *bulletu*. Na tento účel slúžia nasledovné metódy:

```
public void setBulletImage(Bitmap image) {
mBulletImage = image;
}

public void setBulletBackroundImage(Bitmap backgroundImage) {
mBulletBackgroundImage = backgroundImage;
}
```

Vloženie mapy do fragmentu

Na vloženenie komponenty mapu do určitej aktivity musí táto aktivita dediť od `MapActivity`. V našom prípade však vkladáme mapu do `Fragment` ktorý vloženie komponentu mapy nepodporuje. Komponent `Fragment` bol zavedený v Androide od verzie 3.0, a nakoľko naša aplikácia podporuje telefóny s verziou operačného systému od 2.2 vyššie, na využitie fragmentu využívame *Compatibility Library*. Na problém využitia komponentu mapy v rámci Fragmentu existuje špeciálne upravená verzia *Compatibility Library (https://github.com/petedoyle/android-support-v4-googlemaps)*, ktorú sme využili.

Prepínanie medzi obrazovkami

Prepínanie medzi obrazovkami sa uskutočňuje ťahaním prsta po obrazovke. Tu nastáva problém, pretože ak používateľ ťahá prstom nad mapou, tak chceme, aby sa neprepínal medzi obrazovkami ale posúval po mape. `MotionEvent` však spracuje `ViewPager` a preto sa nedostane ku mape. Za

účelom riešenia tohto problému bol vytvorený `ConstrainedViewPager`, ktorý dedí od `ViewPager`. Prostredníctvom metódy:

```
public void addScrollConstrain(int screen, ScrollConstrain constrain);
```

sa pridávajú ohraničenia pre obrazovky. Každé ohraničenie prislúcha obrazovke s id, a constrain určuje pre aký interval y hodnôt na obrazovke nemá `ViewPager` spracovávať `MotionEvent`. Na základe toho môžeme napríklad na obrazovke s mapou pridať interval y hodnôt v ktorých sa nachádza mapa, a ak používateľ ťahá prstom v tomto intervale, `MotionEvent` sa dostáva až ku mape.

Výsledok implementácie

Na nasledujúcom obrázku je zobrazený výsledok implementácie prvej obrazovky:

Obr. 30: Úvodná obrazovka aktívnej activity

6 Zimný semester

6.0.4 Úvod

Základným prvkom je integrácia štyroch častí – merania aktivity, vyhodnotenia aktivity, odporúčania aktivity a motivácie, do jedného celku. Všetky tieto časti sú nevyhnutnou súčasťou riešenia, ktoré si kladie za ciel osloviť širokú verejnosť, poskytnúť jej plnohodnotný nástroj na manažovanie ich fyzickej aktivity a zároveň poskytnúť dostatočnú motiváciu na jeho používanie. Práve tento prístup nás odlišuje od existujúcich riešení, ktoré sú pre fyzicky aktívnych ľudí a teda nevyhnutne nepotrebujú integrovať všetky časti v plnej miere.

Počas zimného semestra sme sa zamerali na dve hlavné časti a to *meranie aktivity* a *vyhodnotenie aktivity*. Cieľom bolo integrovať tieto časti do mobilné telefónu používateľa pre operačný systém Android s nevyhnutnou podporou serverovej časti. V tejto fáze vývoja sme ešte nezačali pracovať na vyhodnocovaní aktivity prostredníctvom webovej aplikácie.

6.0.5 Prehľad Systému

Prototyp sa skladá s troch hlavných častí, ktorá každá plný špecifickú úlohu:

- *ActivityTracking* je zodpovedná za meranie aktivity na mobilnom zariadení.
- *Move2Play* poskytuje vyhodnotenie nameraných údajov a ich synchronizáciu zo serverom.
- *Server* poskytuje základné webové služby, synchronizácia a odporúčanie denného plánu.

Prehľad systému je zobrazený na obrázku. Activity Tracking je vytvorená ako samostatná aplikácia, ktorá beží na pozadí ako Service. Jej úlohou je merať aktivity používateľa počas celého dňa bez toho, aby ju musel používateľ explicitne zapnúť vždy, keď sa začne pohybovať. ActivityTracking postupne napĺňa databázu údajmi o aktivite používateľa, ktorú môžu následne používať iné aplikácie. Práve tým, že sme ActivityTracking oddelili od Move2Play sme dosiahli znovu použiteľnosť, kedy viaceré aplikácie môžu poskytovať vizualizáciu rovnakých dát. Lokálna databáza je sprístupnená prostredníctvom Providera.

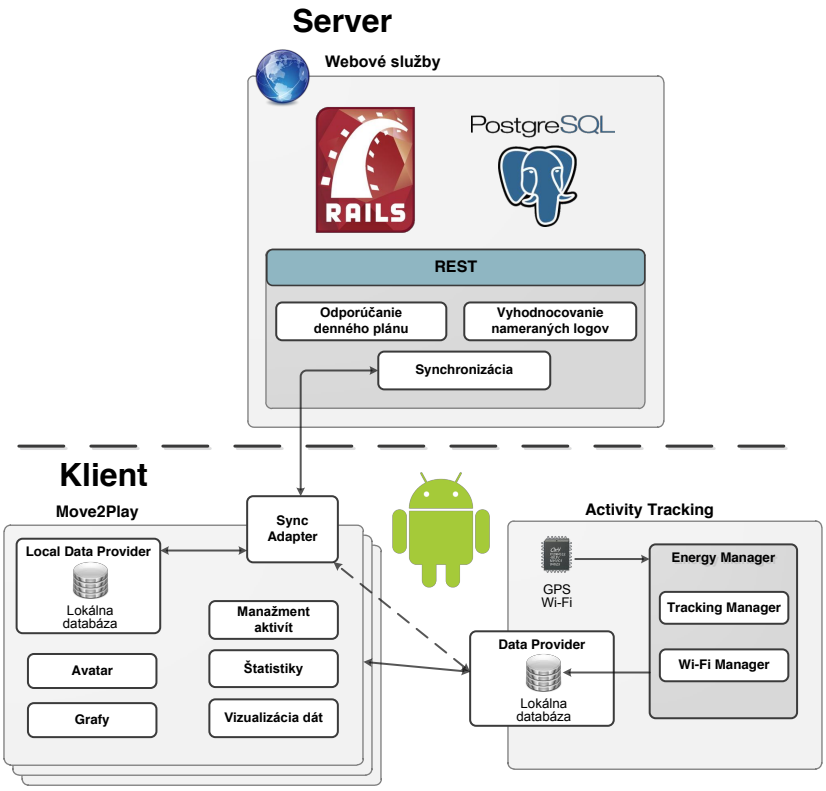
Move2Play pracuje nad dátami z ActivityTracking, pričom vnútorne ani nemusí vedieť, že databáza patrí inej aplikácii. Obmedzenie, ktoré takýto prístup prináša, je horšie využívanie zdrojov a pomalšia odozva. Dáta sú prostredníctvom Move2Play synchronizované s účtom používateľa na servery, pričom využíva aktuálne implementovaný model. Synchronizácia podporuje operácie insert, update, delete ako aj synchronizáciu s viacerými zariadeniami. Hlavných účelom Move2Play aplikácie je vyhodnotenie nameraných údajov. Tá je dosiahnutá vizualizáciou nameranej aktivity pomocou grafov, mapy ako aj poskytnutí rôznorodých štatistík.

6.0.6 ActivityTracking Aplikácia

ActivityTracking je tvorený štyrmi hlavnými časťami:

Data Provider

Zabezpečuje logiku prístupu k databáze externým aplikáciam. Externým aplikáciam poskytuje RESTful API. Poskytuje možnosť zaregistrovať listener na update znolenej URI. Taktiež obsahuje kontrolu prístupových práv k databáze na úrovniach write a read-only.



Obr. 31: Prehľad systému

Tracking Manager

Úlohou Tracking Managera je meranie aktivity pomocou senzora GPS a spracovanie nameraných údajov. Vlastnosti Tracking Managera sú nasledovné:

- ropozná ak sa človek prestal hýbať alebo ako je v budove a vypína senzor GPS za účelom šetrenia batérie
- snaží sa o lepší výkon senzora pravidelnou aktualizáciou pomocných dát
- filtruje aktivity, ktoré boli namerané v dôsledku pohybu vo vozidle
- zlepšuje nepresnosti merania pomocou matematickej metódy Kalamnov filter

Wi-Fi Manager

Na základe analýzy Wi-Fi signálu určujeme stav aktivity používateľa. Hlavnou motiváciou je pritom šetrenie baterky zariadenia. Na analýzu používame jednak algoritmické modely ako aj metódy umelej inteligencie v podobe neurónových sietí.

Energy Manager

Úlohou Energy Managera je riadenie a monitorovanie činnosti Tracking a Wi-Fi manažérov. Taktiež sa ale využíva nadhľad nad činnosťou týchto manažérov a snaží sa dodatočne vylepšovať efektívnosť využitia batérie. Implementované vylepšenia zahŕňajú.

- obmedzenie zapínania Tracking Managera na základe kontextu. Kedy sa snažíme zapamätať úspešnosť zapnutia senzora GPS na základe kontextu v ktorom sme sa senzor snažili zapnúť.
- vypínanie merania v noci
- vypnutie merania v prípade nízkej úrovne baterky zariadenia.

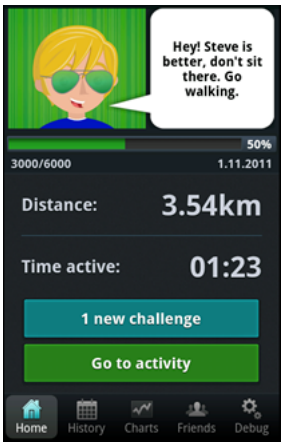
6.0.7 Move2Play Aplikácia

Hlavným cieľom aplikácie Move2Play je v tejto verzii okrem merania aj vizualizácia a vyhodnotenie nameraných údajov. Namerané údaje sa zobrazujú na viacerých obrazovkách:

- Úvodná obrazovka
- Obrazovka s grafom
- Aktuálne vykonávaná aktivita
- Ukončená aktivita

Úvodná obrazovka

Úvodná obrazovka aplikácie. Poskytuje rýchly prehľad o používateľovej aktivite v aktuálnom dni. Vizualizuje vzdialenosť ktorú používateľ prešiel a čas ktorý sa hýbal. Okrem toho zobrazuje odporúčané množstvo aktivity a percento spĺňania denného plánu.



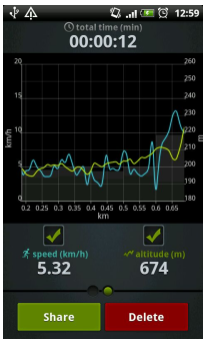
Obr. 32: Úvodná obrazovka Move2Play

Obrazovka s grafom

Obrazovka s grafom používateľovi vo vybranom časovom intervale zobrazuje informácie o množstve vykonanej a odporúčanej aktivity. Používateľ má možnosť výberu časového úseku ktorý chce sledovať:

- Roky
- Mesiace

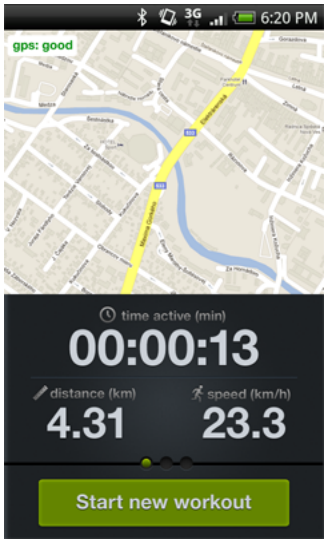
- Dni Graf je interaktívny. Po klepnutí na stĺpec alebo oblasť stĺpca sa zobrazí hodnota vykonanej a odporúčanej aktivity a percento splnenia denného plánu.



Obr. 33: Grafové zobrazenie údajov o aktivite

Aktuálne vykonávaná aktivita

Informácie o aktuálne vykonávanej aktivite sa používateľovi zobrazujú na troch prepínateľných obrazovkách. Používateľ sa medzi nimi prepína ťahaním prsta. Prvá obrazovka zobrazuje komplexné informácie o práve prebiehajúcej aktivite (čas, rýchlosť, priemerná rýchlosť, nadmorská výška, kalórie a iné). Druhá obrazovka zobrazuje aktuálnu polohu a prejdenú trasu na mape a základné informácie o aktivite – čas, rýchlosť, vzdialenosť. Tretia obrazovka zobrazuje graf s rýchlosťou a vzdialenosťou v závislosti na prejdenej vzdialenosti.

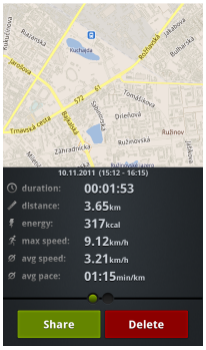


Obr. 34: Obrazovka aktuálne vykonávanej aktivity

Ukončená aktivita

Informácie o ukončenej aktivite sa zobrazujú na dvoch prepínateľných obrazovkách. Na prvej sa nachádzajú sumárne informácie o ukončenej aktivite a mapa s prejdenou trasou. Druhá obrazovka

zobrazuje podobne ako tretia obrazovka v aktuálne vykonávanej aktivite graf s rýchlosťou a nadmorskou výškou.



Obr. 35: Obrazovka ukončenej aktivity aktivity

6.0.8 Server

Server v našom prípade slúži ako dátové úložisko, ktoré poskytuje synchronizáciu a RESTful rozhranie nad potrebnými modelmi. Model synchronizácie je postavený na duplikácii hlavnej databázy zo servera na lokálne zariadenia.

Server ďalej obsahuje základnú funkcionalitu pre spravovanie používateľov. Pri odporúčaní aktivity je vytvorený dátový model a odporúčanie pre tréningový plán chôdze. Pri odporúčaní sme využili pravidlový systém pričom pre každého používateľa sa aktuálne berú do úvahy je vek, pohlavie a fyzická zdatnosť.