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 II
Študijný program:	Softvérové inžinierstvo, Ročník: 1.
Akademický rok:	2011/2012, letný semester
Kontakt:	move2play@gmail.com

Obsah

1 Ružomberok

- Revízia EnergyManager — Pavol Bielik
- Data providery — Peter Krátky
- Pozorovací stav TrackingManagera — Peter Krátky
- Challenges screen — Štefan Mitrík

1.1 Revízia EnergyManager

Zodpovedná osoba: Pavol Bielik*

1.1.1 Analýza

Aktualny stav
Service EnergyManager obsahuje

- stavy OFF, MONITORING, REQUEST_TRACKING, TRACKING, ktore pracuju s Wifi, GSM, a GPS
- ThreadedHandler, zabezpecuje wakeLock a bezanie vo vlastnom vlakne
- BatteryWatcher, vypne meranie ak je slaba baterka
- NightWatcher, vypne meranie ak je noc
- GpsUpdater, pravidelne obnovuje udaje o gps satelitoch
- GpsAvailabilityListener, sleduje ci je zapnute GPS
- ServiceBinding (RemoteNotificationsManager), poskytuje moznost komunikacie s inymy procesmi

Aktualny stav ma niekolko problemov:

- znacna komplikovanost vzhladom na vela komponentov co zahrna
- narocny (a momentalne aj zly) manazment zdrojov, nutnost zabezbecit uvolenie zdrojov vzdy ked sa prestanu pouzivat
- trackingManazer je inicializovany aj ked sa nemeria lebo obsahuje ine komponenty (GpsAvailabilityListener)
- miesanie logiky komponentov do jedneho kodu. Ide najme o manazovanie prechodov medzi stavmy, a kontrolovanym ci ma vobec zmysel meriat.
- Strata kontroly nad moznostou vypnutia sluzby vzhladom na poskytnutie ServiceBinding. Ak je pripojeny aspon jeden klient, tak sa neda sluzba vypnut (aspon sa mi to zatial nepodarilo zo strany sluzby).

1.2 Data providery

1.2.1 Úloha

Na poskytnutie údajov pre grafy je potrebné vytvoriť providery, ktoré vrátia štatistické údaje pre zvolené obdobia

1.2.2 Implementácia

Na rýchle získanie údajov z databázy slúžia tieto providery:
ActivityDataProvider - metódy pre vrátenie záznaov o aktvitách
PeriodDataProvider - metódy pre vrátenie štatistík pre zadané obdobie
PlanFulfilmentDataProvider - metódy pre poskytnutie údajov grafu o plnení plánu (výška stĺpca)
PlanDataProvider - metódy pre poskytnutie údajov grafom o plánoch (horizontálna čiara v grafe)

ActivityDataProvider

poskytované metódy:
getAllActivities() - vráti všetky aktivity vo forme objektov IWorkoutStatistics
getYearActivities(Date) - vráti všetky aktivity za zadaný rok vo forme objektov IWorkoutStatistics
getMonthActivities(Date) - vráti všetky aktivity za zadaný mesiac vo forme objektov IWorkoutStatistics
getDayActivities(Date) - vráti všetky aktivity za zadaný deň vo forme objektov IWorkoutStatistics
getWorkout(int id) - vráti konkrétnu aktivitu
getWorkout() - vráti poslednú aktivitu
getSpeed(int acitivtyId) - vráti pole rýchlostí pre jednotlivé úseky aktivity
getAltitude(int acitivtyId) - vráti pole nadmorských výšok pre jednotlivé úseky aktivity
Rozhranie IWorkoutStatistics poskytuje prístup k týmto metódam:
getWorkoutId()
getScore()
getDistance()
getAvgSpeed()
getMaxSpeed()
getLasting()
getReportId()
getActivityType()
getDateStart()
getDateEnd()
getPace()
getEnergy()
getSteps()
getLocations()

PeriodDataProvider

poskytované metódy:
getYearStatistics(Date) - vráti štatistiky pre rok vo forme objektu IPeriodStatistics
getMonthStatistics(Date) - vráti štatistiky pre mesiac vo forme objektu IPeriodStatistics
getWeekStatistics(Date) - vráti štatistiky pre týždeň vo forme objektu IPeriodStatistics
getDayStatistics(Date) - vráti štatistiky pre deň vo forme objektu IPeriodStatistics

Rozhranie IPeriodStatistics poskytuje prístup k týmto metódam:
getScore()
getDistance()
getAvgSpeed()
getLasting()
getPace()
getEnergy()
getSteps()
getRecommendation() getRecommendationAvailibility()

Poznámka:

Metóda *getRecommendation()* vracia odporúčaný plán za mesiac k aktuálnemu dňu. Predpoklad metódy je, že k aktuálnemu dňu je vždy odporúčanie.
To znamená, že ak je 10. januára a plán je vypočítaný pre každý deň v januári, metóda vráti len súčet plánov pre prvých 10 dní.
Naopak, ak je plán len do 5. januára a je deň 10. januára, plán sa neaproximuje, ale vráti sa súčet pre prvých 5 dní.

Pomocou metódy *getRecommendationAvailibility()*možno zistiť, či ten plán bol dostupný (AVAILIBLE) alebo či nie je dostupný vôbec (NOT_AVAILIBLE)

Toto je možné využitiť v grafoch

PlanFulfilmentDataProvider, DistanceDataProvider, DurationDataProvider a PlanDataProvider

Tieto providery poskytujú metódy:
getYearData()
getMonthData() getDayData()

Vracajú polia vo forme objetov ITimeChartValue. Toto rozhranie predpisuje knižnica grafov. Ide vlastne len o číselné údaje (skóre, vzdialenosť, trvanie, resp. odporúčanie za obdobie)

PlanDataProvider odporúčania funguje rovnako ako pri *PeriodDataProvider*.

PlanFulfilmentDataProvider, *DistanceDataProvider*, *DurationDataProvider* získavajú údaje za obdobie z tabuľky Reports. Každý z providerov však počíta sumu z iného stĺpca (Score, Distance, Duration). Spoločná funkcionalita je vyčlenená v triede *ChartsDataProvider*

1.3 Pozorovací stav TrackingManagera

#2643, Zodpovedná osoba: Peter Krátky

1.3.1 Úloha

Ak sa nachádza používateľ mimo dosahu WiFi prístupových bodov, nemá ActivityTracking informáciu o tom, či sa človek hýbe, pokiaľ nechceme mať zapnuté meranie neustále. Je potrebné vytvoriť stav, ktorý pri neaktivite v určitých časových intervaloch sleduje výraznú zmenu polohy a upozorní na to.

1.3.2 Návrh

TrackingManager sa zapne v špeciálnom režime, kedy v pravidelných intervaloch získa a pozoruje novú súradnicu:

1. pozoruje, či sa zmenila poloha - porovnáva či aktuálna súradnica v je vo výraznej vzdialenosti oproti poslednej v databáze
2. pozoruje, či je zariadenie vo vozidle - porovnáva nameranú rýchlosť s prahom rýchlosti pre vozidlá

Ak sa nepodarí získať súradnicu (nie je možné získať lock), interval medzi jednotlivými pozorovaniami sa predĺži.

Použité konštanty

výrazná vzdialenosť - 30 m
rýchlosť vo vozidle - 3 ms-1
perióda merania - 1 minúta perióda merania po neúspechu získania locku - 3 minúty

1.3.3 Implementácia

Trieda `LocationWatcher` realizuje perodické meranie. GPS senzor sa zapína len na čas potrebný na získanie jednej súradnice, potom sa na určitý čas (perióda pozorovania) vypne. Trieda `TrackingManager` v metóde `onNewLocation` vyhodnocuje získanú súradnicu a notifikuje o udalostiach observera, ktorý implementuje rozhranie IGpsStatusObserver.

```
public interface IGpsStatusObserver {
  // ... other events headers

  // observed events headers
  public void onObservedInVehicle();
  public void onObservedPositionSignificantlyChanged(float distance);
  public void onObservedNoGpsLock(boolean inBuilding);
}
```

spustenie pozorovania - trackingManager.startTracking(PrecisionLevel.Observe)

1.4 Challenges screen

1.4.1 Návrh

Obrazovka challenges zobrazuje používateľovi krátkodobé výzvy. Každá challenge má určenú:

- kategóriu
- dĺžku trvania
- úlohu
- progress
- v prípade skupinových výziev tiež ostatných zúčastnených

Návrh obrazovky je zobrazený na nasledujúcom obrázku:

Obr. 1: Návrh obrazovky challenges

1.4.2 Implementácia

Pre potreby účasti na konferencie bolo implementované demo "Wi-Fi challenge". Pri tejto výzve sa mal používateľ do určitého časového intervalu dostať z dosahu stanovenej wifi siete.

2 Vrútky

- Integrácia GPS šetriaceho stavu do EnergyManagera — Pavol Bielik
- Editovanie trasy vykonanej aktivity — Pavol Bielik
- Pasívne sledovanie Wi-Fi — Pavol Bielik
- Manažovanie notification ikoniek pre startForeground service — Pavol Bielik
- Logovanie stavu batérie — Pavol Bielik
- Rozlišovanie súradníc surových a filtrovaných — Peter Krátky
- Nerozdelovať aktivity, ak sa ide vo vozidle — Peter Krátky
- Vylepšenie spájania aktivít — Peter Krátky
- Refactoring a nová funkcionalita v triede Model — Michal Tomlein
- Pridávanie kamarátov z Facebooku — Michal Tomlein
- Obrazovka Friends — Štefan Mitrík

2.1 Integrácia GPS šetriaceho stavu do EnergyManagera

2.1.1 Úloha

Úlohou je integrácia stavu Observable v TrackingManageri, ktorý je určený na zisťovanie mobility používateľa pomocou GPS v prípade, ak nemáme dostatok údajov pre tento účel z Wi-Fi.

2.1.2 Implementácia

Integrácia pozostáva z vytvorenia nového stavu v EnergyManageri, po ktorého aktivácii a deaktivácií sa spustí, resp. vypne meranie v režime Observable. Pre tento stav je potrebné definovať podmienky jeho aktivácie a deaktivácie.

Podmienky aktivácie

- 3-krát neúspešné zapnutie GPS v priebehu 10 minút.
- Wi-Fi Watcher počas tejto doby klasifikuje mobilitu používateľa hodnotami `WatcherState.UNKNOWN` alebo `WatcherState.NO_DATA`. V prípade inej hodnoty sa počet neúspešných zapnutí vynuluje.

Podmienky deaktivácie

- Wi-Fi Watcher klasifikuje mobilitu hodnotou inou ako `WatcherState.UNKNOWN` alebo `WatcherState.NO_DATA`.
- TrackingManager notifikácia `onObservedPositionSignificantlyChanged`

2.1.3 Testovanie

Testovanie pozostáva z pokrytia možností prechodov medzi stavmi pomocou unit testov.

2.2 Editovanie trasy vykonanej aktivity

#2706, Zodpovedná osoba: Pavol Bielik

2.2.1 Úloha

Umožnenie editovať používateľovu trasu už vykonanej aktivitu. Podporovanie akcie sú pridanie, vymazanie, posunutie bodu ako aj označenie jedného alebo viacerých bodov ako filtrovaných, prípadne nefiltrovaných.

2.2.2 Návrh

Základné akcie

Obrazovka editovania existujúcej aktivity zobrazuje jednotlivé namerané GPS súradnice a umožnuje štyri základné akcie ktoré sa dajú nad bodmy vykonávať - označenie, pridanie, posunutie, vymazanie.

Označenie bodu. Každý bod je možné označiť kliknutím priamo na neho, alebo v jeho blízkom okolý. Po kliknutí sa zmený farba bodu na orazovú. Po kliknutý na už označený bod sa tento bod odznačí. Označených bodov pritom môže byť viacero. Ak je označený aspoň jeden bod, tak sa vysunie aciton bar, na ktorom je možné jedným tlačitkom všetky označené body odznačiť.

Pridanie bodu. Pridanie bodu je vykonané akciou long-click nad prázdnym miestom na mape. Bod sa pridá na nové miesto pričom je s ním možné hýbať a meniť tak s akými bodmi je spojený. Body na mape sú vždy spojené s ich predchodcom a nalsedovných a tak na mape je vždy možné mať iba jeden súvislý segment.

Vybratie susedou nového bodu je vykonané vypočítaným jemu najblizšej hrany v grafe, ktorý tvoria zvyšné body trasy. Bod sa následne pridá medzi dva existujúce body tvoriace najbližsiu hranu. Jedniná výnimka spočíva v pridávaní bodu na začiatok alebo na koniec trasy. Pri pridávaní body je možne tento bod po mape posúvať a sledovať ako sa mu menia susedia v závislosti na jeho polohe.

Posunutie bodu. Posunutie je vykonané akciou long-click nad alebo v blízkosti existujúceho bodu. Po vykonaný takejto akcia sa bod označí a vykreslí sa kruh, ktorého stred tvorý posúvaní bod. Tento kruh tvorý vizuálnu pomôcku pri posúvaní body, aby používateľ vedel kde sa posúvaný bod nachádza, kedže pri posúvaní má väčšinou prst položený práve na posúvanom bode. Posúvanie bodu je zobrazené na nasledovnom obrázku.

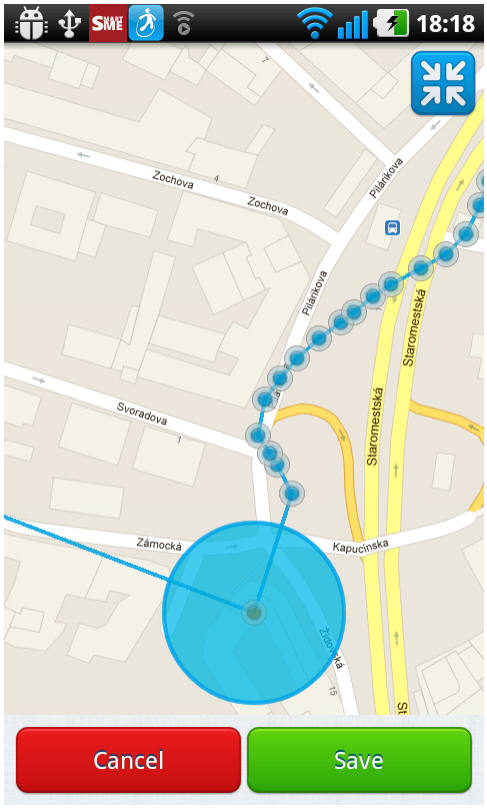
Vymazanie bodu/bodov. Vymazanie jedného alebo viacerých bodov je mozné po ich označený, kedy sa vysunie action bar, v ktorom je možnosť ich vymazania. Action bar a označenie bodov je zobrazné na nasledovnom obrázku.

Doplňujúce akcie

Okrem základných akcií popísaných vyššie sme zahrnulý aj dve pokročilé akcie filtrovania a rýchleho označovania bodov.

Filtrovanie bodov. Po označený bodov na mape je možné nastaviť, či majú byť považované za filtrované alebo ich máme odfiltrovať. Takýmto spôsobom má používateľ možnosť manuálne určovať kedy išiel naprílad autom a taktiež má možnosť opraviť prípadné chybné automatické klasifikovanie vykonané Move2Play.

Rýchle označovanie bodov. Rýchle označovanie umožnuje označiť viacero bodov naraz. V tomto móde sú medzi dvoma označenými bodmi označené, prípadne odznačené, aj všetky body nachádzajúce sa medzi nimy.



Obr. 2: Posúvanie bodu na mape

2.2.3 Implementácia

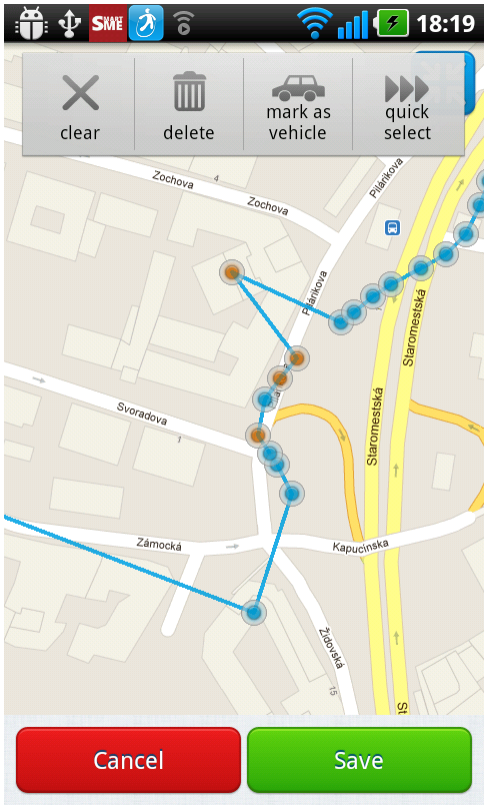
Každá vykonatelná akcia je reprezentovaná triedov, ktorá implementuje rozhranie IChangeAction.

```
private interface IChangeAction {
    public void apply();
    public void revert();
}
```

Pri vykonaní sa akcia pridá na lokálny zásobník akcií a zavolá sa jej metóda apply(). Týmto spôsobom vieme následne jednoducho implementovať akcie undo, pri ktorej jednoducho vyberáme akcie zo zásobníka a zavoláme ich metódu revert(). Undo je implementované ako hardvérové tlačilko back na telefóne.

Implementované akcie sú:

- MovedPoint
- AddedPoint
- RemovedPoints
- FilteredPoints
- SelectedPoint



Obr. 3: Možné akcie vykonateľné nad označenými bodmi

- SelectedPoints
- UnselectedPoint
- UnselectedPoints

Zaznámenavanie Selected a Unselected points je zaznamenávame iba medzi jednotlivými akciami Moved, Added, Removed alebo Filetered.

2.3 Rozlišovanie súradníc surových a filtrovaných

#2806, Zodpovedná osoba: Peter Krátky

2.3.1 Úloha

Návrh a implementácia zobrazenia súradníc na mape tak, aby bolo jednoducho rozlíšiteľné, čo bola aktivita chôdza a čo bola odfiltrovaná aktivita.

Z rôznych dôvodov je dobré, aby sa dalo zistiť, aký filter bol aplikovaný nad jednotlivými súradnicami a tiež, ako vyzerali súradnice pôvodne.

2.3.2 Návrh

Na tento účel slúži systém flagov. Každá súradnica má flagy, či bol alebo nebol konkrétny filter aplikovaný.

Sleduje sa nasledovné:

- Aproximácia rýchlosti
- Aproximácia uhlu
- Vo vozidle
- Slabý signál
- Nehýbe sa
- Dopočítané spojenie
- Kalmanov filter
- Pôvodná

2.3.3 Implementácia

V tabuľke `GpsLogs` sa nachádza stĺpec API a obsahuje flagy označené písmenkami abecedy:
s (aproximácia rýchlosti), **b** (aproximácia uhlu), **v** (vo vozidle), **l** (slabý signál) **m** (nehýbe sa) **c** (dopočítané spojenie) **k** (Kalmanov filter) **o** (pôvodná)
Hodnota v stĺpci pre súradnicu má pevnú dĺžku. Súradnica, ktorá bola odfiltrovaná, kvôli pohybu vo vozidle a bol slabý signál má hodnotu `.v.l....`, pričom bodky označujú neaktívny filter. Tento spôsob reprezentácie je zvolený na úkor pamäťového miesta preto, aby sa nemuselo Prechádzať celým reťazcom, keď sa zisťuje, či bol filter aplikovaný. Každý filter má pevné miesto v reťazci, takže je prístup vždy O(1)

Pozn. stĺpec API bol pôvodne na porovnávanie Google a Skyhook Api, a aby sa vyhlo problémom s kompatibilitou exportovaných databáz, tak je zatiaľ ponechaný na flagy

Mechanizmus nastavovania flagov a zisťovania, či bol filter aplikovaný zabezpečuje trieda `ActivityLocationFlagsHelper`

Objektu `ActivityLocation` je možné nastaviť flag metódou `setFlag(LocationFlags)` alebo `unsetFlag(LocationFlags)`. Zistenie, či je flag aktívny `hasFlag(LocationFlags)`

`LocationFlags` je enumeration flagov a nadobúda nasledovné hodnoty:
`SPEED_APPROXIMATION` (aproximácia rýchlosti), `BEARING_APPROXIMATION` (aproximácia uhlu),

`IN_VEHICLE` (vo vozidle), `LOW_SIGNAL` (slabý signál) `NOT_MOVING` (nehýbe sa) `CALCULATED` (dopočítané spojenie) `KALMAN` (Kalmanov filter) `ORIGIN` (pôvodná)

2.4 Nerozdelovať aktivity, ak sa ide vo vozidle

#2808, Zodpovedná osoba: Peter Krátky

2.4.1 Úloha

Dotaraz bol implementovaný jednoduchý spôsob fitrovania vozidla. Každá aktivita začínala validnými súradnicami. Ak sa prekročila prahová rýchlosť pre vozidlo, tak sa ukladali súradnice v tej istej aktivite. Ak však rýchlosť opäť klesla, tak sa aktivita ukončila a vytvorila sa nová aktivita. Každá aktivita tak mala prvú časť súradníc validných a druhú filtrovaných.

Tento spôsob uľahčoval mazanie tzv. medziaktivít, to znamená napríklad, ak auto zastavilo na križovatke. Jednoducho sa odstránila označila celá aktivita ako odfiltrovaná. Taktiež sa jednoducho zobrazovalo v grafe s rýchlosťou a nadmorskou výškou - zobrazila sa len časť validných súradníc.

Veľká nevýhoda ale bola, že jedna cesta do školy bola rozkúskovaná do niekoľkých malých aktivít.

Je žiadúce zobraziť na mape celú aktivitu v jednom celku aj s filtrovanými úsekmi.

2.4.2 Návrh

Aktivita sa nerozdeľuje, neukončuje sa teda a nevytvára nová. Problémom je však prípad, keď je potrebné mazať medziaktivity. A to aj vtedy, ak sa medzitým aktivita prerušila z dôsledku státia na mieste alebo straty GPS signálu a znova sa nadäzuje na aktivitu.

2.4.3 Implementácia

Odfiltrované úseky sa označia flagmi `IN_VEHICLE`. Kontrola vozidla sa vykonáva vždy pre každú novú pozorovanú súradnicu, ale odhalí sa až po 3 súradniciach. Ukladá sa do databázy nárazovo po 6. Ak chceme spätne filtrovať, je potrebné zabezpečiť update v databáze, o ktorom je napísané v podkapitole.

Ak chceme filtrovať aj medziaktivity, je potrebné si pamätať pozíciu (index) súradnice, kde sa medziaktivita začína. `VehicleFilter` sa môže rozhodnúť odfiltrovať celú medziaktivitu, vtedy sa spätne označia všetky súradnice flagmi `IN_VEHICLE` a následne sa vykoná spätný update.
Udržiava sa zoznam súradníc o poslednej aktivite aj po ukončení aktivity, konrétne sa udržiava zoznam v `LocationProcessor`, keď sa však začína nová aktivita, tento zoznam sa vymaže.

Spätný update súradníc v databáze

Chceme filtre čo najviac oddeliť od ostatných častí kódu a zvlášť od prístupu k databáze.
Každá nová súradnica je obalená do objektu `LocationProcessingData`, ktorý obsahuje informáciu o tom, koľko súradníc sa má spätne updatovať.
Vo filtri `SaveToDbFilter`, ktorý sa aktivuje každých 6 súradníc sa zisťuje, či je potrebné niečo updatovať dozadu (vzhľadom na to, že posledných 6 sa práve ukladá, tak tie nie je potrebné dodatočne updatovať) `TrackingManager`, vyvoláva okrem pôvodnej metódy `saveObjectWithSiblings()` aj novú metódu `updateObjectWithSiblings(count)`, ktorá je implementovaná v triede `ActivityReport`

2.5 Vylepšenie spájania aktivít

#2810, Zodpovedná osoba: Peter Krátky

2.5.1 Úloha

Je potrebné,

- aby sa aktivita po strate GPS locku a znovunájdení spojila,
- aby sa aktivita spojila aj na väčie vzdialenosti, ale dokázalo sa rozlíšiť, či sa išlo v dopravnom prostriedku alebo pešo,
- aby sa aktivita rozdelila vo vhodnom čase,
- Aby sa nová aktivita spojila s najstaršou súradnicu (napríklad po odchode z domu ráno sa spojí s poslednou súradnicou večer, a tak sa približne odmeria aj vzdialenosť, kým sa zapne tracking)

2.5.2 Návrh

Identifikovali sme päť základných prípadov. Prvé štyri opisujú prípad, keď súradnica posledná a nová je vzdialená tak, že pešo bolo možné prejsť vzdialenosť za odmeraný čas. Jednoduchý vzorec je možné prejsť pešo ak : odmeraná vzdialenosť $<$ odmeraný čas$\cdot$všeobecná maximálna rýchlosť chôdze. Piaty prípad opisuje ak nie je splnená podmienka. Všetkých 5 prípadov je postupne znázornených aj na obrázku.

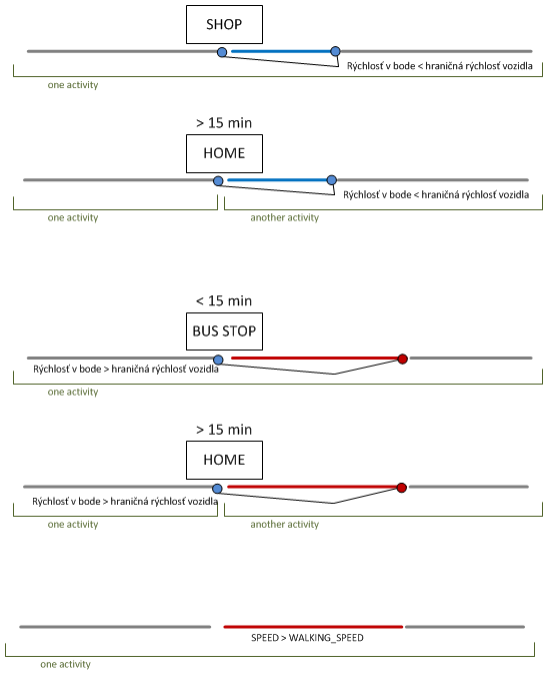
1. Meranie sa preruší a opäť obnoví do 15 minút. Obidve súradnice (posledná aj nová) majú rýchlosť pešej chôdze.
   Aktivity sa spoja (spojenie sa nefiltruje).
   Príkladom je stretnutie kamaráta na ulici, odskočenie do obchodu, strata GPS locku.

2. Meranie sa preruší a opäť obnoví po 15 minútach. Obidve súradnice (posledná aj nová) majú rýchlosť pešej chôdze.
   Vytvorí sa nová aktivita, ale prepojí sa s poslednou súradnicou v databáze (spojenie sa nefiltruje).
   Príkladom je príchod domov večer a odchod z domu ráno.

3. Meranie sa preruší a opäť obnoví do 15 minút. Aspoň jedna zo súradníc (posledná alebo nová) má rýchlosť nad prahom rýchlosti vozidla.
   Aktivity sa spoja, ale spojenie sa filtruje.
   Príkladom státie na autobusovej zastávke, nastúpenie do autobusu a meranie sa spamätá až uprostred jazdy v autobuse - je žiadúce aby sa filtrovalo.

4. Meranie sa preruší a opäť obnoví po 15 minútach. Aspoň jedna zo súradníc (posledná alebo nová) má rýchlosť nad prahom rýchlosti vozidla.
   Vytvorí sa nová aktivita, ale prepojí sa s poslednou súradnicou v databáze. Spojenie sa ale filtruje
   Príkladom je príchod domov večer a odchod z domu ráno autom, ale meranie sa spamätá až uprostred jazdy v autobuse - je žiadúce aby sa filtrovalo.

5. Meranie sa preruší a opäť obnoví. Vzdialenosť však za odmeraný čas nebolo reálne prejsť pešo.
   Aktivity sa spoja, ale spojenie sa filtruje.
   Príkladom je cesta v metre.



Obr. 4: Znazornenie piatich sposobov spajania aktivit

2.5.3 Implementácia

V triede `TrackingManager` je implementovaná metóda `joinActivities(current␣location)`, ktorá sa vyvolá vždy keď sa začína meranie (prišla prvá súradnica). Táto metóda implementuje všetky podmienky z časti Návrh. Vždy sa načíta posledná súradnica z databázy. Ak dochádza k spájaniu, načítava sa aj celá aktivita z databázy.

2.6 Refactoring a nová funkcionalita v triede Model

#3131, Zodpovedná osoba: Michal Tomlein

2.6.1 Úloha

Cieľom tejto úlohy je rozšíriť triedu Model o metódy, ktoré umožnia vykonávať operácie `SELECT`, `INSERT`, `UPDATE` a `DELETE`.

2.6.2 Návrh

Trieda Model bola navrhnutá ako obal (wrapper) triedy Cursor. Definuje tie isté metódy ako trieda Cursor, ale pridáva aj gettery, ktoré vracajú hodnotu pre stĺpec špecifikovaný menom, t.j. namiesto:

```
cursor.getDouble(cursor.getColumnIndex(Report.DISTANCE);
```

je možné použiť:

```
model.getDouble(Report.DISTANCE);
```

Tieto vlastnosti triedy Model sme chceli zachovať. Triedu sme sa rozhodli rozšíriť o nasledujúce metódy:

- `query()` - operácia `SELECT`
- `save()` - operácia `INSERT` alebo `UPDATE`
- `delete()` - operácia `DELETE`

2.6.3 Implementácia a testovanie

Metóda `query()` sa používa nasledovne:

```
Report report = new Report();
report.setId(234); // alebo
report.setSelection("x = ?", new String[] { "y" }); // alebo oboje
report.query(resolver); // je možné ďalšími parametrami špecifikovať projection a order
if (report.moveToFirst()) // nie je nutné volať explicitne
Log.d(TAG, report.getDouble(Report.DISTANCE));
```

Nový záznam sa vytvára nasledovne:

```
Report report = new Report();
report.set(Report.DISTANCE, 10.0);
report.save(resolver);
```

Ten istý objekt je možné ďalej používať (automaticky si zapamätá svoje `_id`):

```
report.set(Report.DISTANCE, 11.0);
report.save(resolver);
```

Aktualizácia existujúceho záznamu sa dá urobiť explicitným nastavením `_id`:

```
report.setId(234);
report.set(Report.DISTANCE, 12.0);
report.save(resolver);
```

Alebo nastavením selection:

```
user.setSelection(User.EMAIL + " = ?", new String[] { "jozef.mrkvicka@gmail.com" });
user.set(User.FIRSTNAME, "Jozef");
user.set(User.LASTNAME, "Mrkvička");
user.save(resolver);
```

2.7 Pridávanie kamarátov z Facebooku

#2651, Zodpovedná osoba: Michal Tomlein

2.7.1 Úloha

Cieľom tejto úlohy je umožniť používateľovi pridávať si v aplikácii kamarátov z Facebooku.

2.7.2 Návrh

Úlohu sme sa rozhodli implementovať použitím Facebook SDK na strane klienta a vytvorením modelu Friendship na serveri, ktorý reprezentuje obojsmerné prepojenie dvoch kamarátov.

Friendship obsahuje okrem ID kamarátov aj stav, ktorý môže byť `PENDING`, `APPROVED`, `IGNORED` alebo `BLOCKED`.

2.7.3 Implementácia a testovanie

Na serveri bol vytvorený model Friendship. Prepojenia kamarátov definované v tabuľke friendships sa používajú v UsersControlleri tak, že `users#index` vracia okrem prihláseného používateľa aj jeho kamarátov.

Bola implementovaná synchronizácia tabuľky friendships a upravená synchronizácia tabuľky users.

Aplikácia bola registrovaná na Facebooku.

Bol vytvorený fork Facebook SDK (http://github.com/move2play/facebook-android-sdk), ktorý bol do projektu integrovaný ako git submodule.

V Android aplikácii sa pred zobrazením obrazovky AddFriendsActivity aplikácia pokúsi použiť Single Sign-On na pripojenie na Facebook. V prípade, že používateľ ešte Facebook na telefóne nepoužil, zobrazí sa prihlasovací dialóg.

Po prihlásení je možné pridávať kamarátov kliknutím na tlačidlo *Add* vedľa ich mien.

2.8 Obrazovka Friends

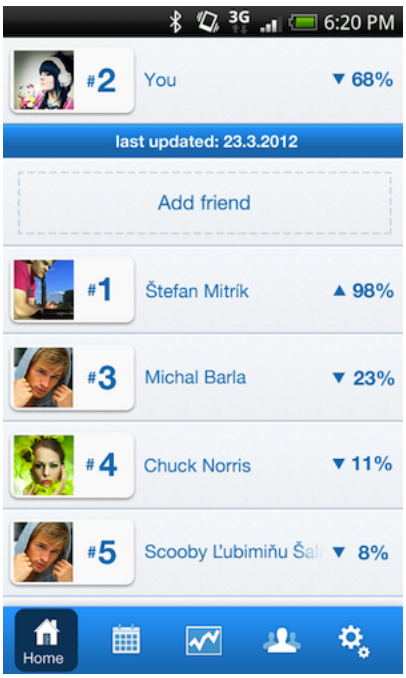
2.8.1 Úloha

Cieľom tejto úlohy je navrhnúť obrazovku Friends, kde používateľ môže sledovať ako sa darí jeho priateľom. Okrem toho by mala umožnovať posielať žiadosti o priateľstvo, prípadne potvrdzovať resp. zamietať žiadosti, ktoré boli poslané používateľovi.

2.8.2 Návrh

Na nasledujúcom obrázku je zobrazený návrh obrazovky friends. Pri každom priateľovi je zobrazené:

- meno
- pozícia v rebríčku
- percento plnenia denného plánu za posledné obdobie
- trend akým sa vyvíja jeho denný plán



Obr. 5: Návrh obrazovky friends

V spodnej časti obrazovky je umiestnené tlačidlo, ktoré používateľa presmeruje na obrazovku kde môže pridávať priateľov.

2.8.3 Implementácia a testovanie

Pri implementácii boli použité štandardné Android komponenty. Programátor má možnosť nastaviť farbu pozadia manuálne alebo automaticky.

3 Žilina

- Šetrenie batérie — Pavol Bielik
- Logovanie stavu merania — Pavol Bielik
- Mapa na úvodnej obrazovke — Pavol Bielik
- Integrácia realtime Kalmanovho filtra — Peter Krátky
- Uloženie zmien pri ručnej úprave súradníc — Peter Krátky
- Nová grafická téma — Štefan Mitrík

3.1 Logovanie stavu merania

#3143, Zodpovedná osoba: Pavol Bielik

3.1.1 Úloha

Vytvoriť logovanie stavu merania pri používaní Move2Play, aby sme vedeli vyhodnotiť a identifikovať prečo sa nachádzame v stave, v akom sme, alebo prečo nám aplikácia nič nenamerala. Zaznamenané dáta sa majú synchronizovať zo serverom.

3.1.2 Návrh a Implementácia

Pre účely logovania stavu baterky sme použili existujúcu tabuľku EnergyManagerDumps na Androide a vytvorili jej kópiu aj na Servery. Tieto dve tabuľky sa následne spolu synchronizujú prostrednictvom SyncAdaptera. Schéma tabuľky na servery je nasledovná:

```
# Table name: energy_manager_dumps
#
#  id          :integer         not null, primary key
#  timestamp   :integer(8)
#  state       :string(256)
#  description :string(512)
#  user_id     :integer
```

Logovanie stavu merania prebieha v triede ActivityTrackingManager a v EnergyManager pri zmene stavu. Pri každom zmene stavu sa úvadza aj dôvod prečo táto zmena nastala, ako napríklad “no gps signal”.

3.2 Integrácia realtime Kalmanovho filtra

#3159, Zodpovedná osoba: Peter Krátky

3.2.1 Úloha

Úlohou je integrovať implementovaný kalmanov filter tak do ActivityTracking tak, aby sa vykonával priamo počas vykonávania merania.

3.2.2 Implementácia

Keďže implementovaný Kalmanov filter využíva jednu nasledujúcu súradnicu, spracúva aktuálne vlastne vždy predposlednú prijatú súradnicu. Toto oneskorenie sa prejaví vždy ak sa aplikuje filter už na uloženú súradnicu.

Preto je potrebné vykonávať dodatočný update jednej súradnice dozadu.

Aplikácia filtra sa vykonáva na väčšej množine súradníc. Konkrétne je zvolený počet 6.

Vylepšenie vstupov filtra

Doteraz boli vstupy, konkrétne uhol (bearing) pomerne dosť nepresný. Preto je upravený `LocationWatcher` tak, že listener na polohu je zaregistrovaný na zmenu jeden meter (namiesto pôvodných 10 metrov). Počíta sa priemer uhlov zo súradníc, kým zmena polohy nenadobudne 10 metrov, až vtedy sa propaguje súradnica na ďalšie spracovanie.

Vplyv takejto implementácie na výdrž batérie by teoreticky nemal mať vplyv, pretože GPS prijímač je zapnutý neustále, ide len o to, ako často upozorňuje na novú súradnicu.

Takéto spriemerované vstupy poskytujú lepšie výsledky po aplikovaní filtra.

Taktiež je potlačené tzv. "zanášanie" filtra pri zmene smeru. Doteraz svoj vstup pre uhol upravoval aj podľa predchádzajúceho uhla ktorý zvierali dve staršie súradnice. To spôsobovalo "zanášanie".

Momentálne je implementovaná stratégia (metóda `bestBearing(prechadzajuca, aktualna, nasledujuca␣suradnica)`), ktorá podľa nasledujúcej súradnice odhaduje, či sa bude meniť smer. Vtedy sa neprihliada na predchádzajúci uhol. Ak nie sa nepredpovedá zmena smeru, ale niektorý zo vstupov má zrejme odchýlku, tak sa prihliada aj na predchádzajúci uhol.

3.3 Uloženie zmien pri ručnej úprave súradníc

#3170, Zodpovedná osoba: Peter Krátky

3.3.1 Úloha

Uloženie zmien do databázy, ktoré sa vykonávajú pri ručnej úprave prejdenej trasy. Je potrebné implementovať spôsob zaradenia novej súradnice na správnu pozíciu v trase a dopočítať niektoré údaje pre novú súradnicu.

3.3.2 Implementácia

Vykonáva sa uloženie zmien v databáze ako transakcia - zmazanie, pridanie novej, úprava súradnice (posúvanie, označenie ako filtrovaná).

Nová súradnica musí mať dopočítané údaje - čas, rýchlosť, nadmorská výška. Predovšetkým čas je dôležitý, lebo udáva umiestnenie súradnice (poradie) v aktivite. Rýchlosť a nadmorská výška sú potrebné kvôli grafom.
Novej súradnici sa vypočítavajú hodnoty na základe susedov (priemerné). Musí sa počítať v momente pridania súradnice do sekvencie. Ak by sa počítalo až na záver, a boli by dve nové súradnice vedľa seba, susedná súradnica bude mať nulové hodnoty.

Na záver sa vykonáva aj prepočet atribútov celého reportu - celková dĺžka, trvanie, priemerná rýchlosť. Uloženie prebieha v rámci jednej transakcie.

3.4 Nová grafická téma

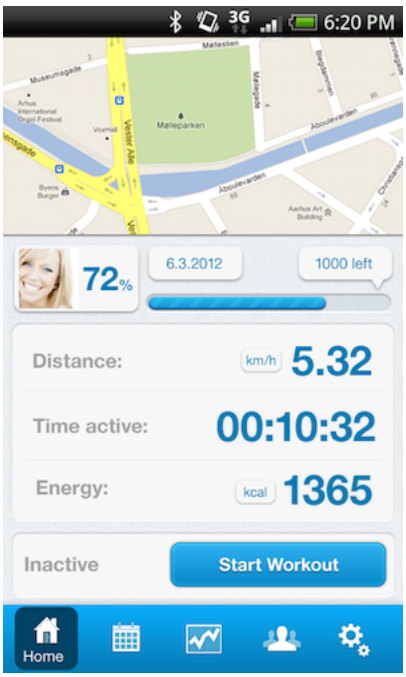
3.4.1 Úloha

Cieľom tejto úlohy je navrhnúť upraviť Home screen, odstrániť z nej newsfeed a patrične k tomu upraviť dizajn aplikácie. K úlohe bola dodatočne pridaná požiadavka na vylepšenie dizajnu aplikácie.

3.4.2 Návrh

Newsfeed na hlavnej obrazovke bol nahradený mapou, ktorá zobrazuje aktuálnu resp. poslednú lokáciu používateľa. Dizajn aplikácie bol upravený tak, aby sa viac držal modro-sivo-bielej farebnej schémy. Takisto boli prepracované detaily ako tieňovanie komponentov z viac rozmazaného na ostrejšie.

Na nasledovnom obrázku je zobrazený návrh Home obrazovky:

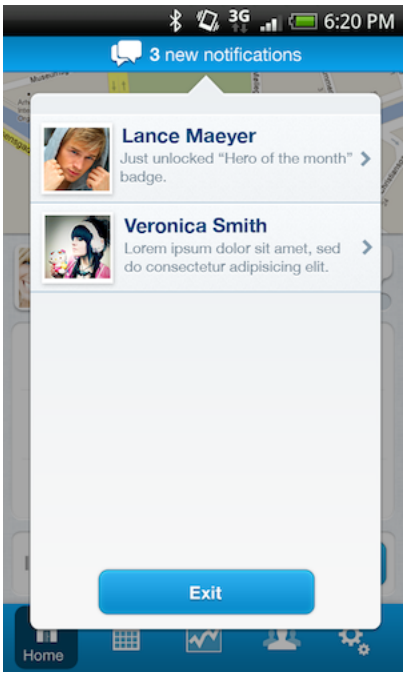


Obr. 6: Home obrazovka

Druhý obrázok zobrazuje Home obrazovku s otvorenými správami/notifikáciami/novinkami.

3.4.3 Implementácia a testovanie

Implementácia Home screen sa uskutočnila podľa návrhu. Tlačidlo start workout a box v ktorom sa nachádza neboli implementované, pretože sa rozhodlo že táto funkcionalita bude presuntá inde. Notifikácie resp. správy implementované neboli. Táto funkcionalita sa totiž bude pridávať neskôr.



Obr. 7: Home obrazovka - notifications

4 Trenčín

- Sťahovanie obrázkov — Pavol Bielik
- Dokončenie obrazovky History s ikonkami športov — Pavol Bielik
- Leaderboard List funcionalita — Pavol Bielik
- Finálne zobrazenie aktivity na mape — Peter Krátky
- Počítanie skóre a energie — Peter Krátky
- Návrh vzhľadu obrazovky Add Activity — Peter Krátky
- Login obrazovka — Štefan Mitrík
- Friend requests — Michal Tomlein
- Fotografie kamarátov — Michal Tomlein

4.1 Sťahovanie obrázkov

4.1.1 Úloha

Úlohou je vytvorenie znovupoužiteľných tried na sťahovanie obrázkov z webu. Obrázky sa sťahujú na základe URL, pričom sa ukladajú do internej pamäti zariadenia, resp. na SD kartu. Okrem sťahovania je potrebné zabezpečiť zobrazovanie obrázkov v list view ako aj ich lazy loading.

4.1.2 Návrh

Pri zobrazovaní obrázku v adaptéri sa obrázok vyhľadá na nasledovných miestach:

1. Lokálna cache, ktorá obsahuje nedávno zobrazené obrázky.
2. SD karta alebo interná pamäť zariadenia.
3. Stiahnutie obrázku z webu.

Obrázky sa sťahujú na základe URL, pričom na zariadení sa ukladajú do priečinka *images*, ktorý sa nachádza na SD karte alebo v internej pamäti, pokiaľ zariadenie nemá SD kartu.

Názov obrázku sa ponecháva, t.j. ak sťahujeme https://example.net/images/1521873546.png, obrázok sa bude volať `1521873546.png`. V tomto návrhu počítame s tým, že názvy obrázkov sú unikátne. Jednoducho sa ale dá upraviť algoritmus tak, aby pre obrázky generoval unikátne cesty.

Sťahovanie prebieha na pozadí vo vlastnom vlákne, pričom obrázky sa sťahujú z fronty.

4.1.3 Implementácia

Riešenie pozostáva zo štyroch tried FileHelper, MemoryCache, ImageDownloader a ImageLoader, ktoré sa nachádzajú v balíku `com.move2play.imagedownloader` v projekte Move2Play.

FileHelper Pomocná trieda pre prácu so súbormi. Poskytuje metódy ako otvorenie, vymazanie alebo vytvorenie súboru.

MemoryCache Cache pre obrázky, aby sa nemuseli načítavať zakaždým z SD karty.

ImageDownloader Zabezpečuje stiahnutie súboru, jeho úpravu (scaling) na požadovanú veľkosť a uloženie.

ImageLoader Poskytuje rozhranie pre prácu s ImageDownloaderom a MemoryCache.

4.1.4 Použitie

Najskôr treba sťahovač obrázkov inicializovať.

```
FileHelper mFileHelper = new FileHelper(getApplicationContext()
                         .getExternalFilesDir(null).getAbsolutePath(),
                         getApplicationContext().getFilesDir().getAbsolutePath(),
                         "images");
mImageLoader = new ImageLoader(mFileHelper, new ImageDownloader(mFileHelper), this);
```

Obrázok stiahneme nasledovne, kde prvý parameter je URL obrázka a druhý je ImageView, v ktorom ho chceme zobraziť.

```
mImageLoader.displayImage(url, (ImageView) convertView.findViewById(R.id.image1));
```

Po ukončení práce treba sťahovač ukončiť.

```
mImageLoader.stopThread();
```

4.2 Leaderboard List funcionalita

#3362, Zodpovedná osoba: Pavol Bielik

4.2.1 Úloha

Vytvorenie listu, v ktorom sa bude zobrazovať pouzívatel a jeho kamaráti. Riadok používatela bude prilepený na spodnej alebo hornej hrane listu, podla toho ako bude posunuty. Ak budú zobrazení kamaráti, ktorý sú lepší ako on, bude na spodnej hrane. Ak bude zobrazení kamaráti, ktorý sú horší ako on, bude na hornej hrane listu.

4.2.2 Návrh a Implementácia

Pri implementácii Leaderboard listu vychádzame z History listu, kde funguje zachytenie sekcii na hornej hrane obrazovku. List sa môže nachádzať v troch stavoch:

- PINNED_ITEM_UP - list je nastavený tak, že sa zobrazujú ludia s horším skóre ako používateľ. Používateľ je vtedy zobrazený na hornej hrane.
- PINNED_ITEM_DOWN - list je nastavený tak, že sa zobrazujú ludia s lepším skóre ako používateľ. Používateľ je vtedy zobrazený na dolnej hrane.
- PINNED_ITEM_GONE - tento stav nastáva vtedy, ak je list nastavený na používateľa a teda nie je potrebné ho dodatočne zobraziť.

Listu vieme nastaviť header a footer pomocou metód setPinnedHeaderView(View view), setPinnedFooterView(View view) alebo obidva naraz setPinnedViews(View header, View footer).

Pri použitý LeaderboardAdapter sa vyžaduje implementácia dvoch metód:

```
public int getPinnedItemPosition();
public void configurePinnedHeader(View header);
```

Prvá metóda musí vrátiť pozíciu používateľa v liste, ktorá sa používa pri výpočte stavu listu. Druhá metóda slúži na konfiguráciu pre header a footer, v prípade, že majú byť zobrazené.

4.3 Finálne zobrazenie aktivity na mape

#3159, Zodpovedná osoba: Peter Krátky

4.3.1 Úloha

Návrh a implementácia zobrazenia súradníc na mape tak, aby bolo jednoducho rozlíšiteľné, čo bola aktivita chôdza a čo bola odfiltrovaná aktivita.

4.3.2 Návrh

Súradnice sú zobrazené dvoma farbami - modrá pre validné súradnice (chôdza) a šedá pre odfiltrované súradnice.
Pre chôdzu nie je vhodné rozlišovať rýchlosť farbami, pretože chôdza má väčšinou charakter rýchlosti, ktorá je okolo 1,5 ms-1. V neskoršej verzií by to malo zmysel pre zobrazenie cyklotrasy.

4.3.3 Implementácia

Vrstvy, ktorá sa zobrazuje na mape a zabezpečujú zobrazenie aktivity sú dve:

- `WalkingMapsOverlay` - zobrazuje validné súradnice (chôdza)
- `FilteredMapsOverlay` - zobrazuje odfiltrované súradnice

`WalkingMapsOverlay` zobrazuje tie súradnice, ktoré nemajú flag `IN_VEHICLE` a `LOW_SIGNAL`

`FilteredMapsOverlay` zobrazuje tie súradnice, ktoré majú flag `IN_VEHICLE`, ale nemajú `LOW_SIGNAL`

Pre debugovacie účely je v kontextovom menu voľba “Origin”, ktorá zobrazí pôvodné surové súradnice.

4.4 Počítanie skóre a energie

#3191, Zodpovedná osoba: Peter Krátky

4.4.1 Úloha

Dokončenie prepočítavania na skóre a spálené kalórie priamo počas vykonávania a uloženie do tabuľky Reports.

4.4.2 Analýza

K dispozícií je obšírna tabuľka s hodnotami METS pre jednotlivé športy. Pre niektoré športy sú dokonca rozlíšené hodnoty aj podľa rýchlosti.
Na výpočet energie je možné použiť jednoduchý vzorec (((METS·3.5·hmotnosť)/200)·(trvanie v minútach)).

4.4.3 Návrh

Energia sa počíta na základe vyššie uvedeného oficálneho vzorca. Je potrebné vypočítať aj skóre. To je momentálne počítané ako METS · trvanie v sekundách. Toto treba prispôsobiť koeficientom.

4.4.4 Implementácia

Prepočítavanie zabezpečuje trieda `ActivityDataCalculator` a `SportsHelper` obsahuje hodnoty METS pre jednotlivé športy.

Tabuľka Reports obsahuje stĺpce Score a Energy, ktoré uchovávajú vypočítané hodnoty

Skóre je potrebné vypočítať aj po ručnom pridaní aktivity, a tiež sa prepočítava aj pre aktivity ktoré boli upravované na mape (edit path)

4.5 Návrh vzhľadu obrazovky Add Activity

#3359, Zodpovedná osoba: Peter Krátky

4.5.1 Úloha

Cieľom tejto úlohy je navrhnúť grafický vzhľad obrazovky Pridanie novej aktivity. Vychádza z implementovanej obrazovky so štandardným štýlom komponentov. Úlohou je navrhnúť rozloženie komponentov a ich vzhľad.

4.5.2 Návrh

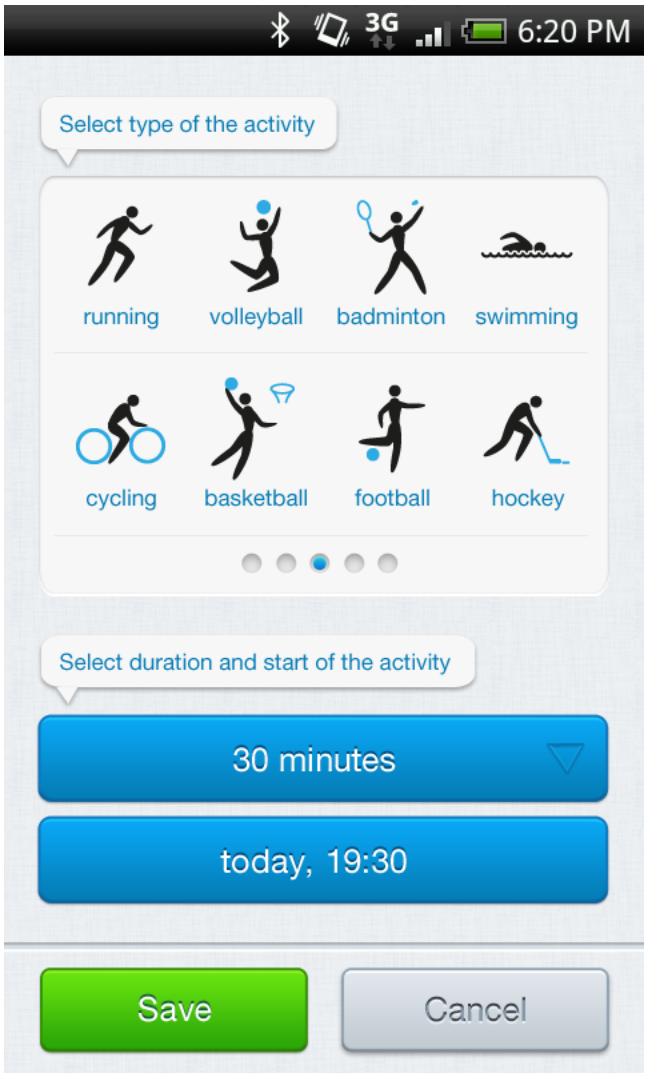
Základné prvky obrazovky by mali byť

- voľba typu aktivity
- voľba dĺžky trvania aktivity
- dátum aktivity

Návrh štýlovania a rozmiestnenia komponentov je možné vidieť na obrázku.

Ako zoznam aktívit sú ikonky aktivít v bullet prepínači. Pri použití obyčajného zoznamu (list box), by sa málo sa využila šírka obrazovky a bolo by potrebne veľa scrollovať. Konkrétne vnorený scroll view by bol nepoužiteľny.

Na začiatku by boli najčastejšie používané aktivity. Posledná ikonka by mala byť voľba “other”, kde by si používateľ vybral ľubovoľnú inú aktivitu. Keď bude väčšie množstvo aktivít, malo by tam byť vyhľadávanie.



Obr. 8: Grafický návrh obrazovky Add Activity

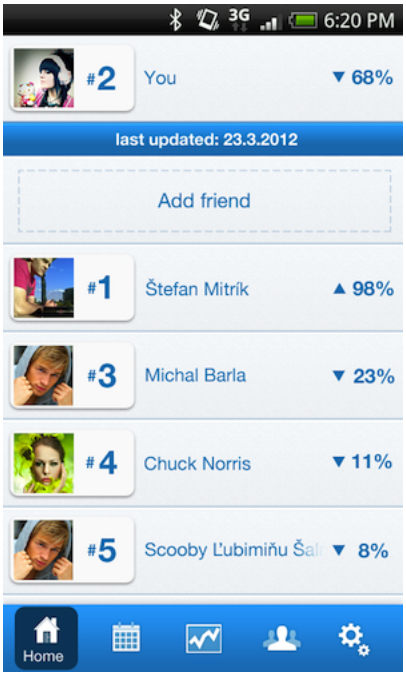
4.6 Friends - pridávanie, potvrdzovanie, úprava grafiky

4.6.1 Úloha

Cieľom tejto úlohy bolo navrhnúť a implementovať vzhľad obrazoviek pre posielanie žiadostí o priateľstvo prípadne pozvánok pre používanie aplikácie. Okrem toho navrhnúť grafické rozhranie pre potvrdzovanie žiadostí.

4.6.2 Návrh

Dospeli sme k záveru, že tlačidlo "Add Friends" podľa pôvodného návrhu zaberá na obrazovke cenné miesto a bude používané relatívne často. Preto sme sa rozhodli že tlačidlo pridáme ako prvý záznam v liste podobne ako je to na obrazovke History. Okrem toho sme sa zhodli natom, že povodný návrh bol až príliš farebný, preto bola grafika upravená podľa novej témy.



Obr. 9: Nový návrh obrazovky friends

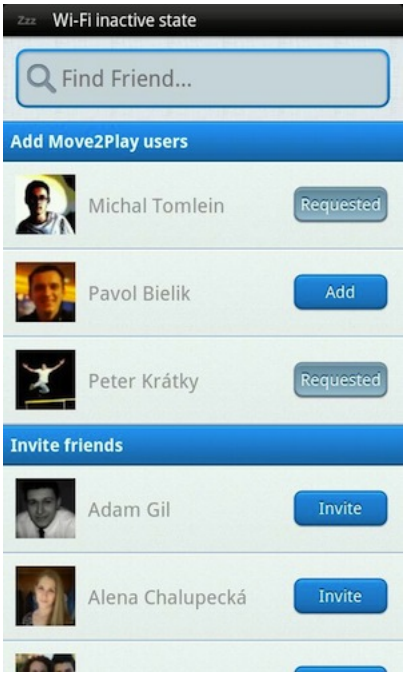
Po stlačení tlačidla "Add Friends" je používateľ presmerovaný na obrazovku kde môže poslať žiadosť o priateľstvo v rámci Move2Play používateľom ktorí už používajú našu aplikáciu. Okrem toho môže pozvať do služby M2P svojich priateľov z Facebooku prípadne kontaktov.

Používateľovi je umožnené vyhľadávanie, ktoré filtruje záznamy v liste.

Pre potvrdzovanie žiadostí sme využili rovnaký mechanizmus, aký bol navrhnutý na prezeranie noviniek a správ na Home screene. Po doručení žiadosti sa z hornej časti obrazovky vysunie notifikácia, po klepnutí na ňu sa zobrazí pop-up kde používateľ môže potvrdiť respektíve zamietnuť žiadosť.

4.6.3 Implementácia a testovanie

Implementácia obrazovky friends bola oproti návrhu mierne upravená. Používateľ aplikácie nie je umiestnený nad listom ale v liste. Pričom pri skrolovaní sa v závislosti od smeru skrolovania prilepí na vrchnú resp. spodnú časť listu. Ostatné obrazovky boli implementované podľa návrhu.



Obr. 10: Nový návrh obrazovky friends

4.7 Friend requests

#3254, Zodpovedná osoba: Michal Tomlein

4.7.1 Úloha

Cieľom tejto úlohy je zmeniť pôvodné správanie aplikácie po kliknutí na tlačidlo Add v zozname kamarátov v AddFriendsActivity. To bolo kvôli snahe o zjednodušenie tohto procesu také, že záznam v tabuľke friendships sa vytvoril priamo v stave `APPROVED`.

Nové správanie má byť také, že sa tieto záznamy budú vytvárať v stave `PENDING` a zobrazia sa cieľovému používateľovi v podobe friend requestov, ktoré bude mať možnosť potvrdiť alebo zamietnuť.

4.7.2 Návrh

Je potrebné zabezpečiť práva na zmenu stavu záznamov v tabuľke friendships tak, aby ich zo stavu `PENDING` do stavu `APPROVED` mohol zmeniť len cieľový používateľ. Zvyšné prechody v rámci stavov môžu realizovať obaja kamaráti, nie však niekto iný.

Po pridaní kamaráta a vytvorení záznamu v tabuľke friendships v stave `PENDING` je potrebné spustiť synchronizáciu, aby sa záznam dostal čo najskôr na server.

Nad tabuľkou friendships na strane klienta je potrebné nastaviť observer, ktorý bude sledovať, či nepribudli nové requesty. Ich počet sa zobrazí v notifikácii na obrazovke Friends, kde kliknutím na Show bude možné v otvorenom dialógu requesty akceptovať alebo zamietnuť kliknutím na jedno z dvoch tlačidiel. Potom je znova potrebné spustiť synchronizáciu.

4.7.3 Implementácia a testovanie

Úloha bola implementovaná podľa návrhu.

Synchronizácia sa zatiaľ spúšťa štandardným spôsobom pomocou dostupného API, nie je preto zatiaľ obmedzená na synchronizáciu tabuliek friendships a users, čo by ju mohlo výrazne zrýchliť.

4.8 Fotografie kamarátov

#3251, Zodpovedná osoba: Michal Tomlein

4.8.1 Úloha

Cieľom tejto úlohy je zobraziť na obrazovkách FriendsActivity a AddFriendsActivity profilové fotografie kamarátov z Facebooku.

4.8.2 Návrh

Na zobrazenie obrázkov sa použije lazy loading a cacheovanie obrázkov implementované v rámci inej úlohy.

4.8.3 Implementácia a testovanie

Keďže všetky adresy profilových obrázkov na Facebooku sa končia na `/picture`, bolo potrebné upraviť triedy MemoryCache, ImageDownloader a ImageLoader tak, aby namiesto mena súboru získaného z adresy ukladali stiahnuté obrázky pod hashom tejto adresy. Zatiaľ sa na tento účel použila metóda `hashCode()` triedy String.

5 Piešťany

- Naplnenie tabuľky DailySummary — Pavol Bielik
- Android akceptačné UI testy — Pavol Bielik
- Friends - pridávanie, potvrdzovanie, úprava grafiky — Štefan Mitrík

5.1 Naplnenie tabuľky DailySummary

5.1.1 Úloha

Prepočítavanie záznamov v tabuľke DailySummary aby sa mohli použit pre porovnávanie kamarátov a ich dosiahnutého výsledku.

5.1.2 Návrh a Implementácia

Tabuľka DailySummary sumarizuje všetky vykonanené aktivity používateľa pre každý deň do jedného záznamu. Schéma tabuľky na servery je nasledovná:

```
# Table name: daily_summaries
#
#  id          :integer         not null, primary key
#  user_id     :integer
#  date        :integer(8)
#  time_active :integer(8)
#  points      :integer
#  recommended :integer
```

Prepočítať tabulku je nutné zakaždím, keď vykoná zmena v tabulke Reports. Prepočítanie sa vykonáva pri každom pridaní, zmene a vymazaní záznamu v tabulke Reports, ale vždy iba pre deň, ktorého sa tento záznam týkal. Implementácia je prostredníctvom ActivityProvider, konkrétne v metódach insert, update a delete. Pre takéto riešenie sme sa rozhodli, kedže to priamo transakciamy nedokázeme vykonať (potrebné tabuľky z ktorých sa počíta DailySummary sa nachádzajú v iných databázach). Pri synchronizácii sa prepočítajú dátumy pre všetky tabulky znova.

5.2 Android akceptačné UI testy

#3487, Zodpovedná osoba: Pavol Bielik

5.2.1 Úloha

Integrovať testovaci nástroj, ktorý umožnuje automatizovane testovať rozhranie Android aplikácie.

5.2.2 Návrh

Ako testovací framework sme zvolili Robotium, ktorý tvorý nadstavbu nad štandardných spôsobom testovania aktivit v Androide. Framework ponúka základnú množinu metód ktoré umožnujú klikať na jednotlivé elementy podla ich indexu v layote alebo priamo na text. Bohuzial nie je podporované klikanie na základe ID komponentu. Ukážka štandardných príkazov, ktoré sa dajú použiť:

```
solo.clickOnText("History");
solo.clickInList(3);
solo.clickOnScreen(50, 200);
solo.clickLongOnText("Walking");
solo.assertCurrentActivity("Should be in AddReportActivity", AddReportActivity.class);
```

5.2.3 Implementácia

Akceptačnými UI testami sme pokryli dve obrazoky - HistoryActivity a EditListMap. Kedže každá pracuje nad databázou, pred každým testom sa nahrá testovacia databáza namiesto pôvodnej, a na konci testy sa obnový pôvodná. Celkovo sme vytvorili cez 30 testov, ktoré ale bežia značne pomaly (okolo 10 minút) a teda bude v budúcnosti potrebné ich nepúštať pri každej zmene v repozitáry ale iba raz za deň.

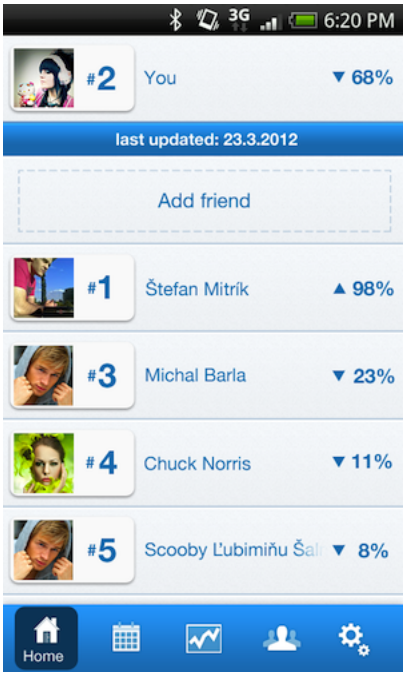
5.3 Friends - pridávanie, potvrdzovanie, úprava grafiky

5.3.1 Úloha

Cieľom tejto úlohy bolo navrhnúť a implementovať vzhľad obrazoviek pre posielanie žiadostí o priateľstvo prípadne pozvánok pre používanie aplikácie. Okrem toho navrhnúť grafické rozhranie pre potvrdzovanie žiadostí.

5.3.2 Návrh

Dospeli sme k záveru, že tlačidlo "Add Friends" podľa pôvodného návrhu zaberá na obrazovke cenné miesto a bude používané relatívne často. Preto sme sa rozhodli že tlačidlo pridáme ako prvý záznam v liste podobne ako je to na obrazovke History. Okrem toho sme sa zhodli natom, že povodný návrh bol až príliš farebný, preto bola grafika upravená podľa novej témy.



Obr. 11: Nový návrh obrazovky friends

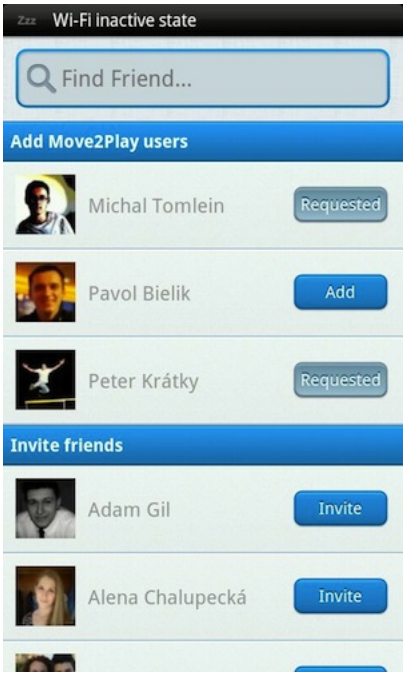
Po stlačení tlačidla "Add Friends" je používateľ presmerovaný na obrazovku kde môže poslať žiadosť o priateľstvo v rámci Move2Play používateľom ktorí už používajú našu aplikáciu. Okrem toho môže pozvať do služby M2P svojich priateľov z Facebooku prípadne kontaktov.

Používateľovi je umožnené vyhľadávanie, ktoré filtruje záznamy v liste.

Pre potvrdzovanie žiadostí sme využili rovnaký mechanizmus, aký bol navrhnutý na prezeranie noviniek a správ na Home screene. Po doručení žiadosti sa z hornej časti obrazovky vysunie notifikácia, po klepnutí na ňu sa zobrazí pop-up kde používateľ môže potvrdiť respektíve zamietnuť žiadosť.

5.3.3 Implementácia a testovanie

Implementácia obrazovky friends bola oproti návrhu mierne upravená. Používateľ aplikácie nie je umiestnený nad listom ale v liste. Pričom pri skrolovaní sa v závislosti od smeru skrolovania prilepí na vrchnú resp. spodnú časť listu. Ostatné obrazovky boli implementované podľa návrhu.



Obr. 12: Nový návrh obrazovky friends