



Универзитет Св. Кирил и Методиј - Скопје
Факултет за информатички науки и компјутерско
инженерство



Дипломска работа

Статистика на повици и одбирање на најдобар тарифен модел (Call Statistics)

Ментор:

Доц. д-р Игор Мишковски

Изработил:

Огнен Костовски 438/2009

Скопје, јули 2013

Содржина

1. Вовед	4
1.1 Краток опис	4
1.2 Чекори за градење на успешна апликација	5
1.2.1 Чекор 1: “Најдобра во” изјава	6
1.2.2 Чекор 2: Избери ги правилните сценарија	6
1.2.3 Чекор 3: Целен пазар и истражување на пазарот	6
1.2.4 Чекор 4: Итеративни подобрувања или пробивни иновации	6
1.3 Отворени податоци	7
2. Психолошки аспект	8
2.1 Маркетинг на апликација	8
2.1.1 Икона	8
2.1.2 Наслов на апликацијата	9
2.1.3 Опис	9
2.1.4 Screenshot	9
2.1.5 Клучни зборови	9
2.1.6 Категории	10
2.2 Маркетинг стратегија	10
2.2.1 Атрактивност на “Call Statistics”	12
2.2.2 Задржување на корисниците	12
2.2.3 Колективна интелигенција	13
3.1 Екран 1: Регистрација	16
3.1.1 Дизајн	17
3.1.2 Функции	18
3.2 Екран 2: Историја на повици	23
3.2.1 Дизајн	24
3.2.2 Функции	24
3.3 Екран 3: Статистика	26
3.3.1 Дизајн	27
3.3.2 Функции	28
3.4 Сервиси	32
3.4.1 Parse база на податоци	33
3.4.2 Parse сервиси	33
3.4.3 АЕК сервис “Провери број”	34

3.4.4 onHandleIntent(Intent intent)	37
3.4.5 calculateBill(callLog, tariffStats)	38
3.4.6 calculateNotifications(callLog, tariffStats)	39
3.4.7 Асинхрон начин на работа.....	39
3.5 Widget.....	41
3.6 Навигација.....	42
4. Можни подобрувања.....	43
4.1 Дизајнерски подобрувања	43
4.2 Функционални подобрувања	44
5. Заклучок	45
6. Листа на референци	46

1. Вовед

1.1 Краток опис

Како што кажува насловот на апликацијата “Call Statistics” првичната идеја за оваа андроид апликација беше прикажување на статистика базирана на историјата од направени повици, која ќе им даде на корисниците јасна слика за телефонските трошкови кои би ги имале доколку биле на некоја друга тарифа. Секако посакуваниот резултат од оваа идеја е да постои и можност за споредба со моменталната тарифа на која е корисникот, се со цел да може да одлучи кој оператор и која тарифа е за него најпогодна. Големiot број на можни тарифи на македонскиот пазар го прави изборот на најсоодветна тарифа многу комплициран и ги принудува луѓето да го направат својот избор базирано на месечните трошкови или услугите кои ги нуди операторот со соодветната тарифа. Постојана промена во цените на тарифите и услугите кои ги нудат уште повеќе го усложнува овој процес. Како последица на изборот на несоодветна тарифа, многу од корисниците имаат многу поголеми трошкови (доколку нивната одлука била базирана на услугите кои ги нуди тарифата), или нивните услуги се ограничени (доколку нивната одлука била базирана на месечните трошкови). Недоволната информираност за воведување на нови тарифи, како и подобрувања на постојните може да доведе до тоа да корисникот не е информиран дека постои подобра тарифа за него.

Комплексноста на овој процес ја зголемува “цената” или подобро кажано корисноста на оваа апликација. Замислете апликација која ќе нуди мониторирање на се што е поврзано со тарифниот модел кој го користите со дополнителна можност да споредите со останатите тарифи и се тоа базирано на вашите потреби. Без разлика дали потребите на корисникот се основани на месечните трошкови или услугите кои ги нуди тарифата, или како што е во најголем број од случаите комбинација од двете. Мое мислење е дека ваква апликација ќе биде многу примамлива за сите корисници на македонски тарифни модели . Воедно тоа е и целта на оваа апликација, да ги енкапсулира најбараните и најкорисните тарифни услуги. Сакам да напоменам дека ова е голем проект кој моментално е сеуште во развивање, што значи дека не сакам да ги зголемам вашите очекувања до степен да мислите дека веднаш ќе може да ја користите апликацијата. Целта ми е низ моето гледиште да ги разберете услугите кои се испланирани за апликацијата, како и да се уверите за поголемиот дел од нив кои се веќе остварени.

- Една од услугите која е испланирана и остварена за оваа апликација е мониторирање на бројот на преостанати бесплатни SMS пораки, како и бројот на преостанати бесплатни минути кон сопствениот оператор и бесплатните минути кон сите (или останатите) мрежи. Оваа услуга е од големо значење за разумно трошење на бесплатните услуги кои доаѓаат со тарифниот модел.

- Исто така друга корисна услуга која е испланирана е мониторирање на преостанатиот бесплатен податочен сообраќај. За оваа услуга моментално нема точно дефинирано решение на андроид платформата, моменталните можни решенија се зависни од мобилниот оператор и без точно дефинирани услови. За решение на овие проблеми потребно е да се потроши

повеќе време и труд, како и повеќе информации од андроид платформата, па останува да се воведат како можно подобрување на апликацијата во блиска иднина.

- За претходно наведените услуги може да се наведе дека би било корисно доколку се воведат нотификации доколку корисникот е во близина на лимитот за бесплатни услуги дефиниран од тарифниот модел. Бидејќи ова е мало подобрување на услугите кои веќе постојат и е базирано повеќе на дизајн на апликацијата одколку функционален аспект, останува да се додаде за време на финалната верзија на апликацијата.

- Прикажување на историјата на повици заедно со точниот оператор на кој припаѓа бројот со кој била воспоставена комуникацијата. Како и проверка на операторот на број внесен преку корисничкиот интерфејс.

- И секако прикажување на статистика базирана на историјата од направени повици и можноста за споредба со моменталната тарифа на корисникот, за која веќе зборувавме.

Сакам да напоменам дека сите овие услуги, како и останатите предлози кои ќе ги добијам од корисниците на бета верзиите на апликацијата ќе ги разгледам и ќе се обидам да ги остварам/подобрам, се со цел да се енкапсулираат најбараните услуги на корисниците на македонските тарифни модели.

Горенаведените услуги се само воведување во проектот, за да може да се добие општа слика за можностите на апликацијата. Тие ќе бидат детално објаснети од психолошки и технички аспект понатаму во документацијата.

1.2 Чекори за градење на успешна апликација

За време на студирањето голем дел од професорите по предметите за софтверско инженерство на своите воведни часови се трудеа да ни ја објаснат важноста на темата “Чекори за градење на успешна апликација”. Главната цел на овие предавања беше да се допре до свеста на студентите и да се објасни дека секоја идеја за нова апликација треба детално да се анализира пред да се одлучи дали таа има шанса да биде успешна. Оваа тема иако е само информативна и нема влијание врз техничкиот аспект на апликацијата, според мене е основа која во никој случај не смее да се прескокне. Главна причина за тоа е времето кое би се потрошило при оваа анализа е многу пократко од времето кое би се потрошило за неуспешна апликација. Секако не е гаранција после овие чекори за анализа на идејата дека апликацијата ќе биде успешна, но колку поголемо искуство има креаторот на апликацијата, толку поточно може да заклучи какви се шансите на апликацијата да успее. Умешноста и искуството играат голема улога во оваа тема и драстично ја менуваат корисноста и “цената” на програмерот. Затоа секој програмер треба да се фокусира да ги прошири своите знаења на оваа тема за да ја зголеми својата вредност.

1.2.1 Чекор 1: “Најдобра во” изјава

Изјава: “Оваа апликација е најдобра во ... “

Ова треба да е специфична изјава во една реченица.

1. “Оваа апликација е најдобра за одбирање на најсоодветен тарифен модел.”
2. “Оваа апликација е најдобра за мониторирање на бесплатните услуги на тарифниот модел.”

1.2.2 Чекор 2: Избери ги правилните сценарија

Избраните сценарија мора да се поврзани со “Најдобра во” изјавата.

Сценарија за изјавата за најсоодветен тарифен модел:

- Најсоодветен тарифен модел базирано на месечните трошкови.
- Најсоодветен тарифен модел базирано на услугите на тарифниот модел.
- Најсоодветен модел базирано на месечните трошкови и услугите на тарифниот модел.

Сценарија за изјавата за мониторирање на бесплатните услуги:

- Мониторирање на преостанатите бесплатни SMS пораки.
- Мониторирање на преостанатите бесплатни минути кон сопствениот оператор.
- Мониторирање на преостанатите бесплатни минути кон сите/останатите оператори.

1.2.3 Чекор 3: Целен пазар и истражување на пазарот

Како целен пазар за мојата апликација претставува македонскиот пазар. Тоа се основа на веб сервисите за проверка на оператор кои работат само за македонски телефонски броеви. По истражување на македонскиот пазар не наидов на апликации кои даваат слични услуги на оваа апликација. Секако постојат апликации кои мониторираат повици и SMS пораки но тие не се базирани на македонските тарифи, па гледано од тој аспект би било подобрувања на нив.

1.2.4 Чекор 4: Итеративни подобрувања или пробивни иновации

Од гледна точка на македонскиот пазар, оваа апликација со сите услуги кои што ги нуди претставува пробивна иновација, бидејќи ги нуди едни од најбараните услуги за македонските тарифи. Особено основната идеја за одбирање на најсоодветен тарифен модел. Додека гледано од глобален аспект, иако постојат слични услуги одделно понудени, нивните резултати не се базирани на македонските мобилни тарифи. Следствено на тоа оваа апликација би претставувала итеративно подобрување на нивните услуги.

Од горенаведените чекори може да се забележи дека оваа апликација поддржува две “Најдобра во..” изјави што секако значително ги подобрува шансите за успех на целиот пазар. Уште поинтересен е различниот и компатибилен психолошки аспект кои тие го имаат кој ќе биде детално објаснет во 2. Психолошки аспект понатаму во документацијата.

1.3 Отворени податоци

Иако податоците кои ги користи “Call Statistics” апликацијата се приватни, а тоа е историјата на воспоставени повици, нивната природа и начинот на обработка е базирана на начинот на обработка на јавни податоци. Со јавните податоци за време на студирањето се запознав на предметот “Веб базирани системи”. На тие предавања ја сфатив важноста на јавните податоци и неверојатните можности и поволности кои тие ги нудат. Тоа беше клучниот момент кога се одлучив да го работам овој проект за мојата дипломска работа и во продолжение ќе се потрудам да објаснам што всушност претставуваат јавните податоци, кој ги воведува и кои се нивните поволности.

Под јавни податоци се подразбираат објективни, неперсонални податоци базирани на факти, врз база на кои се одвиваат и евалуираат јавни услуги и врз база на кои се донесуваат одлуки за политики. Јавни податоци се и оние кои се собрани, односно генерирани при извршувањето на јавните услуги.

Идејата на владата е да ги објават своите јавно достапни информации во отворен формат кој може да биде прочитан од компјутери. Ова претставува нов, модерен пристап кон идеите за е-Влада. Овој пристап влијае на зголемување на транспарентноста на Владите и нивните податоци. Се овозможува креирање групи на податочни корисници и развивачи на софтвер и апликации (десктоп, веб и мобилни). Се искористуваат податоците и им се дава поголема употреблива вредност на самите податоци од јавен карактер.

Главната идеја зад отворањето на јавните владини податоци во формат кој може да биде прочитан од компјутери/машини е креирање на кориснички апликации за мобилни уреди, како и десктоп и веб апликации кои ги искористуваат овие податоци за да на крајните корисници (јавноста) им понудат обработени информации кои се од нивен директен интерес. Преку ваквите апликации, отворените владини податоци добиваат на вредност, поради тоа што нивната употребливост е директно зголемена.

Со отворањето на податоците од јавен карактер во формат кој е товорен и лесно достапен како за луѓето така и за машините, се зголемува интеракцијата на Владата со граѓаните, што води кон развој на е-Демократија. Ваквиот пристап влијае на зголемување на транспарентноста на Владите и нивните податоци, овозможувајќи креирање на разни апликации кои ги интерпретираат податоците, им нудат обработени информации на корисниците (граѓаните) и им даваат поголема употреблива вредност на самите податоци од јавен карактер.

2. Психолошки аспект

Како во секој бизнис, вашиот успех е директно поврзан со вашето познавање на пазарот. App Store-от претставува пазар за апликацискиот бизнис, затоа со цел да се разбере пазарот, мора да се проучи App Store-от. Чекорите за градење на успешна апликација претставуваат само вовед во оваа голема тема. Најдобар начин за да се проучи пазарот е да се мониторираат моментално најдобрите апликации на пазарот. Ова ги вклучува најдобрите платени, најдобрите бесплатни и апликациите со најдобра заработка. Нивното следење ќе го подобри вашето познавање на маркетингот и ќе ве запознае со успешен дизајн, маркетинг и различни модели за наплата. За време на овој процес мора да се фокусирате да дознаете за што се луѓето заинтересирани и типот на апликациите кои ги симнуваат од пазарот, со цел да ја изградите вашата апликација базирано на тоа знаење.

Бидејќи целиот пазар на оваа апликација е многу помал, проучувањето на пазарот е од мала корист за да се дојде до посакуваната информација за типот на услугите за кои се заинтересирани луѓето. Со оглед на тоа потребна е стратегија од друг тип која ефективно ќе успее да ги привлече корисниците да ја симнат апликацијата, а потоа и да одлучат дека таа е корисна за нив и да ја остават на нивните уреди. Ова се двата клучни моменти за успешноста на една апликација. Оваа тема ќе биде детално објаснета во 2.2 Маркетинг стратегија понатаму во документацијата.

2.1 Маркетинг на апликација

Пазарот денес е исполнет со илјадници одлични апликации, но повеќето девелопери не се квалификувани кога станува збор за маркетинг. Во меѓувреме, многу лошо дизајнирани апликации се високо рангирани бидејќи нивните девелопери ја сфатиле играта на маркетинг.

За да ефективно се рекламира една апликација треба да се фокусирате на неколку клучни полиња, што ќе им дозволат на идните корисници да ја откријат и симнат. Разбирањето дека базичните елементи на апликацијата се можности за маркетинг е од суштинско знаење за да се стане успешен во апликацискиот бизнис. Задачата на девелоперот е да креира уникатни елементи од иконата до “Симни” копчето. Во продолжение подетално ќе ги разгледаме овие компоненти, кои може да се менуваат во било кое време на пазарот.

2.1.1 Икона

Првата работа која корисниците ќе ја забележат за време на разгледувањето на апликацијата е нејзината икона. Тоа е исто сликата која корисниците ќе ја гледаат на нивните телефони одкако ќе ја инсталираат апликацијата. Иконата е важна бидејќи преку нејзе корисниците ќе ја идентификуваат апликацијата. Треба да изгледа убаво, а воедно и да успее да ја долови суштината на апликацијата, да го привлече вниманието на корисникот и да го принуди да ја испита дополнително. Многу девелопери како дополнување ја креираат иконата и го фокусираат целиот нивен труд на самата апликација, но иконата е првата импресија која ќе ја направите врз корисникот. Изреката “ Не се добива втора шанса за да се направи прва

импресија “ се однесува и тука. Треба да се потрудите да направите квалитетна икона која ја претставува вашата апликација и да ги натерате корисниците да веруваат дека апликацијата има вредност.

2.1.2 Наслов на апликацијата

Повеќе од 80% од пребарувањата на пазарот за апликации се поврзани со функционалноста на апликацијата, а не по нејзиното име. Затоа е критично да им помогнете на корисниците да ја најдат апликацијата кога извршуваат слични пребарувања по одреден клучен збор. Секој збор во насловот на апликацијата служи како клуч за пребарување, многу слично како клуч за пребарување кај веб пребарувачите. Може да се мисли на насловот на апликацијата како нејзино URL.

2.1.3 Опис

Да се има привлекувачки опис за апликацијата е како да се има одлична прва реченица, лугето повеќе се интересираат за истата одкако ќе го привлечат нивното внимание. Првиот дел од описот треба да ги содржи најрелевантните информации за апликацијата кои корисникот треба да ги знае. Доколку идеата на апликацијата и услугата која ја нуди се софаѓа со барањата на корисникот, ова може да е одлучувачки дел за корисникот да се одлучи да ја симне.

2.1.4 Screenshot

Screenshot-овите се одлични маркетиншки алатки бидејќи тие му даваат на корисникот визуелен ефект од апликацијата, што да очекуваат од истата. Многу луѓе кои бараат апликации нема да го прочитаат нејзиниот опис, наместо тоа ќе се упатат директно до нејзините screenshot-ови. Целта на screenshot-овите е да ја пренесат главната функционалност на апликацијата без покажување на премногу детали кои може да ги збунат корисниците. Доколку тие се преполни со информации тие имаат негативно и збунувачко влијание врз корисниците што може да доведе до тоа купувачите да имаат потешкотии да ја видат вистинската вредност на вашиот продукт. Затоа треба да се потрудите тие да бидат чисти, привлечни и информативни.

2.1.5 Клучни зборови

За разлика од иконата и насловот, клучните зборови не се видливи за корисниците. Со прикачување на апликацијата на апликацискиот пазар, дозволено е да се наведат клучни зборови релевантни на вашата апликација. Кога корисниците пребаруваат за еден од изразите кои ги имате наведено, вашата апликација се појавува во резултатите од пребарувањето. Промената на клучните зборови може да настане и одкако ќе ја прикачите на апликацискиот пазар. Тоа е од голема помош бидејќи тие се од голема важност за добар маркетинг и доколку го испитате добро пазарот може да дојдете до соодветни клучни зборови кои ќе ја подобрат успешноста на апликацијата.

2.1.6 Категории

Апликацискиот пазар ги организира апликациите во специфични категории да им овозможи на корисниците многу полесно да ги најдат бараните апликации. Како дополнување на рангирањето на апликациите, секоја категорија има сопствено рангирање, следствено доколку таргетираат корисници кои се заинтересирани во соодветни категории, овие информации се многу битни за проучување на целниот пазар. Кога ја објавувате апликацијата на пазарот потребно е да изберете категорија во која припаѓа. Доколку апликацијата припаѓа во повеќе категории, во секој момент можете да ја промените категоријата во која припаѓа апликацијата.

Исто како што вашата апликација ќе има потреба од одредени подобрувања и прочистувања заради побарувања од корисниците и конкуренцијата, така и вашиот маркетинг може да има потреба да се промени. Мозни промени од аспект на маркетингот може да се потребни врз иконата, насловот, screenshot-овите и описот. Истото важи и за категоријата во која припаѓа вашата апликација. Најбитно е постојано да се следи развивањето на пазарот и базирано на податоците и резултатите од тоа следење да се вршат соодветни промени за зголемување на популарност и успешноста на вашата апликација.

2.2 Маркетинг стратегија

Првичната идја за оваа апликација беше да се овозможи услуга која ќе им дозволи на корисниците да го најдат најсоодветниот тарифен модел базирано на нивните потреби. Како што беше објаснето во воведот ова е сложен процес, особено доколку сакате да го направите вашиот избор базирано на комбинација од месечните тарифни трошкови и услугите кои ви се потребни. Воведување на статистичка обработка на историјата на воспоставени повици со цел да се претстават резултатите во форма на граф, т.е. лесната читливост за корисникот го решава тој проблем за нив. Иако ова е одлична и многу корисна услуга, таа е примамлива само за луѓето кои моментално се заинтересирани со тој проблем. Многу е мала веројатноста, дури и со помош на добар маркетинг, да се убеди неког дека оваа апликација е корисна за него доколку тој моментално нема потреба од таа услуга. Дополнително разгледувајќи го фактот дека целниот пазар на оваа апликација е многу мал (само македонскиот пазар) претходно наведениот проблем експоненцијално се зголемува и е основа за неуспешна апликација.

Ова е моментот кога треба да се чувствувате горди со себе. Сигурно е малце збунувачки, зошто би требало да се чувствувате горди кога сте стигнале до “dead end” за вашата одлична идеја?

Доколку умеете објективно да ја евалуирате атрактивноста на апликацијата која сте ја замислиле - од очите на корисниците, и за време на таа евалуација се основате на фактите кои сте ги заклучиле за време на фазата на учење на пазарот, а не базирано на вашите мислења и чувства кон идејата, тогаш вие може да се наречете успешен девелопер.

Колку и да е тежок фактот дека вашата идеја не е доволно атрактивна во очите на корисниците, вие со самиот заклучок кој сте успеале да го донесете пред да потрошите многу

време и труд на развивање на апликацијата сте дошле до состојба да бидете повеќе ценет како девелопер.

Со почитување на овие едноставни правила, шансите да успеете да дојдете до успешна апликација се многу поголеми. Фокусирајте се на фактот дека тоа било само една идеја за која пазарот моментално не е подготвен. Можеби во иднина, за време на учење на маркетингот ќе најдете можност за таа апликација и идејата која сте ја добиле нема да биде залудна. Или можеби ќе успеете оваа идеја да ја споите со друга атрактивна идеја од истата област што би ја направила апликацијата поатрактивна во очите на корисниците.

Мој совет е во ваков момент да не донесете избрзана одлука. Искочете надвор, не мислете на идејата и оставете да поминат неколку дена да ви се средат мислите за да може да ја донесете правилната одлука.

За време на овој период од голема корист е да се консултирате со некој од вашите пријатели, без разлика дали се девелопери или не. Многу често само објаснувањето на вашиот проект или вашите проблеми може да ве инспирираат и да успеете да дојдете до некое оригинално решение. Секако земете ги во предвид и советите кои ќе ги добиете. Многу е важно да ги запишете некаде информациите кои сте ги добиле за време на разговорот. Потоа продолжете со вашиот привремен одмор.

Кога ќе се чувствувате подготвени повторно од нула да ја евалуирате идејата потсетете се на планот кој што сте го замислиле и потрудете се овој пат да ги гледате работите од друга перспектива. Погледнете ги информациите кои претходно сте ги запишале за време на консултацискиот период. Можеби ќе дојдете до решение на вашиот проблем.

Овој момент е клучен за успешноста на една апликација и доколку не се најде негово соодветно решение, најдобар избор е да се откажете од идејата. Не се откажувајте и почнете го процесот од почеток. Ова искуство ќе биде од огромно значење за вашата иднина во овој бизнис.

Не мрази – емулирај. Кога ги следите стапките на успешните апликации имате подобра шанса за успех бидејќи овие апликации се докажано барани и имаат постоечка корисничка база. Ова претставува следење на успешни примери, и базирано на тие информации се доаѓа до одлични апликациски идеи.

Не можам доволно да ја нагласам важноста на емулирање постоечки апликации. Лесно е луѓето да се заљубат во сопствената идеја, дури и ако маркетингот не покажува апетит за нејзе. Ова е едена од најскапите грешки кои можете да ги направите.

За жал девелоперите ја прават оваа грешка постојано. Тие се фокусираат на генерирање на оригинални идеи и трошат многу време и труд на креирање на таквите апликации. Кога тие нема да успеат, тие преминуваат на следната нетестирана идеја, наместо да го проучат пазарот. Често тие го повторуваат овој циклус се додека не се откажат од апликацискиот пазар. Тоа е причината зошто денес апликациските маркетинзи се преполни со корисни апликации кои иако може да содржат дури и револуционерни идеи, се непопуларни и неатрактивни за корисниците. Ова не мора да биде и твоето искуство.

2.2.1 Атрактивност на “Call Statistics”

Кога јас наидов на горенаведениот “dead end” на неатрактивност на апликација, ги следев чекорите како да го решам мојот проблем. За време на консултацискиот период добив идеја за нотификации доколку корисниците се доближуваат до лимитот на бесплатни минути. Иако идејата од старт ми се допадна, решив да ја разгледам подетално за време на повторниот евалуациски период, и тоа беше една од најдобрите одлуки кои ги имав дотогаш донесено.

За време на повторната евалуација се трудев да го гледам проблемот од друг аспект и тогаш дојдов до решението на мојот проблем. Размислувајќи за дополнителни идеи кои се поврзани со мојата апликација и идеата да воведам нотификации за лимитот на бесплатни минути, заклучив дека постојано мониторирање на сите бесплатни услуги од тарифните модели би било многу атрактивно за секој корисник на македонски тарифи. Брилијантноста беше во тоа што ова беше многу компатабилно со мојата апликација, а целосно го решаваше проблемот за атрактивноста на мојата апликација. Кој не би сакал да може да ја провери моменталната состојба на бесплатни услуги од неговиот тарифен модел и да ја искористи за сопствена контрола и придобивка? И сето тоа само со клик на едно копче – “Симни”. Со добар маркетинг оваа услуга може да ги привлече сите корисници, и одеднаш повторно максималниот број на корисници се базираше на целиот македонски пазар. Но сега мојата апликација нуди поголеми можности. Корисниците кои се заинтересирани за мониторирање на нивните бесплатни услуги ќе можат да бидат пријатно изненадени од можноста да ја најдат и нивната најсоодветна тарифа. Да ги мониторираат трошковите кои ги имале во последните неколку месеци и да се разбуди нивната совест. Замислете да се воведат можности за автоматизирано преминување на друга тарифа со клик на само едно копче. Со денешните можности за зачувување на телефонскиот број при промена на операторот оваа можност директно преминува во реална идеја. Магнитудата на оваа идеја е од големи размери бидејќи има директно влијание врз темата 2.2.3 Колективна интелигенција. Оваа тема како и идејата за 4.X Автоматизирано преминување на друга тарифа детално ќе се објасни понатаму во документацијата.

2.2.2 Задржување на корисниците

Од како ќе се одреди идеја која ќе ги задоволи вашите критериуми за примамливост на корисниците, следниот чекор за да се подобри успешноста на апликацијата е корисниците да се уверат во нејзините квалитетите и успешно да се задржат. Овој чекор многу зависи од карактерот на апликацијата. Во развивањето на игри апликации, главниот аспект на кој мора да се обрне внимание е динамичноста. Тука влегуваат нови и иновативни моменти кои корисникот не ги очекува. Кај информативните апликации кои се базирани на одредени услуги, овој аспект е поограничен. Доколку тие услуги не се менуваат често тешко е да се задржат корисниците. На пример, доколку апликацијата нуди само услуга за избор на најсоодветна тарифа, корисниците може да се запрашаат: зошто да ја чувам оваа апликација кога потреба од нова промена на тарифата најверојатно нема да имам во блиска иднина? Наша цел како девелопери е да им дадеме причина на корисниците да продолжат да ја

користат нашата апликација. Тоа ќе ја направи користеноста на нашата апликација поголема, а секако колку повеќе луѓе ја користат толку повеќе таа претставува реклама за самата себе.

Тука голема улога има постојење на сервис кој им е постојано потребен на корисниците. Тој сервис во суштина ќе им даде причина да ја задржат апликацијата на нивните машини. Секако тој сервис е од суштинско значење и потребно е на него да се обрне големо внимание. Во случајот со апликацијата “Call Statistics” таков сервис претставува мониторирањето на моменталната состојба на бесплатните услуги. Овој сервис е динамичен и неговите резултати постојано се менуваат, што ќе ги натера корисниците почесто да го користат и повеќе да ја ценат апликацијата. Неговата точност, дизајн и начин на користење се главните аспекти на кои треба да се обрне внимание. Оваа услуга треба да се усоврши до тој степен што ќе ги задржи корисниците и нема да дозволи тие да се решат да најдат апликација од сличен карактер со подобар дизајн или перформанси.

За да се “заразат” корисниците од апликацијата и не се двоумат за нејзината корисност јас решив сервисот за мониторирање на преостанати бесплатни услуги од тарифниот модел да го претставам преку “homescreen widget”. Кој би одолеал на можноста на неговиот home екран да има можност да види уште колку бесплатни услуги му преостануваат? Тоа според мене е важно на ниво на дигиталните часовници кои беа воведени на home екранот уште на првите мобилни телефонски апарати. Според мене популарноста оваа услуга може да ја надмине и популарноста на новите услуги за мониторирање на временската прогноза, но нема да одам до тој степен да се надевам дека ќе биде попопуларно од дигиталниот часовник. Иако факт е дека пред/после секој повик или порака шансите да се погледне на home екранот за да се види моменталната состојба се големи. Уште повеќе оваа популарност ќе се зголеми со воведување на мониторирање на преостанатиот бесплатен податочен сообраќај. Цените на интернет услугите кои ги нудат денес македонските оператори се многу високи, па ова драстично ќе ја зголеми свеста за преостанатиот бесплатен податочен сообраќај што ќе игра и голема улога во рекламирањето на оваа апликација. Постојат многу сценарија кога корисникот ќе има потреба да симне некоја поголема датотека, во тој момент секако дека ќе проработи свеста на корисникот и ќе сака да го провери преостанатиот бесплатен податочен сообраќај. Homescreen widget-ите претставуваат одлична реклама за секоја апликација. Тие се лесно забележливи и при користење на туѓ мобилен телефон кој ја користи оваа апликација големи се шансите да го привлечат вашето внимание.

2.2.3 Колективна интелигенција

За време на студирањето, по предметот Вештачка интелигенција се запознавме со поимот “Колективна интелигенција”. Бидејќи овој поим има директна примена во оваа апликација во продолжение ќе ја нагласам неговата важност. За таа цел прво ќе се запознаеме со неговото значење.

Колективна интелигенција се однесува кога однесувањето на група составена од единки конвергира кон некое пожелно групно однесување.

Основни принципи:

- Однесувањето на група може да реши комплексни проблеми кои ги надминуваат можностите на поединци.

- Групата се состои од одреден број на индивидуи, некои може да ја напуштат, некои да грешат, но тоа не влијае на перформансите на групата.

- Секој член на групата има сопствено однесување, може да изведува едноставни акции.

- Ниту еден член од групата не поседува знаење за глобалната состојба на групата или нејзините цели.

Предизвици кај колективната интелигенција:

- Да се најдат правила на индивидуално однесување кое може да резултира во посакувано групно однесување.

- Треба да се обезбеди условот посакуваното групно однесување да е стабилно.

Моменталните цени на тарифите кај македонските оператори се могу високи во споредба со цените низ европа. Тоа може да се примети особено при појава на нов оператор на македонскиот пазар. За сопствена реклама тие го намалуваат профитот, во споредба со останатите постоечки оператори и нудат многу подобри услови од нив што резултира со нови тарифи кои го привлекуваат вниманието на македонскиот пазар. Секако со тек на време, кога ќе соберат доволно голема база на корисници тие ги нормализираат своите цени и повторно се чувствува рамноправност на пазарот. Посакуваниот исход е да се намали профитот до минимум на постоечките оператори со цел да се добијат подобри услуги за пониски цени на тарифните модели. Тоа е многу тешко бидејќи од остварениот профит на мобилните оператори тие претставуваат лидери во македонскиот бизнис. За да тие останат лидери мора да ги држат цените доста високи и услугите кои ги нудат ограничени на сметка на Македонскиот народ.

За мене претставува големо задоволство тоа што оваа апликација има шанса да направи промена во оваа ситуација, и дополнителен мотив да ја направам оваа апликација успешна.

Од гледна точка на колективната интелигенција, во случајот на мојата апликација:

- поединците кои се членови на групата претставуваат самите корисници на апликацијата.

- нивната едноставна функција претставува избирање на најсоодветен пакет за нивите потреби.

- критичното однесување на поединците кое доведува до посакуваното групно однесување претставува нивната евентуална промена на тарифен пакет.

- групата која целиме да има посакувано групно однесување претставува множеството на сите корисници на апликацијата.

- посакуваното групно однесување претставува одредување на најсоодветен пакет за поголем дел од поединците во групата.

Зошто групното однесување претставува “Одредување на најсоодветен пакет за поголем дел од групата” и како тоа влијае врз намалувањето на цените и подобрувањето на услугите кои ги нудат операторите преку своите тарифни модели?

Одредување на најсоодветен пакет за поголем дел од групата со себе ќе повлече две големи промени:

1. Намалување на профитот на операторите базирано на бесплатни услуги.

Профитот на операторите базиран на бесплатните услуги може да се подели на две категории.

- Профит добиен од нецелосно користење на бесплатните услуги. Јасно е дека тарифните модели нудат подобри бесплатни услуги со поголема месечна претплата. Со нецелосно користење на месечната претплата, корисниците губат профит врз база на кој заработуваат операторите.

- Профит добиен од надминување на бесплатните услуги. Со надминување на границите на бесплатните услуги операторите имаат заработка основана на тарифниот модел.

Услугите кои ги нуди апликацијата директно влијаат врз овие две категории. Со бирање на најсоодветен тарифен модел, корисникот драстично ги намалува своите трошкови при што добива услуги од тарифата кои се за него прифатливи. Тоа доведува неговите трошкови да бидат блиску до рамките на неговите потреби, а со помош на услугата за мониторирање тој добива слобода да се движи во рамките на дозволените бесплатни услуги. Оваа компатибилност на услугите доведува до скоро совршен баланс кој драстично ќе ја намали непотребната потрошувачка кај корисниците, со самото тоа и профитот на операторите.

2. Будење на конкуренцијата помеѓу операторите.

Конкуренцијата помеѓу операторите одсекогаш постоела. Таа беше јасно наведена во погорниот пример при доаѓање на нов оператор на пазарот. Будењето на таа конкуренција директно влијае во подобрување на цените и услугите на тарифните пакети од што корист имаат граѓаните.

Посакуваното групно однесување за “Одредување на најсоодветен пакет за поголем дел од групата” може драстично да ја разбуди конкуренцијата помеѓу операторите. Доколку поголемиот дел од членовите на групата го променат својот моментален пакет со најсоодветниот тоа сигурно ќе биде забележано од страна на операторите. Тие ќе мора да реагираат на тоа со цел да ги задржат своите корисници и да ја искористат оваа шанса на голем број на промени да привлечат што е можно поголем број на нови корисници. Овие реакции ќе предизвикаат поголема конкуренција помеѓу операторите што ќе резултира со големи подобрувања во тарифните пакети. Секако главна корист од тоа ќе имаат нивните корисници.

Овие две промени кои може да ги предизвика посакуваното групно однесување како што беше погоре наведено, ќе доведат до големи промени на моменталната ситуација во која има неприфатливо високи цени за ограничените услуги на тарифните модели.

3. Технички аспект

Во овој дел од документацијата ќе се потрудам детално да го објаснам начинот на кој успеав да ги остварам поголемиот дел од целите кои ги испланирав за оваа апликација. Платформата на која работев е Андроид. Од моментално трите најконкурентни платформи: Apple, Android и Windows 8 базирано на можностите кои ги нудат и бројот на корисници кои се таргетираат според мене тоа беше најдобриот избор. Од гледна точка на пазарот Андроид нуди најголем број на корисници, додека Апле нуди најголем број на корисници кои се подготвени да “Симнат” апликација која не е бесплатна. Сепак за оваа апликација да ја достигне целта за колективна интелигенција, потребно е да биде бесплатна и лесно достапна за секој македонски граѓанин. Секако доколку се покаже како успешна со многу помал труд може да се направи и нејзина Апле верзија. Windows 8 апликациите се нови и модерни, но бројот на корисници и достапниот број на форуми за поддршка на програмерите е драстично помал, па ја направи најнеисплатлива варијанта.

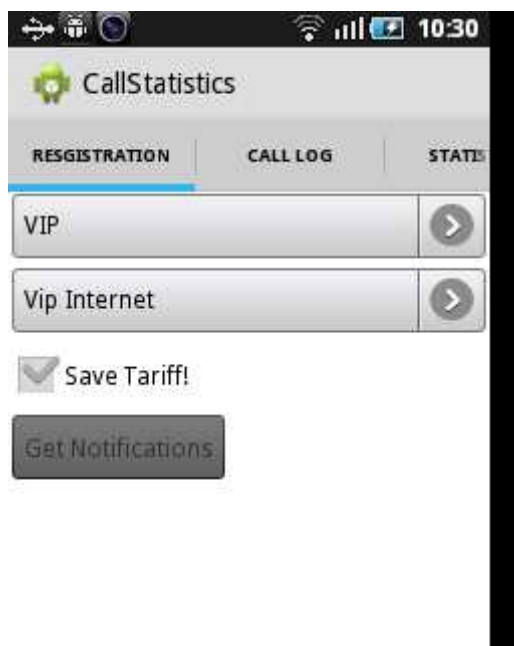
3.1 Екран 1: Регистрација

Секој тип на регистрација потребен за одредена апликација најдобро е да биде на првиот екран. За апликацијата “Call Statistics” не постои потреба од регистрација на корисниците бидејќи таа е базирана на историјата на воспоставени повици. Тие податоци се наоѓаат во меморијата на телефонскиот оператор на корисникот, што на некој начин доведува да корисничката регистрација е поврзана со податоците на меморијата во телефонот. Бидејќи многу се мали шансите друг корисник да ги префрла овие податоци за да ја користи оваа апликација на истиот апарат, не постои потреба од корисничка регистрација.

Потреба од некаков вид на регистрација се појавува за да се реши проблемот за одредување на тарифниот модел кој моментално го користи корисникот. Да се знае на кој тарифен пакет се наоѓа корисникот е од голема важност за двете главни услуги кои ги нуди апликацијата, а тоа се мониторирањето на бесплатните услуги на тарифата и одредувањето на најсоодветниот пакет.

Целта на воведување на оваа регистрација во посебен екран е да се добие добра структура на апликацијата, на принципи на кои се веќе навикнати сите интернет корисници.

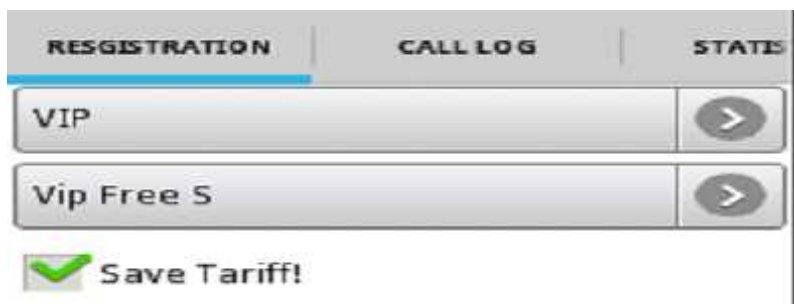
3.1.1 Дизајн



Дизајнот на овој екран не е од голема важност и комплексност. Сепак потребно е да се остави добар прв впечаток кај корисниците одкако ќе ја инсталираат вашата апликација.

Бидејќи бројот на мобилни тарифи на македонскиот пазар е многу голем, добро е да се воведат можности да се филтрираат. Постоенето на сервис кој во зависност од телефонскиот број може да го одреди неговиот мобилен оператор тука може да се искористи. Но тоа не би му дала доволна слобода на корисникот, па воведување на дополнителна селекција на операторот која би ја филтрирала селекцијата на мобилните тарифи врз основа на тој оператор според мене е од поголема вредност.

При секој тип на регистрација, воведување на можност за нејзино зачувување е од голема корист доколку потребата од повторна регистрација е честа појава. На никој корисник не би му се допаднало доколку треба да ја избере сопствената мобилна тарифа секој пат пред да сака да ја провери моменталната состојба на бесплатните услуги. Оваа услуга треба да биде воведена како природен процес за корисниците и тоа е главната цел зошто тој е претставен со помош на homescreen widget.



Рекламирањето на widget на една апликација е проблем со кој се сретнуваат голем број на девелопери, бидејќи за неговото постоење и како да се дојде до него е прилично непознат процес за корисниците кои се без девелоперско искуство. Еден начин за негова реклама претставуваат јасните инструкции наведени во некое "Help" копче од апликацијата. Според мене ова не е доволно за да го привлече вниманието на корисниците, па се потпрев на англиската изрека "What you see is what you get!" за да го привлечам поефективно вниманието на корисниците. Врз основа на таа изрека, доколку нешто им се допадне на корисниците и успее да им го привлече вниманието, тоа ќе биде доволно да ги натера да го искористат целосно. Затоа на овој екран решив да воведам копче "Провери нотификации" кое ќе им ја

прикаже содржината на бесплатните услуги кои може да се мониторираат и од widget-от. Ова подетално ќе се објасни во 3.5 Widget понатаму во документацијата.

3.1.2 Функции

За да се овозможи мониторирање на бесплатните услуги на тарифата, потребно е да се знае кои се тие бесплатни услуги. Одкако ќе се одредат бесплатните услуги за да се одреди колкав дел од нив му преостануваат на корисникот потребно е да се дознае и колку од нив се веќе потрошени.

Бидејќи бесплатните услуги кај мобилните тарифи се нудат за моменталниот месец, потребните пресметките на потрошени услуги се во временскиот период од почетокот на месецот до моментот кога е побарана услугата.

Овие пресметки се базирани на историјата на воспоставени повици, пратени SMS пораки и резултатите од мониторирањето на податочниот сообраќај.

3.1.2.1 Пресметување на моменталните телефонски трошоци, базирани на повици.

За да се добијат моменталните телефонски трошкови, потребно е прво да се добие историјата на воспоставени телефонски повици и за секој телефонски број од оваа историја да се одреди операторот. Врз резултатната листа потребно е да се извршат пресметки кои како резултат ќе ни ги дадат моменталните телефонски трошкови. Со едноставно одземање од бесплатните услуги кои ги нуди тарифниот модел и резултатите од моменталните телефонски трошкови го добиваме посакуваниот резултат, а тоа се преостанатите бесплатни услуги за корисникот. Ова се резултатите кои ќе ги прикажеме на овој екран, и тие резултати ќе обезбедат постојано мониторирање на моменталната состојба на преостанатите бесплатни услуги на корисникот.

Од причината која беше објаснета погоре дека зачувувањето на тарифниот пакет од страна на корисникот е тесно поврзана со мониторирањето на бесплатните услуги на соодветниот тарифен пакет, повикот на функцијата за пресметка на преостанатите бесплатни услуги се активира токму на притиснување на check box-от за зачувување.

По оваа акција најпрво се зачувуваат операторот и тарифниот пакет на корисникот, како и знаменцето дека постои зачувана тарифа од страна на корисникот.

Тука се повикува функција која ја изминува историјата на воспоставени повици од страна на корисникот 1. `getCallLog(Calendar.getInstance())`. Во оваа функција во посебна листа се зачувува историјата на воспоставени повици.

Потоа се повикува функција која во зависност од избраната тарифа од страна на корисникот ги добива нејзините карактеристики од каде може да ги добиеме бесплатните услуги кои ги нуди тарифата. (2. `getTariffStats(userTariff))`

Одкога ќе ги добиеме карактеристиките на соодветната тарифа се повикува функцијата `callLogUpdateOperator()`. Кај оваа функција за секој повик од листата каде ја зачувавме историјата на воспоставени повици, врз основа на телефонскиот број со кој била комуникацијата, ќе го одреди операторот кон кој бил направен повикот.

Преку резултантната листа на воспоставени повици (кон соодветните оператори) и карактеристиките на тарифниот пакет конечно можеме да ги добиеме резултатите за преостанатите бесплатни услуги за корисникот. Ова се извршува во функцијата `calculateNotifications(callLog, myTariffStats)`.

1. `getCallLog(Calendar.getInstance())`

Целта на оваа функција е да пополни листа составена од `Call` објекти која ќе претставува историја на воспоставените повици на корисникот од почетокот на месецот до моментот кога е повикана услугата. За да се добие оваа листа се пристапува до датотека која се наоѓа на меморијата на апаратот на корисникот преку `android.provider.CallLog.Calls.CONTENT_URI`. Бидејќи во оваа датотека се наоѓаат сите воспоставени повици на корисникот, потребно е воведување на филтери кои ќе ни ги дадат само излезните повици (кои ги направил корисникот) во периодот од почетокот на месецот до дадениот момент. За добивање на излезните повици се воведува ограничувањето `CallLog.Calls.TYPE = CallLog.Calls.OUTGOING_TYPE`. За да се добие првиот ден од моменталниот месец, на објектот `Calendar.getInstance()` се поставува ограничувањето `DAY_OF_MONTH = 1`. Со изминување на секој повик од датотеката ги читаме потребните информации за повикот и ги сместуваме во `Call` објекти чија класа е креирана со цел да ги имаме потребните информации за секој повик лесно достапни. Дел од овие податоци се името на контактот, неговиот број и времетраењето на разговорот. Овој објект го додаваме на `callLog` листа врз која ќе бидат направени потребните пресметки.

Бидејќи операторот кон кој бил остварен повикот е од суштинско значење за соодветните пресметки, а овој податок не е дел од оваа датотека, потребно е за секој `Call` објект од листата врз основа на бројот тој да се одреди. Ова е решено со помош на сервис и е објаснето подетално во продолжение.

```
public class Call implements Serializable{
    //http://developer.samsung.com/android/technical docs/CallLogs in Android

    private String number; //The phone number as the user entered it.
    private String type; //The type of the call (incoming, outgoing or missed).
    private long duration; //The duration of the call in seconds
    private Date date; //The date the call occurred in milliseconds since an epoch (String?)
    private String name; //The cached name associated with the phone number, if it exists.
    private String operator;

    public Call(String name, String number, Date date, long duration, int callType) {
        // TODO Auto-generated constructor stub
        this.name = name;
        this.number = number;
        this.date = date;
        this.duration = duration; // in seconds
        if(callType == CallLog.Calls.INCOMING_TYPE) type = "Incoming";
        else if(callType == CallLog.Calls.OUTGOING_TYPE) type = "Outgoing";
        else if(callType == CallLog.Calls.MISSED_TYPE) type = "Missed";
        //operator = Logs.checkOperatorAsync(number);
        operator = "";
    }
}
```

2. getTariffStats(userTariff)

Одкако корисникот ќе ја избере неговата тарифа, со помош на нејзиното име потребно е да стигнеме до нејзините карактеристики, со цел да ги добиеме бесплатните услуги кои таа ги нуди. За жал не постои сервис или база на податоци понудена од АЕК или Македонските оператори која ни ги нуди карактеристиките на Македонските тарифни модели. Една можност е со помош на веб crawler тие да се добијат од http://www.komuniciraj.mk/index.php?option=com_wrapper&view=wrapper&Itemid=115. Сепак тоа беше невозможно заради неструктурираниот формат на податоците и нивната динамичност.

Оператор	Назив	Намен	Единица на мерење	Месечна цена	Цена на позив	Времетраење на разговор	Податоци	Услуги
T-МОБИЛ	Relax Premium	резиденцијален	0.00 ден	1799.00 ден	0.00 ден			
T-МОБИЛ	Relax Surf Start	резиденцијален	0.00 ден	179.00 ден	0.00 ден			

Тука се јавува потреба од креирање на база на податоци каде тие рачно ќе се внесат. За оваа цел ги искористив услугите кои се понудени на Parse.com. Тие услуги како и parse функцијата со која пристапуваме до базата за да ги добиеме карактеристиките на тарифата на корисникот ќе бидат понатаму детално објаснети. Битно е да се знае дека овде повторно имаме податочен трансфер со кој треба да се справиме асинхронно. Одкако ќе ги добиеме карактеристиките на тарифата тие ги запишуваме во објект од класата TariffStats. Оваа класа, слично на класата Call е креирана со цел да ги имаме информациите за тарифата добро структурирани и лесно достапни.

id	name	type	description	price	unit	status	category	parent_id	created_at
1	Call	0		0	0	0	Call	0	2014-01-01 00:00:00
2	Call	0	Call	0	0	0	Call	0	2014-01-01 00:00:00
3	Call	0	Call	0	0	0	Call	0	2014-01-01 00:00:00
4	Call	0	Call	0	0	0	Call	0	2014-01-01 00:00:00
5	Call	0	Call	0	0	0	Call	0	2014-01-01 00:00:00

Одкако ги добивме потребните податоци за тарифата и претходно филтрираната листа за воспоставени повици потребно е да го повикаме веб сервисот кој ќе го одреди операторот на секој од повиците во листата. Битно е да се нагласи дека овој повик мора да биде во нишката каде го добиваме одговорот за карактеристиките на тарифата, зошто понатаму ќе се јави потреба од праќање на соодветниот TariffStats објект како аргумент на сервисот.

3. callLogUpdateOperator()

Во оваа функција потребно е врз основа на бројот за секој Call објект од резултантната листа на воспоставени повици да се одреди соодветниот оператор. За оваа цел се користи веб

сервис понуден од АЕК. За да се дојде до резултатите од овој сервис потребно е да се оствари податочна комуникација која има одредено времетраење, а оваа постапка треба да се повтори за секој број од листата. За да се овозможи корисникот да може слободно да го користи корисничкиот интерфејс на апликацијата додека таа комуницира со веб сервисот и ги добива потребните информации потребно е воведување на асинхрон начин на работа, при што овие две акции ќе се одвиваат на одделни нитки од процесот на апликацијата.

Долгото време потребно за извршување на оваа податочна комуникација и нејзиниот негативен ефект на запирање на корисничкиот интерфејс е главната причина зошто оваа процедура не се извршува директно при читањето на CallLog датотеката во функцијата `getCallLog(..)`.

Едно од можните решенија за воведување на асинхрон начин на работа е имплементирање на сервис. Останатите можни решенија и нивните карактеристики ќе бидат подетално објаснети во продолжение на документацијата.

Со извршување на соодветни пресметки врз резултантната листа на воспоставени повици и карактеристиките на тарифниот пакет конечно можеме да ги добиеме резултатите за преостанатите бесплатни услуги за корисникот.

Бидејќи повикот на сервис воспоставува податочна комуникација, за да нема потреба од одговор во формат на листа на Call објекти со променет оператор, добра програмерска практика е самите пресметки да се извршат во нишката на веб сервисот. Ова доведува големината на податоците кои треба сервисот да ги испрати како одговор драстично да се намалат, на сметка на праќање дополнителен објект од класата `TariffStats` кој ќе ги претставува карактеристиките на тарифата врз која треба да се направат пресметките кои сега треба да се извршат во сервисот.

Од овие иследувања заклучуваме дека во оваа функција ќе се повика соодветниот сервис одкако ќе се постават како аргументи листата на воспоставени повици и карактеристиките на тарифниот пакет. При што ќе се постави и објект од класата `Handler` со чија помош ќе го добиеме одговорот од повиканиот сервис.

4. `handleMessage(Message message)`

Овој метод се наоѓа во `Handler` објектот кој го пративме со повикот на сервисот во функцијата `callLogUpdateOperator()`. Аргументот `message` всушност претставува одговорот на сервисот кој сме го повикале, во нашиот случај тој одговор претставува објект од класата `CallLogMinutes`. Класата `CallLogMinutes` го дава одговорот за преостанатите бесплатни минути кои му се преостанати на корисникот. Во случај бесплатните минути да се веќе преминати, во оваа класа се чуваат и бројот на минути кон соодветните оператори кои дополнително се направени. Ова доведува за можност да му се прикаже на корисникот кон кој оператор колку минути има разговарано после преминувањето на бесплатните услуги, како и можност да се пресметаат дополнителните трошкови кои се остварени до тој момент. Оваа услуга ја прави апликацијата дополнително поатрактивна во очите на нејзините корисници.

```

public class CallLogMinutes{

    public long tmobile = 0;
    public long vip = 0;
    public long one = 0;
    public long wired = 0;
    public long free = 0;
    public long freeOperator = 0;

    public CallLogMinutes(long t, long v, long o, long w) {
        // TODO Auto-generated constructor stub
        tmobile = t; vip = v; one = o; wired = w;
    }
}

```

3.1.2.2 Пресметување на моменталните телефонски трошоци, базирани на SMS пораки

За да се добијат моментално преостанатите бесплатни услуги врз основа на пратените SMS пораки, потребно е прво да се добие историјата на SMS пораките. Тука не се јавува потреба од добивање на операторот кон кој се пратила пораката. Причина за тоа претставува карактерот на бесплатната услуга која ја нудат сите македонски тарифи. Тие нудат одреден број на бесплатни SMS пораки без разлика на операторот кон кој се пратила пораката.

Со помош на изминување на историјата од сите пораки на корисникот, потребно е да се селектираат оние пораки кои ги испратил корисникот во периодот од почеток на моменталниот месец до моментот кога била побарана услугата. Врз овој податок и бројот на бесплатни SMS пораки кои ги нуди тарифниот пакет, ние сме способни да го пресметаме бројот на преостанати бесплатни пораки за корисникот.

Функцијата за одредување на потребните пораки се активира на кликот на копчето за зачувување на тарифниот пакет. При што бројачот на пратени пораки од почетокот на месецот до тој момент потребно е да се ресетира на нула.

Потоа се повикува функцијата 1. calculateSMSMonthRange(Calendar.getInstance()). Оваа функција го сетира бројачот countSMS на бројот на пратени пораки во тој период.

Разликата помеѓу овој бројач и бројот на бесплатни SMS пораки одреден во тарифниот пакет на корисникот дава резултат на преостанати бесплатни SMS пораки. Оваа пресметка се прави во функција која се справува со корисничката акција – клик на копчето “Провери нотификации”.

1. CalculateSMSMonthRange (Calendar.getInstance ())

За да се постави бројачот countSMS на бројот на пратени пораки во периодот од почетокот на месецот до дадениот момент, најпрво потребно е да се пристапи до датотеката во која се зачувани сите SMS пораки на корисникот. Оваа датотека се наоѓа во меморијата на телефонот. Со помош на URI-то content://sms андроид платформата нуди пристап до оваа датотека. И тука како и кај историјата на воспоставени повици потребни се филтри кои ќе ни ја дадат посакуваната резултантна листа на испратени SMS пораки од почетокот на месецот до

моментот кога била побарана услугата. За оваа цел се поставува селекција која при читањето на датотеката за секоја ставка ќе ја проверува содржината на атрибутот “type” чија барана вредност е 2, што претставува константа која кажува дека пораката била испратена од страна на корисникот. Исто така се гледа и атрибутот “date” за кој важи правилото date > првиот ден од моменталниот месец. Првиот ден од моменталниот месец се поставува со функцијата set(Calendar.DAY_OF_MONTH, 1).

Со изминување на бараните ставки од датотеката каде се зачувани SMS пораките на корисникот се зголемува бројач чија вредност, како и вредностите на бесплатните услуги поврзани со тарифниот пакет ќе ни го дадат резултатот на моментално преостанатите бесплатни услуги за корисникот.

3.1.2.3 Пресметување на моменталните телефонски трошоци, базирано на податочниот сообраќај

За разлика од телефонските разговори и SMS пораките, чија историја се зачувува во меморијата на телефонот, тоа не е случајот кај податочниот сообраќај. Тоа е една од главните причини зошто мониторирањето на оваа услуга е потешко за имплементација. Андроид платформата нуди решение со новововедената класа TrafficStats. Оваа класа нуди статистики за мрежниот сообраќај. Овие статистики вклучуваат примени и пратени бајти, како и бројот на примени и пратени мрежни пакети преку сите интерфејси. Не целосната документација на соодветните методи од оваа класа, како и недоволната поддршка на одредени уреди доведе до тоа оваа услуга да се имплементира како можно подобрување на апликацијата.

<http://developer.android.com/reference/android/net/TrafficStats.html>

3.1.2.4 Приказ на резултати од функциите

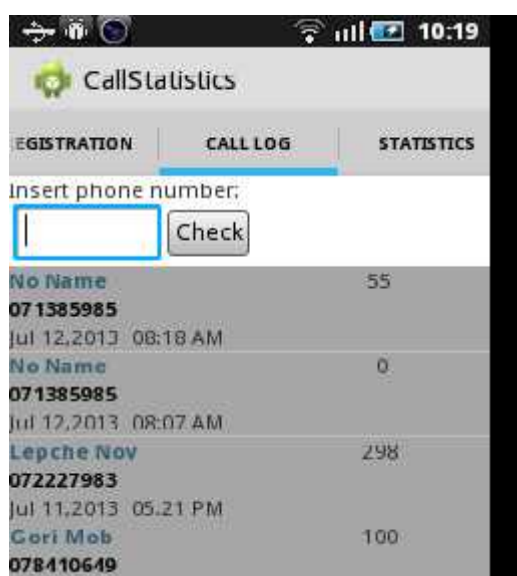
Од како ќе се пресметаат моменталните трошоци базирани на трите фактори: телефонските разговори, SMS пораките и податочниот сообраќај се јавува потреба од приказ на резултатите. Ова треба да е овозможено само кога сите пресметки ќе бидат завршени. Затоа по одговорот на сервисот, во функцијата handleMessage се овозможува корисникот да побара приказ на резултатите преку копчето “Прикажи нотификации”. На акцијата на клик на ова копче најпрво се прави пресметка на преостанатите SMS пораки базирано на податоци чии резултати ги добивме во соодветните функции објаснети погоре. Тој податок, заедно со податоците за преостанатите бесплатни услуги се прикажуваат на корисникот. Ова се истите податоци кои се прикажуваат и преку widget-от па убаво е ова да се напомене тука што би претставувало соодветна реклама за него.

3.2 Екран 2: Историја на повици

Прегледот на историјата на повици е услуга која е достапна на секој мобилен телефон. Пристапот до нејзе е едноставен и од голема корист за корисниците. Причината зошто е потребно да се има пристап до историјата на повиците на корисникот во оваа апликација е дополнителната услуга која може да му се понуди на корисникот, а тоа е можност да го види

операторот за секој од последните броеви со кои комуницирал. Оваа можност е додадена како дополнување на главната карактеристика на овој екран, а тоа е проверка на произволен број внесен од страна на корисникот. Оваа услуга му дозволува на корисникот да ги одреди операторите на телефоните со кои најчесто комуницира, што е од корист за разумно користење на бесплатните услуги кои ги нуди неговата тарифа. Иако ова е единствената допирна точка со главните услуги кои ги нуди апликацијата, таа може да се најде како корисна. Секако во иднина може да се додадат и филтри на историјата на повици, и може тука да бидат прикажани само броевите кои го промениле својот оператор, сортирани по одредена вредност важна за корисникот. Ова подетално ќе биде објаснето понатаму во документацијата во темата 4. Можни подобрувања.

3.2.1 Дизајн



Дизајнот на овој екран е прилично важен за корисниците. Причината за тоа е да им го привлече вниманието на корисниците додека се добие одговор од веб сервисите за соодветните оператори. Доколку овој дизајн е неуспешен корисниците може да заклучат дека овој екран нема поволни услуги за нив и да го променат пред да се добие одговор од веб сервисите, што е голем неуспех од два аспекти.

Прво тоа би создало негативно мислење за апликацијата што во никој случај не смее да се дозволи.

Второ и услугата која ја нуди овој екран ќе биде неискористена, и нема да имате можност дополнително да ги убедите корисниците дека вашата апликација има поголема вредност.

За успешен дизајн на историјата на воспоставени повици треба да се воведат некоја мала иновативност во споредба со дизајнот на историјата на воспоставени повици понудена од оперативниот систем на кој работи мобилниот апарат. Секако да се добие ваква идеја е многу тешко бидејќи оваа услуга постои уште од најстарите телефонски апарати. Мојата првична идеја беше поврзување на телефонскиот број со facebook и добивање на дополнителни податоци за воспоставениот повик. Оваа идеја може дополнително да се разгледа но постојат многу сигурносни полиси кои пречат за нејзино остварување.

3.2.2 Функции

Услугите кои ги нуди овој екран се базични и нивната цел е да ја збогатат и дополнат основната карактеристика за мониторирање на преостанатите бесплатни услуги на корисникот. Тоа го остваруваат со дополнително информирање на корисникот за операторите на телефонските броеви со кои воспоставил комуникација, како и можност за проверка на

операторот на број по негова желба. Како што беше нагласено, потребно е од дизајнерски аспект да се воведат иновативност во овој екран што ќе ја зголеми вредноста на апликацијата.

Од дизајнерски аспект потребно е на повидливо место да се постави можноста за проверката на операторот на бројот кој корисникот доколку посака може да го внесе преку корисничкиот интерфејс. На акцијата на копчето “Провери” се врши повик до сервисот кој го враќа соодветниот оператор.

При отварањето на овој екран од голема важност е да се постават сите податоци за историјата на воспоставени повици пред да се направи повикот кон сервисот кој ги одредува операторите на секој број од оваа листа. Оваа листа без соодветните оператори се добива со помош на функцијата `getCallLog()`.

1. Акција на копчето “Провери”

Можност за надополнување (Screenshot од кодот за `bCheck`)

На клик на копчето “Провери” потребно е да се активира функција која ќе го земе бројот кој корисникот го напишал во текстуалното поле од корисничкиот интерфејс и да се прати барање до веб сервисот на АЕК за проверка на операторот за внесениот број. Бидејќи оваа апликација е сеуште во фаза на развивање, моментално во кодот на ова копче се праќа и барање за проверка на операторите на секој број од листата `callLog` која ја претставува историјата на воспоставени повици.

2. `getCallLog()`

Можност за надополнување (Screenshot од кодот за `getCallLog`)

Во оваа функција, како и во функцијата `getCallLog(Calendar.getInstance())` која се наоѓа во кодот за екранот наменет за регистрација, потребно е да се изимине датотеката во која е зачувана историјата на воспоставени повици, и за секој повик да се запишат неговите информации во објект од класата `Call`. Резултантните објекти се додаваат во листата `callLog` која понатаму ќе ја пратиме до имплементираниот сервис за проверка на операторите на секој нејзин објект. Бидејќи целта ни е да ја прикажеме целата историја на воспоставени повици на корисникот, тука нема потреба од филтрирање на оваа листа. Единствено потребно е да се постави параметар за сортирање на повиците, почнувајќи од повикот кој последно бил воспоставен. Тоа се овозможува со поставување на параметарот `android.provider.CallLog.Calls.DEFAULT_SORT_ORDER` за време на читањето на датотеката. Во оваа функција како и кај сите функции кои ја читаат оваа датотека потребно е да се провери дали прочитаниот број за секој повик ги задоволува условите на веб сервисот понуден од АЕК. Затоа се воведува соодветна модификација на телефонскиот број доколку тој почнува со 00389 или +389. Исто така во оваа листа може да постојат броеви кои почнуваат на 151, повици на броеви кои не припаѓаат на македонски оператори и други специјални броеви. Овој проблем привремено се решава со проверка на бројот на цифри во телефонскиот број, доколку тој број не е еднаков на 9 (бројот на цифри на мобилни и домашни телефони во македонија), тој број се повик се прескокнува во пресметките и се преминува на следниот.

Потребно е прво оваа листа да се постави како извор на податоци за приказ на корисникот без соодветните податоци за операторот на секој повик. Ова се извршува во onCreate методот за овој екран. Како што беше погоре нагласено, оваа листа која ја чува историјата на воспоставени повици се праќа до имплементираниот сервис на акцијаат на клик на копчето “Провери”. Можно подобрување е оваа акција да започне асинхронно да се извршува после прикажувањето на листата на екранот на корисникот. Новата листа со информациите за операторите се поставува по добивање на одговорот од сервисот.

3. handleMessage(Message message)

Можност за надолнување (Screenshot од кодот за handleMessage)

Оваа функција е дел од handler објектот кој се праќа при повикот на имплементираниот сервис. Неговата handleMessage функција се активира одкако ќе се добие одговор од сервисот. Тука се поставува новата листа на корисничкиот екран, што му овозможува на корисникот да ги види и операторите кон кои биле воспоставени соодветните повици. Оваа листа се поставува врз ListView објект со помош на функцијата setAdapter(listAdapter). Објектот listAdapter потребно е однапред да се креира и претставува инстанца од класата CallLogArrayAdapter(Logs.this, R.layout.call_row, callLog).

Можност за надолнување (Screenshot од кодот за CallLogArrayAdapter класата)

CallLogArrayAdapter класата наследува од класата ArrayAdapter<Call>. Со оваа класа се пополнува ListView објектот за секој Call објект од резултантната callLog листа. Изгледот на секоја редица од оваа листа се поставува со помош на објект од LayoutInflater класата и едефиниран преку xml документот R.layout.call_row.

Можност за надолнување (Screenshot од кодот за R.layout.call_row)

Со помош на горенаведените функции успеваме да ја оствариме целта да му овозможиме на корисникот да се запознае со операторите на секој повик од историјата на воспоставени повици сортирана по опаѓачки редослед на датумот на воспоставување.

3.3 Екран 3: Статистика

На овој екран е претставена основната идеја на оваа апликација. Тука треба да се овозможи резултатите од сите статистички пресметки и споредби да бидат претставени на начин кој е лесно разбирлив за корисникот. Изреката “Една слика е вреди илјада зборови” е добра метафора за одлуката која ја донесов. Иако резултатите може да се претстават и со текстуални полиња во кои ќе се споредат резултатите од статистичките пресметки, доколку тие се претстават со убаво нацртан граф тие може полесно да бидат воочени. Од големо значење е тој граф да биде едноставен и лесно разбирлив за корисниците. Неговата главна цел е да им ја олесни споредбата на резултатите на корисниците, а не дополнително да ги збуни што може да предизвика негативни мислења за апликацијата во целост. Од овие причини неговиот дизајн и точност се од суштинско значење. Резултатите од тој граф треба да успеат да им помогнат на корисниците да ја најдат нивната најсоодветна тарифа. Не можам посилно да ја потенцирам важноста на оваа негова цел од наведување на последиците кои би настанале

доколку неговата имплементација е неуспешна. Доколку дојде до тој непосакуван случај, не само што може да дојде до конфузност и нејасност кај корисниците, што негативно би се одразило врз мислењето за апликацијата, нејзината популарност и корисност. Голем проблем претставува губењето на главната идеја на оваа апликација, и сите придобивки кои беа дополнително објаснети во психолошкиот аспект за апликацијата.

За големото влијание кое би го имала неточноста на резултатите кои се претставуваат во оваа апликација ќе биде објаснето во продолжение од документацијата.

3.3.1 Дизајн



Како што беше напоменато во воведот за овој екран, неговиот дизајн е од големо значење за оваа апликација. Воведување на граф во апликација секогаш претставува “нож со две острици”. Што значи дека неговата успешна имплементација би предизвикала позитивна рекација кај корисниците. Доколку тоа не е случајот, негативните впечатоци кај корисниците ќе бидат многу поголеми. Затоа е потребно да се обрне поголемо внимание на дизајнот на овој екран. Едноставноста на изгледот и разбирливоста на графот се многу битни фактори. Доколку се дојде до прекрасен дизајн на граф со тешка разбирливост целосно се губи неговата цел, што може да ја уништи идејата за оваа апликација. Затоа јас пристапив со следното мислење пред да

дојдам до одлуката за типот на графот со кој ќе биде избран:

“Разбирливоста на графот е од многу поголемо значење од убавината на неговиот дизајн”. Знаејќи дека тука треба да се направи споредба од резултатите на статистиката за тарифата на кој се наоѓа корисникот и онаа која моментално ја разгледува, заклучив дека најразбирливи резултати би се добиле со линиски граф (Bar chart). Динамичноста на графот е дополнителен фактор на кој треба да се обрне посебно внимание. Таа динамичност се јавува од потребата за повторна споредба со други тарифи, што е од големо значење за да се одреди најсоодветната тарифа. Еден од начините за решавање на динамичноста е доцртување на резултатите на истиот граф, што би дозволила паралелна споредба на повеќе тарифи и би го олеснила проблемот доколку корисникот се двоуми помеѓу неколку тарифи. Од друга страна, решението кое јас го избрав е прецртување на целиот граф и прикажување на споредба од резултатите на тарифата на која се наоѓа корисникот и новоизбраната тарифа. Причината за овој избор ќе биде детално објаснета во продолжение од документацијата во темата 4. Мозни подобрувања.

3.3.2 Функции

Функциите во овој екран решив да ги структурирам така што креирам посебна класа `CallLogCalculations` во која ќе се извршуваат сите функции кои се потребни за пресметка на податоците преку статистичка обработка. Останатите функции кои се користат за приказ и обработ на резултатите се наоѓаат во самата класа доделена на екранот.

Пресметките се активираат на акцијата на избор на тарифата за споредба од страна на корисникот. Тука се повикува функцијата `getTariffStats(selectedTariff)`. Целта и кодот на оваа функција се исти како и кај истоимената функција во екран 1. Бидејќи оваа функција е асинхрона, пресметките за избраната тарифа се повикуваат откако нитката ќе заврши со податочниот трансфер со `parse` сервисот.

Тука најпрво имаме мал рекурзивен повик на истата функција за добивање на податоците за двете тарифи. Најпрво се повикува за селектираната, врз основа на името кое било одбрано преку `Spinner`-от, а потоа се чита (доколку бил зачуван на Екран 1) тарифниот пакет на кој моментално го користи корисникот на апликацијата. Кога ќе ги добиеме карактеристиките за тарифите, се крира соодветен објект од класата `CallLogCalculations`, во зависност од тоа дали ќе се прикажуваат двете тарифи (доколку била зачувана тарифата на Екран 1), или само моментално селектираната тарифа од страна на корисникот. Потоа се повикува функцијата `calculateMinutesMonthRange(Calendar.getInstance())`.

Во оваа функција се извршуваат најпрво се повикува функцијата `calculateSMSMonthRange(Calendar.getInstance())` каде се пресметуваат сметките за последните четири месеци.

За да се добијат овие сметки најпрво се изминува датотеката за историјата на SMS пораки и се селектираат излезните SMS пораки во последните четири месеци. За секоја од овие пораки се повикува функцијата `updateSMSCount(date)`.

Овде во зависност на датумот на SMS пораката се зголемува еден од локалните четири бројачи креирани за секој од последните четири месеци.

Потоа во функцијата `calculateSMSBill(countSMS, tariffStats)` во зависност од бројачот на SMS пораки и карактеристиките на тарифата се пресметуваат сметките за секој од последните четири месеци. Тие резултати се запишуваат во локални променливи кои се додаваат во листа која понатаму ќе се прати до сервисот каде ќе се добие вкупната сума за сметките направени во тие четири месеци.

Во продолжение на функцијата `calculateMinutesMonthRange(..)` се изминува датотеката со историјата на воспоставени повици, каде се извлекуваат само излезните телефонски повици во периодот од последните четири месеци. За секој од овие повици се повикува функцијата `updateCallLog(call, date)`.

Функцијата `updateCallLog(call, date)` во зависност од датумот го доделува `Call` објектот, кој ги содржи податоците на повикот, во една од четирите листи кои ги претставуваат листите на

излезни податоци за еден од четирите месеци. Одкако секој повик ќе се додаде на соодветната листа се повикува функцијата `callLogUpdateOperator()`.

`callLogUpdateOperator()` всушност претставува повик на имплементираниот сервис со сите податоци кои му се потребни за да во него не само што ќе се одредат операторите за секој од `callLog` листите, туку и ќе се направат целосните пресметки за да се добијат конечните сметки за секој од последните четири месеци. Причината заради која ваквите пресметки се прават во имплементираниот сервис беше и претходно спомената, а таа е да се намали податочниот сообраќај. Пред повикот на овој сервис се поставуваат сите податоци кои ни се потребни за да ги извршиме пресметките во сервисот, а тоа се `callLog` листите, карактеристиките на селектираната тарифа, SMS сметките, како и `handler` објектот кој е креиран во класата поврзана со Екран 3. Дополнително доколку била зачувана и тарифата на која се наоѓа корисникот се поставуваат и нејзините карактеристики, како и SMS сметките за оваа тарифа.

1. `getTariffStats(String tariffName)`

Како што беше погоре објаснето, во оваа функција ги добиваме карактеристиките за избраната и зачуваната тарифа. Самото добивање на овие карактеристики се остварува преку сервис и ќе биде објаснето понатаму. Одкако ќе ги добиеме карактеристиките за селектираната тарифа тие се зачувуваат во `TariffStats` објект, потоа се прави проверка дали е зачувана тарифата на корисникот. Доколку да се прави рекурзивен повик на оваа функција и во зависност од знаменцето `tarifNo` овој пат се зачувуваат во друг `TariffStats` објект новите карактеристики. Потоа се креира `CallLogCalculations` објект и му се предаваат објекти кои се потребни да се стартува сервисот, како и карактеристиките на тарифите. Битно е да се наведе дека тука се предава и `handler` објектот чиј `handleMessage` метод ќе биде повикан по добивање на резултатите од сервисот.

2. `calculateMinutesMonthRange(Calendar.getInstance())`

Во овој метод се повикува функцијата `calculateSMSMonthRange` која ќе ги пресмета месечните сметки за двете тарифи базирано на SMS пораките. Потоа се изминува датотеката која ја чува историјата на воспоставени повици. Од нејзе се селектираат само излезните повици воспоставени во периодот од последните четири месеци. Тие се зачувуваат во `Call` објекти, кои се предаваат заедно со датумот на нивното воспоставување на функцијата `updateCallLog`. Во оваа функција `Call` објектот ќе биде зачуван во `callLog` листата за соодветниот месец. Кога ќе се повтори оваа постапка за сите `Call` објекти се повикува методот `callLogUpdateOperator()` кој ќе го повика имплементираниот сервис за да се извршат пресметките преку кои ќе се добијат конечните месечни сметки кои ќе ги прикажеме на графикот.

3. `calculateSMSMonthRange(Calendar.getInstance())`

За пресметување на бројот на пратени SMS пораки во Екран 1 беше заклучено дека нема потреба од проверка на операторот кон кој била пратена пораката. Бидејќи тука ни е потребен бројот на пратени SMS пораки во последните четири месеци, ја изминуваме датотеката со историја за SMS пораки со филтер за последните четири месеци и влечење само на испратените SMS пораки. За секоја од овие пораки ја повикуваме функцијата

`updateSMSCount(date)` која во зависност од датумот го зголемува бројачот за соодветниот месец. Одредбата ќе го повториме овој процес и ќе ги добиеме конечните бројачи на пратени SMS пораки за секој од последните четири месеци ја повикуваме функцијата `calculateSMSBill(countSMS, tariffStats)`. Оваа функција се повикува по четири пати, за секој месец посебно, за секоја тарифа. Резултатот од оваа функција е сметката од SMS пораките врз соодветната тарифа за бројачот на пратени пораки од одреден месец. Овие сметки се запишуваат во листа на сметки за секоја тарифа. Овие листи ќе ги пратиме при повикот на имплементираните сервис за пресметка на конечните сметки.

Важно е да се напомене дека при повикот на оваа функција, најпрво се ресетираат сите локални бројачи.

4. `updateSMSCount(date)`

Оваа функција се повикува за секоја од SMS пораките кои се прочитани по филтрирањето на датотеката која ја содржи историјата на SMS пораките на корисникот. Нејзината цел е во зависност од датумот на кој е пратена SMS пораката да го провери месецот и да го зголеми бројачот наменет за соодветниот месец. За да се направи оваа проверка потребно е да се знае моменталниот месец и неговите четири претходници. Одредбата тие ќе се одредат се креира `Calendar` објект базиран на времето од објектот `Date` добиен како аргумент, и со функцијата `Calendar.get(Calendar.MONTH)` се добива месецот во кој била пратена соодветната SMS порака. Овој месец се споредува со четирите претходници на моменталниот месец и доколку настане поклопување се зголемува соодветниот бројач.

5. `calculateSMSBill(countSMS, tariffStats)`

Оваа функција ја пресметува месечната сметка, за соодветниот бројач, базирана на SMS пораките. Таа се повикува по четири пати, за секој месец, со различен `countSMS` бројач кој го чува бројот на пратени пораки во тој месец. Бидејќи македонските тарифни пакети имаат еднаква цена за праќање на SMS пораки кон сите оператори, нема потреба од повикување на веб сервисот на АЕК за одредување на операторот кон кој е пратена SMS пораката. Ова ни овозможува локално да ја пресметаме месечната сметка за SMS пораките.

За да ја пресметаме оваа сметка, најпрво ги одземаме бесплатните SMS пораки кои ги нуди тарифата од бројачот на пратени пораки во тој месец. Доколку биле пратени поголем број на SMS пораки од оние кои се доделени како бесплатни од страна на тарифниот пакет, разликата на бројачот и бесплатните пораки се множи со цената за праќање на порака наведена во карактеристиките за соодветната тарифа. Оваа пресметка како резултат ја дава месечната сметка базирана на SMS пораките.

6. `updateCallLog(call, date)`

Овој метод се повикува за секој `Call` објект од листата на излезни повици кои биле воспоставени во последните четири месеци. Неговата функција е базирано на датумот како аргумент да го додели во `callLog` листата за соодветниот месец. Најпрво од датумот на воспоставување на повикот се добива соодветниот месец. Потоа се добиваат информациите за моменталниот месец и неговите четири претходници. Со помош на `if-else` изрази се бара

поклопување на месецот на повикот со еден од претходниците и се додава во соодветната листа која претставува историја на воспоставени повици за одредениот месец.

7. callLogUpdateOperator()

За да се оствари целта на овој метод за добивање на операторите на секој Call објект од четирите месечни листи за историја на воспоставени повици се повикува имплементираниот сервис каде се напишани тие функции. Покрај тоа во овој сервис се прават и пресметките за добивање на конечните сметки кои корисникот ги направил за секој од последните четири месеци. За да се направат овие пресметки потребно е да се испратат соодветните callLog листи, тарифите за кои се потребни сметките како и веќе пресметаните SMS сметки. Ова се сите податоци кои му се потребни на сервисот за да ги добие пресмета конечните сметки кои ќе ги врати како одговор на преку порака до handleMessage функцијата во handler објектот кој се наоѓа во класата за Екран 3.

8. handleMessage(Message message)

Овој метод се активира кога ќе пристигне одговорот од имплементираниот сервис. Тој одговор претставува листа од сметки за секој месец која сакаме да му ја презентираме на корисникот.

*Можност за надополнување (a chart engine) *

За претставување на резултатот во форма на линиски граф се користи надворешната библиотека “A chart engine”. Таа се наоѓа во libs директориумот од проектот, а нејзините класи се референцираат преку директориумот Referenced Libraries.

Графикот се претставува во layout кој е дел од дизајнот на Екран 3. Графикот кој се креира претставува објект од класата GraphicalView која наследува од класата View. Графикот се креира со следниот статички метод ChartFactory.getBarChartView(..). Како аргументи овој метод прима Dataset и Renderer објекти кои ги содржат податоците и изгледот на графикот, соодветно.

Dataset објектот се пополнува во handleMessage методот со резултантните сметки од серверот. Сметките за секоја тарифа (селектираната и корисничката) се доделуваа преку посебни XYSeries објекти. Додека дизајнот за секоја тарифа се претставува преку XYSeriesRenderer објекти. Поврзувањето на соодветните XYSeries и XYSeriesRenderer објекти се остварува преку редоследот кои тие го имаат во Dataset и Renderer објектите, соодветно. Затоа добра програмерска практика е секоја серија која треба да се додаде да биде целосно додадена и од податочен и од дизајнерски аспект.

Во овој метод најпрво потребно е да се избришат претходните податоци запишани во Dataset објектот со помош на clear() методот. Интересна дизајнерска одлука е начинот на првичната претстава на графикот. Една од динамичните карактеристики е решението за претстава на почетниот максимум на Y оската. Бидејќи висината на графикот зависи од цента на месечните сметки што е различно за секоја тарифа, се јавува од потреба за решение кое ќе се основа на цената на сметките. За да се добие на разбирливост на графикот јас го поставив максимумот на Y оската да биде еднаков на 300 + максималната месечна сметка од двете тарифи.

За да се реши проблемот со динамично прецртување на графикот, се јавува потреба од повик на функцијата `repaint()` која ќе го прецрта графикот со новопоставените податоци.

Останатите дизајнерски решенија се наоѓаат во функцијата `initializeMRenderer` и бидејќи тие се статички и имаат прилично самообјаснувачки методи, може да ги видите во продолжение.

```
private void initializeMRenderer() {
    //http://www.fruitson.com/2017/04/android-tutorial-drawing-achartengine-bar-chart-with-example/
    // set some properties on the main renderer
    mRenderer.setChartTitle("Сметки во последните 4 месеци");
    mRenderer.setApplyBackgroundColor(true);
    mRenderer.setBackgroundColor(Color.argb(100, 50, 50, 50));
    mRenderer.setAxisTitleTextSize(10);
    mRenderer.setChartTitleTextSize(15);
    mRenderer.setLabelTextSize(8);
    mRenderer.setLegendTextSize(8);
    mRenderer.setZoomButtonsVisible(true);
    mRenderer.setBarSpacing(1);
    mRenderer.setLabels(0);
    mRenderer.setXAxisMin(-1);
    mRenderer.setXAxisMax(5);
    mRenderer.setYAxisMin(-10);
    mRenderer.setXTitle("Меесу");
    mRenderer.setYTitle("Сметка");
    mRenderer.setBarWidth(13);

    //mRenderer.setLegendHeight(mRenderer.getLegendHeight() + 40);
    mRenderer.setShowLegend(true);
}
```

Сите овие функции се од суштинско значење за пресметките и основниот дизајн на графикот. Иако целата оваа структура на методи е тесно поврзана и тешка за пратење, таа содржи “атомични” методи кои извршуваат по една важна функција. Тоа претставува добра програмерска практика особено за дебагирање на грешки што е тежок процес при сложени програмерски решенија.

3.4 Сервиси

За услугите кои се понудени на секој од екраните на оваа апликација, као и самиот widget, потребна е постојана интернет конекција. Причина за тоа е потреба од веб сервисот “Провери број” понуден од АЕК, како и пристапот со помош на parse сервиси до базата на податоци за карактеристиките на македонските тарифни пакети.

Иако сите потреби од пристап до овие веб сервиси беа споменати до сега, добро е за потсетување да ги наброиме сите нив на едно место.

Пристап преку parse сервиси до parse базата на податоци се јавуваат кај:

- Екран 1 и 3: пристап до сите оператори и сите тарифни пакети
- Екран 1 и 3: пристап до сите тарифни пакети за одреден оператор
- Екран 1 и 3, widget: пристап до сите податоци за одреден тарифен пакет

Пристап преку SOAP барање до АЕК сервисот “Провери број” се јавуваат кај:

- CheckOperatorService, widget: пристап до операторите за callLog листа

3.4.1 Parse база на податоци

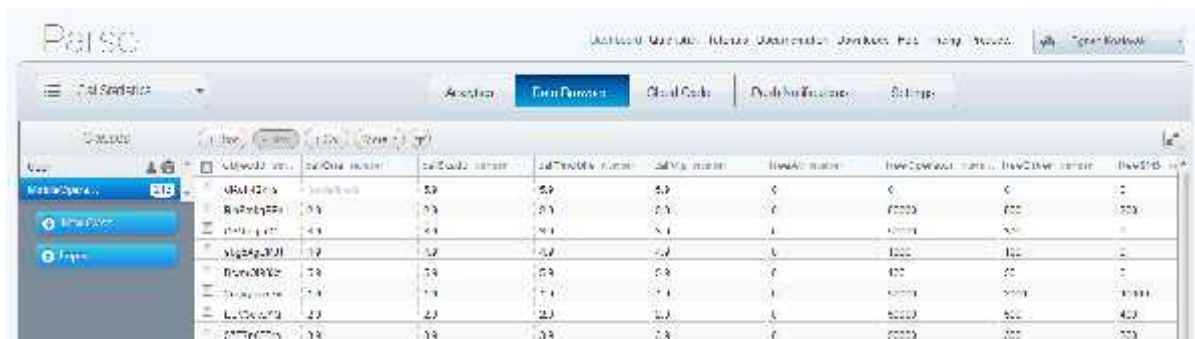
Податоците за сите тарифни пакети рачно ги додадов во база на податоци која се наоѓа на Parse серверите. Причината заради која го избрав Parse се бесплатните услуги кои ги нуди, а тоа се 1GB бесплатна меморија и 1000 бесплатни повици кон сервис. Од голема помош се соодветните SDK кои ги нудат за андроид платформата и нивното едноставно користење. Со помош на понудените решенија кои работат во позадина (асинхроно) оваа платформа целосно ја елиминира потребата од пишување на серверски код и одржување на серверите. Јасната документација за андроид платформата е од голема корист. Таа може да се најде на https://parse.com/docs/android_guide. Пристап до сервисите за пристап до базата на parse се овозможени преку додавање на parse SDK во libs директориумот и користење на неговите класи преку Referenced Libraries директориумот.

Дополнително потребно е да се додаде:

`Parse.initialize(this, "PARSE_KEY1", "PARSE_KEY2");` во `onCreate` методот на екранот каде пристапуваме до базата.

`ParseAnalytics.trackAppOpened(getIntent());` во `onCreate` методот на главниот (MAIN) екран за пратење на статистиката.

За почнување на нов parse проект или додавање во постојан проект може да се погледне: <https://parse.com/apps/quickstart#android/native/existing>.



The screenshot shows the Parse.com dashboard with a table of data for the 'MobileOperators' class. The table has columns for 'objectId', 'name', 'number', 'country', 'operator', 'network', 'type', 'status', 'created_at', and 'updated_at'. The data is as follows:

objectId	name	number	country	operator	network	type	status	created_at	updated_at
1	10000	10000	10000	10000	10000	10000	10000	10000	10000
2	10000	10000	10000	10000	10000	10000	10000	10000	10000
3	10000	10000	10000	10000	10000	10000	10000	10000	10000
4	10000	10000	10000	10000	10000	10000	10000	10000	10000
5	10000	10000	10000	10000	10000	10000	10000	10000	10000
6	10000	10000	10000	10000	10000	10000	10000	10000	10000
7	10000	10000	10000	10000	10000	10000	10000	10000	10000
8	10000	10000	10000	10000	10000	10000	10000	10000	10000
9	10000	10000	10000	10000	10000	10000	10000	10000	10000
10	10000	10000	10000	10000	10000	10000	10000	10000	10000

3.4.2 Parse сервиси

Како што беше горе наведено, parse нуди сервиси кои се имплементирани да ги извршуваат бараните функции на асинхрон начин. Заради ваквата нивна природа, програмерите не се принудени тоа рачно да го воведуваат и тоа ги прави многу популарни. Ваквите сервиси се користат во функциите `getAllOperators()`, `getAllTariffs()` и `getTariffStats()`. Тие се повикуваат преку `ParseQuery` објект кој пристапува до бараната табела од базата на податоци преку статичката функција `ParseQuery.getQuery("MobileOperators")`. Со помош на овој објект се повикува функцијата `findInBackground(..)` која се извршува во позадина што значи дека не го кочи корисничкиот интерфејс.



3.4.3 АЕК сервис “Провери број”

Сервисот “Провери број” има едноставна задача да врати резултат во xml форма за операторот на кој припаѓа внесениот број. Овој сервис е понуден и за низа телефонски броеви и се наоѓа на следната адреса: http://www.aek.mk/ServiceNP1/checkNumbersOperator_WS.aspx. Може да се користи преку два методи: checkNumbers и checknumbersstring.

- checkNumbers: метод кој за влез прима низа телефонски броеви а како резултат враќа xml документ со податоци за секој број – оператор на кој е доделен бројот, оператор на кој е пренесен бројот и статус (енумерација од една од трите можни вредност 0 – не е доделен, 1 – доделен е на оператор и не е пренесен и 2 – пренесен е кај операторот ...)

- checknumbersstring: метод кој за влез прима string од телефонски броеви во формат OXXXXXXX одделени со запирка а како резултат враќа xml документ со податоци за секој број – оператор на кој е доделен бројот, оператор на кој е пренесен бројот и статус (енумерација од една од трите можни вредност 0 – не е доделен, 1 – доделен е на оператор и не е пренесен и 2 – пренесен е кај операторот ...)

За пристап до сервисот потребно е да се внесат параметрите key и secret.

По препорака од АЕК при праќањето на низа од броеви, низата да не биде подолга од 30-40 броеви со цел да не се преоптовари сервисот и да дојде до евентуално блокирање на database серверот. Затоа проверките на поголеми серии се решени со испраќање на повеќе последователни барања до сервисот.

Покрај оваа препорака постојат и повеќе услови за користење на сервисот кон кои мора да се придржува секој негов корисник.

Од двата понудени методи за користење на сервисот јас го избрав методот checknumbersstring.

3.4.3.1 Функции

Услугите на АЕК сервисот се овозможуваат преку методот `callLogUpdate(ArrayList<Call> CallLog)`. Тука најпрво листата која се добива како аргумент се дели на повеќе серии од листи со големина од 30 елементи или помалку. Од овие подлисти се земаат телефонските броеви кон кои биле направени повиците и се адаптираат во бараниот формат. Потоа се повикува функцијата `soapGetOperators(String numbers)` кој го враќа одговорот од веб сервисот “Провери број”.

Во методот `soapGetOperators(String numbers)` се праќа соодветното soap барање до веб сервисот на АЕК. Од xml документот кој се добива како одговор се стигнува до бараниот резултат во кој е сеуште во xml формат и се враќа резултатот како инстанца од класата `SoapObject`.

Следува повик до функцијата `updateOperators(tempLog, phonebook)` каде се парсира резултантниот xml документ и за секој број од подлистата на `callLog` објектот се добива соодветниот оператор и тој се запишува во соодветниот `Call` објект.

1. `callLogUpdate(ArrayList<Call> CallLog)`

По препораката од АЕК, наместо единечно SOAP барање до сервисот каде ќе ги внесеме сите елементи од историјата на воспоставени телефонски повици, потребно е да извршиме поделба на листата на нејзини подлисти кои ќе предизвикаат серии од последователни SOAP барања до сервисот. Оваа поделба се прави со помош на функцијата `CallLog.subList(од, до)`. Резултатот на оваа функција е подлиста од `Call` објекти кои претставуваат референци на вистинските објекти. Тоа значи дека доколку направиме промена врз содржината на елементите од оваа подлиста, таа промена ќе се пренесе во листата од која била добиена. Со изминување на `Call` објектите од подлистата ги додаваме броевите кон кои бил воспоставен повикот во `String numbers` променлива одделени со запирка, за да го добиеме бараниот формат на влезниот параметар според документацијата на методот `checknumbersstring`. Оваа променлива се праќа како аргумент при повик на методот `soapGetOperators(numbers)`. Резултатот од овој метод е објект од класата `SoapObject` наречен `phonebook` и претставува xml документ. Овој документ заедно со подлистата се праќаат како аргументи при повикот на методот `updateOperators(tempLog, phonebook)`. Во овој метод ќе се најде операторот на секој од `Call` објектите во подлистата и соодветно ќе се додаде во него. Бидејќи овие `Call` објекти се само референци од објектите во почетната листа нема потреба од нејзина промена.

Оваа постапка се повторува се додека не се додаде името на операторот за секој `Call` објект од бараната листа.

2. `soapGetOperators(String numbers)`

Овој метод како аргумент ги добива броевите кои сакаме да ги провериме во бараниот формат. Тука се креира SOAP барањето, се додаваат параметрите `key` и `secret` и се додава влезниот параметар за методот како објект од класата `PropertyInfo`. Потоа се креира објект од класата `SoapSerializationEnvelope` и се поставува параметарот `envelope.dotNet = true` што

означува дека SOAP одговорот ќе биде добиен од сервис имплементиран преку ASP.NET. Потоа овој објект се поставува како излезен SOAP објект за креираното SOAP барање со помош на функцијата `envelope.setOutputSoapObject(request)`. Со оваа функција кажуваме дека откако ќе го пратиме барањето до веб сервисот, неговиот одговор ќе биде зачуван во овој објект.

За да го испратиме барањето до сервисот, креираме инстанца од класата `HttpTransportSE` врз даденото URL на сервисот и го повикуваме методот `call(..)` на овој метод каде го специфицираме методот кој го повикуваме и објектот во кој треба да се добие резултатот.

Потоа врз база на описот на сервисот даден на

http://www.aek.mk/ServiceNP1/checkNumbersOperator_WS.asmx?WSDL за да пристигнеме до листата на одговори во xml формат бршиме мало парсирање со помош на функцијата `getProperty(0)`. Оваа листа на одговори ја враќаме како одговор на оваа функција.

Сите класи кои се користат во оваа функција се дел од библиотеката `ksoap2` која е додадена во `libs` директориумот. Пристапот до овие класи е овозможен преку нејзина референца во директориумот `Referenced Libraries`.

3. `updateOperators(tempLog, phonebook)`

Од како го добивме одговорот од веб сервисот, тој одговор заедно со подлистата која ги содржи `Call` објектите за кои го најдовме одговорот се праќаат како аргументи на овој метод. Тука потребно е да ги додадеме операторите во соодветните `Call` објекти. Затоа за секој `Call` објект од подлистата се изминува резултантната `phonebook` листа и се бара поклопување по бројот на повикот. Пристап до содржината на бараното поле во xml документот се прави со помош на функцијата `getPropertyAsString(барано поле)`. Кога ќе се најде на поклопување на бројот на повикот можни се следните три сценарија:

- Бараниот број има статус еднаков на нула, што значи дека тој не бил доделен на ниту еден оператор и може директно да се прекине со пребарувањето за тој објект.

- Бараниот број има статус еднаков на еден, што значи дека тој бил доделен на соодветниот оператор запишан во xml тагот `"dodelenNa"`, оваа вредност ја запишуваме како вредност на операторот и продолжуваме со пребарување бидејќи доколку тој бил префрлен ќе има нов запис со статус еднаков на два за истиот број во одговорот од сервисот.

- Бараниот број има статус еднаков на два, овој случај има два подслучаи:

- до сега сме дошле само до едно поклопување (статусот (знаменце) бил еднаков на 1), тогаш се презапишува името на новиот оператор запишан во xml тагот `"prefrlenVo"` како вредност на операторот, се зачувува вредноста на записот во xml тагот `"prefrlenDatum"` како моментално најстар датум, се обновува статусот (знаменце) за проверка на 2 и продолжуваме со пребарување бидејќи доколку бројот бил повторно префрлен ќе има нов запис за истиот број во одговорот од сервисот.

- До сега сме дошле до две или повеќе поклопувања (статусот (знаменце) бил еднаков на 2), тогаш се прави проверка дали датумот запишан во xml тагот `"prefrlenDatum"` е покасно од најстариот датум од претходните поклопувања. Доколку да се презапишува името

на новиот оператор одзапишан во xml тагот “prefrlenVo” и се зачувува вредноста на записот во xml тагот “prefrlenDatum” како моментално најстар датум и се продолжува со пребарување.

Со сите овие проверки на крај се добива да во објектите на подлистата tempLog биде зачуван моменталниот оператор на кој се наоѓа телефонскиот број кон кој бил воспоставен повикот. За потсетување овие промени директно се одразуваат врз Call објектите на целосната листа на воспоставени повици па не се јавува потреба од нејзино обновување.

3.4.4 onHandleIntent(Intent intent)

Оваа функција се повикува за справување со сите повици на овој сервис. Потреба од повик на овој сервис се јавува на секој од екраните на апликацијата, а причина за тоа е потребата од асинхрон начин на пристап до веб сервисот кој го нуди АЕК.

Бидејќи акциите кои треба да се извршат зависат од екранот од кој е стартуван овој сервис, се појавува потреба од воведување на начин за одредување на класата од која бил повикан повикан овој сервис. Тоа се овозможува со воведување на дополнителен параметар пред повикот на сервисот, кој би се одликувал за секој од класите претставници на соодветните екрани. Затоа пред секој повик на сервисот се воведува параметар наречен ClassPath кој го претставува патот до класата и е единствен за секоја од класите кои го повикуваат сервисот. Вредноста на овој параметар се добива со функциите `getPackageName()` + “.” + `getLocalClassName()`.

Врз основа на овој параметар, во методот onHandleIntent се формираат различни сценарија за обработка на податоците и поставување на одговорот.

Во овој метод има три такви сценарија:

1. Повик на сервисот од com.aek.callstatistics.UserRegistration класата (Екран 1)

При овој повик на сервисот ја праќаме историјата на излезни воспоставени повици од почетокот на моменталниот месец до дадениот момент и карактеристиките на тарифниот пакет на кој моментално се наоѓа корисникот. Целта е да се прикажат резултатите кои може да се мониторираат преку widget-от, а тоа е моменталната состојба на бесплатните услуги понудени во тарифниот пакет. За да се оствари таа цел потребно е да се повика функцијата `callLogUpdate(callLog)` каде со помош на повик до АЕК веб сервисот ги добиваме операторите кон кои биле воспоставени повиците. Потоа со повик на функцијата `calculateNotifications(callLog, myTariffStats)` се добиваат резултатите како објект од класата `CallLogMinutes`. Во овој објект ги чуваме преостанатите бесплатни услуги, а доколку тие се во целост потрошени се чуваат и дополнително направените трошкови. Со ова имаме можност доколку корисникот ги надмине бесплатните услуги да му дадеме приказ преку кој тој ќе може да види на кој оператор по колку дополнителни минути се направени, како и можност за пресметка на моменталната сметка за водење на дополнителна евиденција. Овој објект после го доделуваме преку `msg.obj = clmNotifications` како резултат кој ќе се прати до handler објектот во `UserRegistration` класата.

2. Повик на сервисот од com.aek.callstatistics.Logs класата (Екран 2)

Целта на повик од оваа класа е проверка на операторите на целосната историја од воспоставени повици. Потреба од оваа проверка се јавува при воведувањето на услугата за приказ на историјата на повиците на корисникот (call log) со дополнителна информација за операторот кон кој бил воспоставен тој повик. За оваа цел како параметар се праќа целата таа листа, а во `onHandleIntent` методот се повикува `callLogUpdate(callLog)` каде со помош на повик до АЕК веб сервисот ги добиваме операторите кон кои биле воспоставени сите повици од листата. Соодветно оваа листа претставува одговор кој треба да се врати на `handler` објектот во `Logs` класата, а тоа се остварува со помош на `msg.obj = callLog`.

3. Повик на сервисот од `com.aek.callstatistics.Statistics` класата (Екран 3)

Повикот на сервисот од класата `Statistics` е со цел да се направат целосните пресметки на сметките за секој од последните четири месеци врз база на излезните повици во тој период. Сумата од тие сметки заедно со пресметаните сметки направени преку SMS пораките во овој период ја даваат вкупната сметка за соодветната тарифа. Како параметри потребни за да се остварат овие пресметки, пред повикот на сервисот во `Statistics` класата се додаваат четирите листи кои ја претставуваат историјата на излезните воспоставени повици, секоја за еден од последните четири месеци. Исто така потребно е да се постават карактеристиките на тарифите за кои се бараат пресметките, како и веќе пресметаните сметки направени преку SMS пораките за последните четири месеци. Во методот `onHandleIntent` најпрво се повикува функцијата `callLogUpdate(callLog)` за секоја од листата за четирите месеци за да се добијат соодветните оператори кон кои биле воспоставени повиците. Потоа се пресметува сметката за секој од четирите месеци за тарифите за кои се потребни пресметките преку функцијата `calculateBill(callLog, tariffStats)`. Резултатот на оваа функција е сметката за соодветната тарифа од воспоставените повици за време на соодветниот месец. Сумата на оваа сметка и сметката од SMS пораките ја дава вкупната сметка за тој месец. Со овие пресметки ги добиваме вкупните сметки за секој од четирите месеци за бараните тарифи. Овие сметки се додаваат во листа и таа листа од сметки се враќа како одговор на сервисот преку `msg.obj = dollaBillz`.

3.4.5 `calculateBill(callLog, tariffStats)`

Во овој метод за тарифата која се добива како аргумент се пресметува сметката направена преку листата на воспоставени повици која е претставена со првиот аргумент од методот, `callLog`. За да се пресмета сметката, најпрво потребно е да се одреди карактерот на бесплатните услуги за повиците кои ги нуди тарифата. Тие може да бидат претставени преку бесплатни минути кон сопствениот оператор, бесплатни минути кон останатите мрежи, бесплатни минути кон сите мрежи и бесплатна сума за повици кон сите мрежи.

Во зависност од типот на бесплатните услуги кои ги нуди тарифата, се базираат и соодветните пресметки. Бесплатните услуги се запишани во `tariffStats` објектот. Одкако ќе се утврдат бесплатните услуги нивните вредности се запишуваат во локални променливи. Потоа се изминува историјата на воспоставени повици (`callLog`) и за секој повик кој претставува објект од класата `Call` се прават следните проверки.

Доколку се преостанати некои од бесплатните услуги се проверува типот на преостанатите бесплатни услуги. Доколку повикот ги исполнува условите за бесплатната услуга, во зависност

од операторот кон кој бил воспоставен, се намалува вредноста на локалната променлива за бесплатната услуга за времетраењето (или цената) на тој повик.

Во спротивен случај, кога не постојат повеќе бесплатни услуги или бесплатните услуги не се соодветни за тој повик (зависно од операторот кој ја нуди тарифата и операторот кон кој бил воспоставен повикот), времетраењето на повикот се додава во соодветната променлива, за операторот кон кој бил воспоставен повикот, од CallLogMinutes објект.

Оваа постапка се повторува за секој повик од листата. Резултантниот CallLogMinutes објект заедно со карактеристиките на тарифниот пакет се користат за да се направат финалните пресметки за да се добие резултантната сметка врз основа на воспоставените повици. Таа се добива најпрво со додавање на месечната претплата на тарифата. Потоа се додава производот од минутите потрошени кон секој оператор (кои ги пресметавме во CallLogMinutes објектот) и цената за повици кон тој оператор (дел од карактеристиките на тарифата). Оваа сметка за воспоставените повици се враќа како резултат од овој метод.

Точноста на пресметките во овој метод се од голема важност. До сега во апликацијата можните грешки во најлош случај лошо се одразуваа врз популарноста на апликацијата. Доколку настане грешка во овие пресметки, не само што може да се доведе до негативни мислења за апликацијата, тие може да предизвикаат финансиска загуба кај корисниците. Затоа покрај тоа што треба дополнително внимание да се посвети на тестирање на точноста за овие пресметки, потребно е да се наведе во описот од апликацијата дека таа е од информативен карактер.

3.4.6 calculateNotifications(callLog, tariffStats)

Пресметките кои се извршуваат во овој метод се целосно опфатени во методот calculateBill. Бидејќи неговата цел е да ги претстави како известување на корисникот преостанатите бесплатни услуги или степенот на нивното преминување, пресметките завршуваат со добивање на резултантниот CallLogMinutes објект. Во овој објект се чуваат бараните податоци кои се потребни за приказ на состојбата во зависност од воспоставените повици кои се зачувани во callLog листата. Користењето на овој се разликува од користењето на calculateBill методот. Повикот на овој метод е со листа која ги содржи воспоставените излезни повици во периодот од почетокот на моменталниот месец до дадениот момент, за разлика од повикот на методот calculateBill кој е со иста ваква листа но во временски период на цел месец. Овој метод е од голема важност и се користи кај услугата за мониторирање на моменталната состојба на сметката на корисникот. Таа услуга се корисити во widget-от и на клик на копчето “Провери нотификации” на екран 1.

3.4.7 Асинхрон начин на работа

Користење на сервис претставува само еден од понудените начини за воведување на асинхрон начин на работа на андроид платформата. Постојат пет начини преку кои може тоа да се обезбеди. Секој од тие начини има свои позитивни и негативни карактеристики што

значи дека не постои најдобар начин, најдобриот начин може да се одреди само врз основа на проблемот кој треба да се реши.

1. Нитка:

Негативни карактеристики (со споствен код се воведува):

- синхронизација со главната нитка, ако има потреба од поставување на резултатите на корисничкиот интерфејс.

- не постои стандарден начин за нејзино поништување

- не постои стандарден начин за нивно пордерудање

- не постои стандарден начин за справување со конфигурациски промени во андроид подредување

2. Handler (воведен за андроид платформата):

Претставува подобрување на нитка во:

- корисен при потреба од голем број на пристапи до главната нитка

Негативни карактеристики:

- не се поврзува со класата која го креирала

- потреба од справување memory leak

3. AsyncTask (воведен за андроид платформата):

Позитивни карактеристики:

- го енкапсулира креирањето на процес кој работи во позадина

- синхронизација со главната нитка

- поддржува известување за напредокот на задачата која се извршува

Негативни карактеристики:

- Не се координирани со животниот циклус на активноста (екранот).

- Доведува до memory leak

4. Loaders (воведени во андроид платформата)

Позитивни карактеристики:

- вчитуваат податоци асинхроно

- ги мониторираат изворните податоци и даваат нови резултати при промена на содржината

- ги кешираат податоците = нема memory leak
- координирани со животниот циклус на активност (екранот).

Негативни карактеристики:

- премногу тесно поврзани со животниот циклус на активност (екранот).
- резултатите од барањата се губат со напуштање на активност
- корисниците мора да чекаат за резултатите (да не ја напуштат активност)
- нема поддршка за справување со исклучоци

5. Сервис

Позитивни карактеристики:

- нуди транспарентно кеширање
- нема memory leak
- барањата не се губат кога испраќачот (активност) умира
- резултатите од барањето не се губат кога испраќачот (активност) умира
- тоа овозможува одлична синхронизација со животниот циклус на активност

Од овие иследувања заклучив дека за мојата апликација најбитно е асинхронизираниот процес да биде добро синхронизиран со животниот циклус на активностите. Тоа најдобро се овозможува со помош на Intent Service кој наследува од Service класата. Посебната функција која ја овозможува Intent Service е автоматското уништување на инстанцата на овој објект после извршувањето на позадинската задача.

3.5 Widget

Потребата од widget за оваа апликација беше темелно објаснета во психолошкиот аспект. Неговата функционалност е да ја прикаже моменталната состојба на преостанатите бесплатни услуги за корисникот. Оваа функционалност како и нејзината имплементација е објаснета во темата Екран 1: Корисничка регистрација и таа се активира на акцијата клик на копчето “Прикажи нотификации”. До потребните нотификации се стигнува со помош на функцијата calculateNotifications(callLog, tariffStats) беше порепе објаснета како дел од имплементацијата на сервисот CheckOperatorService.

Рекламирање на widget е од голема корист. Бидејќи корисникот не може да го види во апликацијата, потребно е некако да се потенцира неговото постоење. Можноста да се објасни неговата функција преку Help копче може да не е могу ефективна бидејќи тоа не е најпопуларното копче во очите на корисниците. Добро решение е screenshot при рекламирањето на апликацијата на пазарот. Сепак за да ги потсетиме корисниците за неговото постоење добро е да се овозможи негова ефективна реклама во самата апликација.

Доколку услугата која се нуди преку него е привлечна во очите на корисниците, како што е случајот со оваа апликација, најдобра реклама претставува имплементација на неговата функција во самата апликација. Тоа е овозможено преку копчето “Прикажи нотификации” која се наоѓа на главниот екран. На овој начин widget-от претставува реклама самиот за себе. Можно подобрување е при клик на копчето да му се напомене на корисникот со некоја порака дека може постојано да ја мониторира состојбата на бесплатните услуги преку понудениот widget од апликацијата.

Креирањето на widget се овозможува преку следните чекори:

- Дефинирај layout датотека
- Дефинирај XML датотека (AppWidgetProviderInfo) која ги опишува својствата на widget-от
- Креирај BroadcastReceiver кој се користи за креирање на корисничкиот интерфејс на widget-от
- Внеси ја конфигурацијата на widget-от во AndroidManifest.xml

Ова се едноставни чекори кои треба да се пратат за креирање на widget-от. Битно е да се напомене дека дека обновување на содржината на widget-от настанува при секое приклучување на интернет со помош на android.net.conn.CONNECTIVITY_CHANGE и по изминување на автоматскиот тајмер од 30 минути.

3.6 Навигација

Навигацијата во оваа апликација е овозможена преку AppBar-от. AppBar-от е лоциран најгоре на екранот. Тој ги содржи името на екранот, иконата и акции кои може да се извршат. Дополнителна може да се искористи и за навигација низ апликацијата. За да се овозможи тоа потребно е да се заобиколи ограничувањето дека за навигациска цел тој е воведен во Android 3.0. Тоа е овозможено со помош на библиотеката ActionBar Sherlock library. Документацијата на оваа библиотека може да се најде на <http://actionbarsherlock.com>. Оваа библиотека е додадена во преку нов проект од типот Library. За да може да се пристапи до класите на овој проект потребно е преку тој да се додаде како библиотека на постоечкиот проект преку:

Десен клик на проектот > Properties > Android > Add > Селекција на проектот actionbarsherlock

Потоа потребно е да се адаптираат екраните од проектот да наследуваат од класата SherlockActivity. Пристап до ActionBar-от се добива со помош на методот getSupportActionBar(). Преку овој објект можеме да ги поставиме соодветните табови со функцијата actionBar.addTab(..). Навигацијата се овозможува со помош на методот onTabSelected(..) каде се стартува бараниот екран.

Оваа библиотека ги имплементира и најновите навигациски дизајни, но нејзината популарност е базирана можноста за навигација преку AppBar-от за мобилните оператори кои работат на постара верзија од Android 3.0.

4. Можни подобрувања

Успешноста на оваа апликација сеуште силно зависи од нејзините можни подобрувања. Иако основните функционалности се веќе имплементирани, за да се остварат целите на апликацијата потребно се подобрувања од дизајнерска и функционална гледна точка. Во оваа тема ќе ги наведам сите можни подобрувања кои се замислени за оваа апликација, нивното значење за нејзината успешност како и можните проблеми при нивната имплементација.

4.1 Дизајнерски подобрувања

Во психолошкиот дел од апликацијата е потенцирана важноста на добриот дизајн на секоја апликација. Мое мислење е дека дизајнот е од споредно значење се додека не се устварат посакованите функционални услуги. Затоа на овој дел од апликацијата не е сеуште обренто доволно внимание. Потребно за апликацијата е добро дизајнирана икона која ќе ги привлече корисниците и ќе успее да ја прикаже суштината на апликацијата. Иако навигацијата има добар дизајн можно подобрување е воведување на мали икони во замена на табови со текст за навигација. Исто така може да се воведат мала динамичка update икона во AppBar-от за приказ на корисниците дека нешто се извршува за време на чекање на одговор од веб сервисите.

На првиот екран за регистрација на тарифниот пакет на корисниците потребен е подобар дизајн за прикажување на нотификациите по акцијата на копчето “Прикажи нотификации”. Ова подобрување е потребно за да го привлече поефикасно вниманието на корисниците што има влијание врз успешноста на рекламата дека таа функција е овозможена преку widget од апликацијата.

На вториот екран покрај бројните потребни функционални подобрувања, потребен е и подобар дизајнерски изглед за историјата на воспоставени повици.

Дизајнот на последниот екран за статистика е од огромно значење. Веќе беше потенцирано значењето на успешен дизајн на графот. Иако графот е разбирлив, можни се подобрувања кои можат многу да влијаат врз популарноста на апликацијата.

Најголемо внимание од дизајнерски аспект потребно е да се обрне на widget-от. Бидејќи оваа апликација се наоѓа на homescreen-от на мобилниот оператор, нејзиниот добар дизајн е од голема важност. Услугите кои ги нуди претставуваат најголема атрактивност за корисниците, затоа потребно е да не ги разочараме од дизајнерски аспект.

4.2 Функционални подобрувања

За можните функционални подобрувања беа дадени неколку идеи во текот на оваа документација. Некои од нив се од суштинско значење за апликацијата и нејзината успешност.

Основно подобрување кое може да привлече голем број на корисници е мониторирање на податочниот сообраќај. Имплементацијата на ова решение е зависно од подобрување на документацијата за класата TrafficStats понудена од Андроид платформата за мониторирање на податочен сообраќај. Бидејќи оваа услуга е од голема атрактивност за корисниците, прифатливо е и привремено решение со нецелосна точност.

Воведување на splash екран. Потреба од splash екран се јавува кај апликации кои комуницираат со веб сервиси и доколку на нив се извршуваат комплексни пресметки. Овој екран би го подобрил корисничкиот интерфејс со тоа што ќе се намали времето за чекање на одредени услуги. Таму може да се направат сите пресметки потребни за тарифата на која се наоѓа корисникот, доколку таа била зачувана во процесот на корисничката регистрација. Оваа промена драстично би го забрзала корисничкиот интерфејс.

Воведување на корисни филтри на вториот екран. Бидејќи добивањето на историјата на воспоставени повици е услуга која му е достапна на корисникот, тука може да се воведат филтер кој би ги претставил само телефонските броеви кои биле префрлени на друг оператор. Доколку тие се сортирани според нивната вредност за корисникот, каде највреден е оној број кој воспоставил најголем бројна повици со корисникот, би се добил екран кој нуди интересна услуга за корисникот.

На вториот екран може да се најде и начин за поврзување на телефонските броеви, така што би се добиле повеќе податоци за тие броеви корисни за корисникот. Доколку тој број се поврзе со facebook може да се дојде до голем број на информации од кои може да се дојде до интересни идеи. Поврзувањето на телефонски број со facebook може да нарушува одредени сигурносни полиси, па затоа потребно е подобро да се информираме за оваа можност.

Доколку успешно се имплементира услугата за мониторирање на податочниот сообраќај, тие податоци може да се зачувуваат. По подолго користење на апликацијата истите може да се обработат на начинот на кој ги обработуваме излезните повици и SMS пораки и да се додадат во пресметките за месечните сметки за корисникот.

Причината поради која се прецртува графот во третиот екран, а не се овозможува паралелно споредување на повеќе тарифи е можното подобрување за компјутерско одредување на најсоодветниот тарифен пакет. Тоа може да се добие со пресметка на сметките за секоја тарифа. Тарифата која ќе ги даде најдобрите резултати, тарифата на која се наоѓа моментално корисникот и тарифата која тој ја селектира за споредба според мене би дала најдобра можност за одредување на најсоодветниот тарифен пакет за корисникот, а и би овозможила разбирливост на графикот.

5. Заклучок

Како што беше напоменато во воведот целта на оваа дипломска работа е да се создаде апликација која ќе ги енкапсулира најбараните услуги поврзани со тарифните модели.

Со помош на тие услуги би се дошло до можноста корисниците да имаат контрола врз трошковите кои ги имале во минатото и со помош на позадинската статистичка обработка да ги искористат за сопствена полза.

Не смее да се занемари и придобивката од контролата врз моменталните нивни трошкови, тоа би дозволило целосно да ги искористат бесплатните услуги на тарифата.

Кога моменталните трошкови/услуги ќе варираат од очекуваните, тие со помош на статистиката ќе го решат овој проблем.

Компатабилноста на овие услуги доведува до совршен баланс.

Тој баланс може да задржи голем дел од корисниците. А услугата за мониторирање може постојано да привлекува нови.

Затоа се надевам дека од технички/психолошки и дизајнерски аспект ќе се задоволат вкусовите на корисниците и ќе се префрлат овие услуги во практика.

6. Листа на референци

1. <http://www.vogella.com/android.html>, објаснувања на сите поими за андроид програмирање со понудени примери.
2. https://parse.com/docs/android_guide, база на податоци и сервиси кон нејзе
3. http://komuniciraj.mk/index.php?option=com_wrapper&view=wrapper&Itemid=115, листа на сите понудени тарифи како референца за базата на податоци
4. <http://thenewboston.org/list.php?cat=6>, андроид видео туторијали
5. <http://neilgoodman.net/2011/12/26/modern-techniques-for-implementing-rest-clients-on-android-4-0-and-below-part-1/>, асинхрон POST повик
6. <http://neilgoodman.net/2012/01/01/modern-techniques-for-implementing-rest-clients-on-android-4-0-and-below-part-2/>, асинхрон POST повик
7. <https://github.com/octo-online/robospice>, интент сервис, за разбирање на асинхрон начин на работа и животниот циклус на екран
8. <http://www.fourhourworkweek.com/blog/2012/04/22/how-to-build-an-app-empire-can-you-create-the-next-instagram/>, референца за маркетинг на апликација
9. <http://www.inc.com/aaron-aders/5-steps-to-building-a-million-dollar-app.html>, референца за маркетинг на апликација
10. <http://blogs.msdn.com/b/windowsappdev/archive/2012/07/11/creating-metro-style-apps-that-stand-out-from-the-crowd.aspx>, референца за успешност на апликација
11. <http://www.achartengine.org/index.html>, документација на библиотеката за претставување на граф
12. <http://actionbarsherlock.com/>, документација и примери за користење на библиотеката за навигација.