|  |
| --- |
| **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – София** |
| **ФАКУЛТЕТ Компютърни системи и управление** |

**ДИПЛОМНА РАБОТА**

**НА**

Александър Светославов Иванов, Фак. №121211134

***Специалност: Компютърни системи и технологии***

***Образователно-квалификационна степен:*** Бакалавър

***Тема: Android приложение за съхранение и споделяне на файлове***

**Научен ръководител:**

доц.д-р инж.Сергей Недев

София 2015 г.

Съдържание:

**Лист №**

**Първа глава.** Въведение и обзор

1.1Използване на интернет от частни лица …………………….……………………….……………………………………….4

1.2 Използване на облачни услуги……………………………………………………………….…………………………..…………6

1.3 Cloud Computing…………………………………………………………………………………………………………………………….11

1.4 Избор на среда за реализация на приложението……………………………………………………………………….15

**Втора глава.** Проектиране на приложението

2.1 Backend API …………………………………………………………….……………………………………………………………………..20

2.2 Requests and responses ……………………………………………………….…………………………………………………………23

2.3 Build System…………………………………………………………………………………………………………………………………….26

2.4 Crash/Issue tracking system…………………………………………………………………………………………………………….30

**Трета глава.** Програмна реализация

3.0 Програмна реализация ………………………………………………………….……………………………………………………..33

**Четвърта глава.** Ръководство за работа на потребителя

4.1 Login and Register …………………………………………………………………………………………………………………………..41

4.2 File/Folder Operations ……………….……………………………………………………..……………………………………………44

4.3 Menus …..……………………….……………………………………………………………………………………………………………….45

**Пета глава.** Експериментална част

5.0Source code на написаното програмно осигуряване…………………………………………………..……….…….48

***УВОД***

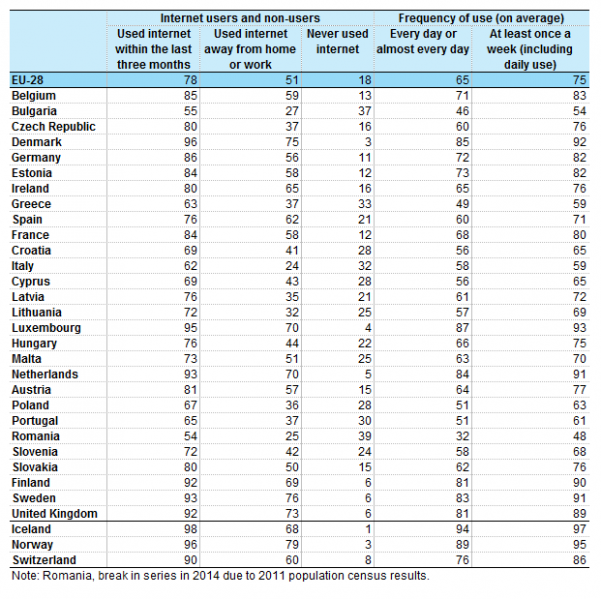
През 2014 г. половината от населението на ЕС на възраст между 16 и 74 години е използвало Интернет от преносими компютри или устройства чрез мобилна мрежа, или безжична връзка, когато не са вкъщи, или на работа. Около една шеста от европейците никога не са използвали интернет. Делът на „не-потребителите“ естествено варира значително между държавите-членки.

Същевременно през изминалата година един от всеки пет граждани на ЕС на възраст между 16 и 74 години е записал файлове в облак~~а~~. Най-високо потребителите на облачни~~те~~ услуги оценяват лекотата на достъпа до файлове от няколко устройства или места едновременно. Значителна част от населението обаче все още не е наясно със съществуването на облачни услуги, въпреки че са потребители на интернет. Сред тези интернет потребители, които са знаели, опасенията за сигурността и неприкосновеността на личния им живот са основен фактор, който им пречи да използват подобен тип услуги.

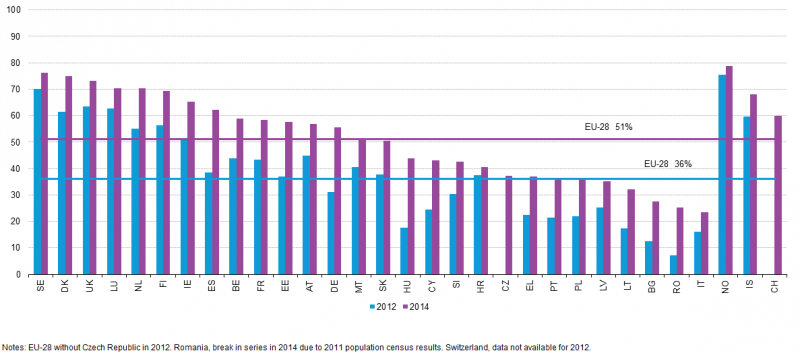
Настоящата дипломна работа се фокусира върху създаването на Android мобилно приложение за съхранение и споделяне на файлове и папки в крауд съръви. Приложението използва екприпшън за достъп и съхранение на данните на потребителите, като в същото време предлага удобен, лесен за употреба и user-friendly интерфейс.

**1.1 Използване на интернет от частни лица**

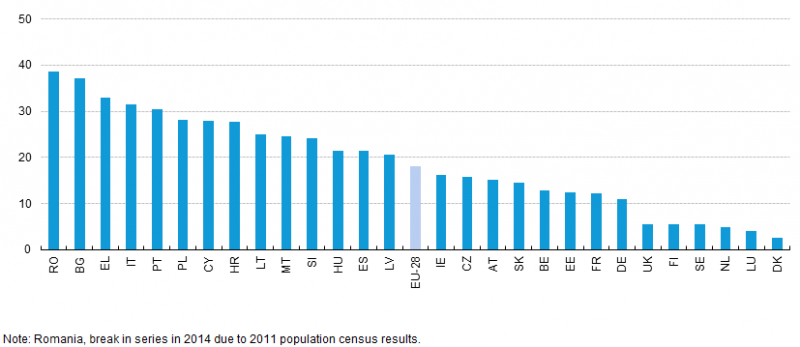
Почти четири от всеки пет лица (78%) в ЕС са използвали интернет поне веднъж през трите месеца преди проучването. Целта Digital Agenda на 75% от населението, които са използвали редовно интернет през 2015 г. (средно най-малко веднъж седмично, у дома, на работното място или другаде) е достигната през 2014 г. Делът на интернет потребителите, които са онлайн на дневна база е висок във всички държави-членки, както и в Исландия, Норвегия и Швейцария. Малко под две трети от всички граждани на Европейския съюз (65%) използват интернет ежедневно или почти всеки ден. Освен това, около половината от населението (51%) правят това в движение на преносими компютри или устройства.



Използването на мобилни интернет услуги в Европа отбелязва забележителен растеж между 2012 г. и 2014 г., скачайки с 16 процентни пункта. Най-бързият растеж се наблюдава в Германия, Естония, Испания и Унгария, където степента на използване се е увеличил с над 20 процента.



Делът на европейските граждани, които никога не са използвали интернет спада до 18% през 2014 г., и е само на три процентни пункта над целта за 2015 Digital Agenda - 15%.

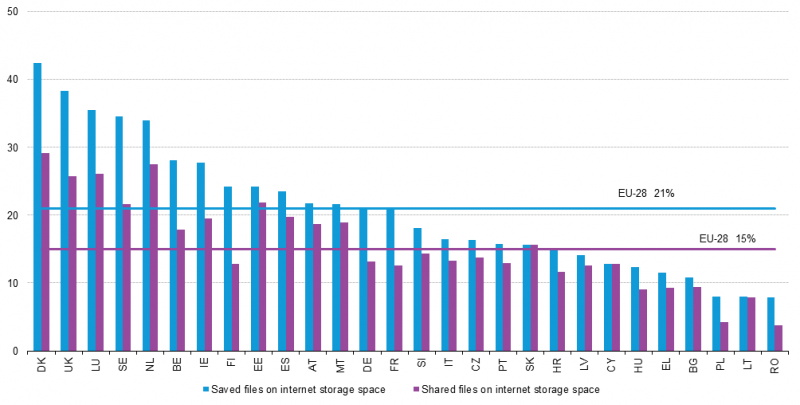


Въпреки това, значителното "цифрово разделение" остава с големи различия в отделните страни. Делът на населението, без опит в използването на интернет (независимо дали у дома, на работното място или някъде другаде) е най-висок в Румъния (39%), България (37%), Гърция (33%), Италия (32%) и Португалия (30%), а най-нисък - в Дания (3%), Люксембург (4%) и Холандия (5%).

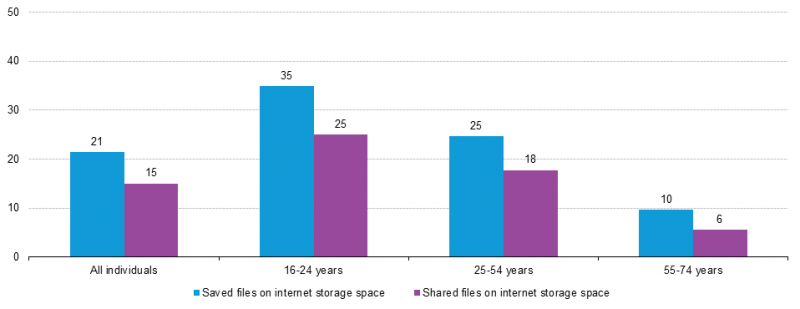
**1.2 Използване на облачни услуги**

Услуги, базирани на облачни технологии позволяват на потребителите да съхраняват големи файлове или да използват софтуер на сървър чрез интернет. Облачните услуги са сравнително ново явление в сравнение с уеб приложенията за социални мрежи, слушане на музика или гледане на филми. Едно от основните предизвикателства, когато се измерва използването на облачни услуги е в състояние да се направи ясно разграничение между тези и други онлайн услуги.

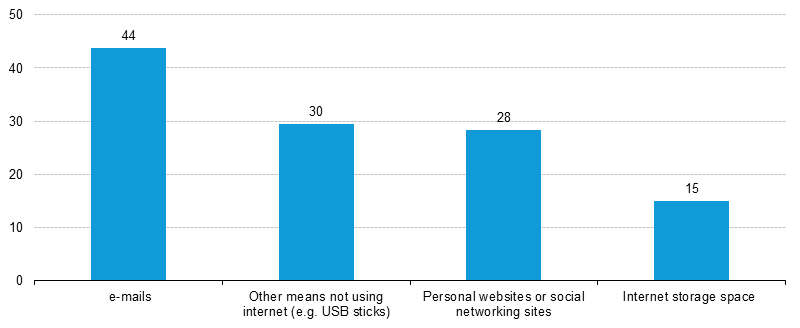
През 2014 г. 21% от населението на ЕС на възраст 16-74г. съобщават, че са използвали интернет пространство за да съхранят документи, снимки, музика, видео или други файлове. Internet пространството за съхранение може и да предостави възможности за споделяне на файлове с други хора. Процентът на хората в ЕС, които допълнително използват интернет пространството за съхранение и споделяне на файлове, е 15%.



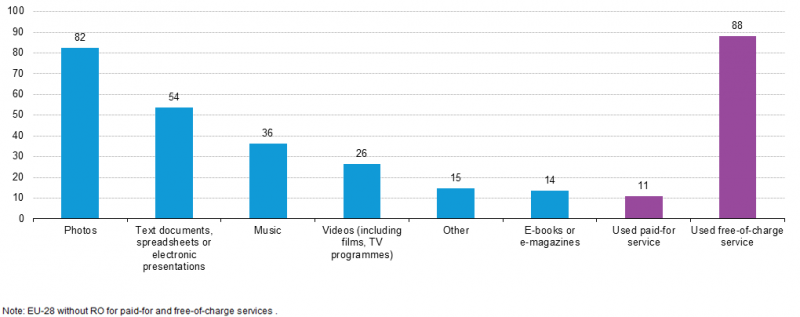
Държавата-членка, с най-висок дял на лицата, използващи облачни услуги за споделяне на файлове, е Дания (42%), следвана от Обединеното кралство (38%), Люксембург,Швеция (35%) и Холандия (34%). Същите страни, плюс Естония, също регистрират най-високите нива на използване на облачни услуги за споделяне на файлове (между 22% и 29%).По-малко от един на всеки десет лица в Полша, Литва и Румъния използват облачните услуги за споделяне на файлове. Около една трета от младите хора в ЕС споделят файлове чрез облачни услуги. Делът на младите хора на възраст 16-24 години които с помощта на облачните услуги споделят файлове е повече от три пъти по-висок от дела на 55-74 годишните, които се възползват от тези услуги (35% в сравнение с 10%). По същия начин, по-младото население прави по-широко използване на интернет пространството за споделяне на файлове.



Делът на 16-24 годишните, които споделят файлове през интернет е повече от четири пъти по-висок от процента да хората в 55-74 възрастова група.Облачни услуги,са използвани по-малко от социалните медии за споделяне на файлове, в сравнение с други начини за електронно споделяне на файлове.



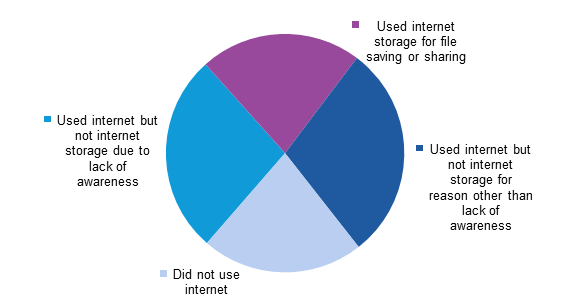
Докато само 15% от населението използва интернет пространство за съхранение и споделяне на файлове, 44% използват приложения за електронна поща да го направи и 30% използват USB флашки, DVD или Bluetooth. Снимки са най-популярният вид на файл за съхранение или споделяне чрез облачни услуги. Сред тези, които използват интернет пространството за съхранение, 82% съхраняват или споделят снимки, докато 54% ​​ съхраняват или споделят на текстови документи, електронни таблици или електронни презентации. Около една трета от потребителите в ЕС споделят музика, една четвърт видео файлове и един от седем електронни книги. Много облачни услуги се предоставят безплатно за потребителите, но в определен размер на съхранение на данни. Само един на всеки десет потребители облачните услуги избра да инвестира в съхранението или споделяне на файлове си.



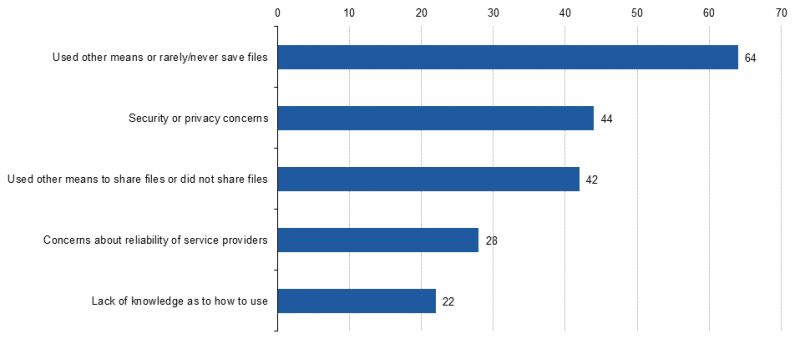
Лесният достъп до файлове от няколко устройства или места, е основната причина за използване на облака. Сред причините за използване на облачни услуги, три от всеки пет потребители на интернет пространството за съхранение, идентифицирани възможността за използване на файлове от няколко съоръжения или места (59%) и за споделяне на файлове с други лица лесно (59%). Освен това, повече от половината от потребителите смятат защита срещу загуба на данни като причина за използване на облака (55%). Потребителите също така оценяват голяма стойност на високата памет (44%) и състоянието за достъп до големи библиотеки от музика, филми или телевизионни програми (22%).



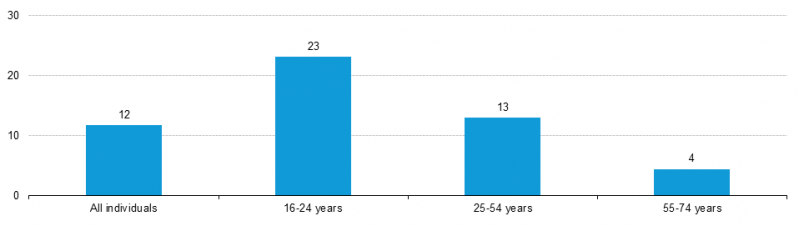
Над една четвърт от населението работят в интернет, но не знаят за облачните услуги. През 2014 г. 55% от населението на ЕС са използвали интернет, но не използват интернет пространството за съхранение или споделяне на файлове.



Опасенията за сигурността и неприкосновеността на личния живот остава основна пречка пред използването на облачни услуги. С поглед към сегмента на населението на тези, които са използвали интернет и са знаели за облачните услуги, но не ги използва, 44% са посочили проблеми със сигурността или поверителността, като причина да не се възползват от тези услуги, 28% споделят притеснения относно надеждността на доставчиците на услуги и 22% липсата на професионални умения. По-голямата част (64%) споделят файлове от собствените си устройства или имейл акаунти. Един на всеки десет лица, използват софтуер през интернет за редактиране на документи, електронни таблици и презентации.



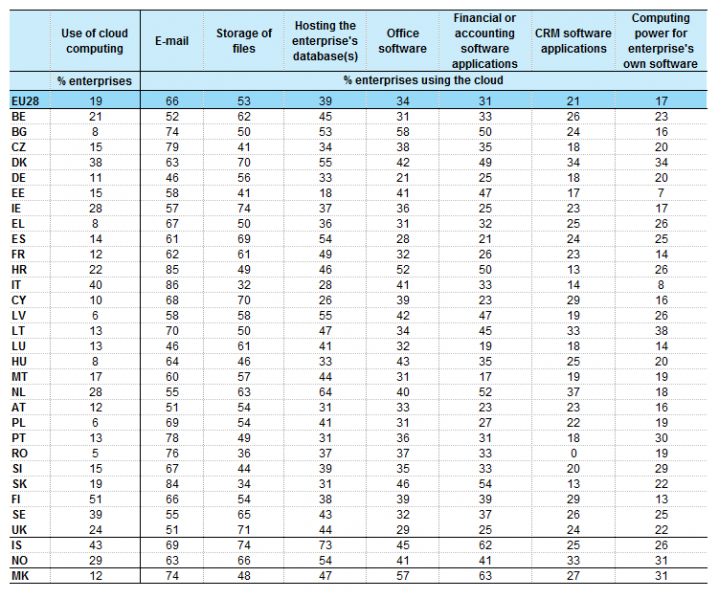
Облачните услуги могат да дадат на потребителите достъп до софтуер за редактиране на текст, таблици или презентации. През 2014 г. 12% от населението на ЕС се възползва от такъв софтуер. Процентът е по-висок сред 16-24-годишните - 23%, отколкото сред по-възрастните групи, 13% от 25-54-годишните и 4% от 55-74-годишните използват на такъв софтуер.



**1.3 Cloud Computing**

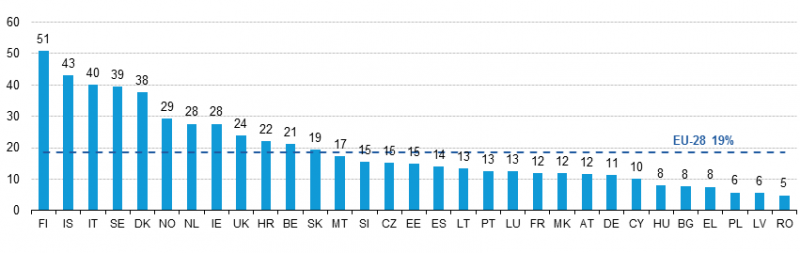
Cloud Computing и сторидж решенията предоставят на потребителите и предприятията с различни способности да съхраняват и обработват данните си в центрове за данни на трети страни. Те разчита на споделянето на ресурси за постигане на съгласуваност и икономии от мащаба, подобно на полезност (като електричеството) през мрежа. В основата на изчисленията в облак е по-широката концепция на пакетна инфраструктура и споделени услуги.

Cloud Computing, или просто "облака", се фокусира върху увеличаване на ефективността на споделени ресурси. Cloud ресурсите обикновено са не само споделени от няколко потребители, но също са динамично преразпределени в зависимост от търсенето. Това може да работи за разпределянето на ресурси между потребителите. Например, един облак (компютър съоръжение), което служи на европейски потребители през европейските бизнес часове с конкретно приложение (например, електронна поща) могат да преразпределят едни и същи ресурси, за да служи на американските потребители в рамките на работното време на Северна Америка с различно приложение (например, уеб сървър). Този подход се използва за да се максимизира използването на изчислителната мощност като по този начин се намалява екологичните щети, както и от по-малкото енергия, климатик, пространство и т.н., който са необходими за различните функции. С изчислителни облаци, много потребители имат достъп до един сървър за извличане и актуализиране на своите данни, без закупуване на лицензи за различни приложения.



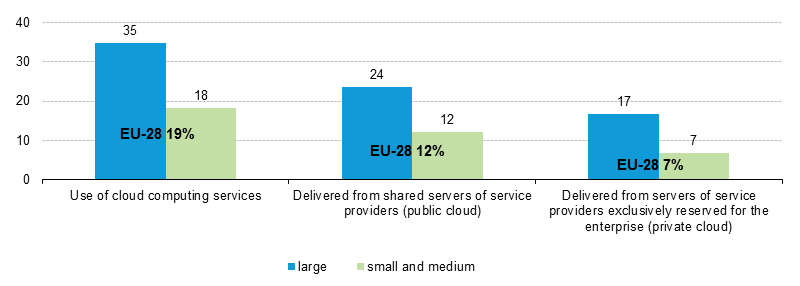
Произходът на терминът облак е неясно. Облакът от израз често се използва в областта на науката, за да опише голяма конгломерация на обекти, които визуално изглеждат от разстояние като облак и описва всеки набор от неща, чиито данни не се проверяват допълнително в даден контекст. Друго обяснение е, че старите програми, които изготвят мрежови схеми, заобиколени от иконите за сървъри с един кръг, и клъстер от сървъри в мрежова схема имат няколко припокриващи се кръгове, които приличат на облак.

По аналогия с използването на думата, облакът се използва като метафора за интернет и стандартизирането на облака с форма се използва за обозначаване на мрежа на телефонни схеми. С тази цел опростяване, изводът е, че спецификата на това как са свързани крайните точки на мрежата не са от значение за целите на разбирането на диаграмата. Символът облак се използва за представяне на интернет още през 1994, в който сървъри биват показани свързани, но външни на облака. Позоваванията на изчислителни облаци в съвременния смисъл на думата се появи най-рано през 1996 г., като най-ранните известни споменавания са в вътрешен документ Compaq.   
Популяризирането на термина може да се проследи до 2006 г., когато Amazon.com въвежда Elastic Compute Cloud.



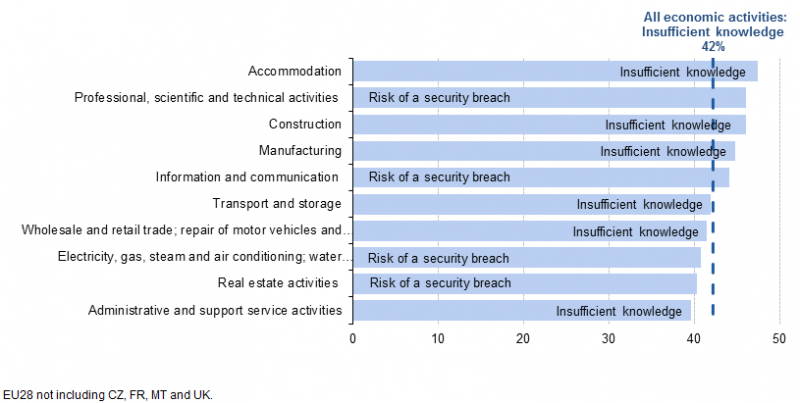
Облачните услуги поставят притеснения когато се отнася до защита на личните данни, тъй като доставчикът на услугата може да получи достъп до данните, които са на облака по всяко време. Той може случайно или умишлено да промени или дори изтрие информация. Много доставчици на облачни могат да споделят информация с трети страни, ако е необходимо за целите на закона и реда, дори без заповед. Това е допустимо в техните декларации за поверителност, които потребителите трябва да се съгласят, преди те да започнат да използват облачни услуги. Решения на неприкосновеност на личния живот включват политика и законодателство, както и избор на крайните потребители за това как се съхраняват данните си. Потребителите могат да криптира данните си, които се обработват или съхраняват в облака, за да се предотврати неоторизиран достъп.

Според Алианса за сигурност на Cloud, трите най-големи заплахи в облака са "несигурни интерфейси - API ", "Data Loss & изтичане на данни" и "хардуерен срив", които възлизат съответно на 29%, 25% и 10% от всички нарушения на сигурност в облака. Заедно те представляват споделените уязвимости на технологията. В облака доставчика на платформата споделя с различни потребители и може да има възможност, че информацията, принадлежащи към различни клиенти се намира на същия сървър на данни. Ето защо, изтичане на информация може да възникне по погрешка, когато информация за един клиент се дава на друг. Освен това, Eugene Шулц, главен технологичен директор на Emagined Security, каза, че хакерите са вложели значително време и усилия, за да намерят начини да проникнат в облака. "Има някои реални Ахилесови пети в облак инфраструктурата, които правят големи дупки за лошите момчета да влязат ". Тъй като данните от стотици или хиляди фирми може да се съхранява на големи клауд сървъри, хакерите могат теоретично да получат контрол над огромни количества информация чрез една единствена атака - процес, който се нарича "hyperjacking".



Много Условия за споразумения за услуги мълчат по въпроса за собствеността. Физически контролът на компютърната техника (частен облак) е по-сигурна, отколкото с оборудването извън обекта и под контрола на някой друг (обществения облак). Това осигурява чудесен стимул за публични клауд компютинг услуги, който да дадат на доставчиците стимул за изграждането и поддържането на силно конфигуруеми и защитени услуги. Някои малки фирми, които нямат опит в ИТ сигурността може да открият, че това е по-безопасно за тях да използват обществения облак.

Съществува риск, че крайните потребители не разберат всички въпроси, свързани с подписването на облак услуги (лица, понякога не четат многото страници на условията на граждански договор, и просто натиснете "Accept", без четене). Това е важно , защото изчислителни облаци се превръщат в популярни и се изискват за някои услуги за да работят, като например за един интелигентен личен асистент (на Apple Siri или Google Now).



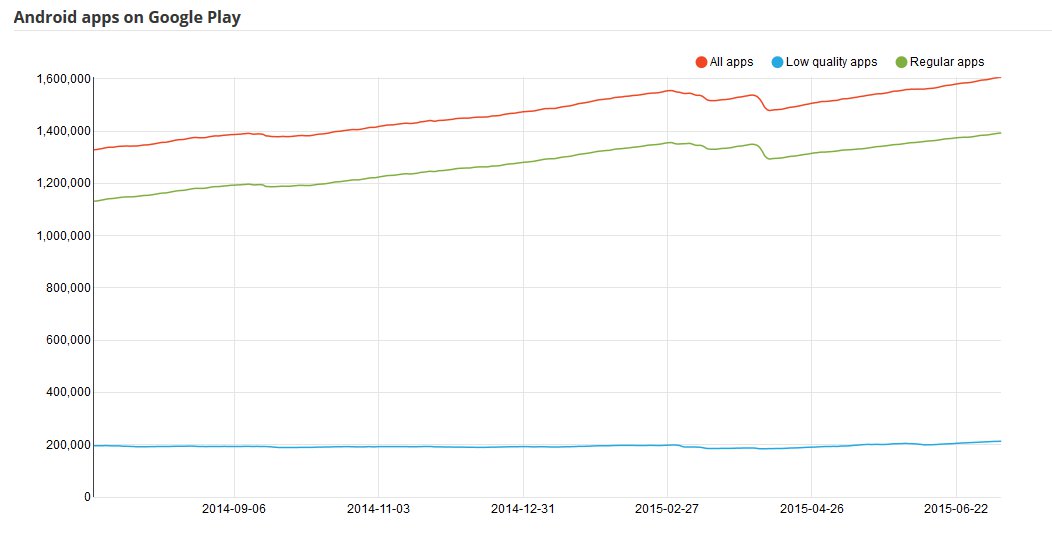
Фундаментално частния облак се възприема като по-сигурен и с по-високи нива на контрол от собственика,но обаче обществения облак се възприема като по-гъвкав и изискващ по-малко време и пари на инвестиции от страна на потребителя.

**1.4 Избор на среда за реализация на приложението**

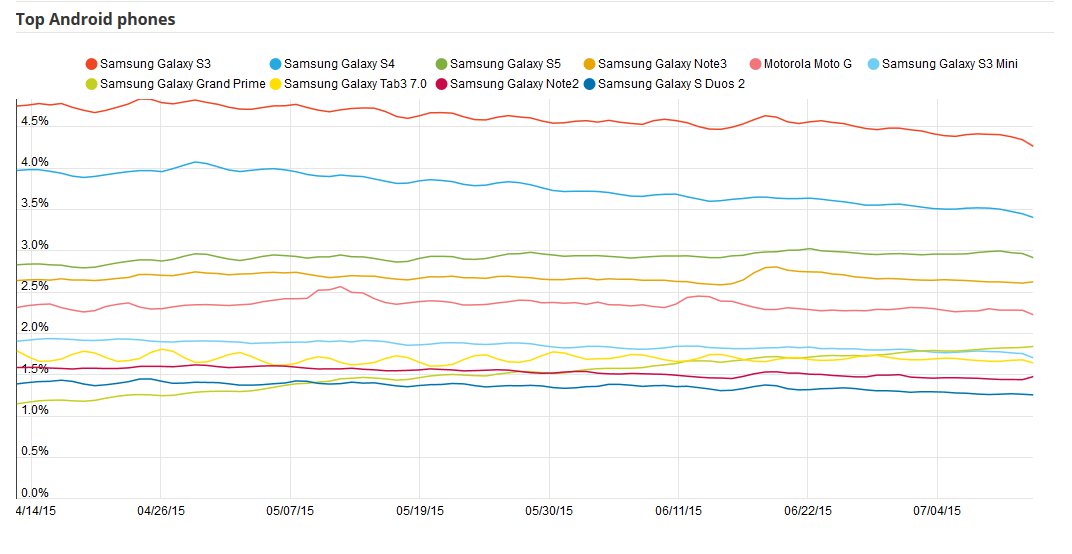
От юли 2014 г., Android е най-популярната мобилна операционна система, когато става дума за сърфиране в интернет, според NetApplications. В Индия и други страни, Android е най-популярната операционна система като цяло (не само да броим "мобилен" такива) за използване уеб браузване. Според StatCounter, "Използването на мобилни услуги вече застига десктоп в няколко страни, сред които Индия, Южна Африка и Саудитска Арабия", и няколко държави в Африка, които са го направили, включително Етиопия и Кения, в които използването на мобилни устройства е 72.23%.

Докато Android телефоните на запад обикновено включват патентовани добавки на Google (като Google Play) към операционната система, все още това не се прави в развиващите се пазари. ABI Research твърди, че 65 милиона са доставените в световен мащаб Android устройства през второто тримесечие на 2014, в сравнение с 54 милиона през първото тримесечие. В зависимост от държавата, процента на телефоните базирани на Android (AOSP), но не на Android марката са: Тайланд (44%), Филипините (38%), Индонезия (31%), Индия (21%), Малайзия (24%), Мексико (18%), Бразилия (9%).

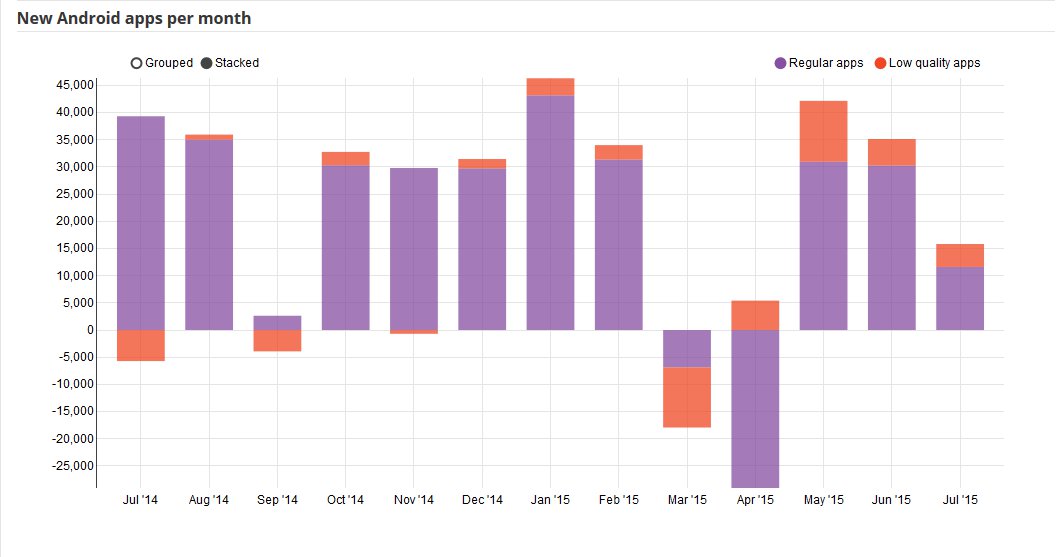
Според доклад от януари 2015 на Gartner, "Android надмина границата от един милиард устройства през 2014 г., и ще продължи да расте с двуцифрени темпове през 2015 г., с 26% увеличение на годишна база." Това го прави за първи път, че всяка операционна система с общо предназначение е достигнал повече от един милиард крайни потребители в рамките на една година: с достигането близо до 1160000000 крайните потребители през 2014 г. Android устройствата са над четири пъти повече от IOS и OS X обединени заедно, и над три пъти повече, отколкото Microsoft Windows. Gartner очаква целия пазар на мобилни телефони да "достигне два милиарда единици през 2016 г.", включително и Android.



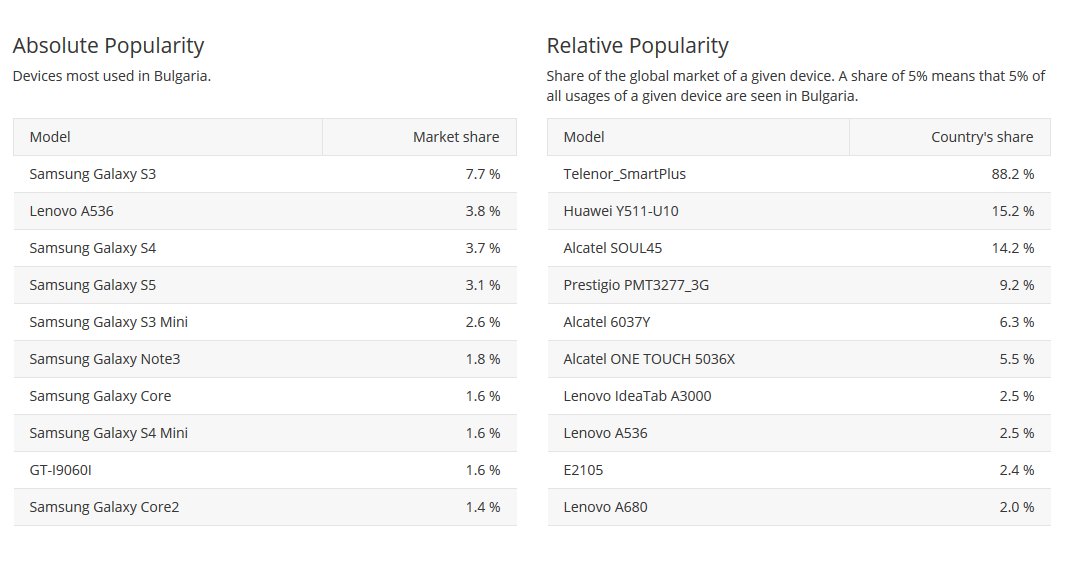
По оценка на СтатСофт, през 2014 г. Android смартфоните имат инсталирана база от 1,6 милиарда смартфона, което е 75% от броя на смартфоните в световен мащаб. Android има най-голямата инсталационна база от всяка мобилна операционна система , считано от 2013 г., и е операционната система с най-големи продажби през 2012 г., 2013 г. и 2014 г. , близо до инсталираната база на всички компютри. През третото тримесечие на 2013 г., делът от световния пазар на смартфони за Android е 81,3%, най-високият, и най - много Android устройства са на Samsung - 81,3%.



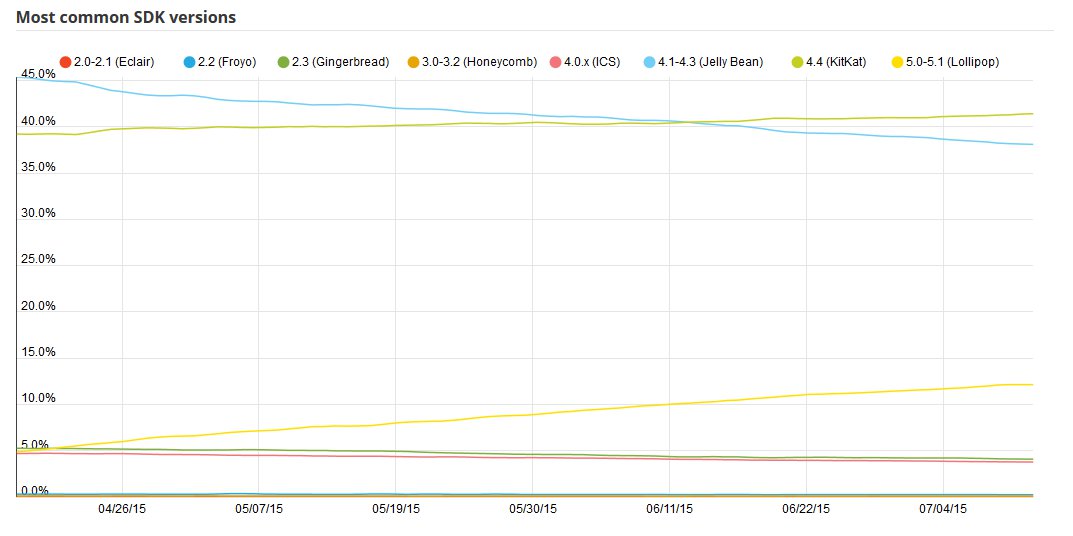
Сорс кодът за Android е с отворен код. Той се развива в частност с Google като изходния код се публикува публично, когато нова версия на Android е пусната. Google публикува повечето от кода (включително мрежови и телефонни стекове) под Apache License версия 2.0, която позволява промяна и преразпределение. Лицензът не предоставя права над "Android" търговската марка, така че производителите на устройства трябва да го лицензира през Google чрез индивидуални поръчки. Промените по Linux ядрото са освободени под Copyleft GNU General Public License версия 2, разработена от Open Handset Alliance, като кода е публично достъпен по всяко време. Обикновено, Google си сътрудничи с производителите на хардуер, за да се произведе флагман устройство (част от серията Nexus) с участието на новата версия на Android, а след това предоставя изходния код на разположение след това устройство е било пуснато. Единствената Android версия, която не е била незабавно на разположение, като изходен код е таблетната 3.0 Honeycomb. Причината, според Andy Rubin в официален Android блог пост, е защото Honeycomb е изпратена за производство на Motorola Xoom, и те не искат трети лица, създаващи "наистина лош потребителски опит", да се опитат да поставят върху смартфони версия на Android, която е предназначена за таблети.



Само на база Android като операционна система (включително някои приложения) е софтуер с отворен код, докато повечето Android устройства са със значителна сума от собствен софтуер, като например Google Mobile Services, която включва приложения като Google Play Store, Google Search и Google Play услуги с софтуерен слой, който осигурява APIs за интеграция с Google-предоставяни услуги. Тези приложения са лицензирани от Google за създателите на устройството, и могат да бъдат инсталирани само на устройства, които отговарят на нейните насоки за съвместимост и други изисквания. По поръчка, сертифицирани дистрибуции на Android, произведени от производители (като TouchWiz и HTC Sense) може също така да заменят някои фондови Android приложения с техните собствени варианти като се добавя допълнителен софтуер който не биват включени в операционната система или в Android Store. Също така могат да биват включени и "двоичен Blob" драйвери, необходими за някои хардуерни компоненти в устройството.



Няколко приложенията в отворен изходен код на Android, използвани от предишните версии (като търсене, музика и календар) също са били ефективно отхвърлени от Google, с развитие като се преориентирали към по-нови, но патентовани версии разпространени и актуализирани чрез Play Store, като например Google Search и Google Play Music. Въпреки тези по-старите приложения остават в изходния код на Android, като те вече не получиват никакви сериозни актуализации. Освен това, патентовани варианти на приложенията за предната камера и галерията включват и определени функции (като фотосферата с панорами и интеграция към Google+ албум), които са изключени от версиите с отворен код. По същия начин, Nexus 5 използва несвободна вариация на Android 4.4 за началния си екран, който е вграден директно в приложението за търсене на Google, като добавя гласово активирано търсене и възможността за достъп до Google Now като страница в главния екран. Въпреки актуализацията за Google Search приложението, съдържащите съответни компоненти са пуснати в Google Play за всички Android устройства, като за новият начален екран се изисква допълнителен widget за да функционира, и не е била предоставена в Android 4.4 актуализацията за всички други устройства (които все още се използва за съществуващия начален екран от Android версия 4.3). Приложението е официално пуснато за разпространение в Google Store като Google Now Launcher през февруари 2014 г., първоначално за Nexus и Google Play Edition устройства с Android версия 4.4.



Ричард Столман и Фондацията за свободен софтуер са били критични към Android и препоръчаха използването на алтернативи като Replicant, защото водачите и фърмуер от жизненоважно значение за правилното функциониране на Android устройства обикновено са патентовани и защото Google Play може насила да инсталирате или деинсталирате приложения и приканва несвободен софтуер.

2.1 **Backend API**

pCloud е услуга за съхраняване на файлове и синхронизация, предоставена от pCloud AG, издаден на 13 септември, 2013 г. Услугата дава възможност на съхранение, споделяне на файлове, архивиране на данни и потребителското сътрудничество в облака. Към юни 2015 г., pCloud има над 2,0 милиона регистрирани потребители от цял свят.pCloud предлага на своите потребители първоначалено безплатно пространство за съхранение на до 20 GB при регистрация (10GB първоначално пространство плюс още допълнително до 10 през нейната схема за разрастване на акауни). Потребителите могат да ъпгрейднат свободното си пространство чрез изплащане на месечно абонаментен план,чрез който може да се получи допълнително пространство за съхранение.



pClouds не налага никакви ограничения на размера на файла на своите потребители. По принцип, потребителите могат да качват файлове от всякакъв размер през уеб платформата, толкова дълго, колкото пространство им за съхранение позволява. pCloud също не налага никакви ограничения върху скоростта на качване или сваляне. Скоростта, в която потребителите ще качват или свалят файловете им изцяло зависи от скоростта на интернета предлагана от доставчика им. Функцията за качване чрез линк е инструмент, чрез който потребителите могат да получават всички видове файлове, онлайн, директно в техните pCloud акаунти от други потребители. За да използват тази функция, потребителят трябва да създаде линк за качване и да го предостави на други потребители, така че те да могат да качват свои файлове в профила на потребителя. pCloud позволява лесно споделяне на файлове и папки между различните потребители. Какво потребители трябва да направите, за да споделят своите цифрови данни е да генерира линк към файла или папката и да я изпрати на други потребители чрез имейл, месинджър или социална платформа, като Facebook или Twitter. Потребителски разрешения могат да бъдат определени при споделяне на линк към файлове или папки. pCloud подкрепя също така използването на файлове в режим офлайн. За да се възползват от тази функция, потребителите трябва да маркират своите файлове като любими. По този начин те се изтеглят на тяхното устройство и са готови за употреба дори и в офлайн режим. Чрез backup функционалността pCloud може автоматично да архивира социално медийното съдържание от Facebook, Instagram и Picasa акаунти директно към техния pCloud акаунт. Като допълнителна средство за backup функция, pCloud позволява на потребителите да архивират съдържанието на Dropbox акаунта си автоматично към техния pCloud профил. Всички потребителски файлове се копират и се съхраняват най-малко на три сървъра в силно защитен център за данни. Платформа за съхранение на pCloud използва протокол TLS / SSL за криптиране, за да се гарантира защитата на личните данни на своите потребители ".

## В общите случаи (например при уеб и мобилно приложение) най-използваният метод за отдалечен достъп до услуги е чрез използването на интерфейс HTTP / HTTPS. Такава услуга използва и REST pCloud API заявките. Услугата REST *(на английски:* *Representational state transfer)* представлява разпределителна системна рамка, базирана на уеб протоколи и технологии. Архитектурният модел Rest включва взаимодействията между сървър и клиент, осъществени по време на трансфера на данни. Уеб е най-мащабната имплементация на REST.

Архитектурният стил на "REST" прилага шест ограничителни условия, като същевременно дава свобода за дизайна и имплементацията на индивидуалните компоненти:

Клиент-сървър

Единният интерфейс разделя клиента и сървъра. Това означава, например, че клиента не се грижи за складирането на данни. Тази задача остава изцяло за сървъра, като по този начин се подобрява портативността на клиентския код (може да се използва в различни среди). Сървърът няма връзка с потребителския интерфейс и по този начин е по-семпъл и лесен за премащабиране. Клиентът и сървърът могат да бъдат заменяни или развивани независимо един от друг, стига това да не налага промяна на единния интерфейс помежду им.

Без статус на сесията *(на английски:* *Stateless)*

Следващото условие е на сървъра да не се запазват статуси на сесиите. Всяка заявка от клиента, съдържа в себе си нужната информация за нейната обработка, статуси на сесии се запазват единствено при клиента.

Кеширане

Клиента има право да кешира (запазва) информация, получена в отговор от сървъра, за да я преизползва при последващи заявки. За тази цел сървъра трябва имплицитно или експлицитно да е посочил дали информацията в отговора може да се кешира, за да се избегнат случаи, в които клиента получава грешна информация при бъдещи заявки. При правилно управление и използване на кеширането могат частично или напълно да се елиминират ненужни взаимодействия между клиента и сървъра, като по този начин се подобрява бързината и производителността.

Многослойна система

Обикновено клиентът не знае дали е свързан с крайния сървър или със сървър-посредник. Сървърите-посредници подобряват ефективността, като увеличават капацитета за обработване на заявки и предоставят споделени кешове. Също така те допринасят да подобряването на сигурността.

Код при поискване (незадължително)

Сървърът може временно да разшири функционалността, изпращайки код, който се изпълнява директно при клиента. Например клиентски скриптове, написани на JavaScript или компилирани компоненти като Java applets.

Единен интерфейс

Единният интерфейс между клиента и сървъра разделя и опростява архитектурата. По този начин всеки компонент може да се развива самостоятелно.

Единственото условие на REST архитектурата, което не е задължително е "Код по поискване". Всяко приложение (услуга), изпълняващо на гореописаните условия, може да се нарече "RESTful". Ако нарушава дори едно от условията, то не може да бъде считано за "RESTful".

Всяка разпространена хипермедийна система, съответстваща на архитектурния стил на "REST" притежава нужната производителност, мащабируемост, опростеност, гъвкавост, видимост, портативност и надеждност

RESTful уеб API (също наричано RESTful уеб service) е уеб приложение,което използва принципите на HTTP и REST. Представлява колекция от ресурси със четири дефинирани аспектa:

* Основният "URL" за уеб приложенията като https://api.pcloud.com/userinfo?
* Internet media типът на данните поддържани от уеб приложенията. Това най-честo е JSON, но може да бъде всеки друг валиден Интернет медиен тип, като се има предвид, че е валиден хипертекст стандарт.
* Операции поддържани от уеб приложението използвайки HTTP методи (примерно: GET, PUT, POST, или DELETE).
* Приложенията трябва да се задвижват от хипертекст.

2.2 **Requests and responses**

Част от използваните методи са както следва:

\* userinfo

\* createfolder

\* listfolder

\* renamefolder

\* deletefolderrecursive

\* downloadfile

\* renamefile

\* register

\* logout

\* getfilepublink

\* getfolderpublink

Името на метода се предава като request path (например можете да направите заявка, за да се извика list метода). Параметрите на дадена заявка се предават чрез GET или POST методите.

## Отговор

### Отговор обикновено е JSON масив с няколко полета. Ключът за резултатът е винаги налице.

### Successful response

result:0 means success.  
  
Successful response example:

{

result: 0,

RESPONSE BODY

}

### Error response

Не нулевия отговор означава грешка.

Резултат е представен като ключ за грешка и съобщение за грешка в ключа грешка. След всяко определение за метод съществува списък на възможните кодове на грешки, които могат да бъдат върнати. Когато има грешка HTTP header X-Error: хххх е изпратен заедно с кода на грешката.  
  
Пример за върната грешка:

{

result: "error code",

error: "error message"

}

JSON, или JavaScript Object Notation, е текстово базиран отворен стандарт създаден за човешки четим обмен на данни. Произлиза от скриптовия език JavaScript, за да представя прости структури от данни и асоциативни масиви, наречени обекти. Въпреки своята връзка с JavaScript, това е езиково независима спецификация, с анализатори, които могат да преобразуват много други езици в JSON.

Форматът на JSON първоначално е бил създаден от Дъглас Крокфорд (Douglas Crockford) и е описан в RFC 4627. Официалният Интернет медия тип за JSON е application/json. Разширението на файловете написани на JSON е .json.

Форматът на JSON често е използван за сериализация и предаване на структурирани данни през Интернет връзка. Използва се главно, за да предаде данни между сървър и Интернет приложение, изпълнявайки функциите на алтернатива на XML.

Базовите типове данни на JSON са:

* Number (число с плаваща запетая, double precision floating-point format в JavaScript)
* String (Низ от символи с Unicode кодиране, затворени в двойни кавички, като „специалните“ символи се представят с т.н. escaping - символни последователности, започващи със символа "\")
* Boolean (true или false)
* Array (наредена поредица от стойности, разделени със запетая и затворени в квадратни скоби; стойностите не е задължително да бъдат от един и същ тип)
* Object (неподредена колекция от двойки ключ:стойност, символът ":" разделя ключът и стойността, разделени със запетая и затворени в къдрави скоби; ключовете трябва да са string-ове и да са различни един от друг)
* null (empty)

Всяко незначимо бяло пространство може да бъде добавено около „структурните символи“ (като скоби “{} []”, двоеточие ‘:’ и запетаи ‘,’).

Следващият пример показва представянето на обект, който описва човек в JSON. Обектът има string полета за първо и последно име, Number поле за години,Object, който представя адресът на човека и Array от телефонни номера представени като Object.

{

"emailverified": true,

"cryptosubscription": false,

"quota": 11811160064,

"cryptosetup": false,

"result": 0,

"userid": 1635486,

"premium": false,

"usedquota": 173470281,

"language": "en",

"business": false,

"email": "…………………………………………",

"registered": "Wed, 01 Apr 2015 16:42:47 +0000"

}

{

"result": 0,

"metadata": {

"path": "\/",

"name": "\/",

"created": "Fri, 03 Apr 2015 14:57:54 +0000",

"ismine": true,

"thumb": false,

"modified": "Fri, 03 Apr 2015 14:57:54 +0000",

"id": "d0",

"isshared": false,

"icon": 20,

"isfolder": true,

"folderid": 0,

"contents": [

{

"name": ".Trash-0",

"created": "Wed, 20 May 2015 17:21:40 +0000",

"ismine": true,

"thumb": false,

"modified": "Thu, 04 Jun 2015 12:12:44 +0000",

"comments": 0,

"id": "d29844535",

"isshared": false,

"icon": 20,

"isfolder": true,

"parentfolderid": 0,

"folderid": 29844535,

"contents": [

{

.

.

.

2.3 **Build System**

Gradle е приложение за автоматизиране на проекти, чиято цел е да сглоби(build) даден проект, използвайки концепциите на Apache Ant и Apache Maven, като също така добавя e и базиран на езика Groovy - DSL вместо традиционния и широко използван XML формат за деклариране на конфигурациите към проекта.

За разлика от Apache Maven, който дефинира жизнен цикъл и Apache Ant, където редът на изпълнение на задачите(targets) се определя от зависимости (depends-on), Gradle си служи с директен ацикличен граф (*DAG*), за да се установи реда, в който отделните задачи (tasks) ще бъдат изпълнени.

Gradle е проектиран за проекти, които се състоят от много билдове(multi-project builds), чиято големина може да се увеличи с времето и да станат значителни по обем и времемки за сглобяване всеки път наново. Поради тази причина той поддържа инкрементални(incremental) билдове. Чрез интелигенто установяване кои части от дървото на билда не са променяни, Gradle съкращава значително времето за билдване, тъй като задачите, които зависят от тях няма нужда да бъдат изпълнявани отново, тъй като дефакто кодът за тях не е променян.

Build конфигурациятя за проекта се определя вътре в build.gradle файловете, които са текстови файлове, които използват синтаксиса и опциите от Gradle и Android приставката, за да настроят следните аспекти на build-a:

Build variants:

Системата за билдване може да генерира множество APK файлове с различени продукт и и конфигурации за един и същ модул. Това е полезно, когато се изгражда различни версии на едно приложение, без да се налага да се създават отделни проекти или модули за всяка версия.

 Dependencies:

Build системата управлява dependencies в проекта и подкрепя dependencies от локалната файлова система и от отдалечени repositories. Това позволява да не се налага да се търси, изтегли и копира binary файлове за всяко dependency в директорията на проекта.

Manifest entries:

Системата за билдване дава възможност за задаване на стойности за някои локални променливи в manifest файла в конфигурацията на билдване. Тези стойности override-ват съществуващите в manifest файла. Това е полезно, ако искате да генерирате множество APK файлове за модули, където всеки от APK файловете има различно име на приложение, минималното SDK версия, или целева SDK версия. Когато множество манифести са налице, manifest настройките се сливат в приоритет на buildType и productFlavor, основния manifest и manifest-а с използваните библиотеки.

Подписване:

Системата за билдване дава възможност да се зададат настройките за подписване в конфигурацията на билдването , и тока може да подпишат APK файлове по време на процеса на билдване.

ProGuard:

Системата за билдване също така дава възможност да се зададат различни правила за ProGuard файла за всеки вариант на билда. Системата за билдване може да работи с ProGuard за да се изчистят класове по време на процеса на билдване.

Тестване:

За повечето templates, системата за билдване създава тест директория, androidTest и генерира тест APK от source директорията на проекта, така че да не е нужно да се създаде отделен тест проект. Системата за билдване може да изпълни тестове по време на процеса на билдване.

Gradle билд файловете използват Domain Specific Language (DSL), за да се опише и да се манипулира билд логиката чрез Groovy синтаксис. Groovy е динамичен език, който може да се използва, за да се определи custom build logic и да си взаимодейства с Android-специфични елементи, предоставени от Android плъгина за Gradle.

buildscript {  
 repositories {  
 maven { url **'https://maven.fabric.io/public'** }  
 }  
  
 dependencies {  
 classpath **'io.fabric.tools:gradle:1.+'** }  
}  
apply plugin: **'com.android.application'**apply plugin: **'io.fabric'**repositories {  
 maven { url **'https://maven.fabric.io/public'** }  
}  
  
  
android {  
 compileSdkVersion 22  
 buildToolsVersion **"22.0.1"** packagingOptions {  
 exclude **'META-INF/DEPENDENCIES'** exclude **'META-INF/NOTICE'** exclude **'META-INF/LICENSE'** }  
  
 defaultConfig {  
 applicationId **"com.chest.blue.filechest"** minSdkVersion 16  
 targetSdkVersion 22  
 versionCode 1  
 versionName **"1.0"** testInstrumentationRunner **"android.support.test.runner.AndroidJUnitRunner"** }  
 buildTypes {  
 release {  
 minifyEnabled **false** proguardFiles getDefaultProguardFile(**'proguard-android.txt'**), **'proguard-rules.pro'** }  
 }  
}  
  
dependencies {  
 compile fileTree(include: [**'\*.jar'**], dir: **'libs'**)  
 compile files(**'lib/httpclient-4.4.jar'**)  
 compile files(**'lib/httpmime-4.4.jar'**)  
 compile files(**'lib/json-simple-1.1.1.jar'**)  
 compile files(**'lib/org.apache.commons.codec-1.8.jar'**)  
 compile files(**'lib/org.apache.commons.httpclient.jar'**)  
 compile(**'com.crashlytics.sdk.android:crashlytics:2.4.0@aar'**) {  
 transitive = **true**;  
 }  
 compile **'com.android.support:appcompat-v7:22.2.0'** *//22.1.1* compile **'com.google.android.gms:play-services:7.3.0'** *// TEST ONLY DEPENDENCIES* androidTestCompile **'com.android.support.test.espresso:espresso-core:2.2'** androidTestCompile **'com.android.support.test:runner:0.3'** androidTestCompile **'com.android.support.test.espresso:espresso-contrib:2.2'**}

Всяка версия на приложението е представен в системата за билдване от билд варианти. Билд вариантите са комбинации от product flavors и build types. Product flavors представляват натрупване на продукт версии на един вариант, като например безплато и платено приложение. Build types представляват версиите пакети, генерирани за всеки пакет, като например debug и release. Системата за билдване генерира APK файлове за всяка комбинация от flavors на продукта и build types.

По подразбиране Android Studio определя стандартните настройки за конфигурация, defaultConfig във файла build.gradle, и два вида build types (debug and release). Това създава два билд варианта, debug and release, и системата за билдване генерира APK за всеки вариант. Добавянето на два product flavors, demo и full заедно с debug and release генерира четири билд варианти, всеки със своя собствен потребителска конфигурация:

  \*  demoDebug

  \* demoRelease

  \*  fullDebug

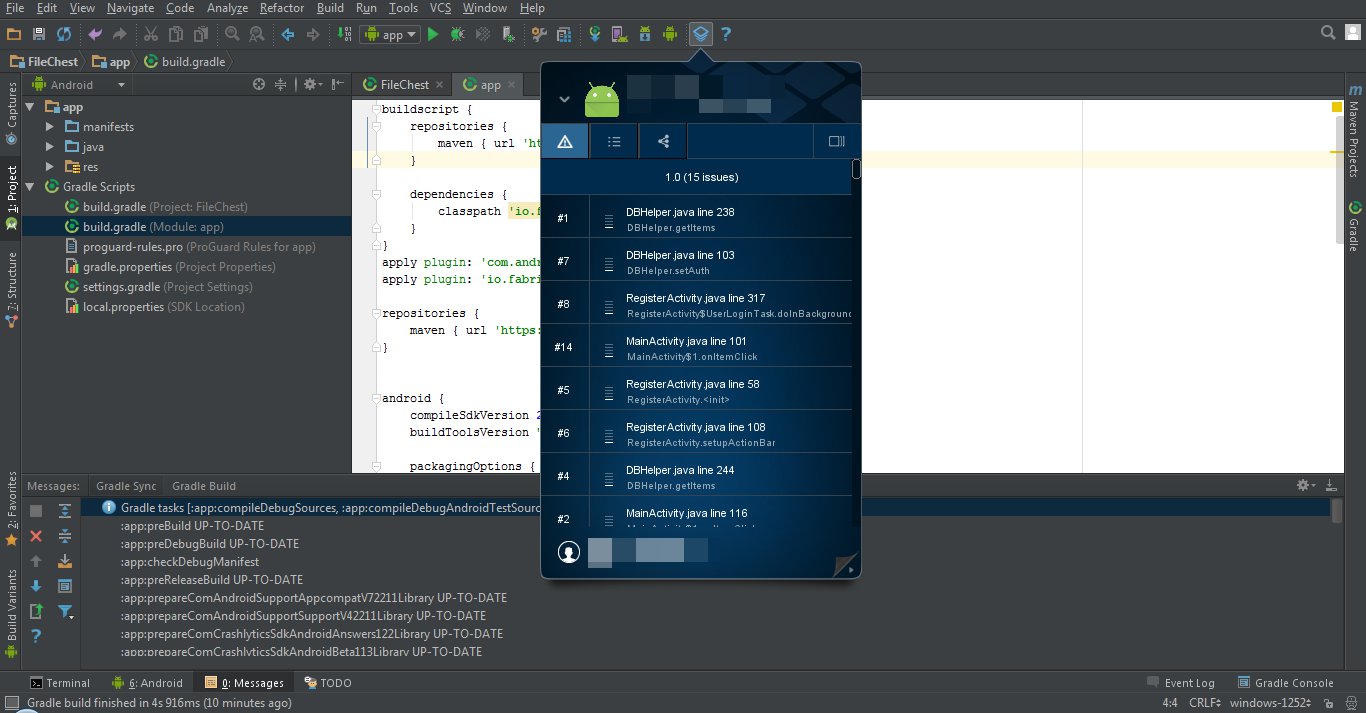
  \*  fullRelease

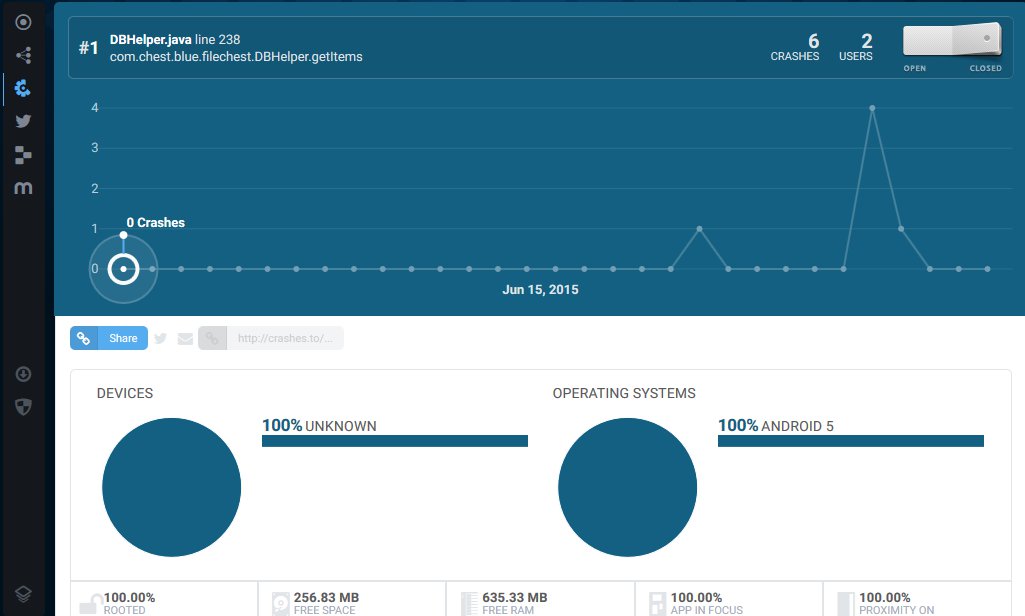
Някои проекти са сложни комбинации от функции в повече от едно измерение, но те все продължават да представляват един и същи апп. Така например, в допълнение демо и пълна версия на приложението, някои апп-ове могат да съдържат изпълними файлове, специфични за конкретен CPU / ABI. Гъвкавостта на системата билдване позволява да се генерира следните варианти за билдване на такъв проект:

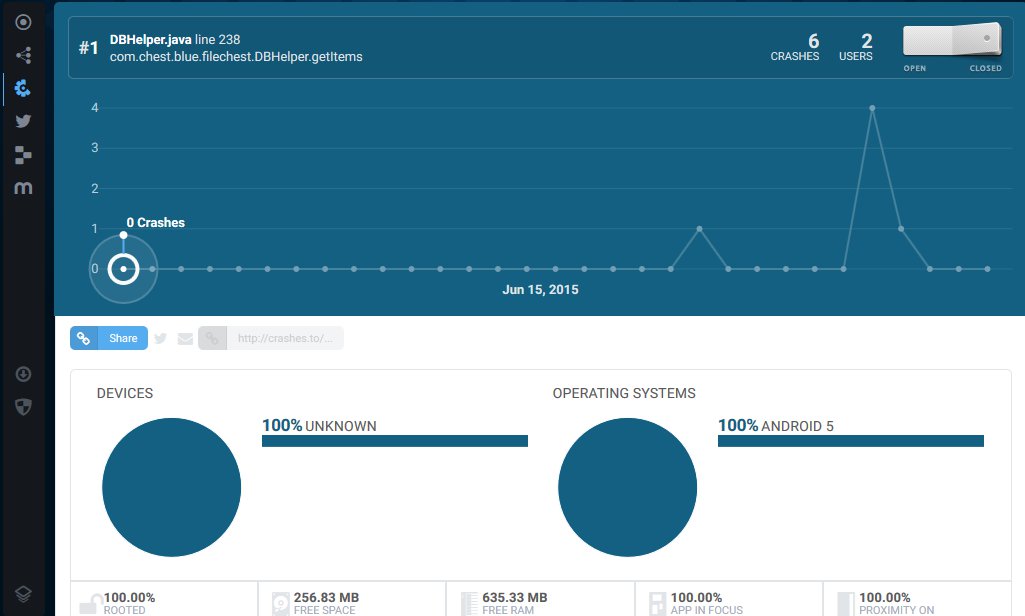
* x86-demoDebug
* x86-demoRelease
* x86-fullDebug
* x86-fullRelease
* ARM-demoDebug
* ARM-demoRelease
* ARM-fullDebug
* ARM-fullRelease
* MIPS-demoDebug
* MIPS-demoRelease
* MIPS-fullDebug
* MIPS-fullRelease

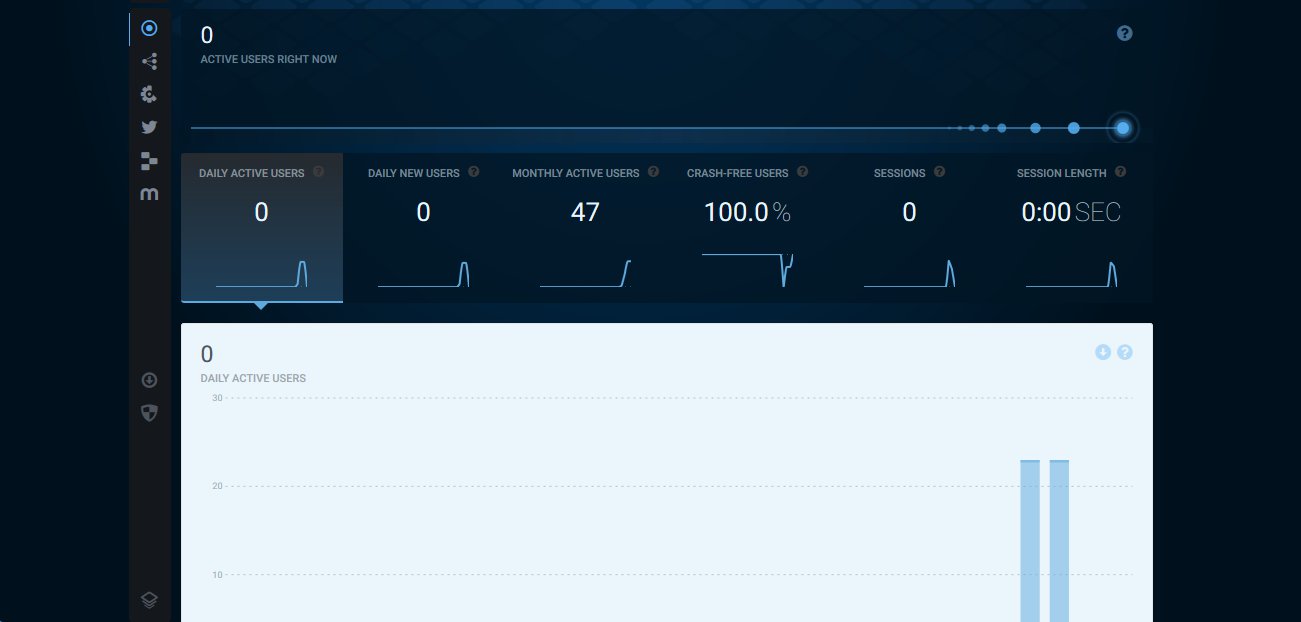
2.4 **Crash/Issue tracking system**

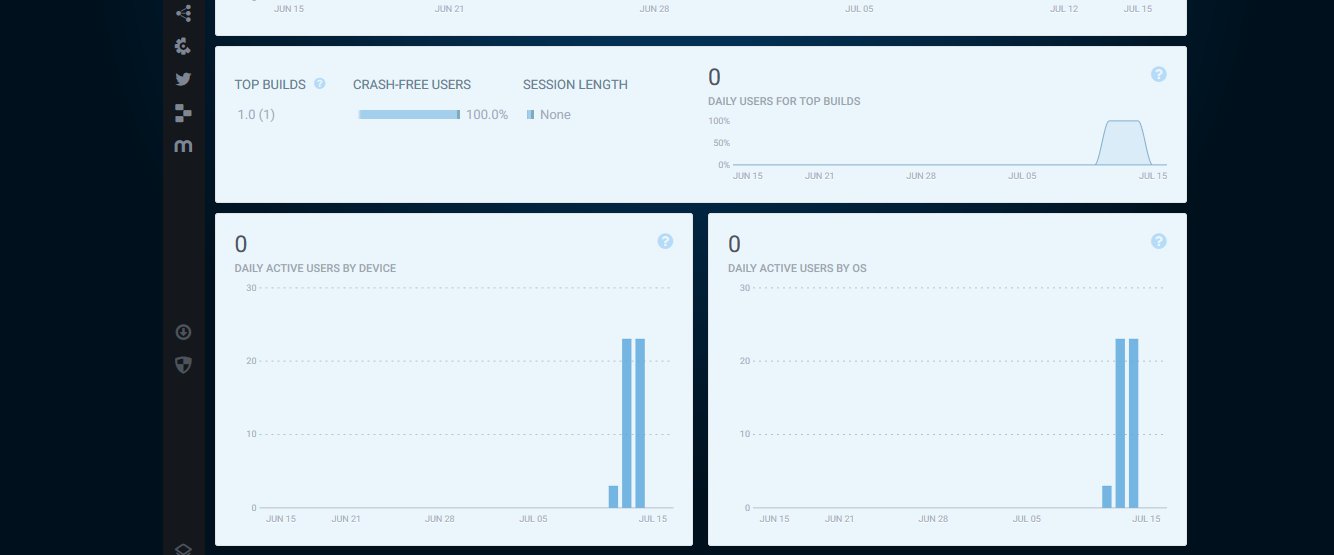
Crashlytics е решение, базирана на отчетния крашове чрез облака, която има за цел да намали сигнали за крашове до най-простият общ знаменател. Тя действа, като внедрява няколко реда прост код, който активира своя лек SDK (около 75KB), така че да се изготвят отчети за крашове който да минат чрез Amazon Web Services и да се върнат във вид на един прост доклад ,който да покаже по точно на реда на код, който прави приложението да крашва. Тя е един от най-простите, но мощни системи Crash/Issue tracking system за мобилни разработчици на разположение. В момента Crashlytics е в частна бета версия и в момента работи на около 100 милиона устройства.











3.0 **Програмна реализация**

Има за цел създаване на LoginActivity и сетване на всички компоненти в него, като използвания layout, fragments, сетване на intents за достъп към следващите Activities, инициализиране на Crashlytics библиотеката, сетване на onClickListeners за слушане за click ивенти.

**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 Fabric.*with*(**this**, **new** Crashlytics());  
 setContentView(R.layout.***activity\_login***);  
  
 **if**(getResources().getConfiguration().**orientation** == Configuration.***ORIENTATION\_LANDSCAPE***){  
 **layout** = (LinearLayout) findViewById(R.id.***email\_login\_form***);  
 **layout**.setPadding(0,40,0,0);  
 }  
  
 Intent intent = **new** Intent(**this**, MainActivity.**class**);  
 Intent registerIntent = **new** Intent(**this**,RegisterActivity.**class**);

……………

Има за цел промяна за layout при смяна на ориентацията на дисплея и правилната подредба на елементите.

**public void** onConfigurationChanged(Configuration newConfig) {  
 **super**.onConfigurationChanged(newConfig);

……………

Има за цел сетване на ивент за Back бутона и неговото правилно управление( излизане от приложението).

**public boolean** onKeyDown(**int** keyCode, KeyEvent event)  
{

…………

Има за цел проверка на въведените данни и показване на грешка при наличието на такава.

**public void** attemptLogin() {

………

Има за цел показване на прогрес диалог при проверката на въведените данни и изчакването за отговор от сървъра с резултата от изпратените заявки.

**public void** showProgress(**final boolean** show) {

…………

Има за цел създаване на background threat за връзка със сървъра, което ще освободи main (UI) threat-a от зацикляне при изчакването от отговор и неговата обработка.

**public class** UserLoginTask **extends** AsyncTask<Void, Void, Boolean> {

**private final** String **mEmail**;  
**private final** String **mPassword**;  
  
UserLoginTask(String email, String password) {  
 **mEmail** = email;  
 **mPassword** = password;  
}

**protected** Boolean doInBackground(Void... params) {

…………

Има за цел спиране на прогрес диалога и показване на грешка при възникнала такава или преминаването към следващото activity.

@Override  
**protected void** onPostExecute(**final** Boolean success) {  
 **mAuthTask** = **null**;  
 showProgress(**false**);  
  
 **if** (success) {

…………

Има за цел проверяване на интернет връзката и наличието на интернет по тази връзка.

*// Check for internet connection***public boolean** isInternetAvailable() {  
 **try** {

…………

Използва се за изчистване на кеша създаден от приложението.

**public class** ClearCache {

**public** ClearCache(Context context){  
 **this**.*context* = context;  
}

**public static void** trimCache() {

…………

Използва се за изчистване на локалната директория създадено от приложението.

**public static boolean** deleteDir(File dir) {

**……………**

Използва се за създаване на SQLite база данни.

**public class** DBHelper **extends** SQLiteOpenHelper {  
  
 **public static final** String ***DATABASE\_NAME*** = **"MyDBName.db"**;  
  
 **public** DBHelper(Context context) {  
 **super**(context, ***DATABASE\_NAME***, **null**, 1);}  
  
 @Override  
 **public void** onCreate(SQLiteDatabase db) {

………………

Използва се за отчитане на upgrade при промяна в базата данни.

**public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {

……………

Използва се за премахване на всички таблици в базата данни.

**public void** DropTables() {

……………

Използва се вмъкване на информация за папки в базата данни.

**public boolean** insertContentsFolder(Context context,String name,  
 String parentfolderid, String folderid,  
 String isfolder,String id, String modified) {

……………

Използва се вмъкване на информация за файлове в базата данни.

**public boolean** insertContentsFiles(String name, String parentfolderid,  
 String fileid, String isfolder,  
 String id, String modified) {

……………

Използва се вмъкване на auth ключа в базата данни, който се генерира при Login или Register.

**public void** setAuth(String auth){

…………

Използва се взимане на ArrayList от елементи от базата данни по parentfolderid.

**public** ArrayList<Items> getItems(String parentfolderid){

……………

Използва се вземане на auth ключа от базата данни.

**public** String getAuth(){

…………

Използва се взимане на ArrayList от елементи от базата данни по folderid.

**public** ArrayList<Items> getParentIdFromId(String folderid){

……………

Използва се изтриване на елементи от базата данни.

**public void** deleteItems(Items items){

………………

Използва се за генериране на публичен линк на файл за сваляне през заявка към сървъра по подадено fileid и auth ключ.

**public class** GetDownloadLink **extends** AsyncTask<String, Void, Boolean> {

**public boolean** getdownloadlink(String fileid, String auth){

……………

Използва се за генериране на публичен линк на папка за сваляне през заявка към сървъра по подадено folderid и auth ключ.

**public class** GetDownloadlinkFolder **extends** AsyncTask<String, Void, Boolean> {

**public boolean** getdownloadlink(String folderid, String auth){

……………

Използва се за създаване на Socket за връзка към сървъра чрез https, който използва SSL за криптиране на връзката между клиента и сървъра.

**сlass** MySSLSocketFactory **extends** SSLSocketFactory {

**public** MySSLSocketFactory(KeyStore truststore) **throws** Exception {

…………

@Override  
**public** Socket createSocket(Socket socket, String host, **int** port, **boolean** autoClose) **throws** IOException,  
 UnknownHostException {

…………

@Override  
**public** Socket createSocket() **throws** IOException {

…………………

javax.net.ssl.SSLSocketFactory getSSLSocketFactory() **throws** Exception {

………………

Използва се за проверка на интернет връзката и връщане на типът на използваната връзка.

**public class** NetworkUtil {  
**public static int** getConnectivityStatus(Context context) {

…………

**public static** String getConnectivityStatusString(Context context) {

……………

Използва се за парсване на JSON респонса от сървъра към базата данни.

**public** ListFolderParser(String json, Context context){

**public void** ParseToDB(){

………….

**public void** ParseContexts(JSONArray contents){

…………..

Използва се изпращане на заявка към сървъра за изреждане на всички налични файлове и папки в акаунта на потребителя.

**public class** ListFolders {

**public void** getFolder(String auth) {

…….

Използва се за генериране на item елементи, които представляват отделен файл или папка за по лесно събиране на наличната информация и нейното презентиране.

**public class** Items {

**public** Items(**final** String name, **final** String id,String fileid, String parentid, String modified ) {

…………

Използва се изпращане на заяка към сървъра за генериране на auth ключ.

**public class** Login {

**public void** logIn(Context context, String email, String password) {

…………

Използва се изпращане на заяка към сървъра за анулиране на auth ключ, който е генериран.

**public class** Logout {  
  
 **public void** logOut(String auth, Context context) {

……………

Използва се за създаване на action bar menu.

@Override  
**public boolean** onCreateActionMode(ActionMode mode, Menu menu) {

…………

Използва се за създаване на Navigation Drawer menu.

@Override  
**public void** onNavigationDrawerItemSelected(**int** position) {

……………

@Override  
**public boolean** onCreateOptionsMenu(Menu menu) {

……………

Използва се за отчитате на tap върху едно от менютата на Navigation Drawer menu.

**public void** onSectionAttached(**int** number) {

………………

Използва се за отчитате на tap върху едно от менютата на action bar menu.

@Override  
**public boolean** onOptionsItemSelected(MenuItem item) {

…………

Използва се за създаване на нотификация с подаден текст.

**private void** notify(String methodName) {

…………

Използва се за създаване на адаптер който спомага за листване на файловете и папките.

**public class** MyItemAdapter **extends** ArrayAdapter<Items> {

**public** MyItemAdapter(Context context, **int** resourceId,  
 List<Items> items) {

…………

Използва се за премахвате на елемент от адаптера, съответно и от list view.

@Override  
**public void** remove(Items object) {

……………

Използва се за генериране на NavigationDrawer менюто и управление на всички негови състояния, в който може да попадне в целия жизнен цикъл на приложението.

**public class** NavigationDrawerFragment **extends** Fragment {

**public void** onCreate(Bundle savedInstanceState) {

……………

@Override  
**public void** onActivityCreated(Bundle savedInstanceState) {

……………

@Override  
**public** View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {

……………

**public void** setUp(**int** fragmentId, DrawerLayout drawerLayout) {

……………

**private void** selectItem(**int** position) {

……………

@Override  
**public void** onAttach(Activity activity) {

……………

@Override  
**public void** onDetach() {

……………

@Override  
**public void** onConfigurationChanged(Configuration newConfig) {

……………

@Override  
**public void** onCreateOptionsMenu(Menu menu, MenuInflater inflater) {

……………

@Override  
**public boolean** onOptionsItemSelected(MenuItem item) {

……………

**private void** showGlobalContextActionBar() {

……………

Използва се за сетване на необходимите софтуерни и хардуерни пърмищъни необходими на приложението.

<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.chest.blue.filechest"** >  
<**uses-permission android:name="android.permission.GET\_ACCOUNTS"** />

………

Използва се за сетване на всички използвани активитита.

<**activity  
 android:name=".LoginActivity"**

**…………**

Използва се за деклариране наNetwork Receiver-a използван за проверка на състоянието на интернет връзката.

<**receiver  
 android:name="com.chest.blue.filechest.NetworkChangeReceiver"  
 android:label="NetworkChangeReceiver"** >

……………

Използва се за сетване на layout-a на Login Activity.

<**LinearLayout xmlns:android=**[**http://schemas.android.com/apk/res/android**](http://schemas.android.com/apk/res/android)

**………**

<**ProgressBar android:id="@+id/login\_progress" style="?android:attr/progressBarStyleLarge"**

**……….**

<**ScrollView android:id="@+id/login\_form" android:layout\_width="match\_parent"**

…………

<**LinearLayout android:id="@+id/email\_login\_form" android:layout\_width="match\_parent"**

…………

Използва се за сетване на layout-a на Action bar менютата.

*<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"** >  
  
 <**item  
 android:id="@+id/delete"  
 android:title="@string/delete"**/>

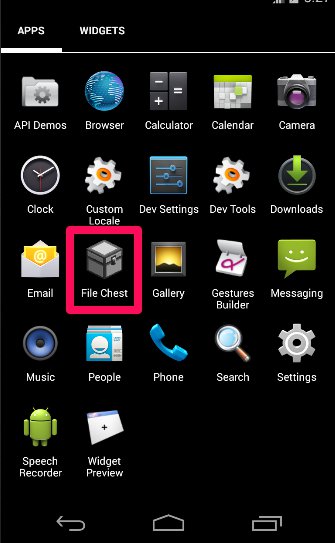
…………

**4.0 Ръководство за работа на потребителя**

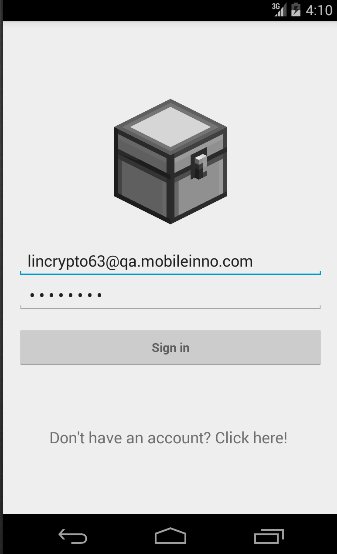
Изтеглете приложението от https://github.com/aleeekc/FileChest и го инсталирайте на мобилното устройство.

**4.1 Login and Register**

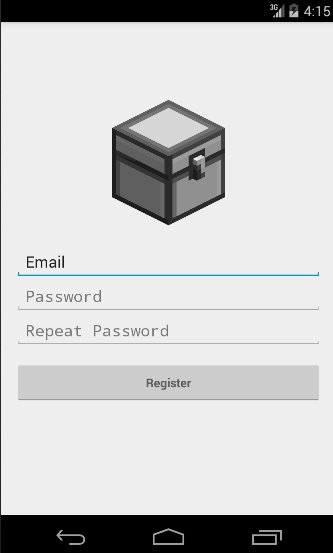
Влезте при вашите приложения и натиснете върху иконката на приложението.



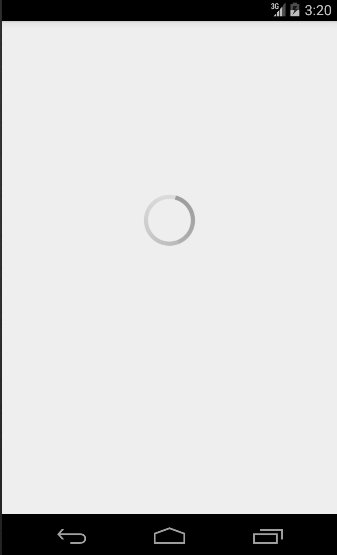
Въведете вашият **email** и **парола** след което натиснете върху **Login** бутона.



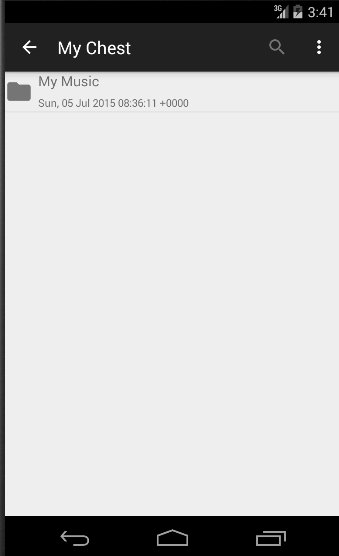
При липса на вас собствен акаунт натиснете върху **„Don’t have an account? Click here!“**, след което попълнете регистрационната форма и натиснете бутона **Register**

****

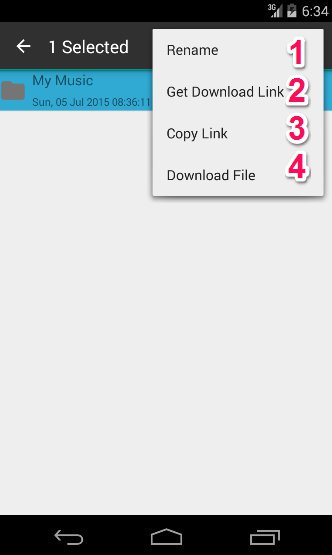
След кратко зареждане Вие успешно ще влезете във вашия акаун. При случай на грешка,като грешно въведена парола, несъществуващ email или други, тя ще бъде посочена към Вас за да може да бъде отсранена.



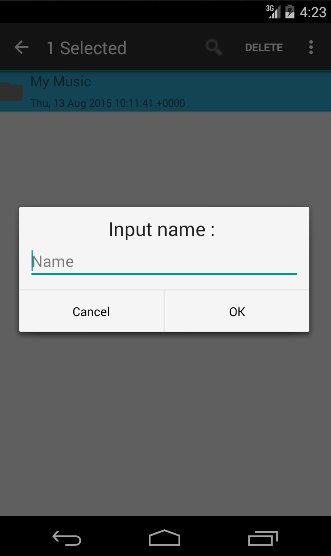
При успешно влизане във вашия профил на екрана ще бъдат изброени вашите файлове и папки.



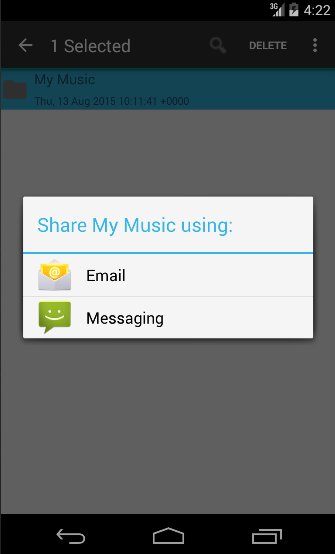
4.2 **File/Folder Operations**

****

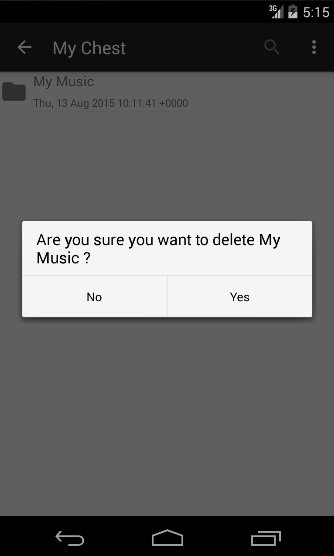
1. Rename – преименува избрания файл/папка



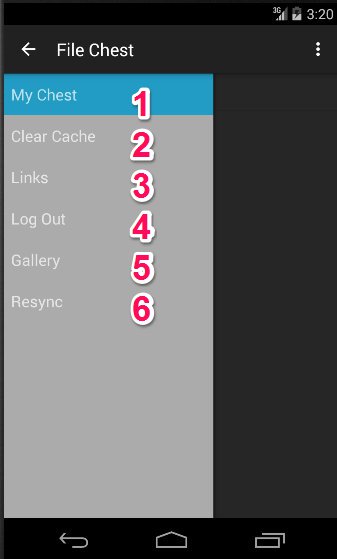
1. Get Download Link – генерира download link за избрания файл/папка, който ще бъде изброен в Links страницата
2. Copy Link – генерира download link за избрания файл/папка и го копира в Clipboard-a



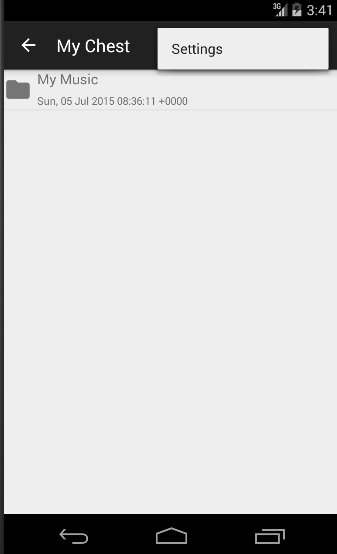
1. Download File – тегли избрания файл в локалната директория на приложението и предлага предлага приложение за неговото отваряне
2. Delete – изтрива избрания файл или папка



* 1. **Menus**

****

1. My Chest – води до главната директория на потребителя с неговите файлове и папки.
2. Clear Cache – изчиства потребителския cache
3. Links – води до всички създадени от потребителя шерове



1. Log Out – изчиства всички потребителски данни и води потребителя до Login страницата
2. Gallery – води до галерията на потребителя с всички негови снимки



1. Resync – презарежда акаунта на потребителя с всички нови актуализации

5.0 **Source code на написаното програмно осигуряване**

LOGIN ACTIVITY

package com.chest.blue.filechest;

import android.animation.Animator;

import android.animation.AnimatorListenerAdapter;

import android.annotation.TargetApi;

import android.app.Activity;

import android.app.LoaderManager.LoaderCallbacks;

import android.app.Notification;

import android.app.NotificationManager;

import android.content.ContentResolver;

import android.content.CursorLoader;

import android.content.Intent;

import android.content.Loader;

import android.content.res.Configuration;

import android.database.Cursor;

import android.net.Uri;

import android.os.AsyncTask;

import android.os.Build;

import android.os.Bundle;

import android.provider.ContactsContract;

import android.text.TextUtils;

import android.view.Display;

import android.view.KeyEvent;

import android.view.View;

import android.view.View.OnClickListener;

import android.view.WindowManager;

import android.view.inputmethod.EditorInfo;

import android.widget.ArrayAdapter;

import android.widget.AutoCompleteTextView;

import android.widget.Button;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.TextView;

import android.widget.Toast;

import com.crashlytics.android.Crashlytics;

import io.fabric.sdk.android.Fabric;

import java.net.InetAddress;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* A login screen that offers login via email/password.

\*/

public class LoginActivity extends Activity implements LoaderCallbacks<Cursor> {

/\*\*

\* A dummy authentication store containing known user names and passwords.

\* TODO: remove after connecting to a real authentication system.

\*/

private static final String[] DUMMY\_CREDENTIALS = new String[]{

"foo@example.com:hello", "bar@example.com:world"

};

/\*\*

\* Keep track of the login task to ensure we can cancel it if requested.

\*/

private UserLoginTask mAuthTask = null;

// UI references.

private AutoCompleteTextView mEmailView;

private EditText mPasswordView;

private View mProgressView;

private View mLoginFormView;

private TextView register;

private Intent intent;

private Intent intentReg;

private LinearLayout layout;

Login LoginObj = new Login();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

Fabric.with(this, new Crashlytics());

setContentView(R.layout.activity\_login);

if(getResources().getConfiguration().orientation == Configuration.ORIENTATION\_LANDSCAPE){

layout = (LinearLayout) findViewById(R.id.email\_login\_form);

layout.setPadding(0,40,0,0);

}

Intent intent = new Intent(this, MainActivity.class);

Intent registerIntent = new Intent(this,RegisterActivity.class);

intent.putExtra("inFolder", "0");

GETIntent(intent);

GETIntentReg(registerIntent);

// Set up the login form.

mEmailView = (AutoCompleteTextView) findViewById(R.id.email);

populateAutoComplete();

register = (TextView) findViewById(R.id.register);

register.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

startActivity(intentReg);

}

});

mPasswordView = (EditText) findViewById(R.id.password);

mPasswordView.setOnEditorActionListener(new TextView.OnEditorActionListener() {

@Override

public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {

if (id == R.id.login || id == EditorInfo.IME\_NULL) {

attemptLogin();

return true;

}

return false;

}

});

Button mEmailSignInButton = (Button) findViewById(R.id.email\_sign\_in\_button);

mEmailSignInButton.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View view) {

register.setVisibility(View.GONE);

attemptLogin();

////// INTERNET CONNECTION CHECK - NOT WORKING PROPERLY

/\*

if(!isInternetAvailable()){

Toast.makeText(getApplicationContext(),"No internet connection!",Toast.LENGTH\_SHORT).show();

}else {

//Hide register textview

register.setVisibility(View.GONE);

attemptLogin();

}\*/

}

});

mLoginFormView = findViewById(R.id.login\_form);

mProgressView = findViewById(R.id.login\_progress);

}

public void onConfigurationChanged(Configuration newConfig) {

super.onConfigurationChanged(newConfig);

Display display = ((WindowManager) getSystemService(WINDOW\_SERVICE)).getDefaultDisplay();

int orientation = display.getOrientation();

switch(orientation) {

case Configuration.ORIENTATION\_PORTRAIT:

layout.setPadding(0,40,0,0);

break;

case Configuration.ORIENTATION\_LANDSCAPE:

layout.setPadding(0,170,0,0);

break;

}

}

@Override

public boolean onKeyDown(int keyCode, KeyEvent event)

{

switch(keyCode)

{

case KeyEvent.KEYCODE\_BACK:

moveTaskToBack(true);

return true;

}

return false;

}

public void GETIntent(Intent intent) {

this.intent = intent;

}

public void GETIntentReg(Intent intent) {

this.intentReg = intent;

}

public Intent ReturnIntent(){

return intent;

}

public Intent ReturnIntentReg(){

return intentReg;

}

private void populateAutoComplete() {

getLoaderManager().initLoader(0, null, this);

}

/\*\*

\* Attempts to sign in or register the account specified by the login form.

\* If there are form errors (invalid email, missing fields, etc.), the

\* errors are presented and no actual login attempt is made.

\*/

public void attemptLogin() {

if (mAuthTask != null) {

return;

}

// Reset errors.

mEmailView.setError(null);

mPasswordView.setError(null);

// Store values at the time of the login attempt.

String email = mEmailView.getText().toString();

String password = mPasswordView.getText().toString();

boolean cancel = false;

View focusView = null;

// Check for a valid password, if the user entered one.

if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {

mPasswordView.setError(getString(R.string.error\_invalid\_password));

focusView = mPasswordView;

cancel = true;

}

// Check for a valid email address.

if (TextUtils.isEmpty(email)) {

mEmailView.setError(getString(R.string.error\_field\_required));

focusView = mEmailView;

cancel = true;

} else if (!isEmailValid(email)) {

mEmailView.setError(getString(R.string.error\_invalid\_email));

focusView = mEmailView;

cancel = true;

}

if (cancel) {

// There was an error; don't attempt login and focus the first

// form field with an error.

focusView.requestFocus();

} else {

// Show a progress spinner, and kick off a background task to

// perform the user login attempt.

showProgress(true);

mAuthTask = new UserLoginTask(email, password);

mAuthTask.execute((Void) null);

}

}

private boolean isEmailValid(String email) {

//TODO: Replace this with your own logic

return email.contains("@");

}

private boolean isPasswordValid(String password) {

//TODO: Replace this with your own logic

return password.length() > 4;

}

/\*\*

\* Shows the progress UI and hides the login form.

\*/

@TargetApi(Build.VERSION\_CODES.HONEYCOMB\_MR2)

public void showProgress(final boolean show) {

// On Honeycomb MR2 we have the ViewPropertyAnimator APIs, which allow

// for very easy animations. If available, use these APIs to fade-in

// the progress spinner.

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.HONEYCOMB\_MR2) {

int shortAnimTime = getResources().getInteger(android.R.integer.config\_shortAnimTime);

mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);

mLoginFormView.animate().setDuration(shortAnimTime).alpha(

show ? 0 : 1).setListener(new AnimatorListenerAdapter() {

@Override

public void onAnimationEnd(Animator animation) {

mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);

}

});

mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);

mProgressView.animate().setDuration(shortAnimTime).alpha(

show ? 1 : 0).setListener(new AnimatorListenerAdapter() {

@Override

public void onAnimationEnd(Animator animation) {

mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);

}

});

} else {

// The ViewPropertyAnimator APIs are not available, so simply show

// and hide the relevant UI components.

mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);

mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);

}

}

@Override

public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {

return new CursorLoader(this,

// Retrieve data rows for the device user's 'profile' contact.

Uri.withAppendedPath(ContactsContract.Profile.CONTENT\_URI,

ContactsContract.Contacts.Data.CONTENT\_DIRECTORY), ProfileQuery.PROJECTION,

// Select only email addresses.

ContactsContract.Contacts.Data.MIMETYPE +

" = ?", new String[]{ContactsContract.CommonDataKinds.Email

.CONTENT\_ITEM\_TYPE},

// Show primary email addresses first. Note that there won't be

// a primary email address if the user hasn't specified one.

ContactsContract.Contacts.Data.IS\_PRIMARY + " DESC");

}

@Override

public void onLoadFinished(Loader<Cursor> cursorLoader, Cursor cursor) {

List<String> emails = new ArrayList<String>();

cursor.moveToFirst();

while (!cursor.isAfterLast()) {

emails.add(cursor.getString(ProfileQuery.ADDRESS));

cursor.moveToNext();

}

addEmailsToAutoComplete(emails);

}

@Override

public void onLoaderReset(Loader<Cursor> cursorLoader) {

}

private interface ProfileQuery {

String[] PROJECTION = {

ContactsContract.CommonDataKinds.Email.ADDRESS,

ContactsContract.CommonDataKinds.Email.IS\_PRIMARY,

};

int ADDRESS = 0;

int IS\_PRIMARY = 1;

}

private void addEmailsToAutoComplete(List<String> emailAddressCollection) {

//Create adapter to tell the AutoCompleteTextView what to show in its dropdown list.

ArrayAdapter<String> adapter =

new ArrayAdapter<String>(LoginActivity.this,

android.R.layout.simple\_dropdown\_item\_1line, emailAddressCollection);

mEmailView.setAdapter(adapter);

}

/\*\*

\* Represents an asynchronous login/registration task used to authenticate

\* the user.

\*/

public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

private final String mEmail;

private final String mPassword;

UserLoginTask(String email, String password) {

mEmail = email;

mPassword = password;

}

@Override

protected Boolean doInBackground(Void... params) {

// TODO: attempt authentication against a network service.

DBHelper db = new DBHelper(getApplicationContext());

// db.DropTables();

try {

LoginObj.logIn(getApplicationContext(), mEmail, mPassword);

}

catch (Exception e)

{

Toast.makeText(getApplicationContext(), "Sorry. Something went wrong!", Toast.LENGTH\_SHORT).show();

return false;

}

try {

// Simulate network access.

Thread.sleep(2000);

} catch (InterruptedException e) {

return false;

}

/\*

for (String credential : DUMMY\_CREDENTIALS) {

String[] pieces = credential.split(":");

if (pieces[0].equals(mEmail)) {

// Account exists, return true if the password matches.

return pieces[1].equals(mPassword);

}

}

\*/

// TODO: register the new account here.

return true;

}

@Override

protected void onPostExecute(final Boolean success) {

mAuthTask = null;

showProgress(false);

if (success) {

startActivity(ReturnIntent().putExtra("auth", LoginObj.auth()));

//finish();

} else {

mPasswordView.setError(getString(R.string.error\_incorrect\_password));

mPasswordView.requestFocus();

}

}

@Override

protected void onCancelled() {

mAuthTask = null;

showProgress(false);

}

}

//--------------------------------LIFECYCLE -------------------------------

@Override

protected void onPause() {

super.onPause();

//notify("onPause");

}

@Override

protected void onResume() {

super.onResume();

//notify("onResume");

}

@Override

protected void onStop() {

super.onStop();

//notify("onStop");

}

@Override

protected void onDestroy() {

super.onDestroy();

//notify("onDestroy");

}

@Override

protected void onRestoreInstanceState(Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);

//notify("onRestoreInstanceState");

}

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

//notify("onSaveInstanceState");

}

private void notify(String methodName) {

String name = this.getClass().getName();

String[] strings = name.split("\\.");

Notification noti = new Notification.Builder(this)

.setContentTitle(methodName + " " + strings[strings.length - 1]).setAutoCancel(true)

.setSmallIcon(R.mipmap.ic\_launcher)

.setContentText(name).build();

NotificationManager notificationManager =

(NotificationManager) getSystemService(NOTIFICATION\_SERVICE);

notificationManager.notify((int) System.currentTimeMillis(), noti);

}

// Check for internet connection

public boolean isInternetAvailable() {

try {

InetAddress ipAddr = InetAddress.getByName("google.com");

if (ipAddr.equals("")) {

return false;

} else {

return true;

}

} catch (Exception e) {

return false;

}

}

}

MAIN ACTIVITY

package com.chest.blue.filechest;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.app.Notification;

import android.app.NotificationManager;

import android.app.SearchManager;

import android.content.ClipData;

import android.content.ClipboardManager;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.AsyncTask;

import android.support.v7.app.ActionBarActivity;

import android.support.v7.app.ActionBar;

import android.support.v4.app.Fragment;

import android.support.v4.app.FragmentManager;

import android.os.Bundle;

import android.util.Log;

import android.view.ActionMode;

import android.view.KeyEvent;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.support.v4.widget.DrawerLayout;

import android.widget.AdapterView;

import android.widget.EditText;

import android.widget.ListView;

import java.io.BufferedInputStream;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.io.OutputStream;

import java.net.URL;

import java.net.URLConnection;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import android.util.SparseBooleanArray;

import android.widget.AbsListView.MultiChoiceModeListener;

import android.widget.SearchView;

import android.widget.Toast;

import android.app.ProgressDialog;

import android.support.v4.app.ActionBarDrawerToggle;

public class MainActivity extends ActionBarActivity

implements NavigationDrawerFragment.NavigationDrawerCallbacks {

/\*\*

\* Fragment managing the behaviors, interactions and presentation of the navigation drawer.

\*/

private NavigationDrawerFragment mNavigationDrawerFragment;

/\*\*

\* Used to store the last screen title. For use in {@link #restoreActionBar()}.

\*/

private CharSequence mTitle;

String inFolder = "0";

List<Integer> selectedItems;

List<Items> itemsList = new ArrayList<Items>();

MyItemAdapter NewnewsEntryAdapter;

String name;

// Progress Dialog

private ProgressDialog pDialog;

// Progress dialog type (0 - for Horizontal progress bar)

public static final int progress\_bar\_type = 0;

private ActionBarDrawerToggle mDrawerToggle;

private DrawerLayout mDrawerLayout;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

Intent intent = getIntent();

inFolder = intent.getStringExtra("inFolder");

Log.v("TAG", "I am in folder with id: " + inFolder);

setContentView(R.layout.activity\_main);

mNavigationDrawerFragment = (NavigationDrawerFragment)

getSupportFragmentManager().findFragmentById(R.id.navigation\_drawer);

mTitle = getTitle();

// Set up the drawer.

mNavigationDrawerFragment.setUp(

R.id.navigation\_drawer,

(DrawerLayout) findViewById(R.id.drawer\_layout));

final ListView newsEntryListView = (ListView) findViewById(R.id.list);

newsEntryListView.setChoiceMode(ListView.CHOICE\_MODE\_MULTIPLE\_MODAL);

final MyItemAdapter newsEntryAdapter = new MyItemAdapter(this, R.layout.listview\_item, getNewsEntries("0"));

SetAdapter(newsEntryAdapter);

newsEntryListView.setAdapter(newsEntryAdapter);

/\*

// Populate the list, through the adapter

for (final Items entry : getNewsEntries("0")) {

newsEntryAdapter.add(entry);

Log.v("TAG", "Name: " + entry.getName());

}

\*/

newsEntryListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view,

final int position, long id) {

//final String item = (String) parent.getItemAtPosition(position);

//final MyItemAdapter newAdapter = new MyItemAdapter(getApplicationContext(), R.layout.listview\_item,getNewsEntries(inFolder));

List<Items> list = getNewsEntries(inFolder);

if (list.get(position).getId() != null) {

inFolder = list.get(position).getId();

}

/\*

// Remove all old items

for(int i=0;i<list.size();i++) {

Log.v("TAG","Removing item: " + list.get(i).getName());

newsEntryAdapter.remove(list.get(i));

}

\*/

NewnewsEntryAdapter.removeAll(list);

for (final Items entry : getNewsEntries(inFolder)/\*list.get(position).getId())\*/) {

NewnewsEntryAdapter.add(entry);

Log.v("TAG", "Name: " + entry.getName());

}

//newsEntryListView.setAdapter(newsEntryAdapter);

//newsEntryAdapter.notifyDataSetChanged();

view.setAlpha(1);

}

});

/\*

newsEntryListView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {

public boolean onItemLongClick(AdapterView<?> parent, View view,

int pos, long id) {

//parent.setItemChecked(position, true);

view.setBackgroundColor(Color.rgb(200, 200, 200));

view.findViewById(R.id.checkicon).setVisibility(View.VISIBLE);

//view.setSelected(true);

Vibrator vb = (Vibrator) getSystemService(Context.VIBRATOR\_SERVICE);

vb.vibrate(100);

//selectedItems.add(pos);

Log.v("TAG", "Item selected at position: " + pos);

newsEntryAdapter.notifyDataSetChanged();

return true;

}

});

\*/

//-----------------------------------------------------------

newsEntryListView.setMultiChoiceModeListener(new MultiChoiceModeListener() {

@Override

public void onItemCheckedStateChanged(ActionMode mode,

int position, long id, boolean checked) {

// Capture total checked items

final int checkedCount = newsEntryListView.getCheckedItemCount();

// Set the CAB title according to total checked items

mode.setTitle(checkedCount + " Selected");

// Calls toggleSelection method from ListViewAdapter Class

newsEntryAdapter.toggleSelection(position);

}

@Override

public boolean onActionItemClicked(ActionMode mode, MenuItem item) {

Boolean Flag\_User\_Choice;

// Calls getSelectedIds method from ListViewAdapter Class

final SparseBooleanArray selected = newsEntryAdapter

.getSelectedIds();

switch (item.getItemId()) {

case R.id.delete:

// Calls getSelectedIds method from ListViewAdapter Class

//SparseBooleanArray selected = newsEntryAdapter

// .getSelectedIds();

DialogInterface.OnClickListener dialogClickListener = new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

switch (which){

case DialogInterface.BUTTON\_POSITIVE:

//Yes button clicked

// Captures all selected ids with a loop

for (int i = (selected.size() - 1); i >= 0; i--) {

if (selected.valueAt(i)) {

Items selecteditem = newsEntryAdapter

.getItem(selected.keyAt(i));

// Remove selected items following the ids

newsEntryAdapter.remove(selecteditem);

DBHelper dbHelper = new DBHelper(MainActivity.this);

// TODO MAKE SOMEKIND OF PROGRESS SO THAT THE ASYNC TASK CAN FINISH THEIR WORK AND THEN REFRESH THE ACTIVITY

if(selecteditem.getFileId().equals("")) {

String[] arg = {"1",selecteditem.getId(),dbHelper.getAuth()};

new DelAsyncTask().execute(arg);

}else {

String[] arg = {"0",selecteditem.getFileId(),dbHelper.getAuth()};

new DelAsyncTask().execute(arg);

}

}

}

dialog.dismiss();

break;

case DialogInterface.BUTTON\_NEGATIVE:

//No button clicked

dialog.dismiss();

break;

}

}

};

Items newselecteditem = newsEntryAdapter.getItem(selected.keyAt(0));

AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);

builder.setMessage("Are you sure you want to delete " + newselecteditem.getName() + " ?").setPositiveButton("Yes", dialogClickListener)

.setNegativeButton("No", dialogClickListener).show();

// Close CAB

mode.finish();

return true;

case R.id.Rename:

if (selected.size() > 1)

Toast.makeText(getApplicationContext(), "Only one item at a time can be renamed!", Toast.LENGTH\_SHORT).show();

else {

// TODO RETEST

// get prompts.xml view

LayoutInflater li = LayoutInflater.from(MainActivity.this);

View promptsView = li.inflate(R.layout.dialog, null);

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(

MainActivity.this);

// set prompts.xml to alertdialog builder

alertDialogBuilder.setView(promptsView);

final EditText userInput = (EditText) promptsView

.findViewById(R.id.editTextDialogUserInput);

// set dialog message

alertDialogBuilder

.setCancelable(false)

.setPositiveButton("OK",

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,int id) {

if(userInput.getText().equals(null) || userInput.getText().equals("")){

Toast.makeText(MainActivity.this,"Please input the new name!",Toast.LENGTH\_SHORT).show();

}else {

//TODO RETEST

Items selectedItem = newsEntryAdapter

.getItem(selected.keyAt(0));

DBHelper DB = new DBHelper(getApplicationContext());

if (selectedItem.getFileId() == "" || selectedItem.getFileId() == null) {

// ITS A FOLDER

String[] arr = {"0", selectedItem.getId(), userInput.getText().toString(), DB.getAuth()};

new RenameAsyncTask().execute(arr);

} else {

// ITS A FILE

String[] arr = {"1", selectedItem.getFileId(), userInput.getText().toString(), DB.getAuth()};

new RenameAsyncTask().execute(arr);

}

// PROGRESS DIALOG

ProgressDialog pdialog = new ProgressDialog(MainActivity.this);

pdialog.setCancelable(true);

pdialog.setMessage("Loading ....");

pdialog.show();

Resync resync = new Resync(MainActivity.this);

pdialog.dismiss();

Log.v("TAG", "The resync was done!");

Intent inten = new Intent(getApplicationContext(), MainActivity.class);

startActivity(inten);

}

}

})

.setNegativeButton("Cancel",

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,int id) {

dialog.cancel();

}

});

// create alert dialog

AlertDialog alertDialog = alertDialogBuilder.create();

// show it

alertDialog.show();

}

case R.id.CopyLink:

if (selected.size() > 1)

Toast.makeText(getApplicationContext(), "Please select one item to copy a link", Toast.LENGTH\_SHORT);

else {

DBHelper db = new DBHelper(getApplicationContext());

for (int i = (selected.size() - 1); i >= 0; i--) {

if (selected.valueAt(i)) {

Items selecteditem = newsEntryAdapter.getItem(selected.keyAt(i));

String text;

//-------------------------------- ////// LATER MAKE THIS WORK WITH THE ASYNCTASK !!!!!!!!!!!!!!!!

if (selecteditem.getFileId().equals("false")) {

Log.v("TAG", "fileid = " + selecteditem.getFileId());

text = "folderid=" + selecteditem.getId();

GetDownloadlinkFolder downloadLink = new GetDownloadlinkFolder(text, db.getAuth());

downloadLink.execute();

if (!downloadLink.Error().equals("")) {

Toast.makeText(getApplicationContext(), downloadLink.Error(), Toast.LENGTH\_SHORT).show();

} else {

ClipboardManager clipboard = (ClipboardManager) getSystemService(CLIPBOARD\_SERVICE);

ClipData clip = ClipData.newPlainText(selecteditem.getName(), downloadLink.link());

clipboard.setPrimaryClip(clip);

Toast.makeText(getApplicationContext(), "Download Link copied!", Toast.LENGTH\_SHORT).show();

}

} else if (selecteditem.getId().equals("false")) {

text = "fileid=" + selecteditem.getFileId();

GetDownloadLink downloadLink = new GetDownloadLink(text, db.getAuth());

downloadLink.execute();

if (!downloadLink.Error().equals("")) {

Toast.makeText(getApplicationContext(), downloadLink.Error(), Toast.LENGTH\_SHORT).show();

} else {

ClipboardManager clipboard = (ClipboardManager) getSystemService(CLIPBOARD\_SERVICE);

ClipData clip = ClipData.newPlainText(selecteditem.getName(), downloadLink.link());

clipboard.setPrimaryClip(clip);

Toast.makeText(getApplicationContext(), "Download Link copied!", Toast.LENGTH\_SHORT).show();

}

}

}

}

}

case R.id.GetDownloadLink:

if (selected.size() > 1)

Toast.makeText(getApplicationContext(), "Please select one item to copy a link", Toast.LENGTH\_SHORT);

else {

DBHelper db = new DBHelper(getApplicationContext());

for (int i = (selected.size() - 1); i >= 0; i--) {

if (selected.valueAt(i)) {

Items selecteditem = newsEntryAdapter.getItem(selected.keyAt(i));

String text;

//-------------------------------- ////// LATER MAKE THIS WORK WITH THE ASYNCTASK !!!!!!!!!!!!!!!!

if (selecteditem.getFileId().equals("false")) {

Log.v("TAG", "fileid = " + selecteditem.getFileId());

text = "folderid=" + selecteditem.getId();

GetDownloadlinkFolder downloadLink = new GetDownloadlinkFolder(text, db.getAuth());

downloadLink.execute();

Log.v("TAG", "error = " + downloadLink.Error());

if (!downloadLink.Error().equals("")) {

Toast.makeText(getApplicationContext(), downloadLink.Error(), Toast.LENGTH\_SHORT).show();

} else {

Intent sendIntent = new Intent();

sendIntent.setAction(Intent.ACTION\_SEND);

sendIntent.putExtra(Intent.EXTRA\_TEXT, downloadLink.link());

sendIntent.setType("text/plain");

startActivity(Intent.createChooser(sendIntent, "Share " + selecteditem.getName() + " using:"));

}

} else if (selecteditem.getId().equals("false")) {

text = "fileid=" + selecteditem.getFileId();

GetDownloadLink downloadLink = new GetDownloadLink(text, db.getAuth());

downloadLink.execute();

if (!downloadLink.Error().equals("")) {

Toast.makeText(getApplicationContext(), downloadLink.Error(), Toast.LENGTH\_SHORT).show();

} else {

Intent sendIntent = new Intent();

sendIntent.setAction(Intent.ACTION\_SEND);

sendIntent.putExtra(Intent.EXTRA\_TEXT, downloadLink.link());

sendIntent.setType("text/plain");

startActivity(sendIntent);

Toast.makeText(getApplicationContext(), "Download Link copied in clipboard!", Toast.LENGTH\_SHORT).show();

}

}

}

}

}

case R.id.DownloadFile:

///////////////////////////////// DOWNLOAD A FILE

if (selected.size() > 1)

Toast.makeText(getApplicationContext(), "Please select one item to copy a link", Toast.LENGTH\_SHORT).show();

else {

DBHelper db = new DBHelper(getApplicationContext());

for (int i = (selected.size() - 1); i >= 0; i--) {

if (selected.valueAt(i)) {

Items selecteditem = newsEntryAdapter.getItem(selected.keyAt(i));

if (!selecteditem.getFileId().equals("false")) {

Download download = new Download();

DBHelper dbHelper = new DBHelper(getApplicationContext());

/\*

download.GetGownLink(getApplicationContext(), dbHelper.getAuth(), selecteditem.getFileId());

String file\_url = download.GetURL();

\*/

String[] arr = {selecteditem.getFileId().toString(), dbHelper.getAuth()};

Don don = new Don(arr, getApplicationContext());

String file\_url = don.GetUrl();

if (file\_url.equals("")) {

Log.v("TAG", "Something went wrong! The download url is: " + file\_url);

} else {

Log.v("TAG", "The download url is: " + file\_url);

}

//new DownloadAsyncTask().execute(arr);

Log.v("TAG", "URL to file: " + file\_url);

new DownloadFileFromURL().execute(file\_url);

}

}

}

}

default:

return false;

}

}

@Override

public boolean onCreateActionMode(ActionMode mode, Menu menu) {

SparseBooleanArray selected = newsEntryAdapter

.getSelectedIds();

if (selected.size() > 1)

mode.getMenuInflater().inflate(R.menu.activity\_main\_menu\_multiple, menu);

else

mode.getMenuInflater().inflate(R.menu.activity\_main\_menu\_single, menu);

return true;

}

@Override

public void onDestroyActionMode(ActionMode mode) {

// TODO Auto-generated method stub

newsEntryAdapter.removeSelection();

}

@Override

public boolean onPrepareActionMode(ActionMode mode, Menu menu) {

// TODO Auto-generated method stub

return false;

}

});

//-----------------------------------------------------------

/\*

newsEntryListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view,

final int position, long id) {

//final String item = (String) parent.getItemAtPosition(position);

newsEntryAdapter.clear();

List<Items> list = getNewsEntries(inFolder);

inFolder = list.get(position).getId();

for (final Items entry : getNewsEntries(list.get(position).getId())) {

newsEntryAdapter.add(entry);

Log.v("TAG", "Name: " + entry.getName());

}

newsEntryAdapter.notifyDataSetChanged();

view.setAlpha(1);

}

});

;

newsEntryListView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {

public boolean onItemLongClick(AdapterView<?> parent, View view,

int pos, long id) {

//parent.setItemChecked(position, true);

if (view.isSelected()) {

view.setBackgroundColor(Color.WHITE);

view.findViewById(R.id.checkicon).setVisibility(View.INVISIBLE);

view.setSelected(false);

Vibrator vb = (Vibrator) getSystemService(Context.VIBRATOR\_SERVICE);

vb.vibrate(100);

Log.v("TAG", "Item unselected at position: " + pos);

newsEntryAdapter.notifyDataSetChanged();

}else {

view.setBackgroundColor(Color.rgb(200, 200, 200));

view.findViewById(R.id.checkicon).setVisibility(View.VISIBLE);

view.setSelected(true);

Vibrator vb = (Vibrator) getSystemService(Context.VIBRATOR\_SERVICE);

vb.vibrate(100);

//selectedItems.add(pos);

Log.v("TAG", "Item selected at position: " + pos);

newsEntryAdapter.notifyDataSetChanged();

}

return true;

}

});

\*/

}

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

switch (keyCode) {

case KeyEvent.KEYCODE\_BACK:

if (inFolder.equals("0")) {

// Set Activity in Background

moveTaskToBack(true);

} else {

// DON"T KNOW IF IT WORKS

ListView newsEntryListView = (ListView) findViewById(R.id.list);

DBHelper db = new DBHelper(getApplicationContext());

List<Items> items = db.getParentIdFromId(inFolder);

/\*

final MyItemAdapter newsEntryAdapter = new MyItemAdapter(this, R.layout.listview\_item,getNewsEntries(items.get(0).getParentId()));

newsEntryListView.setAdapter(newsEntryAdapter);

\*/

String parentfolderid = items.get(0).getParentId();

// Remove all old items

for (int i = 0; i < items.size(); i++) {

Log.v("TAG", "Removing item: " + items.get(i).getName());

NewnewsEntryAdapter.remove(items.get(i));

}

NewnewsEntryAdapter.addAll(getNewsEntries(parentfolderid));

inFolder = items.get(0).getParentId();

Log.v("TAG", "Returning to folderid: " + inFolder);

//newsEntryAdapter.notifyDataSetChanged();

//Intent intent = new Intent(this, MainActivity.class);

//intent.putExtra("inFolder", inFolder);

//startActivity(intent);

}

return true;

}

return false;

}

private List<Items> getNewsEntries(String folderid) {

DBHelper db = new DBHelper(getApplicationContext());

return db.getItems(folderid);

}

public void SetAdapter(MyItemAdapter newsEntryAdapter) {

this.NewnewsEntryAdapter = newsEntryAdapter;

}

public MyItemAdapter ReturnAdapter(MyItemAdapter newsEntryAdapter) {

return NewnewsEntryAdapter;

}

/\*\*

\* Showing Dialog

\*/

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case progress\_bar\_type: // we set this to 0

pDialog = new ProgressDialog(this);

pDialog.setMessage("Downloading file. Please wait...");

pDialog.setIndeterminate(false);

pDialog.setMax(100);

pDialog.setProgressStyle(ProgressDialog.STYLE\_HORIZONTAL);

pDialog.setCancelable(true);

pDialog.show();

return pDialog;

default:

return null;

}

}

@Override

public void onNavigationDrawerItemSelected(int position) {

// update the main content by replacing fragments

FragmentManager fragmentManager = getSupportFragmentManager();

fragmentManager.beginTransaction()

.replace(R.id.container, PlaceholderFragment.newInstance(position + 1))

.commit();

}

public void onSectionAttached(int number) {

switch (number) {

case 1: // HOME ACTIVITY

mTitle = getString(R.string.title\_section1);

Log.v("TAG","In menu: " + mTitle);

break;

case 2: // CLEAR CACHE

//mTitle = getString(R.string.title\_section2);

ClearCache cache = new ClearCache(getApplicationContext());

cache.trimCache();

Log.v("TAG", "The cache was cleared");

break;

case 3: // LINKS

mTitle = getString(R.string.title\_section3);

Log.v("TAG","In menu: " + mTitle);

DBHelper db = new DBHelper(getApplicationContext());

String[] ar = {db.getAuth()};

Log.v("TAG", "The auth from the db is: " + ar);

DonLinks obj = new DonLinks(ar,MainActivity.this);

break;

case 4: //LOGOUT

// WORKS !!!

mTitle = getString(R.string.title\_section4);

Log.v("TAG","In menu: " + mTitle);

//Logout logout = new Logout();

DBHelper dbH = new DBHelper(getApplicationContext());

//logout.logOut(db.getAuth(), getApplicationContext());

String arg = dbH.getAuth();

Log.v("TAG","The auth from the db is: " + arg);

new Logout().execute(arg);

dbH.DropTables();

//Log.v("TAG", "LOGOUT RESPONSE" + logout.logout\_response());

Intent intent = new Intent(this, LoginActivity.class);

startActivity(intent);

break;

case 5: // GALLERY

mTitle = getString(R.string.title\_section5);

Log.v("TAG","In menu: " + mTitle);

//GET IMAGE IDs

ListPhotoes list = new ListPhotoes();

DBHelper dbHelper = new DBHelper(getApplicationContext());

list.listPhoto(getApplicationContext(), dbHelper.getAuth());

//SET IMAGE IDs

ArrayList<String> fileids = list.GetPhotoIds();

Iterator it = fileids.iterator();

ArrayList<String> fileURLs = null;

//GET DOWNLOAD URLs WITH FILE IDs

while (it.hasNext()) {

String[] arr = {it.next().toString(), dbHelper.getAuth()};

Don don = new Don(arr, getApplicationContext());

String file\_url = don.GetUrl();

fileURLs.add(file\_url);

}

//TODO - SEND THE IMAGE URLS TO THE GALLERY

break;

case 6: //RESYNC

// IDK IF IT WORKS

mTitle = getString(R.string.title\_section6);

Log.v("TAG", "In menu: " + mTitle);

// PROGRESS DIALOG

ProgressDialog pdialog = new ProgressDialog(MainActivity.this);

pdialog.setCancelable(false);

pdialog.setMessage("Loading ....");

pdialog.show();

Resync resync = new Resync(getApplicationContext());

// CLOSE PROGRESS

pdialog.dismiss();

Log.v("TAG","The resync is done! It was clicked from the navigation drawer!");

Intent inten = new Intent(getApplicationContext(), MainActivity.class);

startActivity(inten);

Log.v("TAG","THIS MESSAGE SHOULD NEVER BE SHOWN!!! ");

break;

}

}

public void restoreActionBar() {

ActionBar actionBar = getSupportActionBar();

actionBar.setNavigationMode(ActionBar.NAVIGATION\_MODE\_STANDARD);

actionBar.setIcon(R.drawable.treasure\_chest);

actionBar.setDisplayShowTitleEnabled(true);

actionBar.setTitle(mTitle);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

if (!mNavigationDrawerFragment.isDrawerOpen()) {

// Only show items in the action bar relevant to this screen

// if the drawer is not showing. Otherwise, let the drawer

// decide what to show in the action bar.

getMenuInflater().inflate(R.menu.main, menu);

// Associate searchable configuration with the SearchView

SearchManager searchManager = (SearchManager) getSystemService(Context.SEARCH\_SERVICE);

SearchView searchView = (SearchView) menu.findItem(R.id.action\_search)

.getActionView();

// searchView.setSearchableInfo(searchManager

// .getSearchableInfo(getComponentName()));

restoreActionBar();

return true;

}

return super.onCreateOptionsMenu(menu);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

//noinspection SimplifiableIfStatement

if (id == R.id.action\_settings) {

Intent AboutIntent = new Intent(this,About.class);

startActivity(AboutIntent);

return true;

}

if (id == R.id.action\_search) {

// search action

return true;

}

return super.onOptionsItemSelected(item);

}

/\*\*

\* A placeholder fragment containing a simple view.

\*/

public static class PlaceholderFragment extends Fragment {

/\*\*

\* The fragment argument representing the section number for this

\* fragment.

\*/

private static final String ARG\_SECTION\_NUMBER = "section\_number";

/\*\*

\* Returns a new instance of this fragment for the given section

\* number.

\*/

public static PlaceholderFragment newInstance(int sectionNumber) {

PlaceholderFragment fragment = new PlaceholderFragment();

Bundle args = new Bundle();

args.putInt(ARG\_SECTION\_NUMBER, sectionNumber);

fragment.setArguments(args);

return fragment;

}

public PlaceholderFragment() {

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

View rootView = inflater.inflate(R.layout.fragment\_main, container, false);

return rootView;

}

@Override

public void onAttach(Activity activity) {

super.onAttach(activity);

((MainActivity) activity).onSectionAttached(

getArguments().getInt(ARG\_SECTION\_NUMBER));

}

}

//--------------------------------LIFECYCLE -------------------------------

@Override

protected void onPause() {

super.onPause();

//notify("onPause");

}

@Override

protected void onResume() {

super.onResume();

//notify("onResume");

}

@Override

protected void onStop() {

super.onStop();

//notify("onStop");

}

@Override

protected void onDestroy() {

super.onDestroy();

//notify("onDestroy");

}

@Override

protected void onRestoreInstanceState(Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);

//notify("onRestoreInstanceState");

}

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

//notify("onSaveInstanceState");

}

private void notify(String methodName) {

String name = this.getClass().getName();

String[] strings = name.split("\\.");

Notification noti = new Notification.Builder(this)

.setContentTitle(methodName + " " + strings[strings.length - 1]).setAutoCancel(true)

.setSmallIcon(R.mipmap.ic\_launcher)

.setContentText(name).build();

NotificationManager notificationManager =

(NotificationManager) getSystemService(NOTIFICATION\_SERVICE);

notificationManager.notify((int) System.currentTimeMillis(), noti);

}

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//http://www.androidhive.info/2012/04/android-downloading-file-by-showing-progress-bar/

public class DownloadFileFromURL extends AsyncTask<String, String, String> {

/\*\*

\* Before starting background thread

\* Show Progress Bar Dialog

\*/

@Override

protected void onPreExecute() {

super.onPreExecute();

showDialog(progress\_bar\_type);

}

/\*\*

\* Downloading file in background thread

\*/

@Override

protected String doInBackground(String... f\_url) {

int count;

try {

URL url = new URL(f\_url[0]);

URLConnection conection = url.openConnection();

conection.connect();

// this will be useful so that you can show a tipical 0-100% progress bar

int lenghtOfFile = conection.getContentLength();

// download the file

InputStream input = new BufferedInputStream(url.openStream(), 8192);

//!!!!!!!!!!!!!!!!!!!//////////// NEED TO BE ADDED FORMAT OF THE FILE E.G .JPEG AFTER FILENAME

// Output stream

OutputStream output = new FileOutputStream("/sdcard/.FileChest/" /\*+ getName()\*/);

byte data[] = new byte[1024];

long total = 0;

while ((count = input.read(data)) != -1) {

total += count;

// publishing the progress....

// After this onProgressUpdate will be called

publishProgress("" + (int) ((total \* 100) / lenghtOfFile));

// writing data to file

output.write(data, 0, count);

}

// flushing output

output.flush();

// closing streams

output.close();

input.close();

} catch (Exception e) {

Log.e("Error: ", e.getMessage());

}

return null;

}

/\*\*

\* Updating progress bar

\*/

protected void onProgressUpdate(String... progress) {

// setting progress percentage

pDialog.setProgress(Integer.parseInt(progress[0]));

}

/\*\*

\* After completing background task

\* Dismiss the progress dialog

\* \*

\*/

@Override

protected void onPostExecute(String file\_url) {

// dismiss the dialog after the file was downloaded

dismissDialog(progress\_bar\_type);

// Displaying downloaded image into image view

// Reading image path from sdcard

// String imagePath = Environment.getExternalStorageDirectory().toString() + "/downloadedfile.jpg";

// setting downloaded into image view

//my\_image.setImageDrawable(Drawable.createFromPath(imagePath));

}

}

}

NAVIGATION DRAWER FRAGMENT

package com.chest.blue.filechest;

import android.support.v7.app.ActionBarActivity;

import android.app.Activity;

import android.support.v7.app.ActionBar;

import android.support.v4.app.Fragment;

import android.support.v4.app.ActionBarDrawerToggle;

import android.support.v4.view.GravityCompat;

import android.support.v4.widget.DrawerLayout;

import android.content.SharedPreferences;

import android.content.res.Configuration;

import android.os.Bundle;

import android.preference.PreferenceManager;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import android.widget.Toast;

/\*\*

\* Fragment used for managing interactions for and presentation of a navigation drawer.

\* See the <a href="https://developer.android.com/design/patterns/navigation-drawer.html#Interaction">

\* design guidelines</a> for a complete explanation of the behaviors implemented here.

\*/

public class NavigationDrawerFragment extends Fragment {

/\*\*

\* Remember the position of the selected item.

\*/

private static final String STATE\_SELECTED\_POSITION = "selected\_navigation\_drawer\_position";

/\*\*

\* Per the design guidelines, you should show the drawer on launch until the user manually

\* expands it. This shared preference tracks this.

\*/

private static final String PREF\_USER\_LEARNED\_DRAWER = "navigation\_drawer\_learned";

/\*\*

\* A pointer to the current callbacks instance (the Activity).

\*/

private NavigationDrawerCallbacks mCallbacks;

/\*\*

\* Helper component that ties the action bar to the navigation drawer.

\*/

private ActionBarDrawerToggle mDrawerToggle;

private DrawerLayout mDrawerLayout;

private ListView mDrawerListView;

private View mFragmentContainerView;

private int mCurrentSelectedPosition = 0;

private boolean mFromSavedInstanceState;

private boolean mUserLearnedDrawer;

public NavigationDrawerFragment() {

}

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// Read in the flag indicating whether or not the user has demonstrated awareness of the

// drawer. See PREF\_USER\_LEARNED\_DRAWER for details.

SharedPreferences sp = PreferenceManager.getDefaultSharedPreferences(getActivity());

mUserLearnedDrawer = sp.getBoolean(PREF\_USER\_LEARNED\_DRAWER, false);

if (savedInstanceState != null) {

mCurrentSelectedPosition = savedInstanceState.getInt(STATE\_SELECTED\_POSITION);

mFromSavedInstanceState = true;

}

// Select either the default item (0) or the last selected item.

selectItem(mCurrentSelectedPosition);

}

@Override

public void onActivityCreated(Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

// Indicate that this fragment would like to influence the set of actions in the action bar.

setHasOptionsMenu(true);

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

mDrawerListView = (ListView) inflater.inflate(

R.layout.navigation\_drawer, container, false);

mDrawerListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

selectItem(position);

}

});

mDrawerListView.setAdapter(new ArrayAdapter<String>(

getActionBar().getThemedContext(),

android.R.layout.simple\_list\_item\_activated\_1,

android.R.id.text1,

new String[]{

getString(R.string.title\_section1),

getString(R.string.title\_section2),

getString(R.string.title\_section3),

getString(R.string.title\_section4),

getString(R.string.title\_section5),

getString(R.string.title\_section6)

}));

mDrawerListView.setItemChecked(mCurrentSelectedPosition, true);

return mDrawerListView;

}

public boolean isDrawerOpen() {

return mDrawerLayout != null && mDrawerLayout.isDrawerOpen(mFragmentContainerView);

}

/\*\*

\* Users of this fragment must call this method to set up the navigation drawer interactions.

\*

\* @param fragmentId The android:id of this fragment in its activity's layout.

\* @param drawerLayout The DrawerLayout containing this fragment's UI.

\*/

public void setUp(int fragmentId, DrawerLayout drawerLayout) {

mFragmentContainerView = getActivity().findViewById(fragmentId);

mDrawerLayout = drawerLayout;

// set a custom shadow that overlays the main content when the drawer opens

mDrawerLayout.setDrawerShadow(R.drawable.drawer\_shadow, GravityCompat.START);

// set up the drawer's list view with items and click listener

ActionBar actionBar = getActionBar();

actionBar.setDisplayHomeAsUpEnabled(true);

actionBar.setHomeButtonEnabled(true);

actionBar.setIcon(R.drawable.silverchest);

// ActionBarDrawerToggle ties together the the proper interactions

// between the navigation drawer and the action bar app icon.

mDrawerToggle = new ActionBarDrawerToggle(

getActivity(), /\* host Activity \*/

mDrawerLayout, /\* DrawerLayout object \*/

R.drawable.ic\_drawer, /\* nav drawer image to replace 'Up' caret \*/

R.string.navigation\_drawer\_open, /\* "open drawer" description for accessibility \*/

R.string.navigation\_drawer\_close /\* "close drawer" description for accessibility \*/

) {

@Override

public void onDrawerClosed(View drawerView) {

super.onDrawerClosed(drawerView);

if (!isAdded()) {

return;

}

getActivity().supportInvalidateOptionsMenu(); // calls onPrepareOptionsMenu()

}

@Override

public void onDrawerOpened(View drawerView) {

super.onDrawerOpened(drawerView);

if (!isAdded()) {

return;

}

if (!mUserLearnedDrawer) {

// The user manually opened the drawer; store this flag to prevent auto-showing

// the navigation drawer automatically in the future.

mUserLearnedDrawer = true;

SharedPreferences sp = PreferenceManager

.getDefaultSharedPreferences(getActivity());

sp.edit().putBoolean(PREF\_USER\_LEARNED\_DRAWER, true).apply();

}

getActivity().supportInvalidateOptionsMenu(); // calls onPrepareOptionsMenu()

}

};

// If the user hasn't 'learned' about the drawer, open it to introduce them to the drawer,

// per the navigation drawer design guidelines.

if (!mUserLearnedDrawer && !mFromSavedInstanceState) {

mDrawerLayout.openDrawer(mFragmentContainerView);

}

// Defer code dependent on restoration of previous instance state.

mDrawerLayout.post(new Runnable() {

@Override

public void run() {

mDrawerToggle.syncState();

}

});

mDrawerLayout.setDrawerListener(mDrawerToggle);

}

private void selectItem(int position) {

mCurrentSelectedPosition = position;

if (mDrawerListView != null) {

mDrawerListView.setItemChecked(position, true);

}

if (mDrawerLayout != null) {

mDrawerLayout.closeDrawer(mFragmentContainerView);

}

if (mCallbacks != null) {

mCallbacks.onNavigationDrawerItemSelected(position);

}

}

@Override

public void onAttach(Activity activity) {

super.onAttach(activity);

try {

mCallbacks = (NavigationDrawerCallbacks) activity;

} catch (ClassCastException e) {

throw new ClassCastException("Activity must implement NavigationDrawerCallbacks.");

}

}

@Override

public void onDetach() {

super.onDetach();

mCallbacks = null;

}

@Override

public void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

outState.putInt(STATE\_SELECTED\_POSITION, mCurrentSelectedPosition);

}

@Override

public void onConfigurationChanged(Configuration newConfig) {

super.onConfigurationChanged(newConfig);

// Forward the new configuration the drawer toggle component.

mDrawerToggle.onConfigurationChanged(newConfig);

}

@Override

public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {

// If the drawer is open, show the global app actions in the action bar. See also

// showGlobalContextActionBar, which controls the top-left area of the action bar.

if (mDrawerLayout != null && isDrawerOpen()) {

inflater.inflate(R.menu.global, menu);

showGlobalContextActionBar();

}

super.onCreateOptionsMenu(menu, inflater);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (mDrawerToggle.onOptionsItemSelected(item)) {

return true;

}

//EXAMPLE ACTION BAR MENU

/\*

if (item.getItemId() == R.id.action\_example) {

Toast.makeText(getActivity(), "Example action.", Toast.LENGTH\_SHORT).show();

return true;

}

\*/

return super.onOptionsItemSelected(item);

}

/\*\*

\* Per the navigation drawer design guidelines, updates the action bar to show the global app

\* 'context', rather than just what's in the current screen.

\*/

private void showGlobalContextActionBar() {

ActionBar actionBar = getActionBar();

actionBar.setDisplayShowTitleEnabled(true);

actionBar.setNavigationMode(ActionBar.NAVIGATION\_MODE\_STANDARD);

actionBar.setTitle(R.string.app\_name);

}

private ActionBar getActionBar() {

return ((ActionBarActivity) getActivity()).getSupportActionBar();

}

/\*\*

\* Callbacks interface that all activities using this fragment must implement.

\*/

public static interface NavigationDrawerCallbacks {

/\*\*

\* Called when an item in the navigation drawer is selected.

\*/

void onNavigationDrawerItemSelected(int position);

}

}

LOGOUT

package com.chest.blue.filechest;

import android.os.AsyncTask;

import android.util.Log;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.net.UnknownHostException;

import java.security.KeyStore;

import java.security.cert.CertificateException;

import java.security.cert.X509Certificate;

import org.apache.http.HttpResponse;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.conn.scheme.Scheme;

import org.apache.http.conn.ssl.SSLSocketFactory;

import org.apache.http.impl.client.DefaultHttpClient;

import javax.net.ssl.SSLContext;

import javax.net.ssl.TrustManager;

import javax.net.ssl.X509TrustManager;

class Logout extends AsyncTask<String, Void, String> {

private Exception exception = null;

public String ResultString = null;

public int response\_code;

public String response\_string;

protected String doInBackground(String... auth) {

HttpClient client = new DefaultHttpClient();

try {

client.getConnectionManager().getSchemeRegistry().register(getMockedScheme());

HttpGet GET = new HttpGet("https://api.pcloud.com/logout?auth=" + auth[0]);

Log.v("TAG", "Check if it works !!!");

Log.v("TAG", "The send request is: " + "https://api.pcloud.com/logout?auth=" + auth[0]);

HttpResponse response = client.execute(GET);

Log.v("TAG", response.toString());

BufferedReader rd = new BufferedReader(new InputStreamReader(response.getEntity().getContent()));

response\_code = response.getStatusLine().getStatusCode();

String line = "";

while ((line = rd.readLine()) != null) {

response\_string = response\_string + line;

}

Log.v("TAG", "Response string: " + response\_string);

} catch (Exception e) {

Log.v("TAG", "SSL Error " + e.toString());

}

client.getConnectionManager().shutdown();

return response\_string;

}

protected void onPostExecute(String stream\_url) {

// TODO: check this.exception

// TODO: do something with the feed

if (this.exception != null)

this.exception.printStackTrace();

this.ResultString = stream\_url;

}

public Scheme getMockedScheme() throws Exception {

MySSLSocketFactory mySSLSocketFactory = new MySSLSocketFactory();

return new Scheme("https", mySSLSocketFactory, 443);

}

class MySSLSocketFactory extends SSLSocketFactory {

javax.net.ssl.SSLSocketFactory socketFactory = null;

public MySSLSocketFactory(KeyStore truststore) throws Exception {

super(truststore);

socketFactory = getSSLSocketFactory();

}

public MySSLSocketFactory() throws Exception {

this(null);

}

@Override

public Socket createSocket(Socket socket, String host, int port, boolean autoClose) throws IOException,

UnknownHostException {

return socketFactory.createSocket(socket, host, port, autoClose);

}

@Override

public Socket createSocket() throws IOException {

return socketFactory.createSocket();

}

javax.net.ssl.SSLSocketFactory getSSLSocketFactory() throws Exception {

SSLContext sslContext = SSLContext.getInstance("TLS");

TrustManager tm = new X509TrustManager() {

public void checkClientTrusted(X509Certificate[] chain, String authType) throws CertificateException {

}

public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException {

}

public X509Certificate[] getAcceptedIssuers() {

return null;

}

};

sslContext.init(null, new TrustManager[]{tm}, null);

return sslContext.getSocketFactory();

}

}

}

LIST FOLDERS

package com.chest.blue.filechest;

/\*\*

\* Created by Blue on 6/6/2015.

\*/

import android.util.Log;

import org.apache.http.HttpResponse;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.conn.scheme.Scheme;

import org.apache.http.conn.ssl.SSLSocketFactory;

import org.apache.http.impl.client.DefaultHttpClient;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.net.UnknownHostException;

import java.security.KeyStore;

import java.security.cert.CertificateException;

import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;

import javax.net.ssl.TrustManager;

import javax.net.ssl.X509TrustManager;

@SuppressWarnings("deprecation")

public class ListFolders {

String response\_string = "";

public void getFolder(String auth) {

HttpClient client = new DefaultHttpClient();

try {

client.getConnectionManager().getSchemeRegistry().register(getMockedScheme());

HttpGet GET\_folders = new HttpGet("https://api.pcloud.com/listfolder?folderid=0&recursive=1&nofiles=0&auth=" + auth);

HttpResponse response = client.execute(GET\_folders);

BufferedReader rd = new BufferedReader(new InputStreamReader(response.getEntity().getContent()));

String line = "";

while ((line = rd.readLine()) != null) {

System.out.println(line);

response\_string = response\_string + line;

}

// REMOVE HASH BECAUSE OF NumberFormatException

for (int index = response\_string.indexOf("hash");

index >= 0;

index = response\_string.indexOf("hash", index + 1))

{

String straaa = response\_string.substring(index + 7, index + 24);

response\_string = response\_string.replaceAll(straaa, "");

//System.out.println(straaa);

}

} catch (Exception e) {

Log.v("TAG", "SSL Expection: " + e.toString());

} finally {

client.getConnectionManager().shutdown();

}

}

public Scheme getMockedScheme() throws Exception {

MySSLSocketFactory mySSLSocketFactory = new MySSLSocketFactory();

return new Scheme("https", mySSLSocketFactory, 443);

}

public String foldersList() {

return response\_string;

}

class MySSLSocketFactory extends SSLSocketFactory {

javax.net.ssl.SSLSocketFactory socketFactory = null;

public MySSLSocketFactory(KeyStore truststore) throws Exception {

super(truststore);

socketFactory = getSSLSocketFactory();

}

public MySSLSocketFactory() throws Exception {

this(null);

}

@Override

public Socket createSocket(Socket socket, String host, int port, boolean autoClose) throws IOException,

UnknownHostException {

return socketFactory.createSocket(socket, host, port, autoClose);

}

@Override

public Socket createSocket() throws IOException {

return socketFactory.createSocket();

}

javax.net.ssl.SSLSocketFactory getSSLSocketFactory() throws Exception {

SSLContext sslContext = SSLContext.getInstance("TLS");

TrustManager tm = new X509TrustManager() {

public void checkClientTrusted(X509Certificate[] chain, String authType) throws CertificateException {

}

public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException {

}

public X509Certificate[] getAcceptedIssuers() {

return null;

}

};

sslContext.init(null, new TrustManager[]{tm}, null);

return sslContext.getSocketFactory();

}

}

}

ITEMS

package com.chest.blue.filechest;

/\*\*

\* Created by Blue on 6/28/2015.

\*/

public class Items {

private final String name;

private final String id;

private final String fileid;

private final String parentid;

private final String modified;

public Items(final String name, final String id,String fileid, String parentid, String modified ) {

this.name = name;

this.id = id;

this.fileid = fileid;

this.parentid = parentid;

this.modified = modified;

}

public String getName() {return name;}

public String getId() {

return id;

}

public String getFileId() {

return fileid;

}

public String getParentId() {return parentid;}

public String getModified() {return modified;}

}

DOWNLOAD ASYNC TASK

package com.chest.blue.filechest;

/\*\*

\* Created by Elysi on 8/9/2015.

\*/

import android.content.Context;

import android.os.AsyncTask;

import android.util.Log;

import org.apache.http.HttpResponse;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.conn.scheme.Scheme;

import org.apache.http.conn.ssl.SSLSocketFactory;

import org.apache.http.impl.client.DefaultHttpClient;

import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.net.UnknownHostException;

import java.security.KeyStore;

import java.security.cert.CertificateException;

import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;

import javax.net.ssl.TrustManager;

import javax.net.ssl.X509TrustManager;

//http://stackoverflow.com/questions/18870065/retrieving-a-returned-string-from-asynctask-in-android

class DownloadAsyncTask extends AsyncTask<String[], Void, String> {

TheInterface listener;

private Exception exception = null;

public String ResultString = null;

public int response\_code;

public String response\_string;

public String URL = "";

public DownloadAsyncTask (Context context)

{

listener = (TheInterface) context;

}

public interface TheInterface {

public void theMethod(String result);

}

protected String doInBackground(String[]... arr) {

String fileid = arr[0].toString();

String auth = arr[1].toString();

HttpClient client = new DefaultHttpClient();

try {

client.getConnectionManager().getSchemeRegistry().register(getMockedScheme());

HttpGet GET = new HttpGet("https://api.pcloud.com/getfilelink?fileid=" + fileid + "&forcedownload=1&auth=" + auth);

Log.v("TAG", "Check if it works !!!");

HttpResponse response = client.execute(GET);

Log.v("TAG", response.toString());

BufferedReader rd = new BufferedReader(new InputStreamReader(response.getEntity().getContent()));

response\_code = response.getStatusLine().getStatusCode();

String line = "";

while ((line = rd.readLine()) != null) {

response\_string = response\_string + line;

}

} catch (Exception e) {

Log.v("TAG", "SSL Errror " + e.toString());

}

JSONParser parser = new JSONParser();

try {

JSONObject jsonObject = (JSONObject) parser.parse(response\_string);

String path = jsonObject.get("path").toString();

path = path.replaceAll("\\/", "/");

Log.v("TAG", "Path: " + path);

JSONArray host = (JSONArray) jsonObject.get("hosts");

Log.v("TAG", "host: " + host.toString());

URL = (String) host.get(0);

URL = URL + path;

Log.v("TAG", "URL: " + URL);

} catch (Exception e1) {

e1.printStackTrace();

try {

JSONObject jsonObject = (JSONObject) parser.parse(response\_string);

String error = (String) jsonObject.get("error");

return "";

//Toast.makeText(context, error, Toast.LENGTH\_SHORT).show();

} catch (Exception e) {

e.printStackTrace();

}

}

client.getConnectionManager().shutdown();

return URL;

}

protected void onPostExecute(String stream\_url) {

// TODO: check this.exception

// TODO: do something with the feed

if (this.exception != null)

this.exception.printStackTrace();

this.ResultString = stream\_url;

if (listener != null)

{

listener.theMethod(stream\_url);

}

}

public Scheme getMockedScheme() throws Exception {

MySSLSocketFactory mySSLSocketFactory = new MySSLSocketFactory();

return new Scheme("https", mySSLSocketFactory, 443);

}

class MySSLSocketFactory extends SSLSocketFactory {

javax.net.ssl.SSLSocketFactory socketFactory = null;

public MySSLSocketFactory(KeyStore truststore) throws Exception {

super(truststore);

socketFactory = getSSLSocketFactory();

}

public MySSLSocketFactory() throws Exception {

this(null);

}

@Override

public Socket createSocket(Socket socket, String host, int port, boolean autoClose) throws IOException,

UnknownHostException {

return socketFactory.createSocket(socket, host, port, autoClose);

}

@Override

public Socket createSocket() throws IOException {

return socketFactory.createSocket();

}

javax.net.ssl.SSLSocketFactory getSSLSocketFactory() throws Exception {

SSLContext sslContext = SSLContext.getInstance("TLS");

TrustManager tm = new X509TrustManager() {

public void checkClientTrusted(X509Certificate[] chain, String authType) throws CertificateException {

}

public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException {

}

public X509Certificate[] getAcceptedIssuers() {

return null;

}

};

sslContext.init(null, new TrustManager[]{tm}, null);

return sslContext.getSocketFactory();

}

}

}

GRADLE BUILD

buildscript {

repositories {

maven { url 'https://maven.fabric.io/public' }

}

dependencies {

classpath 'io.fabric.tools:gradle:1.+'

}

}

apply plugin: 'com.android.application'

apply plugin: 'io.fabric'

repositories {

maven { url 'https://maven.fabric.io/public' }

}

android {

compileSdkVersion 22

buildToolsVersion "22.0.1"

packagingOptions {

exclude 'META-INF/DEPENDENCIES'

exclude 'META-INF/NOTICE'

exclude 'META-INF/LICENSE'

}

defaultConfig {

applicationId "com.chest.blue.filechest"

minSdkVersion 16

targetSdkVersion 22

versionCode 1

versionName "1.0"

testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

}

buildTypes {

release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

}

dependencies {

compile fileTree(include: ['\*.jar'], dir: 'libs')

compile files('lib/httpclient-4.4.jar')

compile files('lib/httpmime-4.4.jar')

compile files('lib/json-simple-1.1.1.jar')

compile files('lib/org.apache.commons.codec-1.8.jar')

compile files('lib/org.apache.commons.httpclient.jar')

compile('com.crashlytics.sdk.android:crashlytics:2.4.0@aar') {

transitive = true;

}

//22.1.1

// TEST ONLY DEPENDENCIES

androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2'

androidTestCompile 'com.android.support.test:runner:0.3'

androidTestCompile 'com.android.support.test.espresso:espresso-contrib:2.2'

compile 'com.android.support:appcompat-v7:22.2.0'

compile 'com.google.android.gms:play-services:7.5.0'

compile 'com.android.support:support-v4:22.2.0'

}

MANIFEST

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.chest.blue.filechest" >

<!-- To auto-complete the email text field in the login form with the user's emails -->

<uses-permission android:name="android.permission.GET\_ACCOUNTS" />

<uses-permission android:name="android.permission.READ\_PROFILE" />

<uses-permission android:name="android.permission.READ\_CONTACTS" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.VIBRATE" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.WRITE\_INTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<application

android:allowBackup="true"

android:icon="@drawable/silverchest"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name=".LoginActivity"

android:configChanges="keyboardHidden|orientation|screenSize"

android:label="@string/app\_name"

android:screenOrientation="portrait"

android:windowSoftInputMode="adjustResize|stateVisible" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity

android:name=".MainActivity"

android:label="@string/title\_activity\_main"

android:parentActivityName=".LoginActivity" >

<meta-data

android:name="android.support.PARENT\_ACTIVITY"

android:value="com.chest.blue.filechest.LoginActivity" />

</activity>

<!--

<activity

android:name=".ui.MainPhotoActivity"

android:label="@string/app\_name"

android:launchMode="singleTop">

</activity>

-->

<meta-data

android:name="io.fabric.ApiKey"

android:value="04efbdb942af5e4a781952bafbea27b01114a118" />

<activity

android:name=".RegisterActivity"

android:configChanges="keyboardHidden|orientation|screenSize"

android:label="@string/title\_activity\_register"

android:parentActivityName=".LoginActivity"

android:screenOrientation="portrait"

android:windowSoftInputMode="adjustResize|stateVisible" >

<meta-data

android:name="android.support.PARENT\_ACTIVITY"

android:value="com.chest.blue.filechest.LoginActivity" />

</activity>

<activity

android:name=".SearchResultsActivity"

android:parentActivityName=".MainActivity" >

<intent-filter>

<action android:name="android.intent.action.SEARCH" />

</intent-filter>

<meta-data

android:name="android.app.searchable"

android:resource="@xml/searchable" />

</activity>

<receiver

android:name=".NetworkChangeReceiver"

android:label="NetworkChangeReceiver" >

<intent-filter>

<action android:name="android.net.conn.CONNECTIVITY\_CHANGE" />

<action android:name="android.net.wifi.WIFI\_STATE\_CHANGED" />

</intent-filter>

</receiver>

<activity

android:name=".LinksActivity"

android:label="@string/title\_activity\_links" >

</activity>

<activity

android:name=".SettingsActivity"

android:label="@string/title\_activity\_settings"

android:parentActivityName=".MainActivity" >

<meta-data

android:name="android.support.PARENT\_ACTIVITY"

android:value="com.chest.blue.filechest.MainActivity" />

</activity>

<activity

android:name=".About"

android:label="@string/title\_activity\_test"

android:parentActivityName=".MainActivity" >

<meta-data

android:name="android.support.PARENT\_ACTIVITY"

android:value="com.chest.blue.filechest.MainActivity" />

</activity>

</application>

</manifest>

**Използвани източници**

1. https://docs.pcloud.com/
2. http://developer.android.com/develop/index.html
3. https://try.crashlytics.com/
4. http://ec.europa.eu/eurostat/statistics-explained/index.php/
5. https://docs.gradle.org/current/userguide/userguide
6. http://www.w3schools.com/json/