REVIEW CODE VNC

Version code : 17

Contents

[A.Cấu trúc code 2](#_Toc352071416)

[B. Cấu hình server Utils 2](#_Toc352071417)

[C.Follow của Google Analytics 3](#_Toc352071418)

[C.1. Follow các màn hình chính(Tạo tại com.xing.joy.common.CoreActivity, và com.xing.joy.common.CoreGameActivity) 3](#_Toc352071419)

[C.2.Follow của Web(Tại màn hình com.xing.joy.others.Top) 4](#_Toc352071420)

[C.3.Follow của clear game 1 và clear game 3(Tại com.xing.joy.play.DynamicSong) 5](#_Toc352071421)

[C4.Follow của update ứng dụng hay cài mới ứng dụng(Tại jp.co.xing.utaehon.VNCStartUpActivity) 5](#_Toc352071422)

[D.Follow di chuyển các màn hình 6](#_Toc352071423)

[E. Follow các màn hình 6](#_Toc352071424)

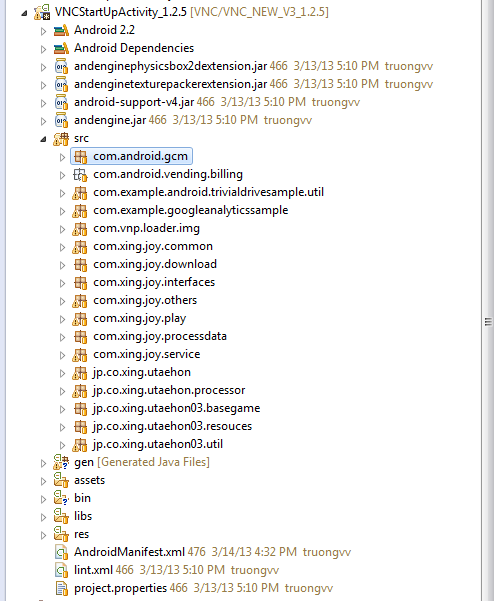
[E.1. jp.co.xing.utaehonStartUp Screen(Đảm bảo các icon tạo từ bản 1.2.3 trở xuống hoạt động bình thường) 6](#_Toc352071425)

[E.2. VNCStartUpActivity Screen(của jp.co.xing.utaehon. VNCStartUpActivity) 6](#_Toc352071426)

[E.3. Top Screen(com.xing.joy.others.Top) 7](#_Toc352071427)

[E.4. Buy Screen(extends com.xing.joy.others.BaseInappActivity)(class com.xing.joy.others.Buy) 10](#_Toc352071428)

# A.Cấu trúc code



# B. Cấu hình server Utils

Thông tin server lưu trữ tại res/values/url.xml

Thông tin server được quản lý bởi class “jp.co.xing.utaehon03.util. UrlUtils”

**public** **class** UrlUtils {

**private** Context context;

**public** UrlUtils(Context context) {

**this**.context = context;

}

**private** String getServer() {

**return** context.getResources().getString(R.string.*server\_real*);

}

**public** String getUrl(**int** res) {

String folderServer = context.getResources().getString(res);

**if** (folderServer.startsWith("http://")) {

**return** folderServer;

}

**return** getServer() + context.getResources().getString(res);

}

}

Trong đó : R.string.*server\_real* được lưu trữ tại res/values/url.xml, là domain hoặc ip của server.

Sử dụng bằng cách :

String serverUrl = **new** UrlUtils(context).getUrl(res);

Trong đó res được lưu trữ tại res/values/url ví dụ “R.string.*url\_package\_download*”, context là đối số truyền vào là màn hình đang sử dụng.

# C.Follow của Google Analytics

### C.1. Follow các màn hình chính(Tạo tại com.xing.joy.common.CoreActivity, và com.xing.joy.common.CoreGameActivity)

Tạo biến class lưu thông tin

**protected** GAUTils gauTils = **new** GAUTils();

**protected** Logtime logtime;

Khởi tạo khi vào màn hình, tại onCreate

Logtime.*deviceId* = Secure.*getString*(getContentResolver(),Secure.*ANDROID\_ID*);

Logtime.*urlLogPassword* = getString(R.string.*url\_log\_password*);

Logtime.*urlLogDevice* = getString(R.string.*url\_log\_device*);

logtime = **new** Logtime(**this**);

logtime.Logstart(getNameCount());

logtime.LogPageStart = getNameCount();

gauTils.onCreate(**this**);

**if** (!"".equals(getNameCount()) && getNameCount() != **null**)

gauTils.sendView(getNameCount());

Tại onStart khởi động google analytics

gauTils.onStart();

Tại onResume, khởi động lại logTime

logtime.onResume();

Tại onPause tạm thời dừng logTime

logtime.onPause();

Hàm logEnd dùng để đẩy thông tin google analytics

**public** **void** logEnd() {

logtime.Logend(getNameCount());

Log.*e*("GAV2", logtime.getTimeTotal() + " " + getNameCount());

**if** (!"".equals(getNameCount()) && getNameCount() != **null**) {

String category = getResources().getString(

R.string.*namescreen\_Access\_Count*);

String name = getResources().getString(

R.string.*namescreen\_Play\_time*);

Log.*d*("NAME : ", getNameCount() + "sssss");

gauTils.sendTiming(logtime.getTimeTotal(), category, name,

getNameCount());

}

gauTils.onStop();

}

Hàm public abstract String getNameCount() Lấy tên tương ứng tại các màn hình. Hàm này được viết lại tại các màn hình tương ứng.

### C.2.Follow của Web(Tại màn hình com.xing.joy.others.Top)

Sử dùng biến class

**private** Logtime logtimeForWeb;

Tại onResume khởi động lại nếu đã có

**if** (logtimeForWeb != **null**) {

logtimeForWeb.onResume();

}

Tại onPause tạm dừng tính thời gian

**if** (logtimeForWeb != **null**)

logtimeForWeb.onPause();

Tại onCreateWeb khởi động

logWEBD = getResources().getString(R.string.*namescreen\_webView*);

logtimeForWeb = **new** Logtime(**this**);

logtimeForWeb.Logstart(logWEBD);

logtimeForWeb.LogPageStart = logWEBD;

gauTils.sendView(logWEBD);

Khi đóng webView lại thì thực hiện push data google analytics

**if** (logtimeForWeb != **null**) {

logtimeForWeb.Logend(logWEBD);

**long** time = logtimeForWeb.getTimeTotal();

**if** (time > 0) {

String category = logWEBD;

String name = "";

category = getResources().getString(R.string.*namescreen\_Access\_Count*);

name = getResources().getString(R.string.*namescreen\_Play\_time*);

String label = category;

gauTils.sendTiming(time, category, name, label);

}

}

logtimeForWeb = **null**;

gauTils.sendView(getNameCount());

### C.3.Follow của clear game 1 và clear game 3(Tại com.xing.joy.play.DynamicSong)

Tạo BroadCast đợi thông tin

BroadcastReceiver broadcastReceiver2 = **new** BroadcastReceiver() {

@Override

**public** **void** onReceive(Context context, Intent intent) {

String category = intent.getStringExtra("category");

**if** (category != **null** && !"".equals(category.trim())) {

gauTils.sendEvent(category, category, category);

gauTils.sendTiming(1l, category, category, category);

}

}

};

Add thông tin broadcast vào màn hình tại onCreate

registerReceiver(broadcastReceiver2, **new** IntentFilter("jp.co.xing.utaehon03.action"));

Tại onStop loại bỏ Broadcast, Khi rời màn hình thì disable chức năng này.

**try** {

unregisterReceiver(broadcastReceiver2);

} **catch** (Exception exception) {

}

### C4.Follow của update ứng dụng hay cài mới ứng dụng(Tại jp.co.xing.utaehon.VNCStartUpActivity)

Tại hàm onResume check việc cài đặt ứng dụng mới hay update thông qua hàm countDownload.

Thông tin version được lưu trong **“**\_now\_version\_name**”** ShareReference. Thực hiện so sánh version code hiện tại của ứng dụng và version code được lưu trong **“**\_now\_version\_name**”**. Nếu versionCode == 0 là trường hợp cài mới ứng dụng, ngược lại là trường hợp update ứng dụng.

**private** **void** countDownload() {

DataStore.*getInstance*().init(**this**);

String keyName = "\_now\_version\_name";

**int** versionCode = DataStore.*getInstance*().get(keyName, 0);

**int** versionCodeOfApp = -1;

**try** {

versionCodeOfApp = **this**.getPackageManager().getPackageInfo(**this**.getPackageName(), 0).versionCode;

} **catch** (NameNotFoundException e) {

}

**if** (versionCode != versionCodeOfApp && versionCodeOfApp != -1) {

**if** (versionCode != 0) {

**if** (versionCodeOfApp > versionCode) {

String category = getResources().getString(R.string.*namescreen\_count\_update*);

String action = "";

String label = "";

gauTils.sendEvent(category, category, category);

gauTils.sendTiming(1l, category, category, category);

}

} **else** {

String category = getResources().getString(R.string.*namescreen\_count\_downnload*);

String action = "";

String label = "";

gauTils.sendEvent(category, category, category);

gauTils.sendTiming(1l, category, category, category);

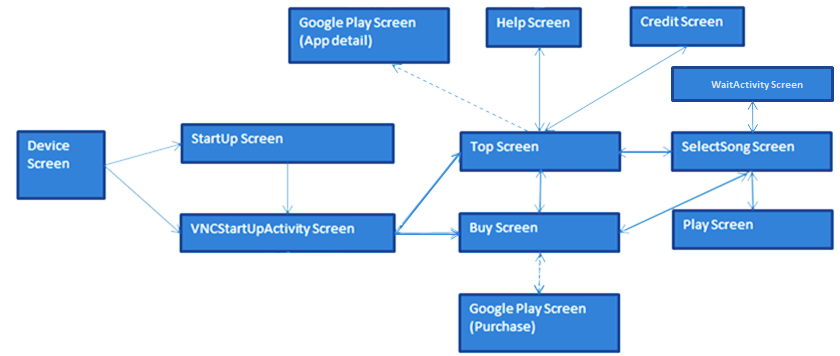
}

DataStore.*getInstance*().save(keyName, versionCodeOfApp);

}

}

# D.Follow di chuyển các màn hình



Từ màn hình thiết bị khi click vào icon của ứng dụng(shortcut) có thể khởi động ứng dụng vào theo 1 trong 2 cách.

1. Nếu là icon cũ(Được tạo trước bản 1.2.3) sẽ khởi động “StartUp Screen”,ngay lập tực sẽ vào “VNCStartUpActicity Screen”.
2. Icon được tạo từ các bản từ 1.2.3, sẽ khởi động vào “VNCStartUpActicity Screen”.

* Trong cả 2 trường hợp, app đều hoạt động bình thường.
* Thêm follow xử lý khi “SelectSong Screen” bị lỗi outOfMemory, khi đó chuyển qua màn hình “WaitActivity Screen”.
* Các follow khác giữ nguyên.

# E. Follow các màn hình

### E.1. jp.co.xing.utaehonStartUp Screen(Đảm bảo các icon tạo từ bản 1.2.3 trở xuống hoạt động bình thường)

Tại onCreate của jp.co.xing.utaehon.StartUp ngay khi khởi tạo màn hình, Chuyển qua màn hình VNCStartUpActivity Screen và đóng màn hình lại.

startActivity(**new** Intent(**this**, VNCStartUpActivity.**class**));

finish();

Quá trình này không gây delay, Người dùng nhìn thấy ngay màn hình VNCStartUpActivity.

### E.2. VNCStartUpActivity Screen(của jp.co.xing.utaehon. VNCStartUpActivity)

1. Thực hiện chức năng google analytics và tạo shortcut nếu cài lần đầu hoặc mới được update nhờ hàm countDownload, Được gọi tại hàm onResume.

2. Nếu màn hình màn hình cũ nhất được sử dụng là màn hình Buy và app vừa được sử dụng thì vào luôn màn hình Buy.

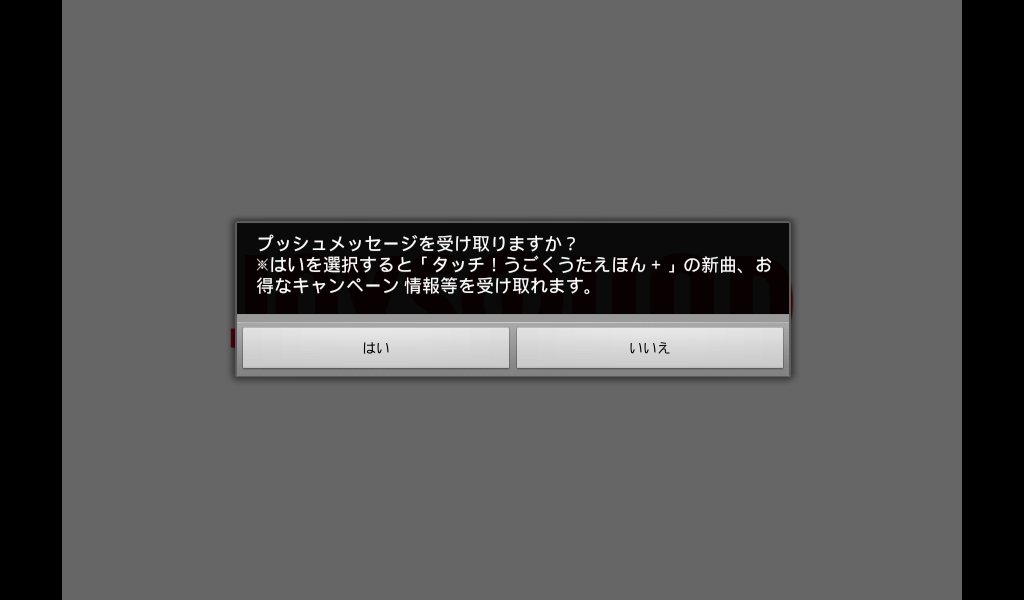
**if** (appData.getStringData(IDataActions.*CLASS\_NAME*).contains("com.xing.joy.others.Buy") && appData.getIntData(IDataActions.*PROCESS\_ID*) == android.os.Process.*myPid*()) {

CommonUtils.*startNewActivity*(**this**, Buy.**class**, "");

}

3. Nếu thông tin config “AllowReceivePush” chưa được lưu thì hiển thị popUp chọn lựa.

appData.getStringData("AllowReceivePush").equalsIgnoreCase("")



Nếu thông tin “AllowReceivePush” là yes thì thực hiện register google analytics.

CommonUtils.*registerGCM*(**true**, getApplicationContext());

Nếu thông tin “AllowReceivePush” đã được đăng ký thì chuyển qua màn hình Top Screen.

### E.3. Top Screen(com.xing.joy.others.Top)



Màn hình “Top Screen”, luồng follow chính không thay đổi.

1. Hiển thị thông tin mới tại icon(Bên trái góc trên, cho phép hiển thị webView).Thực hiện sau 1,5s vào màn hình.

**if** (checkFileExists(getFilesDir().getPath() + "/" + "top\_news\_iphone.png")) {

setImageTopNew(**false**);

}

2. Check thông tin version mới, sau khi vào màn hình 1,5s.

Thông tin có version mới lấy thông qua hàm checkNewVersionAPK, version mới nhất lấy từ “server/vol3general/versions.ini” và lưu vào “version\_on\_market” trong shareReference. Hàm checkNewVersionAPK trả về true nếu có version mới.

* Nếu “NotShowPopupNewVersion” lưu trong ShareRefrence trả về false, sẽ thực hiện gọi đến hàm checkNewVersionAPK() để lấy thông tin versin mới hay không.
* Sau khi check “NotShowPopupNewVersion” (có thể check thêm version mới) sẽ gọi hàm checkDownloadData.
  + Nếu có version.
    - Hiển thị popup thông báo version mới.
    - Tạo callback Cho màn hình Top.



* + Nếu không có version mới. Gọi getInfo = (GetNewInformation) **new** GetNewInformation().execute(); để lấy thông tin các file sẽ download.

3. Lấy thông tin các file sẽ download.Tại GetNewInformation.

* Tại doInBackground,Lấy thông tin từ các file “server/vol3general/update\_free.ini”, “server/vol3general/update\_freesong.ini”,“server/vol3general/update\_webview.ini" được lưu trong listDownload.
* Trong trường hợp file “a\_otf\_jun501pro\_bold.otf” chưa tồn tại thì đánh dấu các file đã lấy từ “server/vol3general/update\_free.ini”, “server/vol3general/update\_freesong.ini” là đã được download. Thông tin sẽ download sẽ là "vol3\_part0.zip" và thông tin lấy từ “server/vol3general/update\_webview.ini".
* Tại onPostExecute.
  + Nếu là lần đầu và có listDownload sẽ thực hiện download.

**if** (!**new** File(memory.getPathFileInternalMemory() + "/" + "a\_otf\_jun501pro\_bold.otf").exists()) {

**if** (download == **null** && listDownload != **null** && listDownload.size() > 0) {

download = (TopDataDownload) **new**

TopDataDownload(Top.**this**).execute((FilesModel[]) listDownload.toArray(**new**

FilesModel[listDownload.size()]));

}

}

* + Nếu đã từng download sẽ hiện thị popup thông báo có muốn update content không.Khi click đồng ý sẽ thực hiện download content.

download = (TopDataDownload) **new** TopDataDownload(Top.**this**).execute((FilesModel[]) listDownload.toArray(**new** FilesModel[listDownload.size()]));

4. Thực hiện download content về và unZip data (TopDataDownload, TopDataUnzip).

* TopDataDownload extends com.xing.joy.processdata.Download;
  + Khi download thành công thực hiện unZip(TopDataUnzip), gọi tại onPostExecute;
  + Khi download không thành công Và file “a\_otf\_jun501pro\_bold.otf” không tồn tại thì hiển thị redownload và ẩn process đi(Theo trường hợp dưới).

**if** (!**new** File(memory.getPathFileInternalMemory() + "/" + "a\_otf\_jun501pro\_bold.otf").exists()) {

setBacgroundResource(imgRedownload, R.drawable.*redownload\_iphone*);

isTouchRedownload = **true**;

}

* + Khi dowload không thành công Và file “a\_otf\_jun501pro\_bold.otf” tồn tại hiển thị như khi đã có content. Ẩn process đi.

**if** (downloadBar != **null**) {

downloadBar.setVisibility(View.*GONE*);

downloadBar.reset();

download = **null**;

}

**if** (imgCancel != **null**) {

imgCancel.setVisibility(View.*GONE*);

}

* TopDataUnzip extends com.xing.joy.processdata.Unzip.Unzip xong thực hiện onPostExecute
  + Unzip thành công . Đánh dấu các file đã download cho hiển thị webView, cho hiển thị topLink và ẩn process download

downloadBar.setVisibility(View.*GONE*);

imgCancel.setVisibility(View.*GONE*);

setAlphaButton(1f);

setImageTopNew(**true**);

setImageTopLink();

isTouch = **true**;

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

dHistory.setStringData(listDownload.get(i).getFileName() + "isDownloaded", "OK");

}

* + Unzip không thành công.
    - Nếu file “a\_otf\_jun501pro\_bold.otf” tồn tại. Ẩn downloadBar và imgCancel

downloadBar.setVisibility(View.*GONE*);

imgCancel.setVisibility(View.*GONE*);

* + - Nếu file “a\_otf\_jun501pro\_bold.otf” không tồn tại. Ẩn downloadBar và imgCancel và hiển thị nút redownload

downloadBar.setVisibility(View.*GONE*);

imgCancel.setVisibility(View.*GONE*);

**if** (!**new** File(memory.getPathFileInternalMemory() + "/" + "a\_otf\_jun501pro\_bold.otf").exists()) {

setBacgroundResource(imgRedownload,R.drawable.*redownload\_iphone*);

isTouchRedownload = **true**;

}

### E.4. Buy Screen(extends com.xing.joy.others.BaseInappActivity)(class com.xing.joy.others.Buy)



1.Lấy thông tin

Sau khi vào màn hình 1s thì thực hiện lấy thông tin qua hàm loadData(), hàm này được gọi ở onCreate().

* Nếu không support version 3 thì hiển thị message với ID *DIALOG\_CANNOT\_CONNECT\_ID*

**if** (!isBillingSupported()) {

showDialog(*DIALOG\_CANNOT\_CONNECT\_ID*);

}

* Thực hiện parser data lấy thông tin nhờ hàm parsePackageInfoXML().content được lưu vào các biến singleV1Songs, singleV2Songs, packageSongs, singleV1\_2Songs
* Hiển thị content nhờ hàm changeModeDisplay().

Hàm changeModeDisplay lấy content từ hàm getListPackageSongs() dựa vào các thông số modeDisplay và *volDisplay. Content sẽ là* singleV1Songs hoặc singleV2Songs hoặc packageSongs hoặc singleV1\_2Songs.

**private** List<PackageSongs> getListPackageSongs() {

**switch** (modeDisplay) {

**case** 0:

**return** packageSongs;

**case** 1:

**switch** (*volDisplay*) {

**case** 0:

**return** singleV1\_2Songs;

**case** 1:

**return** singleV1Songs;

**case** 2:

**return** singleV2Songs;

}

**break**;

}

**return** packageSongs;

}

Content sẽ được đưa vào adapter và tính lại số page đang hiển thị

packagesAdapter.addData(getListPackageSongs());

totalPage = packagesAdapter.getCount();

**if** (page != **null**) {

page.setText((currentPage + 1) + "/" + totalPage + " "+ getString(R.string.*page*));

}

**if** (pager != **null**)

pager.setCurrentItem(currentPage, **false**);

2. LastUpdate.

Thực hiện lấy thông tin lastUpdate bắt đầu từ onCreate()

**new** Thread(getDateModifierPackBought).start();

Thông tin được lấy từ server nhờ “server/ vol3gles2 /checkLastUpdate.php” hoặc “server/ vol3gles2hd /checkLastUpdate.php”.

Thông tin lấy về được lưu vào listSongDateModifier; listSongDateModifier được dùng tại hàm allowUserRedownloadPackage và hàm downloadedButRedownload để check xem có data mới của một bài không.

3. Hiển icon.

Sau khi content được add vào packagesAdapter (thực hiện tại 1 packagesAdapter.addData(getListPackageSongs());) sẽ được hiển thị nhờ hàm createOnPage(**int** position, List<PackageSongs> packages).

Bằng cách lấy lần lượt bộ 6 content

**int** countPackage = 6 \* position;

**for** (**int** j = 0; j < 6; j++) {

icon được lấy qua

String path = memory.getPathFileInternalMemory() + "img\_buy/"

+ packageSongs.getIconImage();

Và được lấy thông qua

imageClientLoader.newpath(imgPackageButton, path);

Kiểm tra các điều kiện để hiển thị mờ icon

Nếu đã mua và đã download tại version cũ

isAppInstalled(memory.getPathFileExternalMemory() + packageSongs.getPackageName(), packageSongs.getPackageName())

Nếu package hoặc bài đơn có mối quan hệ đã được mua ở cả version cũ và mới

isAppRelationInstalled(packageSongs.getPackageName(), packageSongs.getPackagesRelation())

Với bài mua ở version mới hoặc bài free đã download

**boolean** isPurchased = (isPurchased(packageName) || checkFreeSongDownload(packageName));

**if** (isPurchased && !allowUserRedownloadPackage(packageName)) { imgPackageButton.startAnimation(alphaAnimation);

}

Với các bài trả phí, nhưng bài free tương ứng đã được download

IDFreePurchaseComparseProcessor comparseProcessor = **new** IDFreePurchaseComparseProcessor(**this**, packageName);

Intent intent = comparseProcessor.onProcessor();

REUSULT result = (REUSULT) intent.getSerializableExtra(IDFreePurchaseComparseProcessor.*STR\_RESULT*);

**if** (result == REUSULT.*ISPURCHARSE\_AND\_DOWNLOADEDFREE*) {

imgPackageButton.startAnimation(alphaAnimation);

}

Setting action khi click vào icon

imgPackageButton.setOnClickListener(**new** OnClickSong(packageSongs));

Khi click vào icon sẽ chuyển quan detail nhờ gọi hàm processPackagePurchased

**private** **class** OnClickSong **implements** View.OnClickListener {

**private** PackageSongs songs;

**public** OnClickSong(PackageSongs songs) {

**this**.songs = songs;

}

@Override

**public** **void** onClick(View v) {

viewClick = v;

processPackagePurchased(songs);

}

}

4. Hiển thị detail.(Khi click vào icon và gọi hàm processPackagePurchased)

Hiển thị info

showView(imgInfo, **true**);

Lưu thông tin của bài được chọn

purchasedPackage = ps.getPackageName();

*purchasedImage* = ps.getImage();

\_package = purchasedPackage;

Lấy thông tin và hiển thị ảnh detail

String path = memory.getPathFileInternalMemory() + "img\_buy/" + ps.getImageInfo();

imageClientLoader.newpath(imgInfo, path);

imgInfo.setBackgroundResource(R.drawable.*a10\_yagiyubin\_detail\_iphone1*);

Nếu bài có mối quan hệ đã được mua thì hiển thị popup tương ứng, không hiển thị các button download, mua, đã mua, đã download và ngừng xử lý khác.

**if** (!isPurchased(purchasedPackage) && isAppRelationInstalled(purchasedPackage, purchasedPackageRelation)) {

**if** (!isDebug()) {

hideBtnBuy.setVisibility(View.*INVISIBLE*);

imgOK = (ImageView) findViewById(R.id.*ok\_download*);

setBacground3Resource(imgOK, R.drawable.*buy\_ok*);

imgOK.setEnabled(**true**);

showViewWhenTime(imgOK);

}

}

**if** (!isPurchased(purchasedPackage) && isAppRelationInstalled(purchasedPackage, purchasedPackageRelation)) {

**if** (!isDebug()) {

showViewWhenTime(imgOK);

imgCancelInfo.setImageResource(0);

imgCancelInfo.setEnabled(**false**);

imgBuy.setImageResource(0);

imgBuy.setEnabled(**false**);

String message = getString((ps.getPackageNumberSongs() == 1) ? R.string.*note\_package\_is\_purchased* : R.string.*note\_single\_is\_purchased*);

CommonUtils.*popupAlert*(Buy.**this**, message);

**return** **false**;

}

}

Đối với các bài trả phí mà có bài free tương ứng đã download. Thì xử lý như bài đã mua. Nếu bài free có content mới thì hiển thị download free ngược lại hiển thị đã download và ngừng xử lý khác.

**if** (result == REUSULT.*ISPURCHARSE\_AND\_DOWNLOADEDFREE*) {

// imgOK.setVisibility(View.VISIBLE);

purchasedPackage = purchasedPackage.replace(IDFreePurchaseComparseProcessor.*PUCHARSEID*, IDFreePurchaseComparseProcessor.*FREEID*);

**if** (downloadedButRedownload(purchasedPackage)) {

setBacground3Resource(imgCancelInfo, R.drawable.*cancel\_iphone*);

imgCancelInfo.setEnabled(**true**);

imgBuy.setImageResource(0);

imgBuy.setEnabled(**false**);

showViewWhenTime(imgDownload);

setBacground3Resource(imgDownload, checkFreeSongDownload(purchasedPackage) ? R.drawable.*dl\_bt\_iphone* : R.drawable.*re\_dl*);

showViewWhenTime(imgCancelInfo);

} **else** {

showViewWhenTime(imgOK);

setBacground3Resource(imgOK, R.drawable.*dlok\_bt\_iphone*);

imgCancelInfo.setImageResource(0);

imgCancelInfo.setEnabled(**false**);

imgBuy.setImageResource(0);

imgBuy.setEnabled(**false**);

}

**return** **true**;

}

Nếu ngoài 2 trường hợp trên.

**boolean** isPurchased = (isPurchased(purchasedPackage) || checkFreeSongDownload(purchasedPackage)) || isDebug();

**boolean** isExistFile = **new** File(pat1h + ".ini").exists();

Nếu bài đã mua hoặc là bài free và có content mới thì hiển thị redownload hoặc download free(với bài free).Nếu đã mua mà chưa có download hiển thị redownload hoặc download free(với bài free).Trường hợp bài đã mua hoặc free đã có content và không có content mới thì hiển thị đã mua hoặc đã download(bài free).

**if** (isExistFile) {

**if** (downloadedButRedownload(purchasedPackage)) {

setBacground3Resource(imgCancelInfo, R.drawable.*cancel\_iphone*);

imgCancelInfo.setEnabled(**true**);

imgBuy.setImageResource(0);

imgBuy.setEnabled(**false**);

showViewWhenTime(imgDownload);

setBacground3Resource(imgDownload, checkFreeSongDownload(purchasedPackage) ? R.drawable.*dl\_bt\_iphone* : R.drawable.*re\_dl*);

showViewWhenTime(imgCancelInfo);

} **else** {

showViewWhenTime(imgOK);

setBacground3Resource(imgOK, checkFreeSongDownload(purchasedPackage) ? R.drawable.*dlok\_bt\_iphone* : R.drawable.*buy\_ok*);

imgCancelInfo.setImageResource(0);

imgCancelInfo.setEnabled(**false**);

imgBuy.setImageResource(0);

imgBuy.setEnabled(**false**);

}

} **else** {

setBacground3Resource(imgCancelInfo, R.drawable.*cancel\_iphone*);

imgCancelInfo.setEnabled(**true**);

imgBuy.setImageResource(0);

imgBuy.setEnabled(**false**);

showViewWhenTime(imgDownload);

setBacground3Resource(imgDownload, checkFreeSongDownload(purchasedPackage) ? R.drawable.*dl\_bt\_iphone* : R.drawable.*re\_dl*);

showViewWhenTime(imgCancelInfo);

}

Đối với bài chưa mua. Thực hiện lấy thông tin giá thông qua.

**new** GetPackagePrice().execute();

Và kiểm tra sdCard và khả năng support inapp v3

**if** (!isSDCardExisted || !isBillingSupported()) {

imgBuy.setAlpha(50);

imgBuy.setEnabled(**false**);

}

\* Giá của bài được lấy qua link “server/ vol3general/getprice.php?package\_id="

Kiểm tra có support v3 không thông qua hàm isBillingSupported()

**public** **boolean** isBillingSupported() {

**try** {

**return** mService.isBillingSupported(3, getPackageName(), "inapp") == 0;

} **catch** (Exception e) {

**return** **false**;

}

}

\*Kiểm tra một bài có content mới hay không nhờ hàm downloadedButRedownload. So sánh thời gian thay đổi của file từ listSongDateModifier và thời gian lưu trong máy(lưu tại thời điểm download package) dlHistory.getStringData(purchasedPackage + ".zip");

**public** **boolean** downloadedButRedownload(String purchasedPackage) {

**try** {

String timeHis = dlHistory.getStringData(purchasedPackage + ".zip");

String timeMod = **null**;

**if** (listSongDateModifier.containsKey(purchasedPackage)) {

timeMod = listSongDateModifier.get(purchasedPackage);

}

**if** (timeMod != **null** && !timeHis.equalsIgnoreCase(timeMod)) {

**return** **true**;

}

} **catch** (Exception e) {

}

**return** **false**;

}

\*Kiểm tra một bài đã mua chưa thông qua hàm isPurchased

**protected** **boolean** isPurchased(String sku) {

**return** queryingForPurchasedItems(sku) || isPurcharseOld(sku);

}

Hàm isPurchaseOld lấy thông tin một bài được mua từ version cũ.

**protected** **boolean** isPurcharseOld(String pack) {

**return** Boolean.*parseBoolean*(prefs.getString(pack, getResources().getBoolean(R.bool.*debug\_version*) + ";0").split(";")[0]);

}

Hàm queryingForPurchasedItems lấy thông tin một bài được mua với inappv3 và cả inapp v2.

**protected** **boolean** queryingForPurchasedItems(String sku1) {

Bundle ownedItems;

**try** {

ownedItems = mService.getPurchases(3, getPackageName(), "inapp", **null**);

**int** response = ownedItems.getInt("RESPONSE\_CODE");

**if** (response == 0) {

ArrayList ownedSkus = ownedItems.getStringArrayList("INAPP\_PURCHASE\_ITEM\_LIST");

ArrayList purchaseDataList = ownedItems.getStringArrayList("INAPP\_PURCHASE\_DATA\_LIST");

ArrayList signatureList = ownedItems.getStringArrayList("INAPP\_DATA\_SIGNATURE");

String continuationToken = ownedItems.getString("INAPP\_CONTINUATION\_TOKEN");

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

String sku = ownedSkus.get(i).toString();

**if** (sku.equals(sku1)) {

**return** **true**;

}

}

}

} **catch** (Exception e) {

}

**return** **false**;

}

\*Hàm checkFreeSongDownload kiểm tra 1 bài là free download hay không

**public** **boolean** checkFreeSongDownload(String packageName) {

**if** (packageName.contains("free.song.relation")) {

**return** **true**;

}

**return** **false**;

}

5. Purchase.

Khi click vào button mua bài hát thì sẽ gọi hàm **public** **void** onClick(View view) { và kiểm tra điều kiện **if** (view == imgBuy)

Và gọi đến hàm launchPurchaseFlow(purchasedPackage); để thực hiện việc mua bán với google Play.

Thiết lập hiển thị cho các item cần thiết

imgBuy.setEnabled(**false**);

imgBuy.setAlpha(50);

imgCancelInfo.setEnabled(**false**);

imgCancelInfo.setAlpha(50);

mProgressCloseX.setEnabled(**false**);

mProgressCloseX.setAlpha(50);

màn hình thanh toán hiển thị cho phép người dùng thực hiện việc mua bán. Khi mua bán thành công hoặc không thành công, hoặc kết thúc việc mua bán thì gọi vào hàm onIabPurchaseFinish.

Hàm onIabPurchaseFinish

* Nếu việc mua bán không thành công(người dùng dừng việc mua bán, mua bán không thực hiện được, phát sinh lỗi). thực hiện hiển thị lại màn hình.

setEnableAndAlPha(**true**, 255, imgCancelInfo);

setEnableAndAlPha(**true**, 255, mProgressCloseX);

setEnableAndAlPha(**true**, 255, imgBuy);

* Khi mua bán thành công.Hiển thị lại màn hình và thực hiện download ngay gói bài hát.

setEnableAndAlPha(**false**, 100, imgCancelInfo);

setEnableAndAlPha(**false**, 50, mProgressCloseX);

setBacground3Resource(imgBuy, 0);

dlPackage = (DownloadSongPackage) **new** DownloadSongPackage(**this**).execute();

setEnableAndAlPha(**true**, 255, imgBuy);

isTouch = **true**;

6. Download content và unZip.( DownloadSongPackage và SongPackageUnzip)

Khi mua bài hát xong, hoặc khi click vào button redownload hoặc download free sẽ gọi đến DownloadSongPackage để download content về.

* DownloadSongPackage **extends** Download
  + Tại doInBackground config đường dẫn file và thực hiện downlaod

String pathFile = **new** UrlUtils(Buy.**this**).getUrl(R.string.*url\_package\_download*);

**if** (isTablet()) {

pathFile = **new** UrlUtils(Buy.**this**).getUrl(R.string.*url\_package\_hd\_download*);

}

* Tại onPostExecute, Nếu download thành công thực hiện unzip và thực hiện log data lên server.

spUnzip = (SongPackageUnzip) **new** SongPackageUnzip().execute(fm);

Thông tin đẩy lên server qua link

String url = **new** UrlUtils(Buy.**this**).getUrl(R.string.*url\_top\_free\_download*) + "download.php";

* Tại onPostExecute, nếu download không thành công thì hiển thị lại màn hình. Nhờ gọi lại action của cancelDownload.

cancelDownload.onTouch(**null**, **null**);

* SongPackageUnzip **extends** Unzip
  + Tại onPostExecute kiểm tra giải nén thành công thì hiển thị lại màn hình, lưu lại thông tin của bài vừa mua.

**if** (result != **null** && result && imgOK != **null**) {

imgBuy.setAlpha(255);

imgCancelInfo.setAlpha(255);

downloadBar.setVisibility(View.*GONE*);

imgCancel.setVisibility(View.*GONE*);

downloadBar.reset();

imgBuy.setImageResource(0);

imgCancelInfo.setVisibility(View.*GONE*);

imgDownload.setVisibility(View.*GONE*);

Toast.*makeText*(Buy.**this**, getString(R.string.*dlp\_dl\_completed*), Toast.*LENGTH\_LONG*).show();

ReportHistory report = **new** ReportHistory(Buy.**this**);

report.resetReportCountByPackage(purchasedPackage);

// **FIXME**

**if** (checkFreeSongDownload(purchasedPackage)) {

SharedPreferences.Editor edit = prefs.edit();

edit.putString(purchasedPackage, "true;" + System.*currentTimeMillis*());

edit.commit();

}

String timeMode = listSongDateModifier.get(purchasedPackage);

**if** (timeMode != **null** && !timeMode.trim().equals("")) {

dlHistory.setStringData(purchasedPackage + ".zip", timeMode);

}

imgOK.postDelayed(**new** Runnable() {

@Override

**public** **void** run() {

**if** (imgOK != **null**) {

imgOK.setVisibility(View.*VISIBLE*);

**if** (checkFreeSongDownload(purchasedPackage)) {

setBacground3Resource(imgOK, R.drawable.*dlok\_bt\_iphone*);

} **else** {

setBacground3Resource(imgOK, R.drawable.*buy\_ok*);

}

isTouch = **true**;

mProgressCloseX.setAlpha(255);

mProgressCloseX.setEnabled(**true**);

}

}

}, 3000);

**if** (*purchasedImage*.getBackground() != **null**)

*purchasedImage*.getBackground().setAlpha(100);

* + Tại onPostExecute nếu giải nén thất bại, không thực hiện.