

PAPERBOARD

by Jahir Fiquitiva

Contents:

- Info about the Dashboard.
- Info about the License.
- Requeriments.
- Getting started.
- Files to be edited.

Info about Dashboard:

Features:

- Material Design inspired/based dashboard.
- Cloud based (only) wallpapers.
- Wallpapers can be applied, cropped-and-applied, or downloaded.
- In-App Icon Request tool.
- Previews section, where user could see themed icons.
- License Checker.
- Changelog shown with every update.
- Apply section with 16 supported launchers.
- Credits section.
- Requires API15+ or Android 4.0.3+

Future features (No ETA):

- Clickable icons in Previews section, showing icon and app name in a dialog.
- Replace GridView and ListView with RecyclerView to improve performance.
- Docks support.
- Muzei Support.
- Themeable UI.
- Option to delete cache directly inside the app.
- Option to choose downloads folder (for wallpapers).
- In-App-Purchases for Premium Icon Requests.

Info about the License:

This Dashboard is licensed under CreativeCommons-Attribution-ShareAlike license. **What does that means?** That means you can use and modify the provided source however and wherever you want, **but** you must mention always its original creator (me, Jahir Fiquitiva). And you can share your source keeping the same license.

By the way, you can contribute to the development of this dashboard by helping improve its current features and/or helping to add the future features, by simply doing some commits to GitHub repository. I will provide credits, obviously.

Requirement to use: Keep Credits section as it is. That's it.

Requirements:

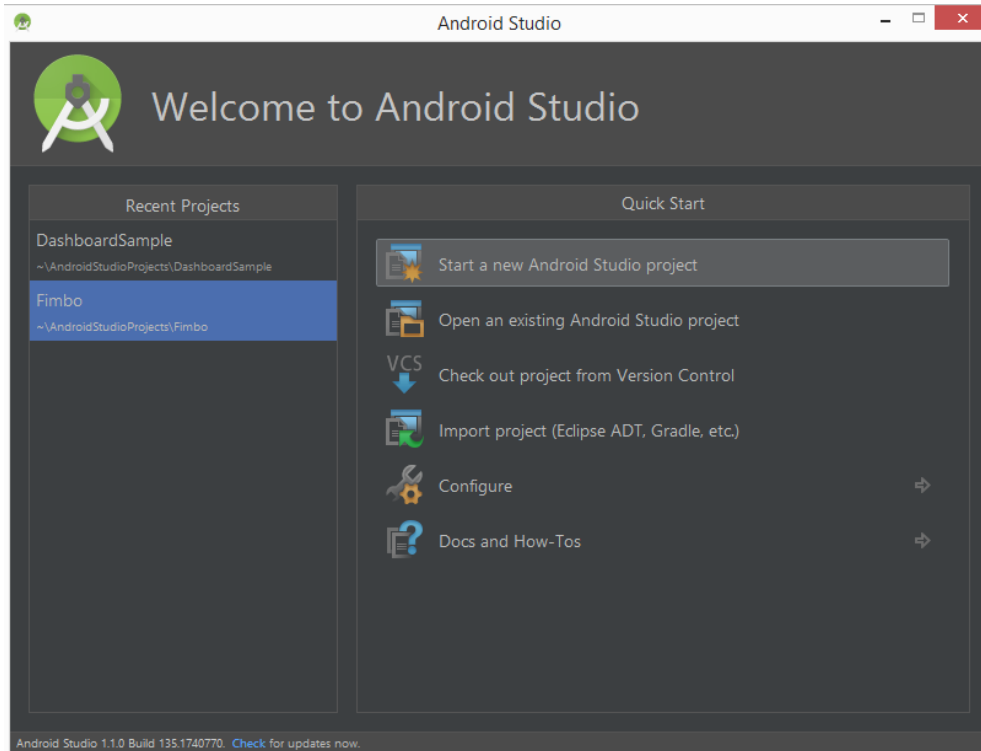
- Latest version of Android Studio and it's requirements
<http://developer.android.com/sdk/index.html>
- Latest Java SE Development Kit.
- Android 5.0 and 5.1 Lollipop SDK.
- Latest version of Android Support Library
- And a text editor. I recommend Notepad++

Packages			
Name	API	Rev.	Status
Tools			
Android SDK Tools		24.1.2	Installed
Android SDK Platform-tools		22	Installed
Android SDK Build-tools		22	Installed
Android SDK Build-tools		21.1.2	Installed
Android SDK Build-tools		20	Not installed
Android SDK Build-tools		19.1	Not installed
Android 5.1 (API 22)			
Documentation for Android SDK	22	1	Installed
SDK Platform	22	1	Installed
Samples for SDK	22	5	Installed
Android TV ARM EABI v7a System Image	22	1	Not installed
Android TV Intel x86 Atom System Image	22	1	Not installed
ARM EABI v7a System Image	22	1	Not installed
Intel x86 Atom_64 System Image	22	1	Not installed
Intel x86 Atom System Image	22	1	Not installed
Google APIs	22	1	Installed
Google APIs ARM EABI v7a System Image	22	1	Not installed
Google APIs Intel x86 Atom_64 System Image	22	1	Not installed
Google APIs Intel x86 Atom System Image	22	1	Not installed
Sources for Android SDK	22	1	Installed
Android 5.0.1 (API 21)			
SDK Platform	21	2	Installed
Samples for SDK	21	4	Installed
Android TV ARM EABI v7a System Image	21	1	Not installed
Android TV Intel x86 Atom System Image	21	1	Not installed
Android 4.4W.2 (API 20)			
Android 4.4.2 (API 19)			
Android 4.3.1 (API 18)			
Android 4.2.2 (API 17)			
Android 4.1.2 (API 16)			
Android 4.0.3 (API 15)			
Android 2.3.3 (API 10)			
Android 2.2 (API 8)			
Extras			
Android Support Repository		12	Installed
Android Support Library		22	Installed
Google Play services		22	Installed
Google Repository		15	Installed
Google Play APK Expansion Library		3	Installed
Google Play Billing Library		5	Installed
Google Play Licensing Library		2	Installed
Android Auto API Simulators		1	Installed
Google USB Driver		11	Installed
Google Web Driver		2	Installed
Intel x86 Emulator Accelerator (HAXM installer)		5.2	Installed

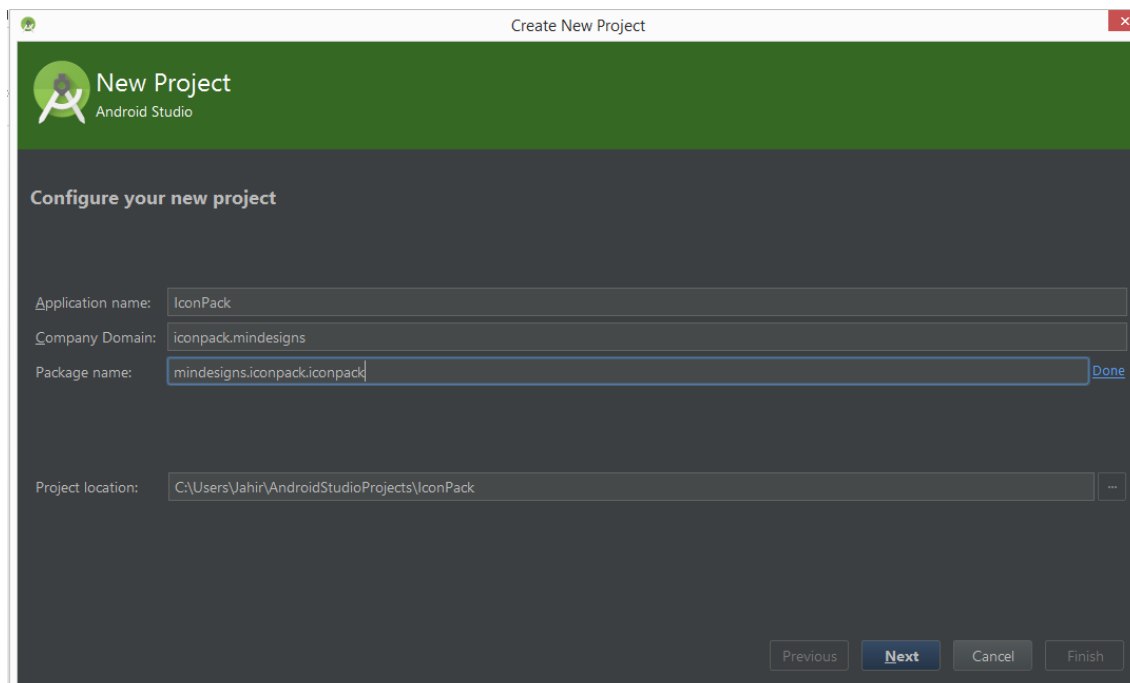
Packages			
Name	API	Rev.	Status
Intel x86 Atom System Image	21	1	Not installed
Google APIs	21	1	Installed
Google APIs ARM EABI v7a System Image	21	3	Not installed
Google APIs Intel x86 Atom_64 System Image	21	3	Not installed
Google APIs Intel x86 Atom System Image	21	3	Installed
Sources for Android SDK	21	1	Installed
Android 4.4W.2 (API 20)			
Android 4.4.2 (API 19)			
Android 4.3.1 (API 18)			
Android 4.2.2 (API 17)			
Android 4.1.2 (API 16)			
Android 4.0.3 (API 15)			
Android 2.3.3 (API 10)			
Android 2.2 (API 8)			
Extras			
Android Support Repository		12	Installed
Android Support Library		22	Installed
Google Play services		22	Installed
Google Repository		15	Installed
Google Play APK Expansion Library		3	Installed
Google Play Billing Library		5	Installed
Google Play Licensing Library		2	Installed
Android Auto API Simulators		1	Installed
Google USB Driver		11	Installed
Google Web Driver		2	Installed
Intel x86 Emulator Accelerator (HAXM installer)		5.2	Installed

Getting Started:

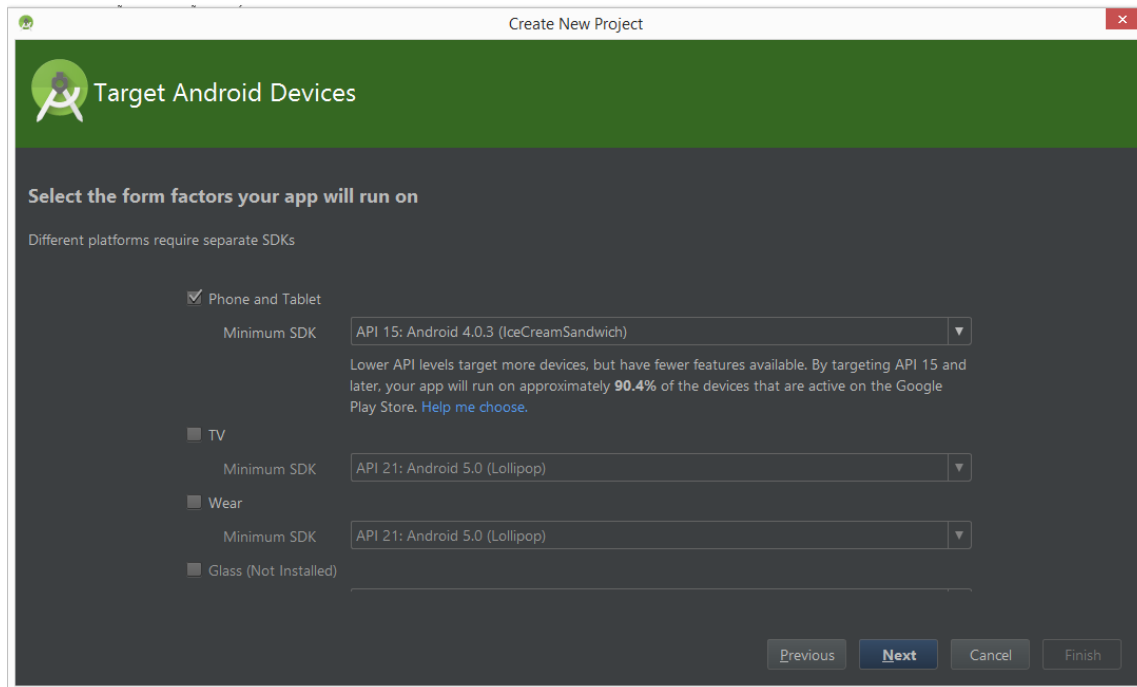
- Open Android Studio and create a new Project



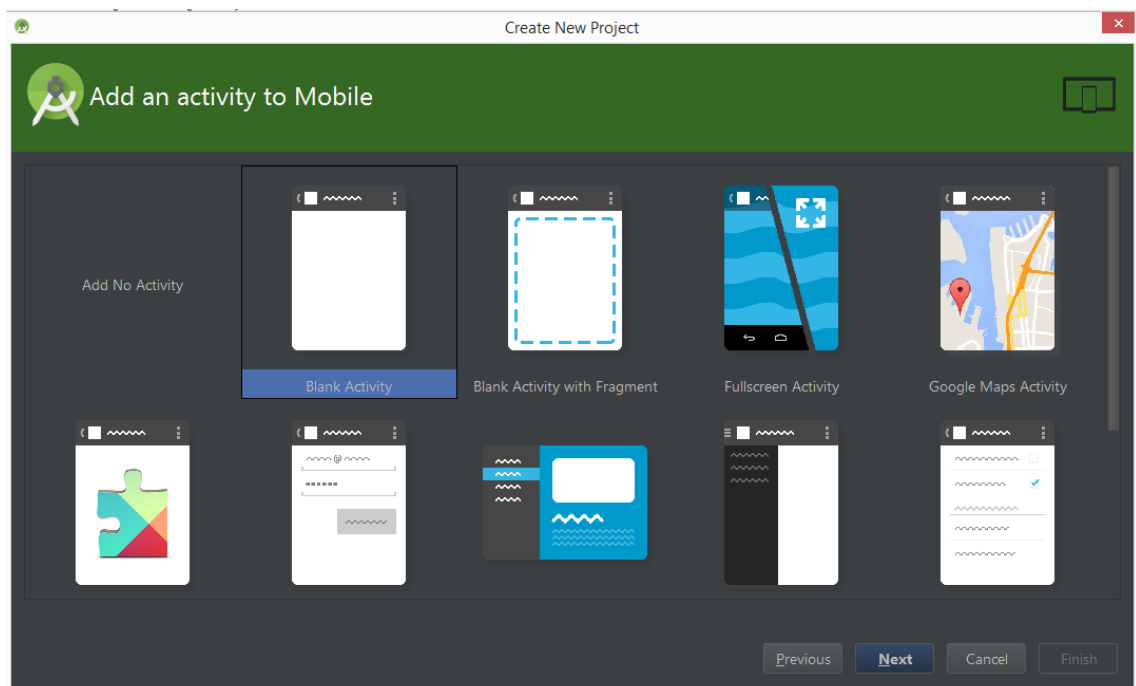
- Name it however you want

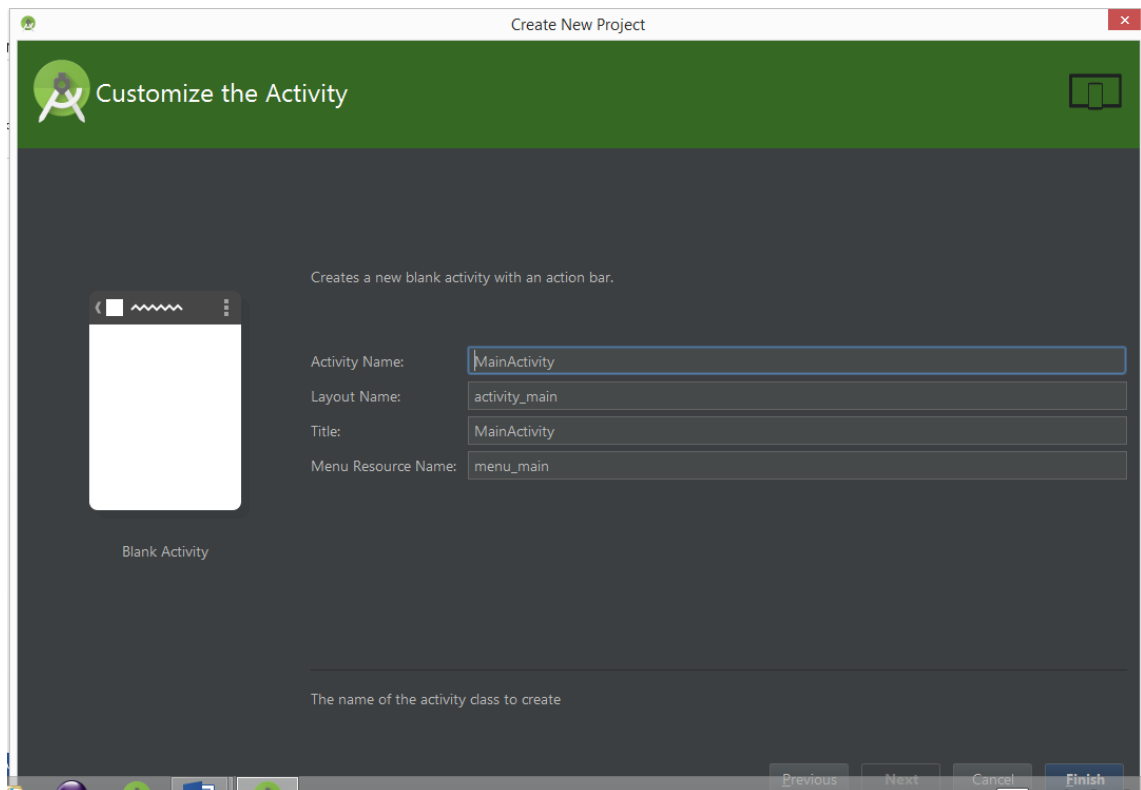


- Choose Minimum SDK to API15



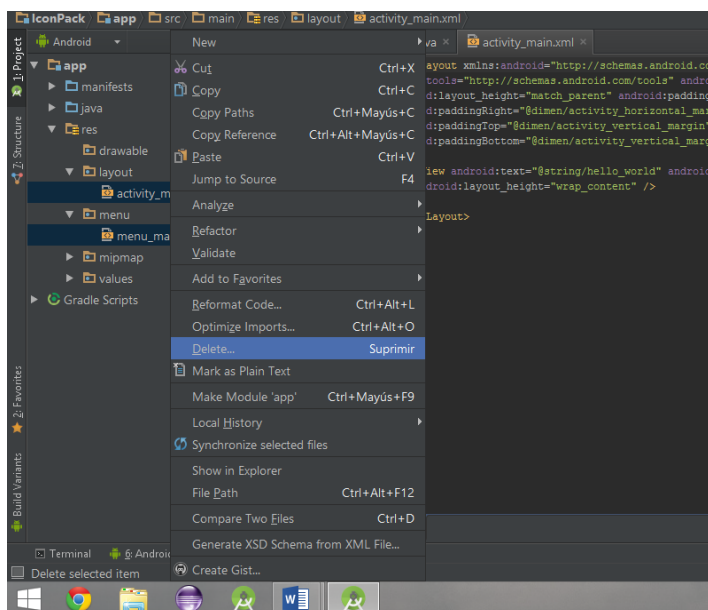
- Create a new blank activity with any name you want. Just **don't** use Main.





- Hit Finish
- Locate these files:
res/layout/activity_main
res/menu/menu_main

And delete them.



- Open you MainActivity.java file and delete the

```
onCreateOptionsMenu()  
onOptionsItemSelected()
```

and in onCreate void and delete the line

```
setContentView(R.layout.activity_main);
```

at the end it should look like this:

```
1 package com.jahirfiquitiva.dashboardsample;  
2  
3 import android.content.Intent;  
4 import android.os.Bundle;  
5 import android.support.v7.app.AppCompatActivity;  
6  
7 public class Home extends AppCompatActivity {  
8  
9     @Override  
10    protected void onCreate(Bundle savedInstanceState) {  
11        super.onCreate(savedInstanceState);  
12    }  
13  
14 }  
15  
16  
17  
18
```

- Now open your build.gradle files and add:
 - mavenCentral() to repositories
 - and these lines to dependencies{} :


```
compile fileTree(dir: 'libs', include: ['*.jar'])  
compile 'com.android.support:appcompat-v7:22.0.0'  
compile 'com.android.support:support-v4:22.0.0'  
compile 'com.android.support:cardview-v7:22.0.0'  
  
//Material Dialogs ftw  
compile 'com.afollestad:material-dialogs:0.6.4.7'  
  
//Material Drawer ftw  
compile('com.mikepenz:materialdrawer:library:2.5.5@aar') {  
    transitive = true  
}  
  
//Needed for wallpapers section  
compile 'com.squareup.picasso:picasso:2.5.0'  
compile 'com.squareup.okhttp:okhttp:2.2.0'
```

```
compile 'com.squareup.okhttp:okhttp-urlconnection:2.2.0'
```

```
//Muzei Support
```

```
compile 'com.google.android.apps.muzei:muzei-api:+'
```

```
//FloatingActionButtons
```

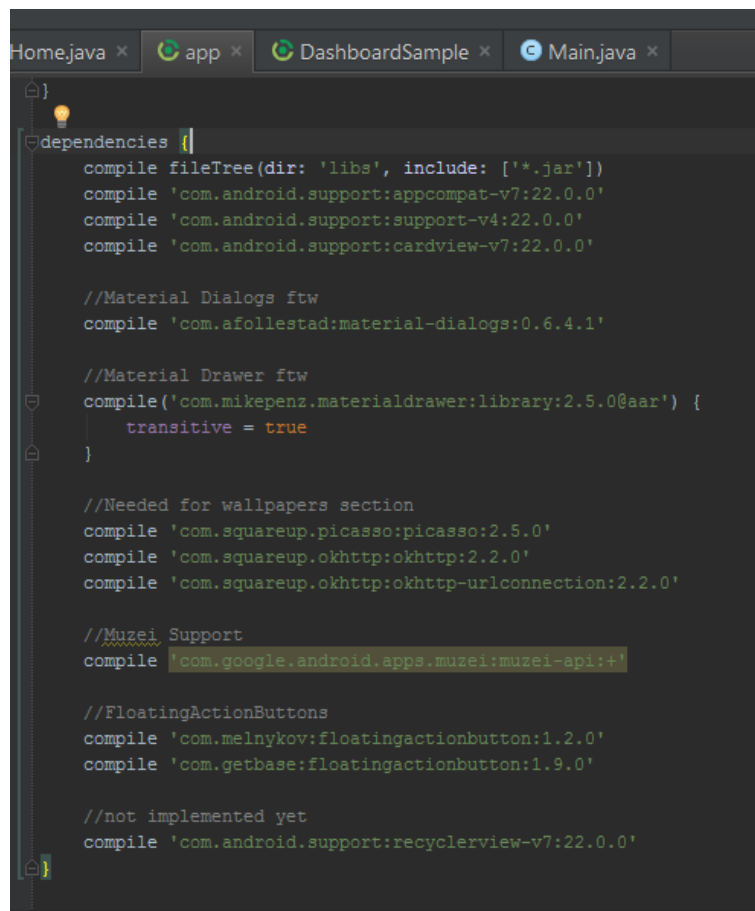
```
compile 'com.melnykov:floatingactionbutton:1.2.0'
```

```
compile 'com.getbase:floatingactionbutton:1.9.0'
```

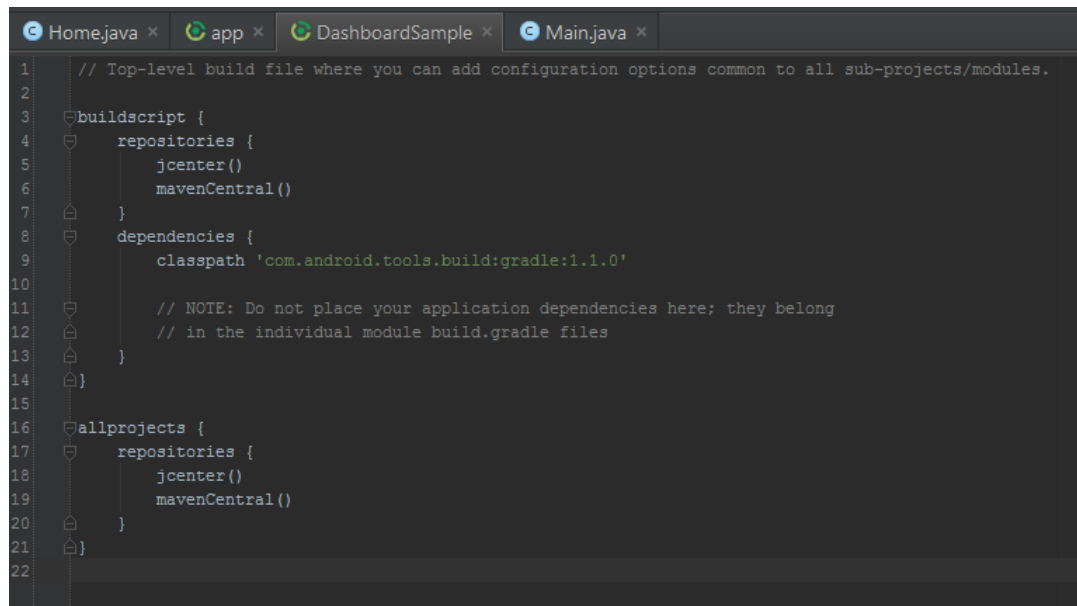
```
//not implemented yet
```

```
compile 'com.android.support:recyclerview-v7:22.0.0'
```

At the end they should look like these:

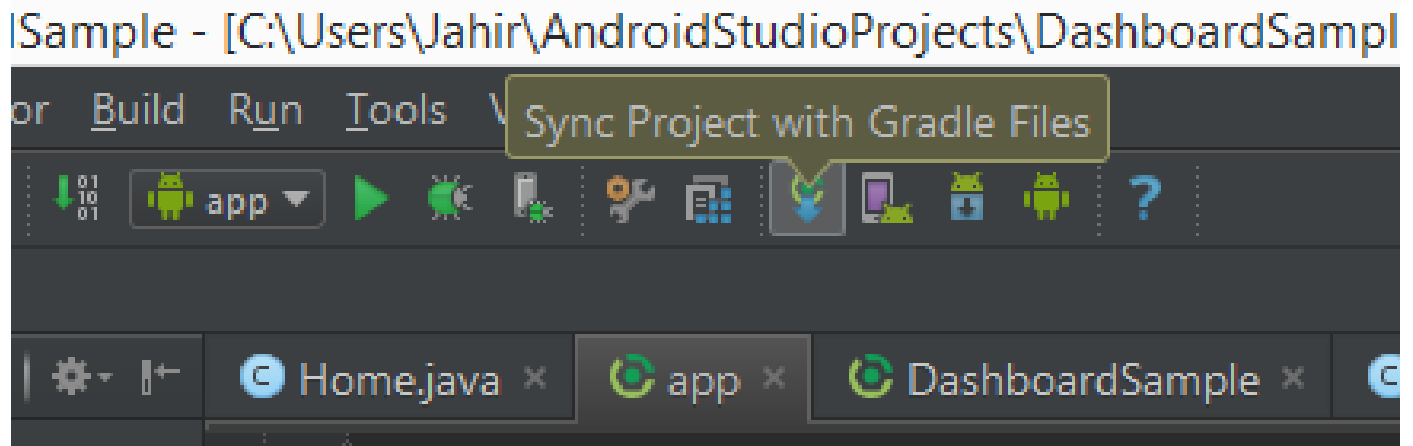
A screenshot of an IDE window showing a build file with a 'dependencies' section. The window has tabs for 'Home.java', 'app', 'DashboardSample', and 'Main.java'. The 'dependencies' section is expanded, showing a list of compile dependencies. The code is as follows:

```
}  
  
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    compile 'com.android.support:appcompat-v7:22.0.0'  
    compile 'com.android.support:support-v4:22.0.0'  
    compile 'com.android.support:cardview-v7:22.0.0'  
  
    //Material Dialogs ftw  
    compile 'com.afollestad:material-dialogs:0.6.4.1'  
  
    //Material Drawer ftw  
    compile('com.mikepenz:materialdrawer:library:2.5.0@aar') {  
        transitive = true  
    }  
  
    //Needed for wallpapers section  
    compile 'com.squareup.picasso:picasso:2.5.0'  
    compile 'com.squareup.okhttp:okhttp:2.2.0'  
    compile 'com.squareup.okhttp:okhttp-urlconnection:2.2.0'  
  
    //Muzei Support  
    compile 'com.google.android.apps.muzei:muzei-api:+'  
  
    //FloatingActionButtons  
    compile 'com.melnykov:floatingactionbutton:1.2.0'  
    compile 'com.getbase:floatingactionbutton:1.9.0'  
  
    //not implemented yet  
    compile 'com.android.support:recyclerview-v7:22.0.0'  
}
```

A screenshot of an IDE window with four tabs: 'Home.java', 'app', 'DashboardSample', and 'Main.java'. The 'app' tab is active, displaying a Gradle build script. The script is a top-level build file with line numbers 1 through 22 on the left. It contains a 'buildscript' block with repositories (jcenter, mavenCentral) and a dependency on 'com.android.tools.build:gradle:1.1.0'. It also includes a comment about not placing application dependencies here. Finally, it has an 'allprojects' block with the same repositories. The code is as follows:

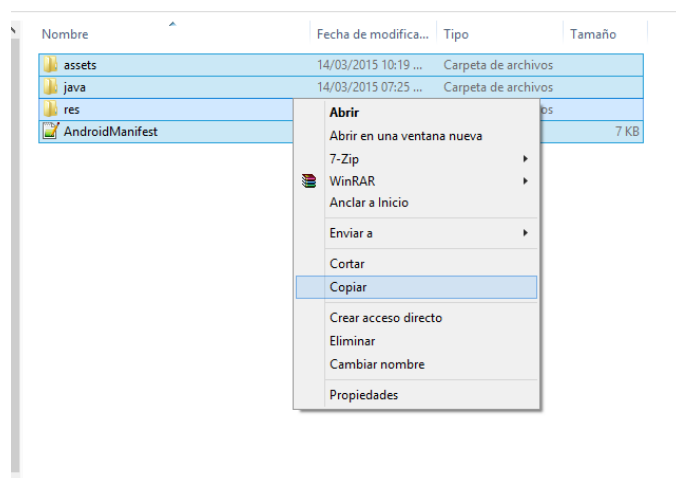
```
1 // Top-level build file where you can add configuration options common to all sub-projects/modules.
2
3 buildscript {
4     repositories {
5         jcenter()
6         mavenCentral()
7     }
8     dependencies {
9         classpath 'com.android.tools.build:gradle:1.1.0'
10
11         // NOTE: Do not place your application dependencies here; they belong
12         // in the individual module build.gradle files
13     }
14 }
15
16 allprojects {
17     repositories {
18         jcenter()
19         mavenCentral()
20     }
21 }
22
```

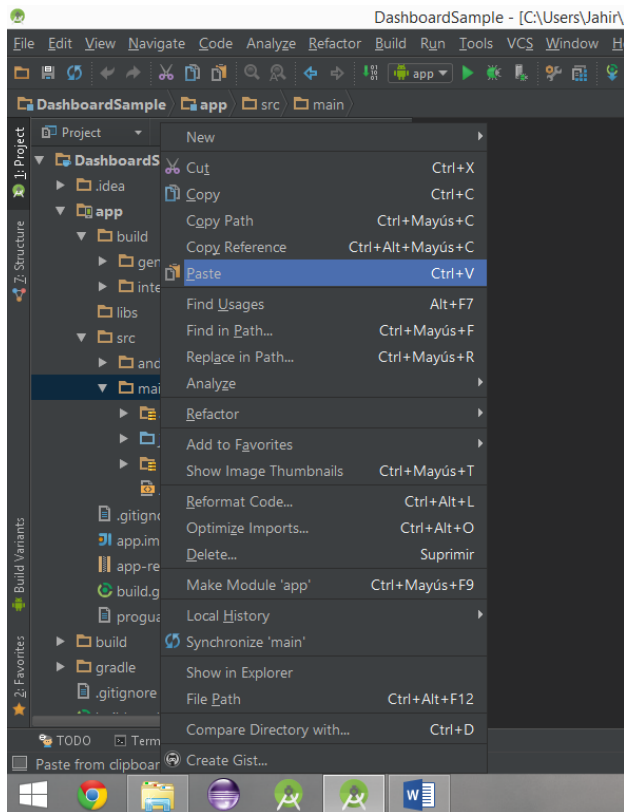

- Click “Sync project with gradle files” button at the top of the window.



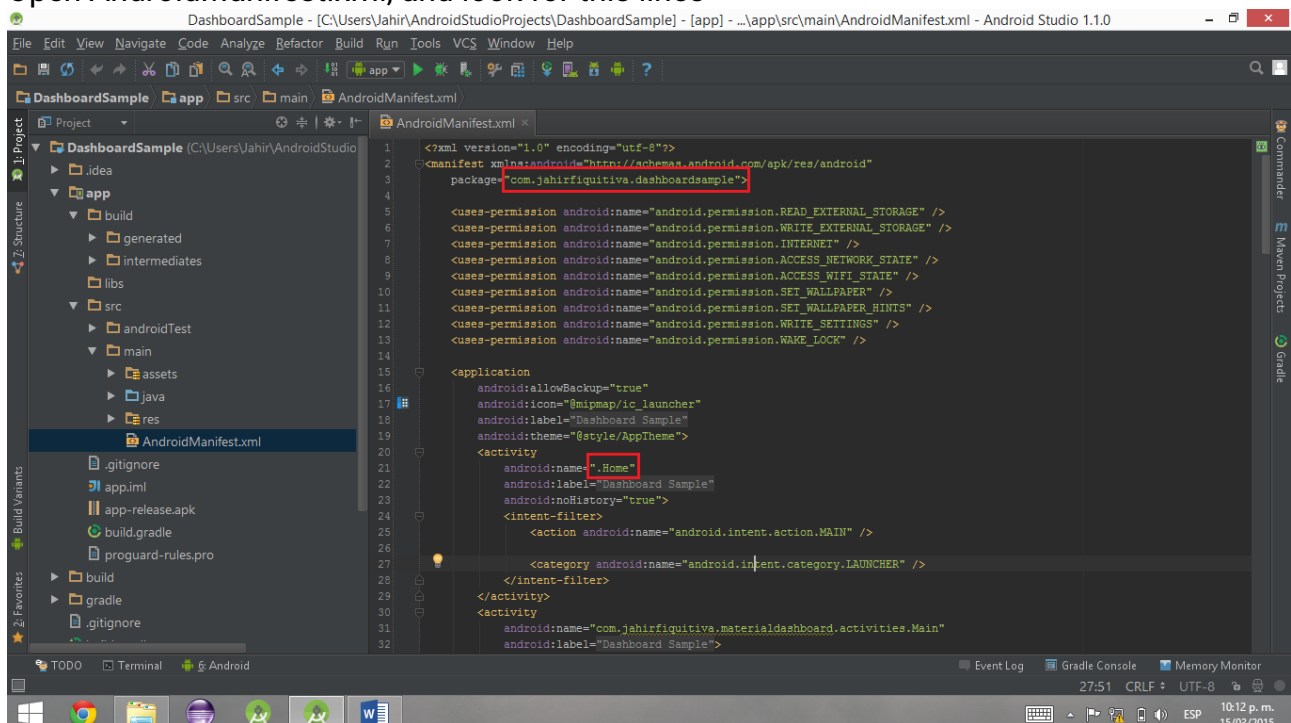
And wait for it to download all the libraries that we need to make this dashboard work properly. You won't be able to run anything yet, but you will be able to add the other files without getting errors.

- Once it gets built. Copy files from source/main folder into your app's main folder.





- Open AndroidManifest.xml, and look for this lines



Replace

com.jahirfiquitiva.dashboardsample
with your package name i.e.
com.package.name

And change
.Home

With the name of the activity you created at the beginning i.e.
.MainActivity

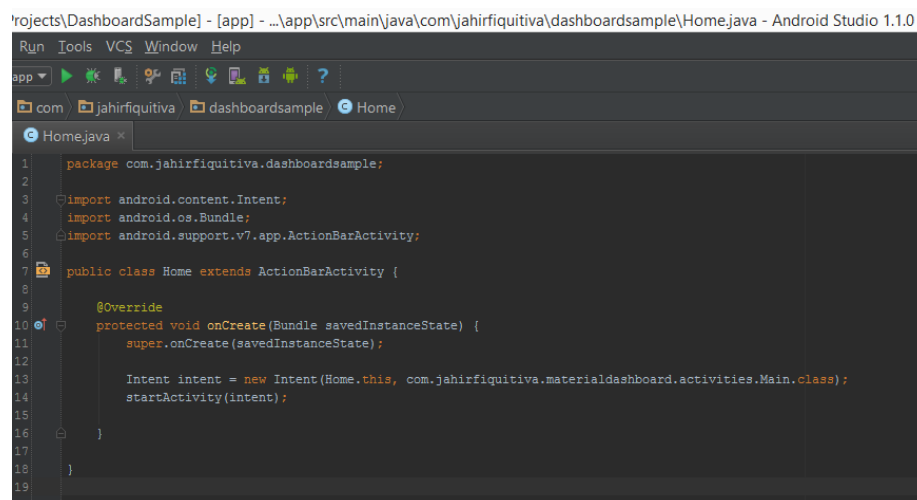
- Now open the java file for the activity you create at the beginning i.e. MainActivity.java

And add these lines to onCreate void.

```
Intent intent = new Intent(Home.this, com.jahirfiquitiva.paperboard.activities.Main.class);  
startActivity(intent);
```

Replace Home.this with the name of your activity, i.e. MainActivity.this

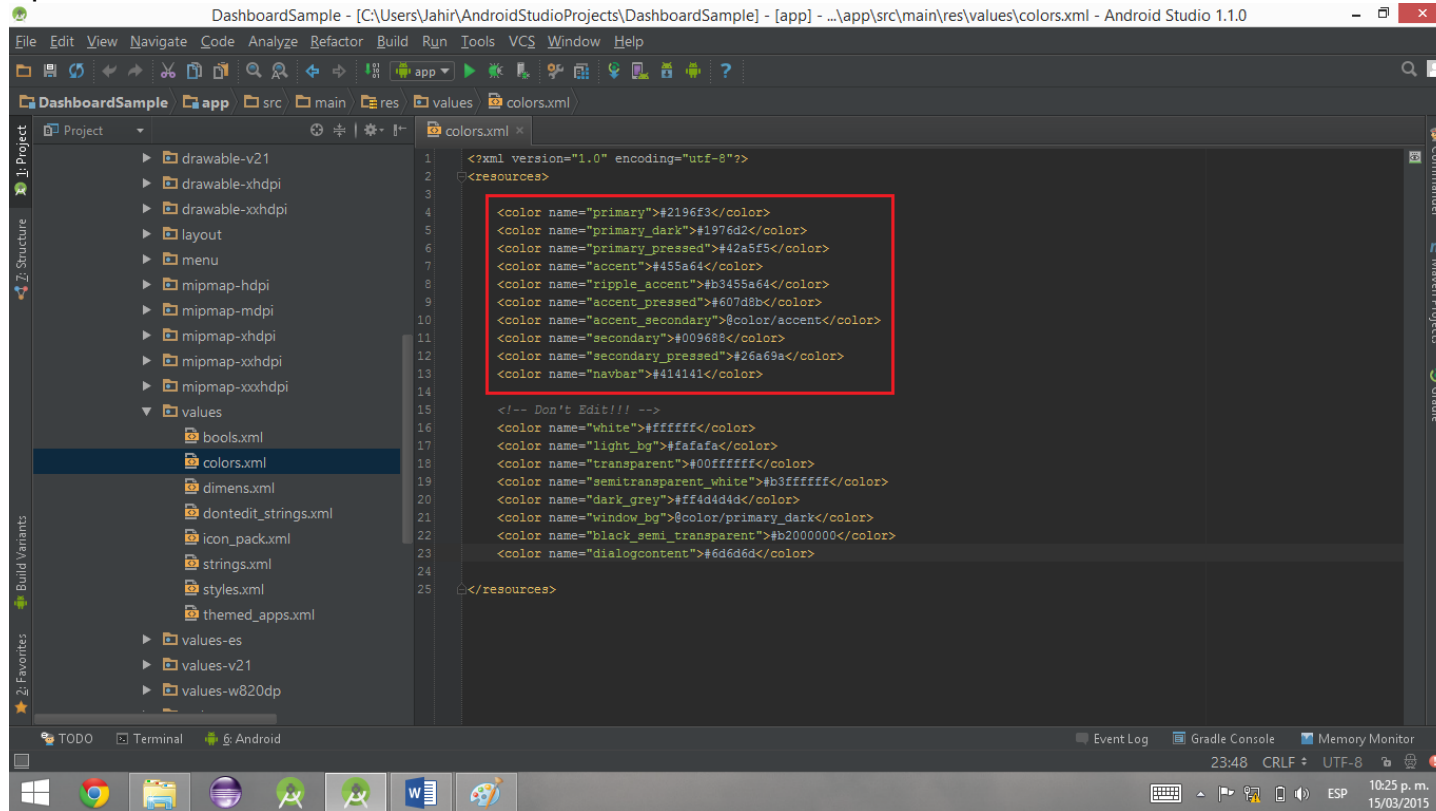
At the end should look like this:



- Now open all the java files inside com.jahirfiquitiva.paperboard. In the import section, look for a line that says:
import com.jahirfiquitiva.dashboardsample.R;

replace it with:
import com.yourpackage.name.R;

- And that's it. Project is correctly set and you can run it in your device already.
Next step is to edit the files to match the look you want in your app, and match your icon pack.
Please, edit **only** the files mentioned here.
- Let's start with colors file. Look for colors.xml file at res/values folder
Open it and edit these colors:



Primary goes to ActionBar color

Primary_Dark goes to status bar color (API 21+)

Primary_Pressed must be a lighter color from primary.

Accent color is used in dialog buttons, cards titles and fabs background.

Ripple_Accent is the same as Accent just set with a hex transparency of #b3

Accent_Pressed must be a lighter color from accent.

Secondary is a secondary color, different from primary and accent, which will be used only in the fab menu.

Secondary_Pressed must be a lighter color from secondary.

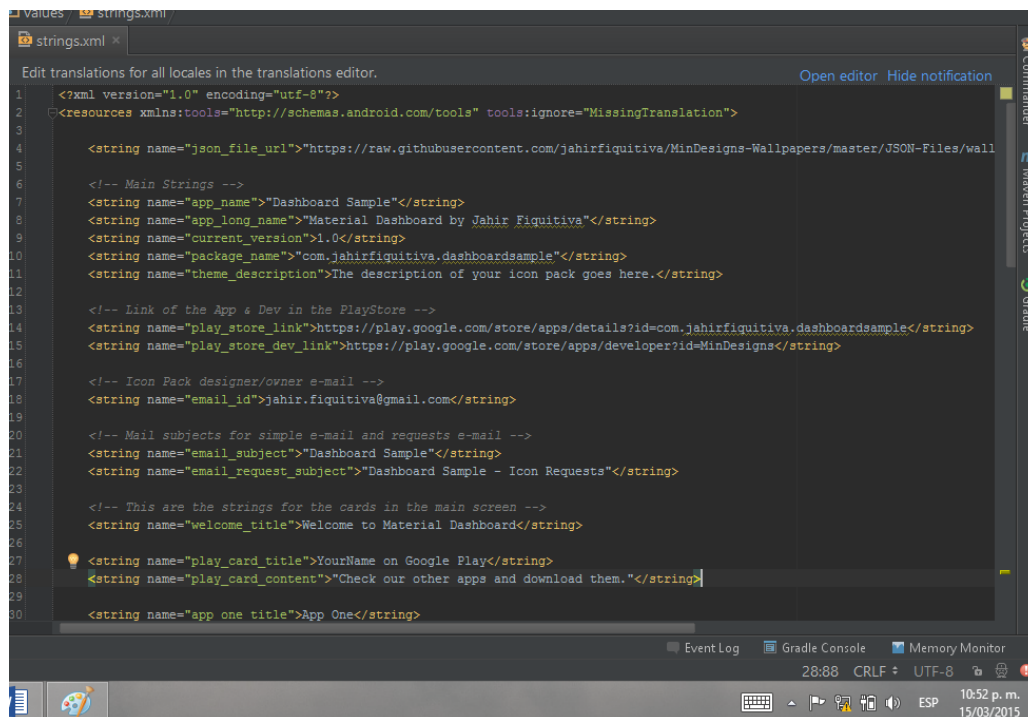
NavBar color is the one for the NavigationBar (only in API 21+)

Done

- Let's go with strings. Strings are texts shown around the app. Every string is used somewhere so you must keep them all. There's a file with many strings that shouldn't be edited, its name is dontedit_strings.xml, I added translation to Spanish for those strings.

The file you must edit is strings.xml that file contains mostly your info and the info related to you.

Each string name identifies what it is used for, though I will explain each one here.



json_file_url is the url of your json file where the info about your wallpapers will be stored.

App_name is the name of the app. And the text shown in launcher.

App_long_name is the text shown in drawer. I.e. if your icon pack name is “Sky”, then the app_long_name should be “Sky – Icon Pack”.

If you don't like it, simply replace the text with @string/app_name and it will show the short name.

Current_version is the current version of your icon pack, it is used to the changelog stuff, and it must match the version shown in build.gradle file.

Package_name is the same text from your app package

Theme_description is a brief description of your icon pack and it will be shown in the first card in home section and in some launchers.

Play_store_link is the link to your app in the PlayStore

Play_store_dev_link is the link to your developer account in PlayStore.

Email_id is your e-mail, where people can contact you and send the icon requests.

Email_subject is the default subject for the emails when people want just to contact you.

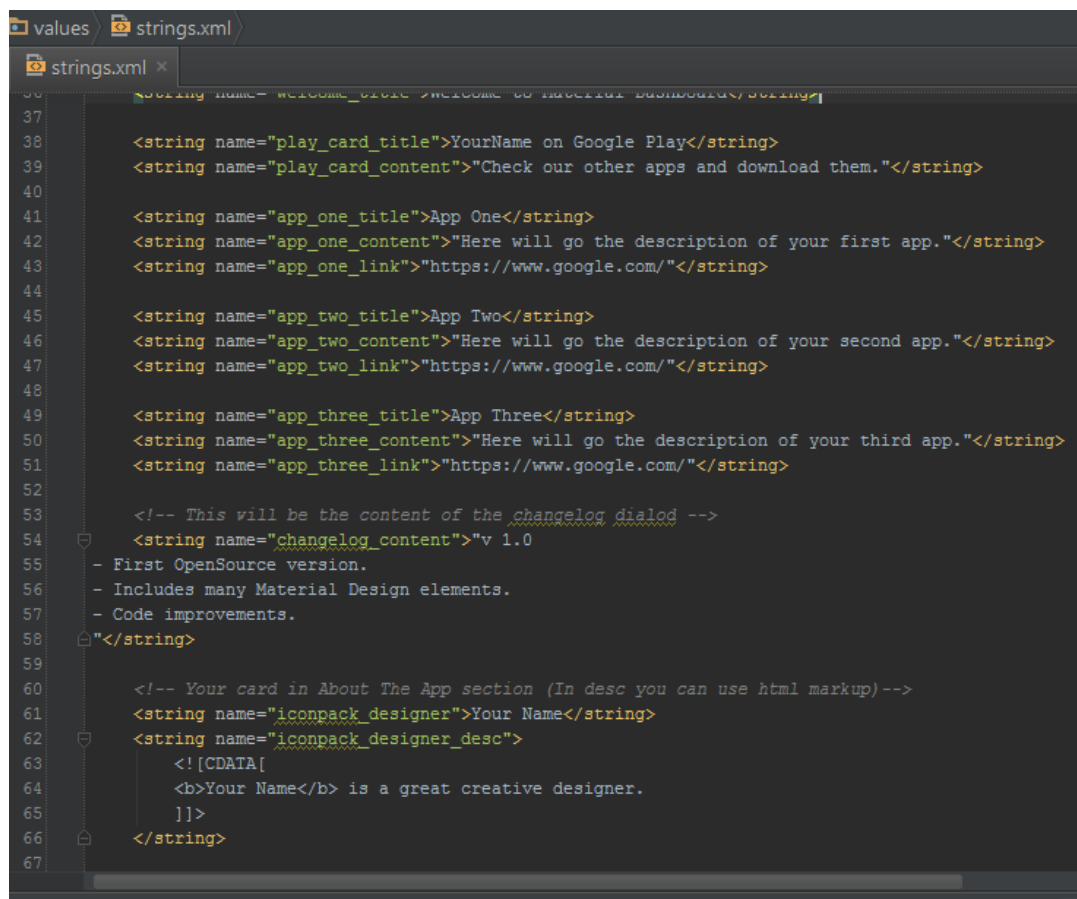
Email_request_subject is the default subject for the emails when people want to send you an icon request.

Welcome_title is the text shown as title in the first card in home section. It's like a welcome greeting for your users.

Play_card_title is the title of a card that when clicked, goes to your dev account in PlayStore and shows all your apps.

Play_card_content is the content of that card.

App_one_title is the title of a card that shows info about one of your apps. The title could be that app's name.



```
36 <string name="welcome_title">Welcome to Material Dashboard</string>
37
38 <string name="play_card_title">YourName on Google Play</string>
39 <string name="play_card_content">"Check our other apps and download them."</string>
40
41 <string name="app_one_title">App One</string>
42 <string name="app_one_content">"Here will go the description of your first app."</string>
43 <string name="app_one_link">"https://www.google.com/"</string>
44
45 <string name="app_two_title">App Two</string>
46 <string name="app_two_content">"Here will go the description of your second app."</string>
47 <string name="app_two_link">"https://www.google.com/"</string>
48
49 <string name="app_three_title">App Three</string>
50 <string name="app_three_content">"Here will go the description of your third app."</string>
51 <string name="app_three_link">"https://www.google.com/"</string>
52
53 <!-- This will be the content of the changelog dialog -->
54 <string name="changelog_content">"v 1.0
55 - First OpenSource version.
56 - Includes many Material Design elements.
57 - Code improvements.
58 "</string>
59
60 <!-- Your card in About The App section (In desc you can use html markup)-->
61 <string name="iconpack_designer">Your Name</string>
62 <string name="iconpack_designer_desc">
63 <![CDATA[
64 <b>Your Name</b> is a great creative designer.
65 ]]>
66 </string>
67
```

App_one_content is the content of the card that shows info about one of your apps. It could be the description of it.

App_one_link is the link to download that app. Where the user will be redirected after clicking "download" button.

App_two_XXXXX
App_three_XXXXX

Those six strings have the same function as the previous ones, just that goes to different apps.

If you don't want to show all the three cards, I will explain later how to "delete" any of them. I won't explain how to add more, because is a little more difficult, and also it affects the UX. By the way, if you still want to add more, then hangout me, and I will explain it to you.

Changelog_content is the content of every changelog, you can add changelog for every update or just the latest. The problem comes with a long changelog because it becomes a little laggy on scrolling.

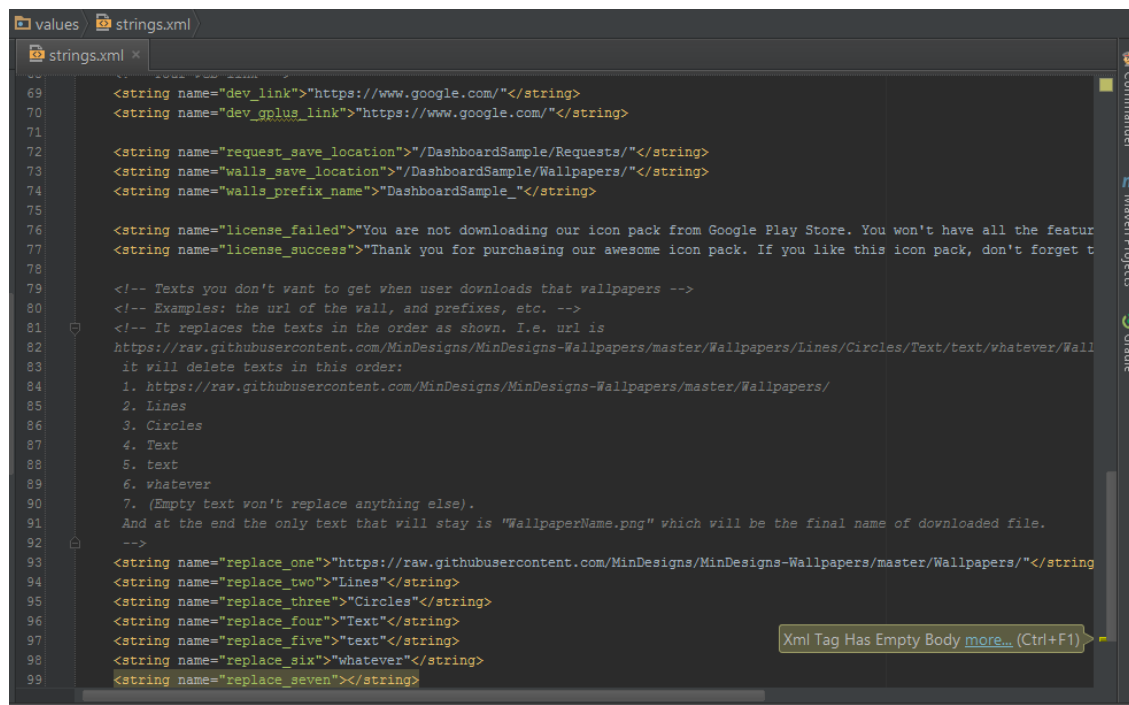
Using an html file should be faster, I know, but the reason I prefer a string is that you can add translations.

The "structure" of the changelog, or the format it must be written, is as shown, because that way it won't show weird spaces.

Iconpack_designer is your name, or your design group or team name.

Iconpack_designer_desc is a little description/bio about you. It has html format so it must keep the `<![CDATA[text]>` structure.

Here you can use `text` to make text **bold**. Or `<i>text</i>` to make text with *Italic font*. There may be many other options, but I won't explain them here.



```
69 <string name="dev_link">"https://www.google.com/"</string>
70 <string name="dev_gplus_link">"https://www.google.com/"</string>
71
72 <string name="request_save_location">"/DashboardSample/Requests/"</string>
73 <string name="walls_save_location">"/DashboardSample/Wallpapers/"</string>
74 <string name="walls_prefix_name">"DashboardSample_"</string>
75
76 <string name="license_failed">"You are not downloading our icon pack from Google Play Store. You won't have all the featur
77 <string name="license_success">"Thank you for purchasing our awesome icon pack. If you like this icon pack, don't forget t
78
79 <!-- Texts you don't want to get when user downloads that wallpapers -->
80 <!-- Examples: the url of the wall, and prefixes, etc. -->
81 <!-- It replaces the texts in the order as shown. I.e. url is
82 https://raw.githubusercontent.com/MinDesigns/MinDesigns-Wallpapers/master/Wallpapers/Lines/Circles/Text/text/whatever/Wall
83 it will delete texts in this order:
84 1. https://raw.githubusercontent.com/MinDesigns/MinDesigns-Wallpapers/master/Wallpapers/
85 2. Lines
86 3. Ciroles
87 4. Text
88 5. text
89 6. whatever
90 7. (Empty text won't replace anything else).
91 And at the end the only text that will stay is "WallpaperName.png" which will be the final name of downloaded file.
92 -->
93 <string name="replace_one">"https://raw.githubusercontent.com/MinDesigns/MinDesigns-Wallpapers/master/Wallpapers/"</string>
94 <string name="replace_two">"Lines"</string>
95 <string name="replace_three">"Ciroles"</string>
96 <string name="replace_four">"Text"</string>
97 <string name="replace_five">"text"</string>
98 <string name="replace_six">"whatever"</string>
99 <string name="replace_seven">"</string>
```

Dev_link is the link to your website.

Dev_gplus_link is the link to your google+ profile, or page, or community.

Request_save_location is where the requests files will be stored. For now it has only support for Internal Storage.

It will create that folder if the user doesn't have it.

It must begin and end always with / . And also, it must begin and end with the quotes.

i.e. */youriconpackname/requests/*

That will create a folder named youriconpackname and inside of it, there will be another folder named requests.

Walls_save_location has the same functionality as the previous strings, just that this folder will be used for the wallpapers only.

Walls_prefix_name is a prefix name that the wallpaper will have after being downloaded.

If the wall in the cloud it's named asdfgh and you specify the prefix as MylconPack_, after download the wall name will be

MylconPack_asdfgh

If you don't want to add this prefix just delete the text inside the quotes, like

```
<string name="walls_prefix_name">""</string>
```

License_success

License_failed

Are the messages you will show to user if the license check is success, or fail.

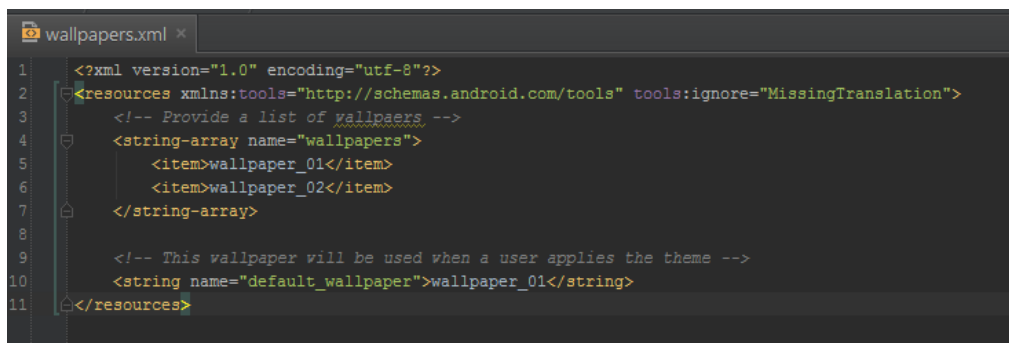
There is an example text, you can keep it or change it.

The last part is explained in the xml file, so just read it, and you will know how to use that.

Sorry for not rewriting it.

- Now let's edit wallpapers.xml also found at res/values/ folder
That xml file contains the name of the walls **included** in the app. (Not the names of the walls to be downloaded or something, only the ones **included**.)

I recommend to add them in res/drawable-nodpi folder, but you're free of adding them in any drawable folder you want.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources xmlns:tools="http://schemas.android.com/tools" tools:ignore="MissingTranslation">
3   <!-- Provide a list of wallpaers -->
4   <string-array name="wallpapers">
5     <item>wallpaper_01</item>
6     <item>wallpaper_02</item>
7   </string-array>
8
9   <!-- This wallpaper will be used when a user applies the theme -->
10  <string name="default_wallpaper">wallpaper_01</string>
11 </resources>
```


The array wallpapers will include the name of all the wallpapers included, and the array default_wallpaper is **obviously** the default wallpaper that will be applied with the skin.

You can name the wallpapers however you want, just skip spaces with underscores _ And try not to add Uppercase letters.

In the array replace the names with the actual name of the wall, without extension. I mean without .jpeg, .jpg, .png etc.

You can add as many walls as you want, just keep in mind it increases the app size, and that's why there's cloud wallpapers support. Also, remember you can set just **one** wallpaper as default.

This stuff is for Apex launcher, and maybe others, but I'm not sure.

- Next step: edit skin_colors.xml found at res/values/ folder
Each color is explained with its name, and you're free to edit them as you want. I don't really use them so I just keep them as there is, which is default, but you can play with those values and test the app and see how it changes the UI of some launchers.
- Final values step:
The icon_pack.xml file
This file contains the names of the icons drawables included in your pack.

They are separated in five sections:

Latest, system, google, games, and icon_pack

Latest, well, the icons you added in latest update.

System, icons for system apps (settings, calculator, browser, etc.)

Google, icons for Google apps (inbox, gmail, keep, chrome, etc.)

Games, icons for Games (asphalt, temple run, plants vs. zombies, etc.)

Icon_pack are all the icons included in the pack, so if you have 2000 icons, well, there will be 2000 items.

How to add a new icon there?

Find the section you want to add it.

And add <item>name</item>

Replace name with the actual name of your icon without extensions.

Keep the structure shown in sample.

If you add an incorrect name but keep the structure it won't make the app crash or something, it just won't be shown in the Icons section.

- Values files you **must not** edit:
Dontedit_strings.xml
Bools.xml
Dimens.xml
Styles.xml

Or, well, you can edit them, but this will affect the app UI, and functionality, so I won't explain them.

- Let's start with some drawables.

For icons, I recommend to save them as .png, and size of 192x192, but you're free to make them however you want.

Besides, I recommend to add them in drawable-nodpi folder so there won't be any problems with any dpi device.

Name it without uppercase text, replacing spaces with underscores _ and don't use numbers. Examples:

- Icon 1.png **wrong**
- Icon_1.png **wrong**
- icon1.png **wrong**
- icon_one.png **right**
- icon_two.png **right**
- icontwo.png **right**

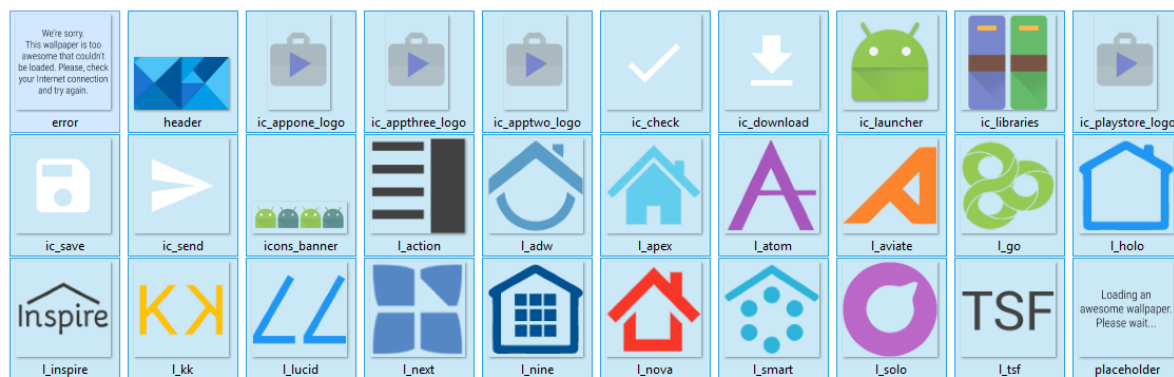
- App UI drawables.

I recommend to keep your icons, wallpapers, and previews pictures inside the folder drawables-nodpi.

And the other drawables in the other folders. I also recommend to keep those drawables names as they are, just change its content (and size if necessary), so you won't get issues with the app.

Most of the App UI drawables are located at drawable-xxhdpi folder.

Files you will find:



Error.png is a squared pic with an error pic. You can edit it however you want and set the size however you want. It will be used in wallpapers section, if there's an error loading the wallpaper.

Placeholder.png is also a squared pic, it contains something that will be shown **only** while loading the wallpaper.

These two drawables are not used by default, but I will explain how to enable them later.

Header.png is the picture shown in the drawer header.

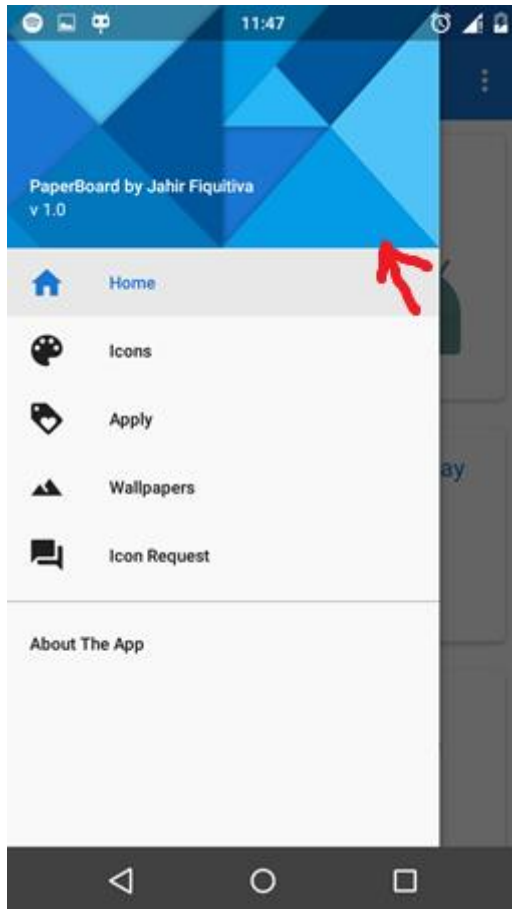
Recommended sizes (not mandatory).

xxhdpi → 900x500

xhdpi → 600x333

hdpi → 450x250

mdpi → 300x167



About...

lc_check.png

lc_save.png

lc_download.png

lc_send.png

I don't recommend to edit or modify them as they are the proper Material Design icons, and have the correct sizes for every dpi.

Btw, feel free to do that under your own responsibility.

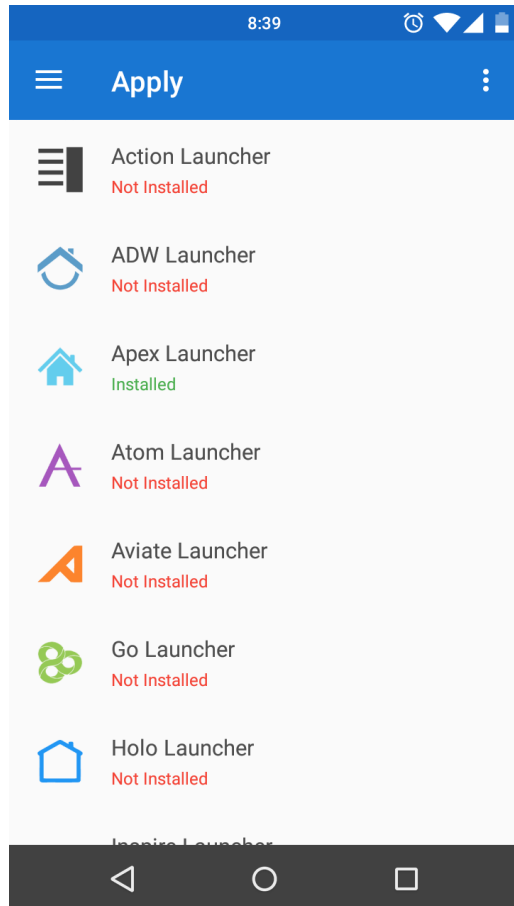
L_xxxxx.png

Is an icon for every launcher.

Perfect size to keep Material Design guidelines: 96x96

And make the icon inside almost touch the borders.

Edit the png, but keep their names.



lc_launcher.png is not the icon of your app shown in launcher. Is just a placeholder for icon request. You can edit it or keep it, anyways, it won't be really shown anywhere.

The icon of your app is found at mipmap-xxxx folders.

Sizes for your app icon:

Mipmap-mdpi → 48x48

Mipmap-hdpi → 72x72

Mipmap-xhdpi → 96x96

Mipmap-xxhdpi → 144x144

Mipmap-xxxhdpi → 192x192

Icons_banner.png is a banner with a preview of your icons shown in home section under the "welcome" card.

Recommended size: 900x300

Feel free to modify its size and content, just keep in mind the size may affect the UI. And always keep its name.

lc_appone_logo.png

lc_apptwo_logo.png

lc_appthree_logo.png

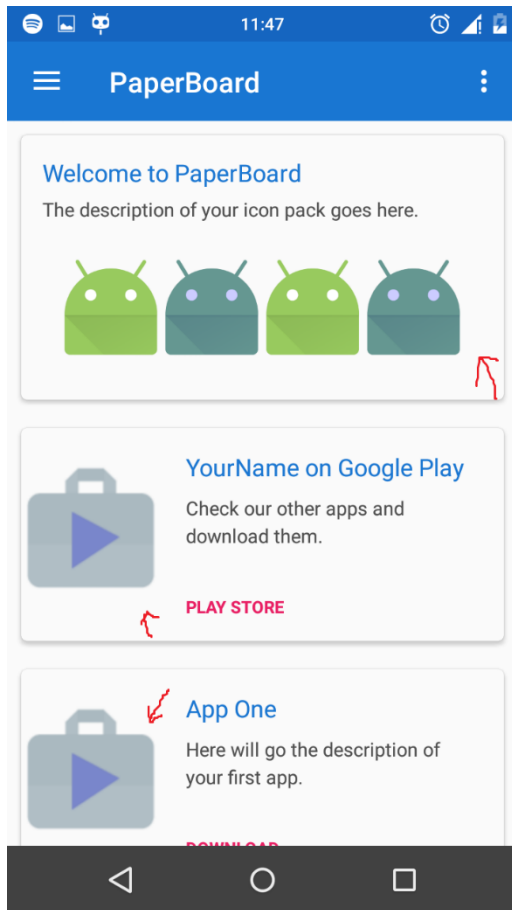
These icons are shown in the cards at home section where you put the info about some of your other apps.

Recommended size: 178x272

lc_playstore_logo.png

Is your logo as developer or designer.

Recommended size: 178x272



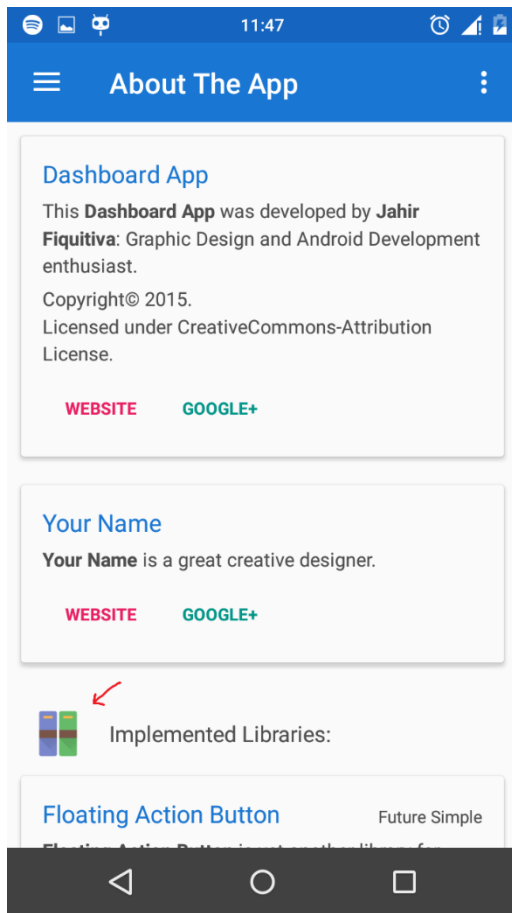
lc_libraries.png

Is an icon that represents libraries, feel free to modify it. Recommended size to keep

Guidelines: 96x96

It's shown in About The App or Credits section.

And it's made by Maximilian Keppeler.



- Now the most important files:
Found at res/xml folder

Those are:

Appfilter.xml
Appmap.xml
Drawable.xml
Noshader.xml
Themecfg.xml
Themeinfo.xml

I can't explain how noshader.xml works, but if you look to the1dynasty's template for it, you may find some useful info.

Why do I don't explain it? Just because I have not so many experience with it, neither I include it in my icon pack. Excuses.

Appfilter.xml

It contains the info about activities and its respective icon.

Just replace `ComponentInfo{apppackage/activityname}` `drawable="iconname"`

Keep the structure shown in source.

You will receive this file with every request, so you will know the correct activity everytime.

The stuff below is not necessary but you can add it if you want to.

This xml files also contains some elements set as default in Nova launcher (I'm not sure in others).

LAUNCHER_ACTION_APP_DRAWER → it's the default drawer icon

SMS and BROWSER → Default icon for messaging and browser apps, respectively.

Examples in source.

It also contains some other things, you can add to your icon pack. I recommend to add them if your icon pack has an specified shape, don't do it if they are shapeless:

- iconback is a background shape to the unthemed icons. You can add as many as you want with different colors that will be applied randomly. There are some in the source, though I don't use them as I set those lines as comments.
- Iconupon is like an overlay to the unthemed icons, add it if you use some shadows or extra elements.
- Iconmask, is how your icons will be "cutted". It's like a black picture, with a transparent shape that matches the shape of your icon pack. See the picture for "details".
- Scale, if you set this, the icons will be scaled as default. Use values of 0.5, 0.7, or 1.2, 1.5 Whatever you need but keeping that format.

If you don't want the icons to be scaled, just delete those lines.

Appmap.xml

It has the same functionality as Appfilter.xml, though, it's for other launchers and it has another but similar structure.

```
<item class="activityname" name="iconname"/>
```

Example included in source.

Drawable.xml

Is a list of all your icons separated by categories.

Keep version number as it is, just replace

Structure

```
<category title="categoryname" />
```

```
<item name="iconname" />
```

```
<item name="anothericonname" />
```

```
<category title="anothercategoryname" />
```

```
<item name="anothericonname" />
```

```
<item name="anotheranothericonname" />
```

Example in source.

ThemeInfo.xml and Themecfg.xml

Those are files including some info about the icon pack. Keep the strings as they are, just change its content, their names are clear enough to understand so just replace with the info of your icon pack.

An extra for this stuff

I found the tool png2xml.jar by the great Pkmmte Xeleon, which generates the drawable.xml and icon_pack.xml (found at values folder), automatically if you add it to your icons folder. (Where you put the pngs).

Well, I modified it a little so that it creates also the appfilter.xml. The only thing you must do is to add the correct activity name to it.

Hope it helps a little more to you. Feel free to use it or not.
Found at source/tools.

- Another important files:
These will be found at assets folder
 - App_func_theme.xml
 - Appfilter.xml
 - Desk.xml
 - Drawable.xml
 - Themecfg.xml
 - Themeinfo.xml
 - Themefont.ttf

Appfilter.xml, drawable.xml, themecfg.xml and themeinfo.xml
Are exactly the same found at xml folder, just copy, paste and replace the old.

Themefont.ttf is a font used to apply the skin in Apex (not tested in other launchers).
You only have to keep its name as it is "themefont.ttf"
Some fonts may not be supported by the launcher, i.e. the included in source.

App_func_theme.xml and desk.xml are for Go Launcher support, though I'm not sure how to use them, but feel free to modify them, under your own responsibility, or you can leave them as they are, which is default.

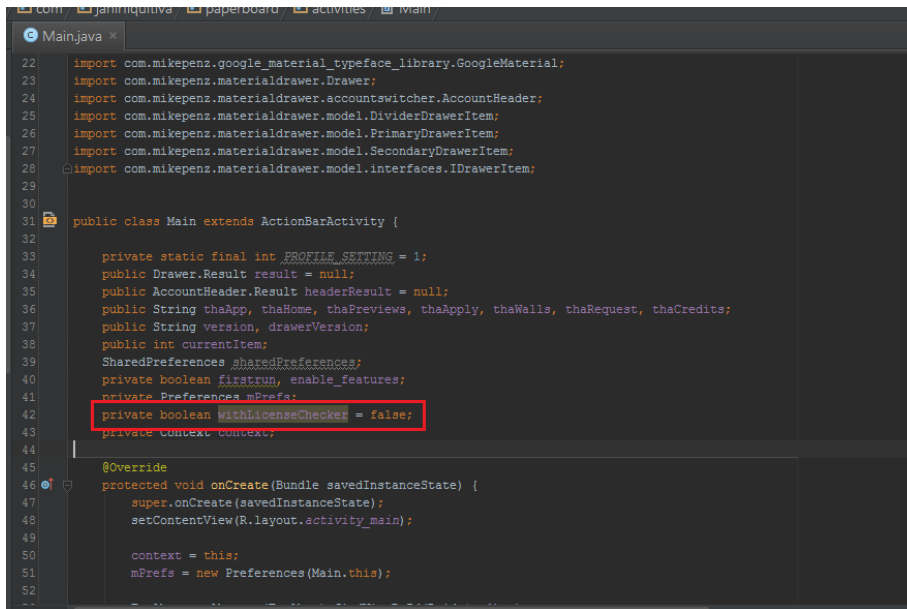
- Now some coding needed stuff:
 - **How to enable/disable the LicenseChecker?**
Go to `com.jahirfiquitiva.paperboard.activities.main`

Search for the line:
`Private Boolean withLicenseChecker = true;`

Just edit it with
True – to enable licenseChecker
False – to disable it.

That's it.

Important: don't enable it while testing, or you won't be actually able to test, as the app will close if it's pirate.



```
22 import com.mikepenz.google_material_typeface_library.GoogleMaterial;
23 import com.mikepenz.materialdrawer.Drawer;
24 import com.mikepenz.materialdrawer.accountswitcher.AccountHeader;
25 import com.mikepenz.materialdrawer.model.DividerDrawerItem;
26 import com.mikepenz.materialdrawer.model.PrimaryDrawerItem;
27 import com.mikepenz.materialdrawer.model.SecondaryDrawerItem;
28 import com.mikepenz.materialdrawer.model.interfaces.IDrawerItem;
29
30
31 public class Main extends ActionBarActivity {
32
33     private static final int PROFILE_SETTING = 1;
34     public Drawer.Result result = null;
35     public AccountHeader.Result headerResult = null;
36     public String thaApp, thaHome, thaPreviews, thaApply, thaWalls, thaRequest, thaCredits;
37     public String version, drawerVersion;
38     public int currentItem;
39     SharedPreferences sharedPreferences;
40     private boolean firstrun, enable_features;
41     private Preferences mPrefs;
42     private boolean withLicenseChecked = false;
43     private Context context;
44
45
46     @Override
47     protected void onCreate(Bundle savedInstanceState) {
48         super.onCreate(savedInstanceState);
49         setContentView(R.layout.activity_main);
50
51         context = this;
52         mPrefs = new Preferences(Main.this);
53
54     }
55 }
```

- **How to enable the use of Error and Placeholder drawables in Wallpapers section?**
Look for
com.jahirfiquitiva.paperboard.fragments.WallsGridAdapter
Now, look for:

```
Picasso.with(context)
    .load(wallurl)
    .resize(imageWidth, imageWidth)
    .centerCrop()
    .into(wall, new Callback.EmptyCallback() {
        @Override
        public void onSuccess() {
            if (mProgress != null) {
                mProgress.setVisibility(View.GONE);
            }
        }
    });
```

And replace with:

```
Picasso.with(context)
    .load(wallurl)
    .error(R.drawable.error);
    .placeholder(R.drawable.placeholder);
    .resize(imageWidth, imageWidth)
    .centerCrop()
    .into(wall);
```

Before

```
Picasso.with(context)
    .load(jsondata.get(Wallpapers.WALL))
    .resize(imageWidth, imageWidth)
    .centerCrop()
    .into(wall, (EmptyCallback) onSuccess() -> {
        if (mProgress != null) {
            mProgress.setVisibility(View.GONE);
        }
    });
```

After

```
Picasso.with(context)
    .load(jsondata.get(Wallpapers.WALL))
    .error(R.drawable.error)
    .placeholder(R.drawable.placeholder)
    .resize(imageWidth, imageWidth)
    .centerCrop()
    .into(wall);
```

- **How to delete cards from Home section:**
Open section_home.xml inside res/layout/ folder
Look for the card you want to delete and add the line:
android:visibility="gone"

That's it.

Before

```
<android.support.v7.widget.CardView
    android:id="@+id/cardView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    card_view:cardCornerRadius="4dp"
    card_view:cardElevation="3dp">
```

After

```
<android.support.v7.widget.CardView
    android:id="@+id/cardView"
    android:visibility="gone"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    card_view:cardCornerRadius="4dp"
    card_view:cardElevation="3dp">
```

- The last file you must create is the JSON file.
Content:
JSON Array name = "wallpapers"
JSON Object names:
name="nameofyourwall"
author="nameoftheauthor"
wall="urlofthewall"

There's an example at source/json_example.

You don't need to worry about the thumb_url as Picasso do the stuff to make the preview of the wallpaper look like a square.

- **Further support.**
If you think I missed something, send a hangout and I will add it to this tutorial as soon as possible.

If you think everything is explained, but still having issues or don't understand anything, also contact me via hangouts, I will try to answer as soon as possible.

Sorry if there are some out dated screenshots, or screenshots not showing the actual code but I tried to make it compatible even with the latest changes. I will update them soon.

If you think that your questions or suggestions may help others too, **don't** hangout me, but post the question in the Google+ Community.

About including libraries, always check the latest version of each one. I added links to them in the GitHub README.MD file.

- **Known issues.**

If user "enters" wallpapers section, and is loading, and he suddenly loses the Internet connection, then app will FC.

App can be a little laggy if there are many icons to request. So, theme them all. ;)

- **Some screenshots:**

