

PAPERBOARD

by Jahir Fiquitiva

Contents:

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

Info about Dashboard:

Features:

- Material Design inspired/based dashboard.
- Cloud based (only) wallpapers.
- Wallpapers can be applied, cropped-and-applied, or downloaded.
- Muzei Support.
- In-App Icon Request tool.
- Previews section, where user could see themed icons.
- Clickable icons in Previews section, showing icon and app name in a dialog.
- License Checker.
- Changelog shown with every update.
- Apply section with 26 supported launchers.
- Credits section.
- Docks support.
- Requires API15+ or Android 4.0.3+

Future features (No ETA):

- Replace GridView and ListView with RecyclerView to improve performance.
- Themeable UI.
- Option to delete cache directly inside the app.
- Option to choose downloads folder (for wallpapers).
- In-App-Purchases for Premium Icon Requests.
- Searchable Icons and Launchers to apply pack.

Info about the License:

This Dashboard is licensed under MIT license. **What does that means?** This means that you may use the work commercially, make changes to the work, distribute the compiled code and/or source and use the work for private use. This also means you may not hold the author liable. Because the work is provided "as is". And you must include the copyright notice in all copies or substantial uses of the work and include the license notice in all copies or substantial uses of the work.

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.

Additional requirement to use: Keep Credits section as it is and my name in very first card. That's it.

PaperBoard by Jahir Fiquitiva.

Copyright © 2015. Licensed under MIT License.

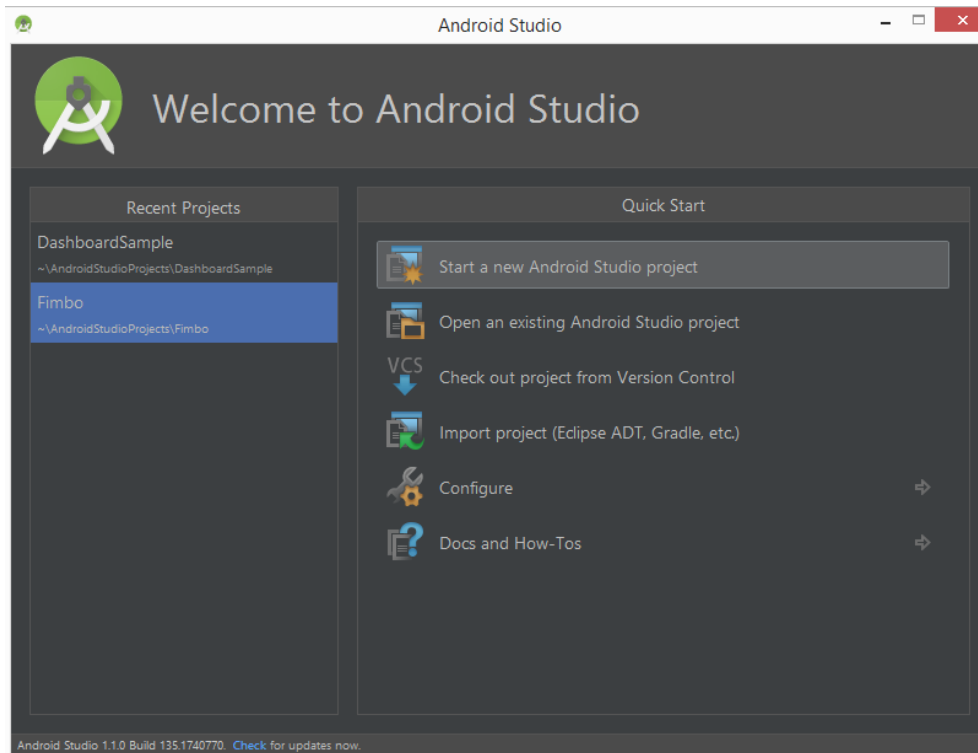
Requirements:

- 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++

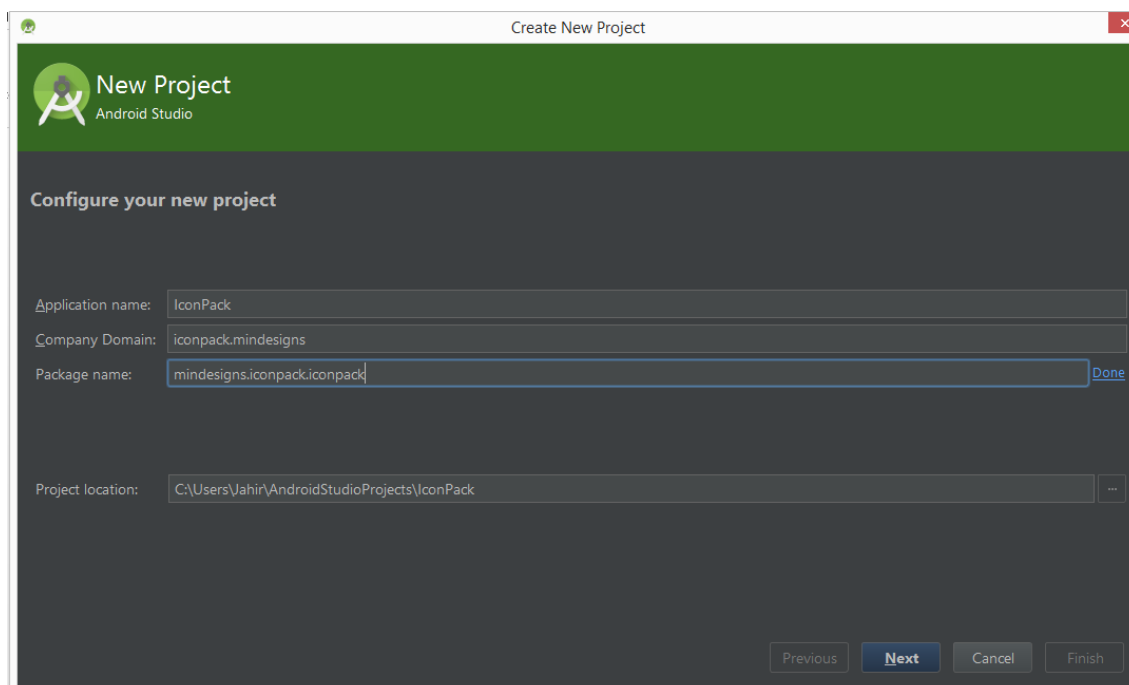
Tools			
Android SDK Tools	24.1.2		Installed
Android SDK Platform-tools	22		Installed
Android SDK Build-tools	22.0.1		Installed
Android SDK Build-tools	21.1.2		Installed
Android SDK Build-tools	20		Not installed
Android SDK Build-tools	19.1		Not installed
Android 5.1.1 (API 22)			
Documentation for Android SDK	22	1	Not installed
SDK Platform	22	2	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	Not 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	3	Not installed
Android TV Intel x86 Atom System Image	21	3	Not installed
Android Wear ARM EABI v7a System Image	21	2	Not installed
ARM EABI v7a System Image	21	3	Not installed
Intel x86 Atom_64 System Image	21	3	Not installed
Intel x86 Atom System Image	21	3	Not installed
Google APIs	21	1	Not installed
Google APIs ARM EABI v7a System Image	21	4	Not installed
Google APIs Intel x86 Atom_64 System Image	21	4	Not installed
Google APIs Intel x86 Atom System Image	21	4	Not 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	13		Installed
Android Support Library	22.1		Installed
Google Play services	23		Installed
Google Repository	16		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

Getting Started:

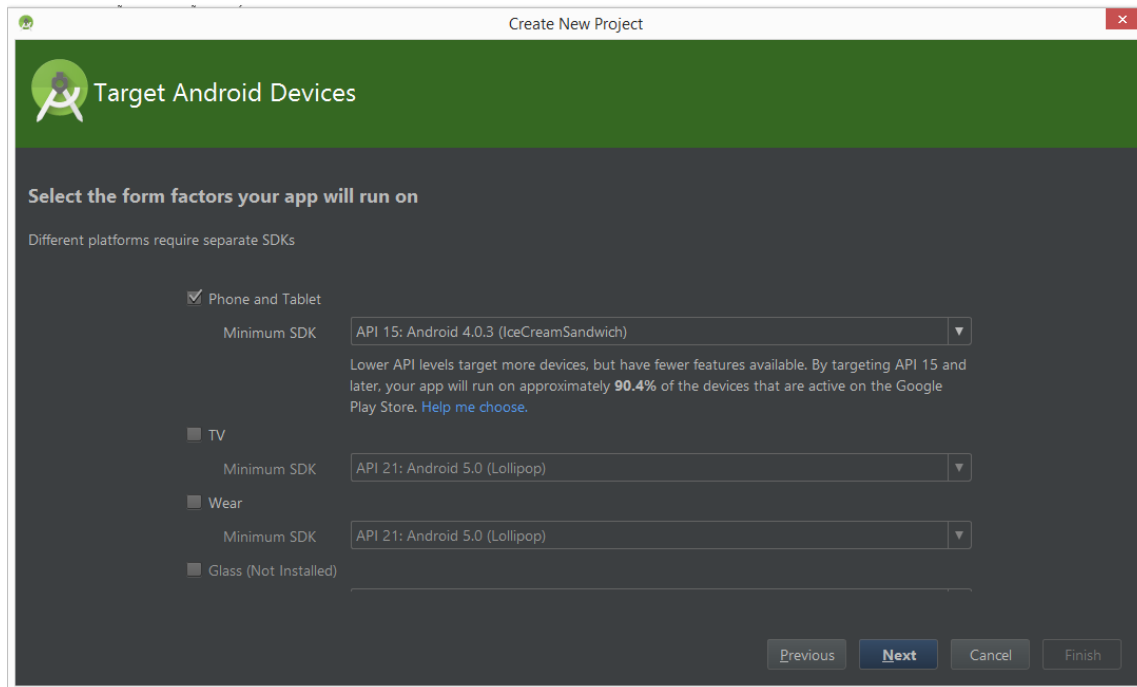
- Open Android Studio and create a new Project



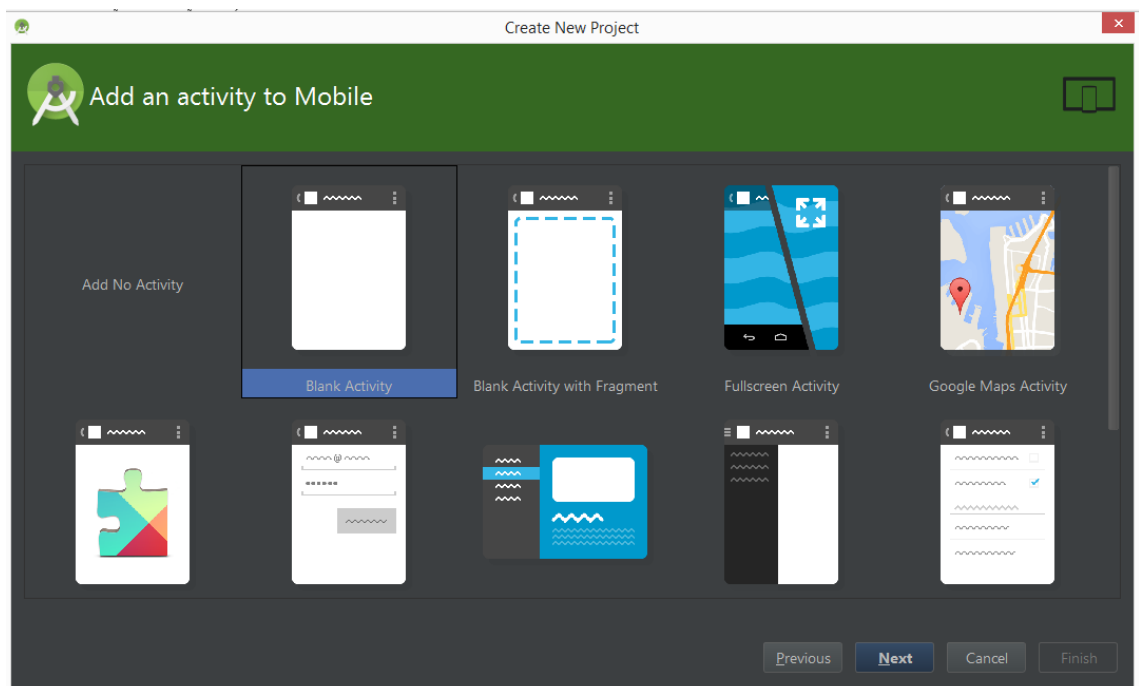
- Name it however you want

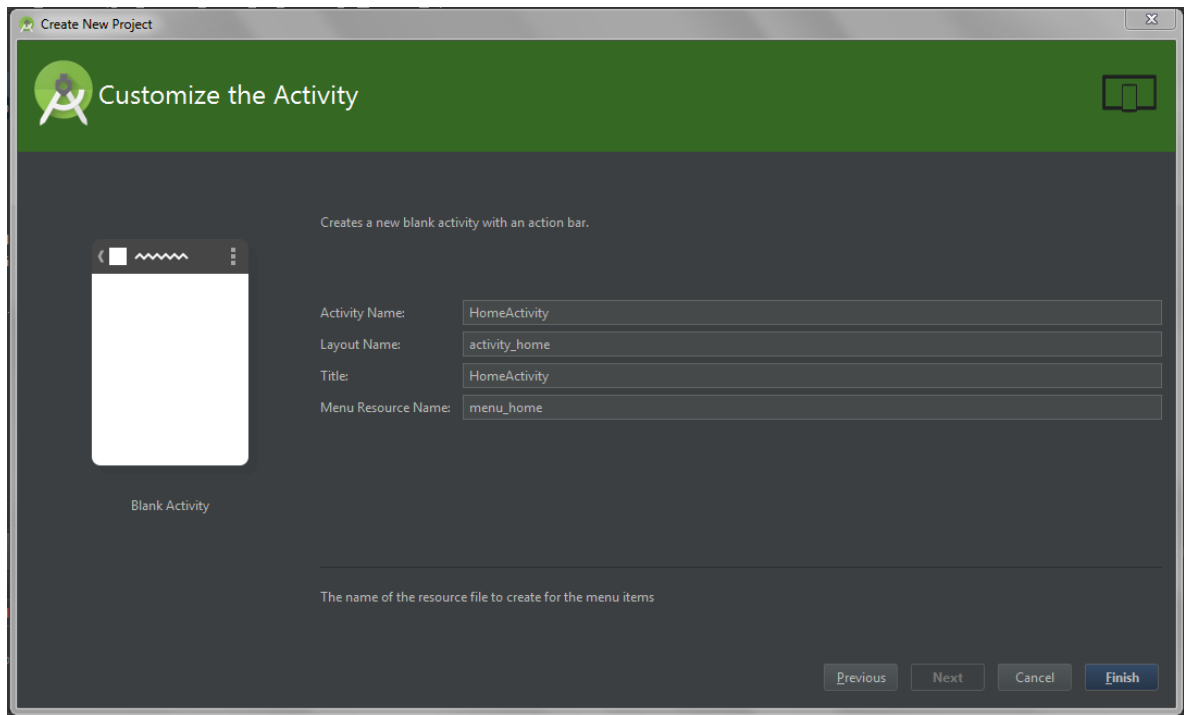


- Choose Minimum SDK to API15



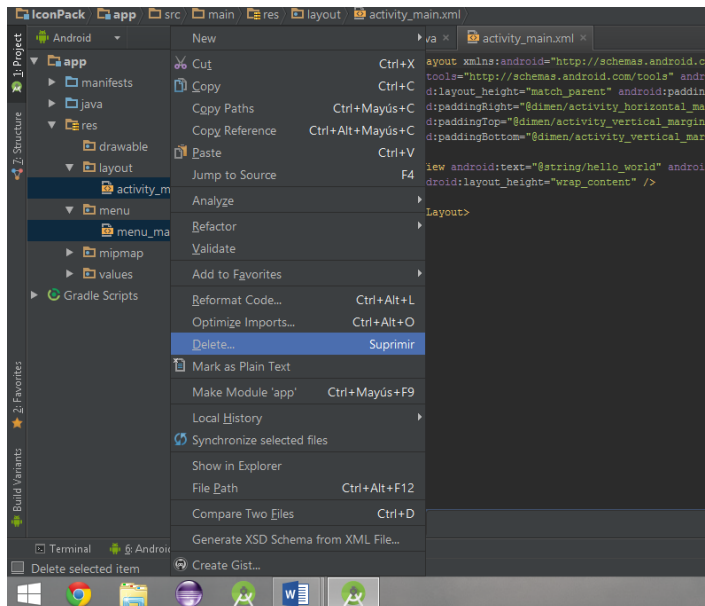
- Create a new blank activity with any name you want. Just **don't** use MainActivity.





- Hit Finish
- Locate these files:
res/layout/activity_home
res/menu/menu_home

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:

```
import android.content.Intent;  
import android.os.Bundle;  
import android.support.v7.app.AppCompatActivity;  
  
import com.jahirfiquitiva.paperboard.activities.Main;  
  
public class Home extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
}
```

- Now open your build.gradle file and fill it like shown in these pictures:

```
apply plugin: 'com.android.application'  
  
repositories {  
    jcenter()  
    maven { url 'https://dl.bintray.com/drummer-aidan/maven' }  
}  
  
android {  
    compileSdkVersion 22  
    buildToolsVersion "21.1.2"  
  
    defaultConfig {  
        applicationId "jahirfiquitiva.paperboard.sample"  
        minSdkVersion 15  
        targetSdkVersion 22  
        versionCode 7  
        versionName "7.0"  
    }  
  
    buildTypes {  
        debug {  
            minifyEnabled true  
            shrinkResources false  
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
        }  
        release {  
            minifyEnabled true  
            shrinkResources false  
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
        }  
    }  
}
```

```

    }
    release {
        minifyEnabled true
        shrinkResources false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile 'com.android.support:appcompat-v7:22.1.0@aar'
    compile 'com.android.support:support-v4:22.1.0@aar'
    compile 'com.android.support:cardview-v7:22.1.0@aar'
    compile 'com.android.support:palette-v7:22.1.0@aar'
    compile 'com.android.support:recyclerview-v7:22.1.0@aar'

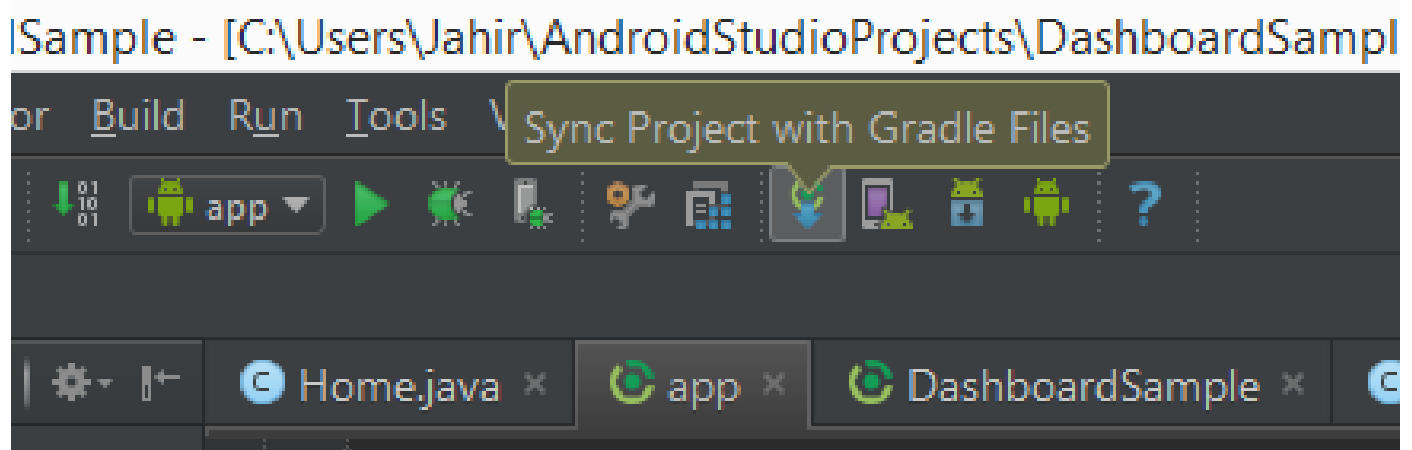
    compile 'com.afollestad:material-dialogs:0.7.2.7'
    compile('com.mikepenz:materialdrawer:library:2.8.1@aar') {
        transitive = true
    }
    compile 'com.melnykov:floatingactionbutton:1.3.0'
    compile 'com.balysv:material-ripple:1.0.1'

    compile 'com.squareup.picasso:picasso:2.5.2'
    compile 'com.squareup.okhttp:okhttp:2.3.0'
    compile 'com.squareup.okhttp:okhttp-urlconnection:2.3.0'
    compile 'com.google.android.apps.muzei:muzei-api:2.0'
}

```

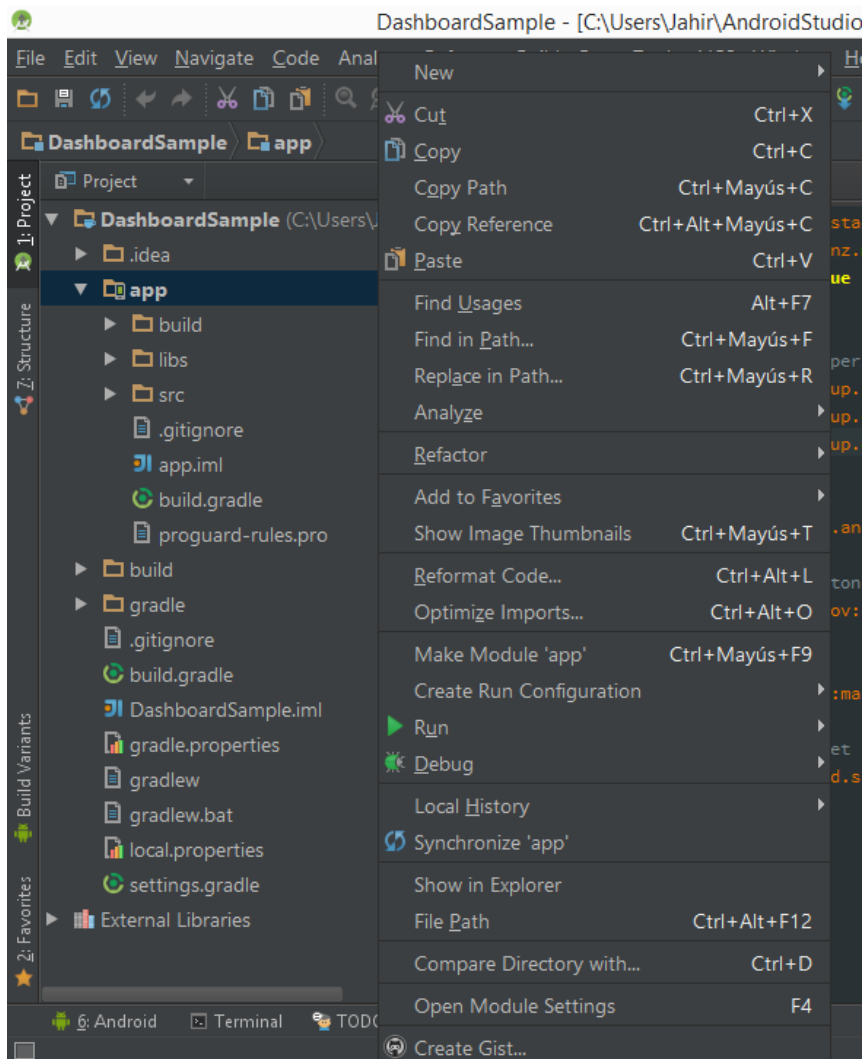
Screenshots may be out of date. So you must check the libs in GitHub repositories and add the latest version for each one.

- 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/app folder into your app's app folder.



- Open AndroidManifest.xml, and look for this lines

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  package="com.jahirfiquitiva.dashboardsample">
```

```
<!-- MainActivity -->
<activity
  android:name=".HomeActivity"
  android:label="PaperBoard"
  android:noHistory="true">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
    <category android:name="android.intent.category.MULTIWINDOW_LAUNCHER" />
  </intent-filter>
</activity>
```

Replace

jahirfiquitiva.paperboard.sample
with your package name i.e.
com.package.name

And change
.HomeActivity

With the name of the activity you created at the beginning i.e.
.HomeActivity (I know is redundant but is the one I used in the beginning of tutorial.)

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

And add these lines to onCreate void.

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

```
finish();
```

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

At the end should look like this:

```

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

import com.jahirfiquitiva.paperboard.activities.MainActivity;

public class HomeActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

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

        finish();
    }
}

```

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

```
colors.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3
4      <color name="primary">#1976d2</color>
5      <color name="primary_dark">#1565c0</color>
6      <color name="primary_pressed">#039be5</color>
7      <color name="accent">#e91e63</color>
8      <color name="ripple_accent">#80e91e63</color>
9      <color name="accent_pressed">#ec407a</color>
10     <color name="accent_secondary">@color/accent</color>
11     <color name="secondary">#009688</color>
12     <color name="secondary_pressed">#26a69a</color>
13     <color name="nav_bar">#424242</color>
14     <!-- Ctrl + H and search for color/cardsbg if you want to customize card colors individually-->
15     <color name="cardsbg">#ffffff</color>
16     <!-- Color for Backgrounds -->
17     <color name="light_bg">#fafafa</color>
18     <!-- Color for texts in Dialogs -->
19     <color name="dialog_content">#6d6d6d</color>
20     <!-- Color for some ripples -->
21     <color name="semitransparent_white">#80ffffff</color>
22     <!-- Color for texts -->
23     <color name="dark_grey">#4d4d4d</color>
```

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 #80

Accent_Pressed must be a lighter color from accent.

Secondary is a secondary color, different from primary and accent, which will be used for some buttons.

Secondary_Pressed must be a lighter color from secondary.

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

Cards_bg is the background color to cards at Home and Credits sections.

Light_bg is the background for all windows.

Dialog_content is the color for the text in the Changelog dialog.

Semitransparent_white is the color for some ripples and other stuff.

Dark_grey is the color for texts in all the app.

You don't have to edit them all. Just the ones you want.

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 `donedit_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.

```

<!-- MainActivity Strings -->
<string name="app_name">"PaperBoard"</string>
<string name="app_long_name">"PaperBoard - Dashboard Sample"</string>
<string name="package_name">"jahirfiquitiva.paperboard.sample"</string>
<string name="theme_description">"PaperBoard is a Material Design based Icon Pack dashboard template. Free and Open-Sourced.</string>

<!-- Muzei Stuff -->
<string name="artsource_name">"PaperBoard"</string>
<string name="artsource_desc">"Wallpapers made by MinDesigns"</string>
<string name="muzei_settings">"Muzei Settings"</string>
<string name="json_file_url">"https://raw.githubusercontent.com/jahirfiquitiva/MinDesigns-Wallpapers/master/JSON-Files/walls_test.json"
<!-- Sharing wallpaper from Muzei -->
<string name="partone">"My wallpaper for today is "</string>
<string name="parttwo">" by "</string>
<string name="partthree">" from "</string>
<string name="partfour">"\nGet it now at "</string>

<!-- Developer Account Link -->
<string name="play_store_dev_link">"https://play.google.com/store/apps/developer?id=Jahir+Fiquitiva"</string>

<!-- Icon Pack designer/owner e-mail -->
<string name="email_id">"someone@email.com"</string>

```

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 could 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, or write whatever you want.

Package_name is 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.

Artsource_name is the name shown in muzei.

Artsource_desc is the description shown in muzei.

Muzei_settings is the title for the activity of muzei settings, you don’t have to change it but do it if you want to.

Json_file_url is the url where your json file is located

The strings partxxxx are just constructors for the content when an user wants to share muzei wallpaper. You can change them or leave them as they are.

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.

```

<!-- Mail subjects for simple e-mail and requests e-mail -->
<string name="email_subject">"PaperBoard"</string>
<string name="email_request_subject">"PaperBoard - Icon Requests"</string>

<!-- This are the strings for the cards in the main screen -->
<string name="welcome_title">Welcome to PaperBoard</string>

<string name="play_card_title">Jahir Fiquitiva on Google Play</string>
<string name="play_card_content">"Check my other apps and download them."</string>

<string name="app_one_title">Ideal Themes</string>
<string name="app_one_content">"Ideal is a set of four themes to give your KitKat device a full re-design and look with Material Design"</string>
<string name="app_one_package">"com.jaydvl.idealthemes.light"</string>

<string name="app_two_title">Flaterial</string>
<string name="app_two_content">"Flaterial is a Zooper skin with the awesome collab of Juan Saracho and Dany Flores. Material Design"</string>
<string name="app_two_package">"com.mindesigns.zwskin.flaterial"</string>

<string name="app_three_title">Pop It!</string>
<string name="app_three_content">"Pop It! is a Zooper skin with the awesome collab of Juan Saracho and Dany Flores. Material Design"</string>
<string name="app_three_package">"com.mindesigns.popit.zwskin"</string>

```

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.

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_package is the package name for that app.

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.

```

<string name="iconpack_designer">Your Name</string>
<string name="iconpack_designer_desc">
    <![CDATA[
        I\'m a <b>designer</b>. I like <b>Android</b>. I love this <b>dashboard</b>.
    ]]>
</string>
<!-- A link to your web page, if you have one -->
<string name="dev_link">https://jahirfiquitiva.github.io/me/</string>
<string name="dev_gplus_link">"https://www.google.com/+JahirFiquitivaJDev"</string>

<!-- Feature picture that appears above the description -->
<string name="theme_feature">icons_banner</string>

<!-- Previews Pictures Names -->
<string name="theme_preview1">preview1</string>
<string name="theme_preview2">preview2</string>

<!-- Tabs Names list -->
<string-array name="tabs">
    <item>LATEST</item>
    <item>SYSTEM</item>
    <item>GOOGLE APPS</item>
    <item>GAMES</item>
    <item>ALL</item>
    <item>DRAWERS</item>
</string-array>

```

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.

Dev_link is the link to your website.

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

The string array "tabs" are the names for the tabs shown in Icons/Previews section, you can modify them, or delete some, or add some, just make sure to fix the necessary code in its java file, otherwise you will get a FC. (It will be explained at the end).

```

<!-- Strings for Wallpapers and Requests -->
<string name="request_save_location">"/PaperBoard/Requests/"</string>
<string name="walls_save_location">"/PaperBoard/Wallpapers/"</string>
<string name="walls_prefix_name">"PaperBoard_"</string>

<!-- Messages shown when using LicenseChecker -->
<string name="license_failed">"You are not downloading PaperBoard from Google Play Store. You have two options only:"</string>
<string name="license_success">"Thank you for downloading my Icon Pack Dashboard Sample. If you like it, don't forget to rate </string>

```

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.

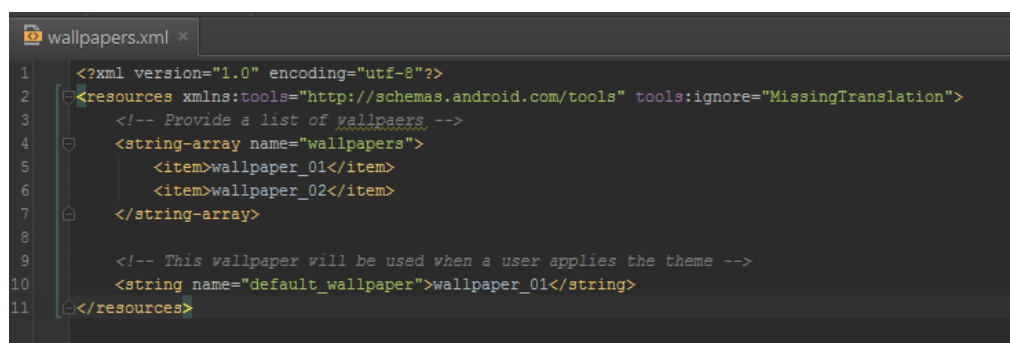
There is an example text, you can keep it or change it. They will be only used if you enabled the license checking.

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 wallpapers -->
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.

- You want to have a shadow below the toolbar? Or, the correct term: elevation?
Go to `dimens.xml`
Find this line and edit it

```
<!-- Elevation of toolbar. Play with values until you find the one you want -->  
<dimen name="toolbar_elevation">@dp</dimen>
```

Afaik, default elevation is 6dp, but as it says, play with that value.

All other `dimens` values can be edited, but I'm not responsible if you ruin the UI after doing it.

- 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.
- Let's update the changelog!
Go to `res/values/changelog.xml` You will see something like this

```
<?xml version="1.0" encoding="utf-8"?>  
<resources xmlns:tools="http://schemas.android.com/tools" tools:ignore="MissingTranslation">  
  
    <string-array name="fullchangelog">  
        <item>@array/newstuff</item>  
        <item>@array/improvements</item>  
        <item>@array/fixes</item>  
    </string-array>  
  
    <string-array name="newstuff">  
        <item>New</item>  
        <item>Ripples in tabs in Icons section.</item>  
        <item>Ripples when pressing icons and wallpapers.</item>  
        <item>FAB simplified to just one lib.</item>  
        <item>Requests FAB icon changed.</item>  
        <item>Added Nova Launcher Wallpapers support.</item>  
        <item>Added Docks support.</item>  
        <item>Cards in Home section will hide if app is installed.</item>  
        <item>Added FAB in Home section to Apply icons.</item>  
        <item>Added 8 new launchers to launchers list. Total: 26.</item>  
        <item>Added support for LG Home Launcher.</item>  
        <item>Added Drawers tab to Icons section.</item>  
    </string-array>  
  
    <string-array name="improvements">  
        <item>Improvements</item>  
        <item>Home section cards buttons alignments improved.</item>  
        <item>FAB deleted from viewing walls.</item>  
    </string-array>  
</resources>
```


The array fullchangelog will include all the arrays for every update.

The array vX will include the version number as first item, and the other items are just a new change.

Will be shown in that exact order.

Follow the example to add you own.

- What about docks?

You will need to edit docks.xml

Adding docks is simple

```
<?xml version="1.0" encoding="UTF-8"?>
<resources
  xmlns:tools="http://schemas.android.com/tools"
  tools:ignore="MissingTranslation"
>

  <string-array name="dock_backgroundlist">
    <item>dock_a</item>
    <item>dock_b</item>
  </string-array>

</resources>
```

Keep the string array name as it is because some launchers detect it by it, to add items just write
<item>drawable_name</item>

Try to follow the example and you will be done.

- 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 six sections:

Latest, system, google, games, icon_pack and drawer

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.

Drawer was created as a new section for you to show only the icons you created for app drawer.

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

Libraries_strings.xml

Bools.xml

Styles.xml

Launchers.xml

Or, well, you can edit them, but this will affect the app UI, and functionality, so I won't explain them. Do it under your own responsibility and/or risk.

- 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**
- icon_alt.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 neither added by default, but I will explain how to enable them later.

Ic_muzei_logo is the logo of your app shown in muzei.
Recommended size 144x144

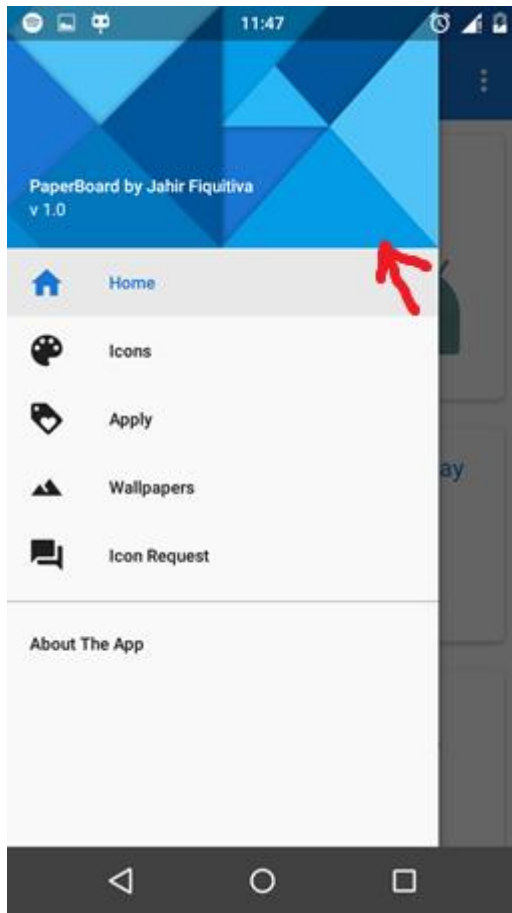
Header.png is the picture shown in the drawer header.
Recommended sizes (not mandatory).

xxhdpi → 900x500

xhdpi → 600x333

hdpi → 450x250

mdpi → 300x167



About...

lc_check.png

lc_save.png

lc_download.png

lc_send.png

lc_apply_icons.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.

ic_xxxx_launcher.png

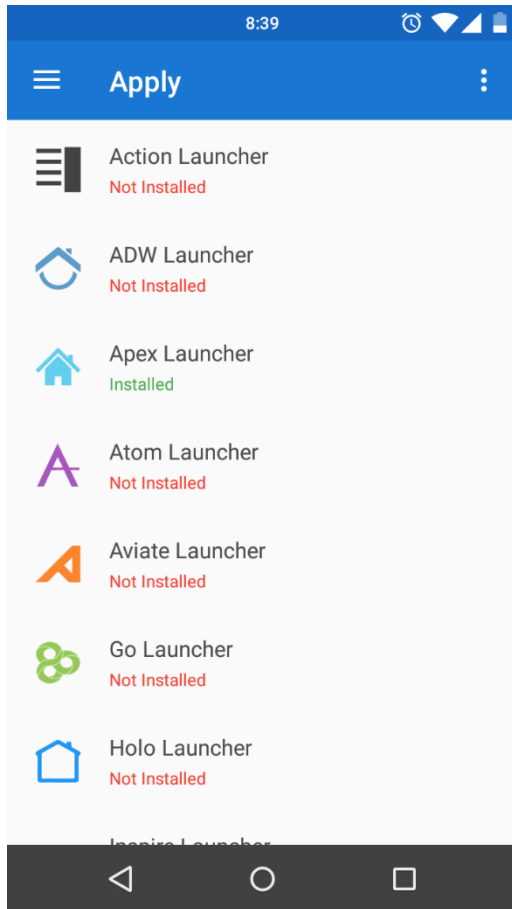
Is an icon for every launcher.

Perfect size to keep Material Design guidelines: 96x96

And make the icon inside almost touch the borders.

Icon for CyanogenMod Theme Engine was made by Maximilian Keppeler.

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 in the upper part of 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

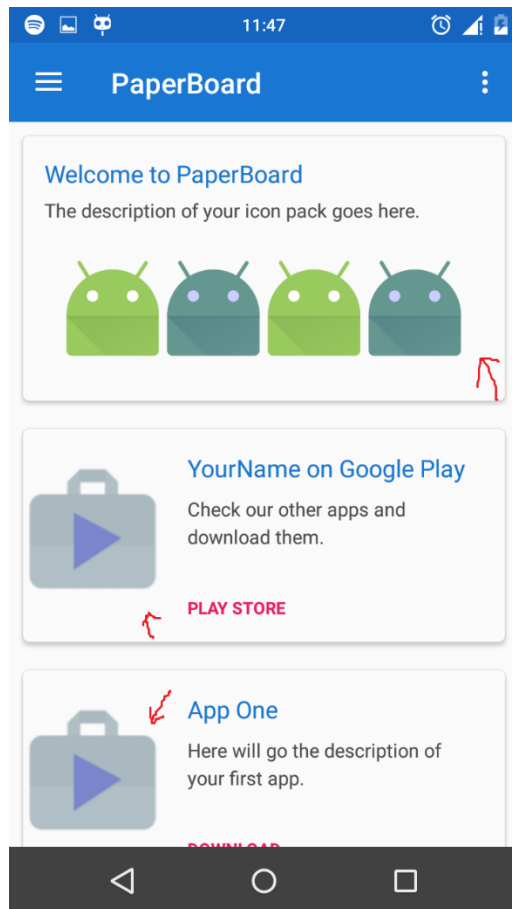
These icons are shown in the cards at home section where you put the info about some of your other apps.

Recommended size: 178x272

lc_playstore_logo.png

Is your logo as developer or designer.

Recommended size: 178x272

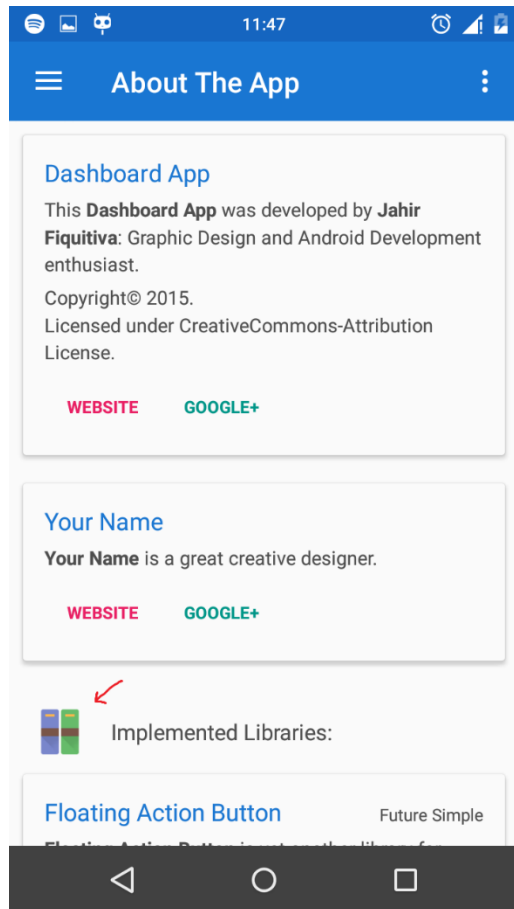


lc_libraries.png

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

Theme_resources.xml

Theme_wallpapers.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.

There are more keywords available. Examples can be found 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.

Theme_resources.xml

Again, the same function as Appfilter.xml, just that this one goes for Lg Home Launcher and it has different structure. More examples can be found in source.

```
<AppIcon name="packageName/activityName" image="iconname" />
```

Drawable.xml

Is a list of all your icons separated by categories.

Keep version number as it is, just replace the things below.

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 DEV/tools/xmlgen.jar.

- 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.MainActivity`

Search for the line:

```
private static final boolean WITH_LICENSE_CHECKER = 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.

```
public class MainActivity extends AppCompatActivity {  
  
    private static final boolean WITH_LICENSE_CHECKER = false;  
    private static final String MARKET_URL = "https://play.google.com/sto
```

- **How to enable the use of Error and Placeholder drawables in Wallpapers section?**

Look for

`com.jahirfiquitiva.paperboard.adapters.WallsGridAdapter`

Now, look for:

```
Picasso.with(context)  
    .load(wallurl)  
    .resize(imageWidth, imageWidth)  
    .centerCrop()  
    .noFade()  
    .transform(PaletteTransformation.instance())  
    .into(holder.wall,  
        new PaletteCallback(holder.wall){  
            @Override  
            public void onSuccess(Palette palette){  
                holder.progressBar.setVisibility(View.GONE);  
  
                if (usePalete) {  
                    if (palette != null) {  
                        Palette.Swatch wallSwatch = palette.getVibrantSwatch();  
                        if (wallSwatch != null) {  
                            holder.titleBg.setBackgroundColor(wallSwatch.getRgb());  
                            holder.titleBg.setAlpha(1);  
                            holder.name.setTextColor(wallSwatch.getTitleTextColor());
```

```

        holder.name.setAlpha(1);
    }
}

}

}

@Override
public void onError() {

}

});

```

And add:

```

.error(R.drawable.error);
.placeholder(R.drawable.placeholder);

```

Before

```

holder.wall.startAnimation(anim);
Picasso.with(context)
    .load(wallurl)
    .resize(imageWidth, imageWidth)
    .centerCrop()
    .noFade()
    .transform(PaletteTransformation.instance())
    .into(holder.wall,
        new PaletteCallback(holder.wall){
            @Override
            public void onSuccess(Palette palette){
                holder.progressBar.setVisibility(View.GONE);

                if (usePalette) {
                    if (palette != null) {
                        Palette.Swatch wallSwatch = palette.getVibrantSwatch();
                        if (wallSwatch != null) {
                            holder.titleBg.setBackgroundColor(wallSwatch.getRgb());
                            holder.name.setTextColor(wallSwatch.getTitleTextColor());
                            holder.name.setAlpha(1);
                        }
                    }
                }
            }
        }
    );
}

```

After

```

holder.wall.startAnimation(anim);
Picasso.with(context)
    .load(wallurl)
    .resize(imageWidth, imageWidth)
    .centerCrop()
    .noFade()
    .error(R.drawable.error)
    .placeholder(R.drawable.placeholder)
    .transform(PaletteTransformation.instance())
    .into(holder.wall,
        new PaletteCallback(holder.wall){
            @Override
            public void onSuccess(Palette palette){
                holder.progressBar.setVisibility(View.GONE);

                if (usePalette) {
                    if (palette != null) {
                        Palette.Swatch wallSwatch = palette.getVibrantSwatch();
                        if (wallSwatch != null) {
                            holder.titleBg.setBackgroundColor(wallSwatch.getRgb());
                            holder.name.setTextColor(wallSwatch.getTitleTextColor());
                            holder.name.setAlpha(1);
                        }
                    }
                }
            }
        }
    );
}

```

- **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">
```

- **Creating your JSON file.**

Content:

JSON Array name = "wallpapers"

JSON Object names:

name="nameofyourwall"

author="nameoftheauthor"

wall="urlofthewall"

There's an example at DEV/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.

- **Providing Muzei support.**

Go to com.jahirfiquitiva.paperboard.muzei

Open ArtSource.java

Look for the line:

private static final String ARTSOURCE_NAME = "PaperBoard";
and replace with the Artsource name you wrote in strings.

Also, look for the line:

private static final String JSON_URL =
"https://raw.githubusercontent.com/jahirfiquitiva/MinDesigns-Wallpapers/master/JSON-Files/walls_test.json";

and replace with the URL for your Json file. (The same you put in strings.xml).

- **Enabling/disabling use of Palette API in Wallpapers Section:**

Go to com.jahirfiquitiva.paperboard.adapters

Open WallsGridAdapter.java

And look for the line

```
private boolean usePalette = true;
```

set it to true if you want to use the Palette API, or false if you don't want to.

```
public class WallsGridAdapter extends BaseAdapter {  
  
    private final ArrayList<HashMap<String, String>> data;  
    private final Context context;  
    private final int numColumns;  
    private boolean usePalette = false;
```

- **How to add or delete tabs in Icons section:**

Go to com.jahirfiquitiva.paperboard.fragments

Open PreviewsFragment.java

At the end you will find this:

```
@Override  
public Fragment getItem(int position) {  
    Fragment f = new Fragment();  
    switch (position) {  
        case 0:  
            f = IconsFragment.newInstance(R.array.latest);  
            break;  
        case 1:  
            f = IconsFragment.newInstance(R.array.system);  
            break;  
        case 2:  
            f = IconsFragment.newInstance(R.array.google);  
            break;  
        case 3:  
            f = IconsFragment.newInstance(R.array.games);  
            break;  
        case 4:  
            f = IconsFragment.newInstance(R.array.icon_pack);  
            break;  
        case 5:  
            f = IconsFragment.newInstance(R.array.drawer);  
            break;  
    }  
    return f;  
}
```

Each case is a section. Cases must always start with 0 (zero).

Use the same code to create a new section, just replace `R.array.xxxx` for the array you will use there. (The array for the icons you will show there.)

To delete a section you can just delete the code:

case X:

```
f = IconsFragment.newInstance(R.array.xxxx);  
break;
```

Cases must always be the exact number of the sections you wrote in tabs array, otherwise, as I said before, you will get a FC.

- **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 something, 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. ;)