Material Palette

GitHub Username: Daxgirl

Introduction

The truth is, I have been in love with material design since the first images flooded the web. Even before the support design library became available, before the first material design guides popped up here and there, I was utterly fascinated by the concept. My only issue was with available resources for choosing material palette. Our world is full of colors, and yet all the resources available online for creating material palette are either limited with choosing basic colors or are not quite intuitive.

I am no web developer (yet), so I can't have my web engine to produce material scheme for any chosen color (i.e. calculating the proper value of primary dark color based on any primary color). So, awhile back, I have created a little android application for my own usage that can create a simple basic material palette from any color.

Since we don't do development on our android devices (pity…), the application has a function to extract colors.xml file from the created palette and share it through mail, upload to dropbox, drive and so on.

For me it was one of those little projects that are my little piece of heaven. Something extremely useful. Something I made simply because I needed it and I COULD.

As my final project for the Udacity Nanodegree I would like to productionalize that concept and make it available on Google Play for other devs.

Description

This application allows you to create material design palette for your project, by using simple color picker or loading colors from your favourite images on your device or on Google Drive.

Once you have chosen your colors, a preview of basic android app interface is presented to you (dynamically updated as you choose colors), and should you choose to save it, I will cache your palette locally on your device, on Google Drive, or both.

In addition, you will be able to send your colors.xml file using sharing functionality, to your email address, to your dropbox, to any other cloud available on your device. You can also share your palette preview with your fellow developers, contractors or friends to ask their opinion.

I will create palettes on device or Google Drive using the explicit name of your choice for each palette, so you can reuse them later, and yet the file inside each folder on Drive will remain colors.xml, with proper standard format so you can just copy/paste it to your android studio project and have your favourite colors available to you. Just as simple as that.

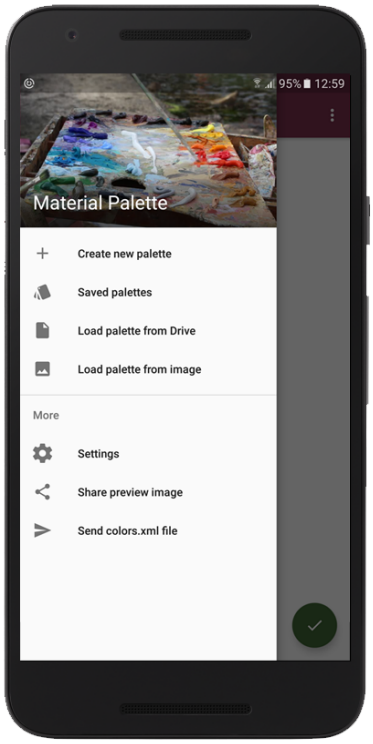
Intended User

Developers for Android Platform

Features

- Creates basic Material Palette based on user preferred colors
- Uses color picker or loads palette from existing image
- Sets dynamic preview of the interface mocks using chosen colors
- Allows the user to share the preview, actual colors.xml file, create palette directory for keeping palettes on Drive for easy access from PC, save palettes on device for future use.
- Contains option for Collection Widget on home screen with Grid view of palette preview images. Where clicking on a preview will open the app and load the preview into editor

User Interface Mocks

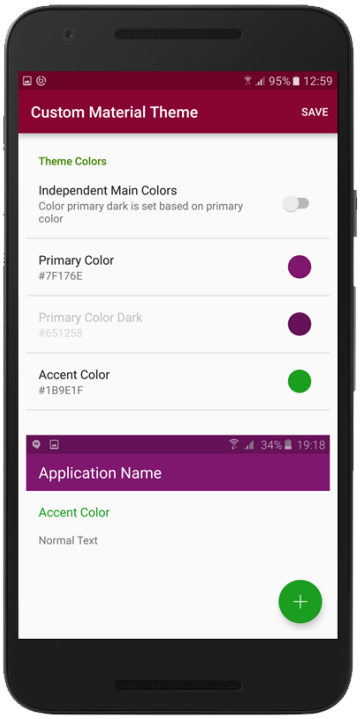


Screen 1 - Main View

This is the main interface. For small devices the application implements Navigation Drawer strategy, allowing intuitive navigation between creating new palette, loading palette from database on device, using Content Provider, loading one of the palettes saved to Drive (using distinct custom content in metadata, created solely using this app, so no mixup can be done), or loading color palette from user defined image on the device, using GET_CONTENT intent and Palette library.

In the secondary menu of the navigation drawer the user is given options to configure the app, share preview image view share content intent or send actual colors.xml file through any application allowing sharing of “application/xml” mime type, i.e. email, dropbox, Drive, Mega, OneCloud, etc.

For larger devices dual panel strategy will be implemented.



Screen 2 - Palette Editor

This is an actual screen from my existing application, displaying the main palette editor view. This is the basic idea. The preview on the bottom part of the screen (which I might move to the upper part), is changing dynamically once a color is selected. Primary dark color is being calculated according to Google guidelines based on primary color.

Key Considerations

Data persistence on device:

Palettes saved on device will be stored in sqlite table, which can be queried from and written into using Content Provider. The database will contain single table, with id field(int, auto incremented), 3 hexadecimal string color fields and a bitmap field (saved as blob) for providing preview. When loading from database the user will be given a RecyclerView populated by preview images, for intuitive and easy presentation. CursorLoaders will be used to load data into views.

Data persistence on Drive:

Provided user's agreement to enable connection to Drive, I will use Google Drive API for android to store palettes on Drive. I will create my app's specific folder on Drive and add our custom properties to its metadata. In the main folder I will create subfolders based on user's input of preferred name. Inside each folder I will create 2 files: colors.xml file and png image with preview for that palette.

Once chosen to load palette from drive, user will be taken to choose a palette using newOpenFileActivityBuilder in IntentSender. I will add filters for choosing only colors.xml files and only APP's colors.xml files. Upon getting input stream I decode it and read from it, using BufferedReader and extract the info I need for displaying and previewing the palette. Palette later can be modified and saved back to drive and/or the database on the device.

Corner cases in the UX

The main idea is to have fragments being replaced inside fragment container inside main activity, based on navigation choices. I will implement concise system of callbacks between fragment and activity, through variety of our custom interfaces and existing tools in android sdk. Be that android's onActivityResult or custom OnFragmentCallbackListener.

Main colors currently in use will be saved in shared preferences and dynamically utilized for updating views by implementing a OnSharedPreferencesChangedListener.

Compiled libraries

1. **com.android.support:appcompat-v7:23.2.1** including utilizing vector support
2. **com.android.support:cardview-v7:23.2.1** to display items in recycler views for choosing colors or previews from the database
3. **com.android.support:recyclerview-v7:23.2.1**… obviously
4. **com.google.android.gms:play-services-drive:8.4.0** to provide connection to user's Drive contents
5. **com.google.android.gms:play-services-ads:8.4.0** frankly speaking (very frankly), mainly because you guys requested we use at least 2 google services and for my app I only need one. So I will add interstitial ad support for the free flavor.
6. **com.android.support:palette-v7:23.2.1** to extract swatches from existing image of user's choice
7. **Daniel Nilsson's and Sergei Margaritov's ColorPickerPreference library**, as android library module included in project.
8. Last but not the least… **com.android.support:design:23.2.1** for UI elements and coordinator layouts

Required Tasks

Task 1: Project Setup & Implement UI for Each Activity and Fragment

- Creating main UX for small device
- Creating all layouts and implementing main necessary libraries
- Adding mock navigational steps
- Making sure all callbacks are functional within preliminary parameters

Task 2: Setting up dynamic UI and view's interactions

- Implementing listeners for preference change and updating UI
- Finalizing all fragments and child activities with inits and new instances where necessary

Task 3: Creating local data persistence utility

- Creating Content Provider
- Creating SQLite helper class and database contract
- Adding cursor loaders to classes that require it

Task 4: Adding functionality to extract swatches from image

- Creating activity that will be started for result to display chosen image and it's color palette. Where users can choose color and set them as result and return to main activity to populate the palette editor view and preview.
- Implement all required callbacks and data persistence.

Task 5: Connect to Google Drive

- Set up conditions for connections.
- Create and use GoogleApiClient with Drive.API with necessary callbacks and data utilization and parsing.

Task 6: Unit and connected tests if needed for asynchronous data loading

Task 7: Utilize sharing engine

- Set up sharing intents for sharing preview image
- Set up sharing intent for sending colors.xml file

Task 8: Create widget

- Create widget required elements and bind views
- Set up widget updating concept

Task 9: Flavourize and Finalize

- Add free and paid flavours
- Compile google ads service only to free flavor
- Create signing key
- Test release build task
- Done