**Building Android Apps — 30 things that experience made me learn the hard way**

There are two kinds of people — those who learn the hard way and those who learn by taking someone’s advice. Here are some of the things I’ve learned along the way that I want to share with you:

1. Think twice before adding any third party library, it’s a **really** **serious**commitment;
2. If the user can’t see it, [**don’t draw it**](http://riggaroo.co.za/optimizing-layouts-in-android-reducing-overdraw/)!;
3. Don’t use a database unless you **really** need to;
4. Hitting the 65k method count mark is gonna happen fast, I mean really fast! And **[multidexing](https://medium.com/@rotxed/dex-skys-the-limit-no-65k-methods-is-28e6cb40cf71" \t "_blank)**[can save you](https://medium.com/@rotxed/dex-skys-the-limit-no-65k-methods-is-28e6cb40cf71" \t "_blank);
5. [RxJava](https://github.com/ReactiveX/RxJava) is the **best** alternative to [AsyncTasks and so much more](https://medium.com/swlh/party-tricks-with-rxjava-rxandroid-retrolambda-1b06ed7cd29c" \t "_blank);
6. [Retrofit](http://square.github.io/retrofit/) is the **best** **networking** **library** there is;
7. Shorten your code with **[Retrolambda](https://medium.com/android-news/retrolambda-on-android-191cc8151f85" \t "_blank)**;
8. Combine **[RxJava with Retrofit and Retrolambda](https://medium.com/swlh/party-tricks-with-rxjava-rxandroid-retrolambda-1b06ed7cd29c" \t "_blank)** for maximum awesomeness!;
9. I use [EventBus](https://github.com/greenrobot/EventBus" \t "_blank) and it’s great, but I **don’t**use it too much because the codebase would get really messy;
10. [Package by Feature, not layers](https://medium.com/the-engineering-team/package-by-features-not-layers-2d076df1964d);
11. Move *everything* off the application thread;
12. [lint](http://developer.android.com/tools/help/layoutopt.html) your views to help you optimize the layouts and layout hierarchies so you can identify redundant views that could perhaps be removed;
13. If you’re using *gradle*, speed it up anyway you [can](https://medium.com/the-engineering-team/speeding-up-gradle-builds-619c442113cb);
14. Do [profile reports](https://medium.com/the-engineering-team/speeding-up-gradle-builds-619c442113cb) of your builds to see what is taking the build time;
15. Use a [well known](http://fernandocejas.com/2015/07/18/architecting-android-the-evolution/" \t "_blank) architecture;
16. [Testing takes time but it’s faster and more robust than coding without tests once you’ve got the hang of it](http://stackoverflow.com/a/67500/794485);
17. Use [dependency injection](http://fernandocejas.com/2015/04/11/tasting-dagger-2-on-android/) to make your app more modular and therefore easier to test;
18. Listening to [fragmented podcast](http://fragmentedpodcast.com/) will be great for you;
19. [**Never** use your personal email for your android market publisher account](https://www.reddit.com/r/Android/comments/2hywu9/google_play_only_one_strike_is_needed_to_ruin_you/);
20. **Always** use [appropriate](http://developer.android.com/training/keyboard-input/style.html) input types;
21. Use **analytics** to find usage patterns and isolate bugs;
22. Stay on top of new [libraries](http://android-arsenal.com/) (use [dryrun](https://github.com/cesarferreira/dryrun" \t "_blank) to test them out faster);
23. Your services should do what they need to do and **die** as quickly as possible;
24. Use the [Account Manager](http://developer.android.com/reference/android/accounts/AccountManager.html) to suggest login usernames and email addresses;
25. Use **CI**(Continuous Integration) to build and distribute your beta and production .apk’s;
26. Don’t run your own **CI**server, maintaining the server is time consuming because of disk space/security issues/updating the server to protect from SSL attacks, etc. Use circleci, travis or shippable, they’re cheap and it’s one less thing to worry about;
27. [Automate your deployments to the playstore;](https://github.com/Triple-T/gradle-play-publisher)
28. If a library is massive and you are only using a small subset of its functions you should find an alternative **smaller** option (rely on [proguard](http://developer.android.com/tools/help/proguard.html" \t "_blank) for instance);
29. Don’t use more modules than you actually need. If *that* modules are not constantly modified, it’s important to have into consideration that the time needed to compile them from scratch (**CI**builds are a good example), or even to check if the previous individual module build is up-to-date, can be up to almost 4x greater than to simply load that dependency as a binary .jar/.aar.
30. Start [thinking about ditching PNGs for SVGs](http://developer.android.com/tools/help/vector-asset-studio.html);
31. Make library abstraction classes, it’ll be way easier to switch to a new library if you only need to switch in one place (e.g. ***AppLogger.d(“message”)*** can contain ***Log.d(TAG, message)*** and later realise that ***[Timber.d(message)](https://github.com/JakeWharton/timber" \t "_blank)*** is a better option);
32. Monitor connectivity and type of connection (**more** **data** **updates** while on **wifi**?);
33. Monitor power source and battery (**more data updates** while **charging**? **Suspend updates** when**battery is low**?);
34. A user interface is like a joke. If you have to explain it, it’s not that good;
35. [Tests are great for performance: Write slow (but correct) implementation then verify optimizations don’t break anything with tests](https://twitter.com/danlew42/status/677151453476032512).