Skyrocketing Android with Kotlin

Fully embracing advanced Kotlin features to create a scalable and modern Android application

Filip Babić

1st Edition

1

Intro

The idea of this book is to provide a well documented and thoroughly explained process of creating an advanced Android application using intermediate Kotlin features and language constructs.

This is not a toy project or some code whipped up together just to work and show off. It's a full process, from the debate of why use Kotlin, and not Java, to finer things in Kotlin like using extensions and generics to build your own DSL-like syntax for some Android specific things.

Every single feature, library, plugin and construct used in this book is what I personally favor currently in Android development, and the current Kotlin ecosystem. I won't enforce you to use these tools if you prefer something else, but be advised that I won't provide examples for every single tool there is, I'll stick some specific tools.

It is my only wish that you fall in love with Kotlin as much as I did, and that you broaden and strengthen your Android development knowledge while reading this book, so you may once refer it to others as a good learning resource.

Getting Started

You must have heard of all of Kotlin's awesomeness in the past year or two since it's been out, and you might familiar with it's syntax, or you may know a few things but haven't quite started learning it.

Everything we'll cover will be elaborated extensively, but I'd advise you, if you aren't already familiar with Kotlin, to go over Kotlin's documentation[1], and the Kotlin Online Koans[2] examples, which will give you a rough image of what this beautiful language can do.

Since the topics we'll be covering are intermediate to say the very least, if you do get lost, be sure to check a few things before fully delving into this book. Check out Kotlin in Action[3] and Kotlin for Android Developers[4] by Antonio Leiva[5] books

---

[1] https://kotlinlang.org/docs/reference/ - Kotlin official documentation

[2] https://try.kotlinlang.org/ - Kotlin online console and compiler with practical examples

[3] https://www.manning.com/books/kotlin-in-action - Kotlin in Action book

[4] https://antonioleiva.com/kotlin-android-developers-book/ - Kotlin for Android Developers book

[5] https://antonioleiva.com/ - Antonio Leiva's website

This book's lifecycle

Wow, we haven't even gotten to the App, and I'm already mentioning Lifecycle? Don't worry, this isn't the pain-in-the-ass Android Lifecycle, but just an overview of what I've divided this book into.

Visualising whatever you're trying to learn and combining previously gained knowledge makes you understand and remember it far better. This is great when you're building an application, because you can visualize whatever you build and reuse a lot of things learned along the way. This is why this book will be followed by a full working project rather than plain simple examples.

You'll go through the tough task of creating a Social network, something you could really be proud of and something you can easily add to your portfolio.

You'll see every single detail of what it takes to build such an app, from the concept and organizational process to the code implementation. The process, or rather the methodology you'll be using is Agile development[6].

Being agile

Agile is an iterative form of building products, in which you work in time periods of *sprints* which last about two to four weeks. Each sprint you take a certain set of epics and stories which you want to complete. An epic represents one solid feature in the application, like *user authentication,* where each story is a jigsaw piece within that epic, like *login*, *registration* and *social login*.

The idea is not to plan everything ahead, but rather be agile! React to feedbacks, developer insights, client wishes and so on. If you develop using this approach, you don't have to think about everything at once and even if you stop somewhere in the middle of development, you'll have a working version of stories up to the point of stopping.

Software companies use this approach because it's easier to bugfix only a sprint at a time, you can test each iteration of the application with a user group and get invaluable feedback and it's cheaper to change or even cancel a project if you're not satisfied with the results up to that point.

Agile however is not only a way to optimize the development process. It also focuses a lot on building applications that your users will love. The goal is not to build what you want to have in the app, but to create a general concept of what you want to achieve, research the market and the personas that would use your application and sculpt the features according to their needs or suggestions.

After all, we are researching *user* stories, not *developer* stories. :]

---

[6] https://www.qasymphony.com/blog/agile-methodology-guide-agile-testing/ - Agile manifesto

Audience level

It wouldn't really be logical if I was writing the book just for myself, right? I really do hope many a person choose this book as their next source of information of all things Kotlin.

However, given my explanation of what you'll be building you can realise I haven't written a tutorial, or an article on a cute little Kotlin language feature I've just found out about. I've written a book filled with hardcore stuff like generics, mapping operators, coroutines and monads, and I expect of you hold on for dear life.

You *will* be annoyed, frustrated, going crazy, bloodthirsty even when I drop crazy ass functions for networking who have a bunch of these "<T : Mappable<R>, R : Any>" in their signature. If you don't get annoyed, and figure them out momentarily, I'll be even happier knowing you, yourself, are badass (yes I'll know because I'm watching you while you're reading).

The reason I've chosen this type of learning is because, when I first started, I had a very crazy way of learning things. I was literally thrown in fire (as we say in Croatia) and I had to find my way through the smoke. Which is what I want to do with you.

Though this might seem weird at first, it really was the quickest and best way to learn for me. A good developer will find his own way around, without much help, and he'll become even better for it. A great developer will do the same, but he won't stop at that, he'll continue to teach others how to do it (yes, I'm implying I'm great).

2

Brainstorming our App

Let's socialize

Most of you have probably used a Social network, but for those of you who haven't it's a platform on which people can communicate in one way or another. Usually there is a place on which you can post your thoughts, memories, life events and questions - either by creating a thread or by posting an image or a textual post.

The concept is pretty straightforward, try to entice your user-base to generate new content daily, in order to gain new followers, likes/hearts, shares and comments on their posts. This way people could always come to the application and see something new in the feed or on their posts.

Why a social network

Social networks are a really popular thing nowadays. Most developers think each has it's own enormous codebase which is composed of spaghetti code, and that only tough veterans can do anything productive in such an environment.

I'm not really buying it, because even if that were true, each network had to start somewhere. I believe anyone could create their own social network, which you'll see by following this book!

I've personally never built a social network (up until now), so this will be something new for me too. But I wanted it to be this kind of app because it has many components, it's extensible and you can always go back to it to experiment with something new, like a new architecture pattern, new networking paradigm and so on.

What will it do?

As previously mentioned, you'll be able to create posts shown on a global feed and like other people's posts. You'll have some form of communication with other users, a profile which other users can check out if they want to get in contact with you, and whatever else I think would be a good feature to implement on the way.

Since there are nearly infinite possibilities, I wouldn't plan absolutely everything ahead, let's try to go step by step and see what comes out!

The first door is the hardest

You'll be learning how to use the aforementioned Agile methodology to achieve great UX (User experience). So your first step should be to take a look at what a social network is composed of, and try to derive user stories from that. Each user story will begin with: "As a user I want to... so that I can..."

An example user story would be: "As a user I want to to log in so that I can be recognized as a unique individual." Try to write down a few yourself, before heading to the next section. I'll try to walk you through this process as I would do it, but try to write down a few .

We have to build for someone

We've already said that each Social network needs users - therefore we need an Authentication component (epic). Now you'd say it's easy, we have Register and Login stories, we've built them for so many projects (if you have some experience).

But what I want you to understand is that most people will want an easy path within your application, so that in only a few clicks they could participate in the full experience. With this in mind, a regular register/login screen would take 3-4 steps for the user to participate, could we reduce this somehow? //TODO add some illustrations here

Most applications have integrated other Social networks' login for this reason only. Which is why we will do too.

Connections all around

It's great UX (User experience) to have a quick log in option by using accounts from other Social networks. Be it Google Sign In or Facebook Login, or more often both, it's an amazing way to draw users to use your application.

Think about it, if you had the option to fill in a ton of forms, confirm your e-mail, and only then enter the app, or the option to click a button and choose an account, which one would you take?

For navigation, you'll have social login on a welcome screen, so that the first entry point is available as soon as possible.

Welcoming our guests

Most applications welcome their users by showing a short splash screen depicting their product's logo and core theme. It's a great way to leave a first impression by using live colors and a logo by which your application will be recognized.

After the splash we have to show something to our Users. A compact welcome screen, with all the entry points into the core application is required. Regular authentication and social authentication will take place here because we want to let the user choose how he'd want to join the community, rather than providing only one type of login.

Another thing you'll have to consider now is the fluidity of the App. If your application has unclear navigational flows or a repetitive process, your users won't be too happy using it. For example if a user has to authenticate every time he opens the app, they will quickly be annoyed and leave negative reviews on the Play Store.

With that said, once the User is authenticated, there is no reason to show a Welcome screen, you should simply show them the feed, so they don't have to log in each time they open your app. A small thing, albeit important to the users.

Restraining order

Notice how we've gone through one component of the app without mentioning the case when a user doesn't want to sign up. Simply put, it's a huge effort to block entire features of your application when you're not authenticated, which is why I've decided to only let the user use the app if he's signed up.

Feeding our users

From an economic perspective, each application has to have something that'll make users want to use it. People just won't use your app because it's something not so much different from the competition. Each app has to have a "killer feature" to stand out from the vast sea of similar applications. Something that will draw users to use your app. Like Instagram[7] did with its "stories".

But because I'm not creating an app that each of you will be able to sell and make a living off, this Social network won't exactly have a killer feature, but the concept stays the same. Just make sure that if you're building a product, you have that one thing that differentiates you from everyone else!

Since your users will need a reason to use the app, you will build a feed where all the new things people post will be visible and you could explore new things. You can like posts as previously mentioned, and leave comments on them.

Improving the posts

//TODO add illustrations

As before, you want to improve the experience for the user. If you've been a part of some Social network, you know how tedious it can be to find a post so you can show it to your friends. Having a favorites page can simplify this process a lot.

Furthermore, having a page just for your posts, so you can see their progress (number of likes) and so you could delete them if you want is also something that's crucial. This covers the posts section of feed, so you can move on to writing user stories.

Try to come up with the user stories that best fit these features yourself, before checking what I came up below.

The stories I believe best describe what the users want, are next:

- "As a user I want to be able to preview posts from other people, so I can find interesting topics."
- "As a user I want to be able to like interesting posts so I can browse them later."
- "As a user I want to be able to leave comments on posts, so I can communicate with other people."
- "As a user I want to be able to preview my posts."
- "As a user I want to be able to delete my unwanted posts."

---

[7] https://www.instagram.com/ - Instagram application website

Notice how I split up the my posts page into two stories, one to view your posts and one to delete them. If you come up with a story that seems to have two small tasks, it's best to split them up and work on them separately.

What else do we need to wrap up feed? We will work on lazy loading (loading items page by page rather than all at once), refreshing, updating lists when liking posts and so on, but all of that will be covered deeply in the implementation part, for now we're just conceptualizing.

Now, we keep talking about the users, but we haven't said how they'll interact. I know it seems that it's not a programming book so far, but from my experience, coding is just about 5-15% of a programmer's job, the rest is talking, planning, discussing and brainstorming! Let's move to the profile feature.

Profiling

Each user likes to feel special, which is why they need their own section in an application. By having a user profile we give them an overview of their current status in the community (number of collected likes, number of posts). Furthermore we give them a way to check out other Users as well, by looking at their public profiles.

We've come to a new possibility now, should we add a "Friends" section as well? Why not! Since we will allow users to communicate and look at their profiles, you'll also add a Friends section for them to be able to stalk people they like.

On the profile we will give the user a way to change their names and profile images, passwords (if they have any), and to log out or disable their account. But more on that later when we implement it.

One last thing left to think about is feed details and the comments section, which we will do in the next section. It's really important to structure your features into logically separate units, so you can finish them one by one, without breaking the rest of the app.

Details matter

Having just posts, without the ability to see them with a detailed description feels incomplete. Even more so because we're adding comments. You don't want to scroll endlessly through a feed of comments, you want to open up a post and then see what it's about, and comment if you like what you see.

When creating a post, you'll add the ability to write a detailed description, if there would be any, and to add a picture for the post. This way the app won't be a huge wall of text, the users will be able to see images describing posts before reading them.

Any comments?

Comments will be a vital part of each post's details page, because the users can express their opinion for each post. For the sake of simplicity, you'll leave the comments to be textual only, having tons of images in the comments section will make you scroll endlessly to find what you're looking for.

We can make this quite easy, and Facebook-like. Each person can comment, but you won't have nested comments. Often it is best to stick to simplicity to provide the best experience for your users. If you overcomplicate the code, it only affects your team. But if you overcomplicate how the application works, it affects every single user.

Conclusion

It's very easy to brainstorm all the features and what you want to see, but it's important not to think too much ahead. Overthinking, or rather over-engineering, is a key step to failure and process breakdown. Once you start going too far ahead, you'll overcomplicate even the simplest things, end up with tunnel vision and lose productivity, and your users won't be happy with the app complexity.