The Best Way To Structure Your iOS Project

Learn how to keep it clean

Artiom Khalilyaev · Follow

Published in Level Up Coding · 5 min read · Apr 12

77 1

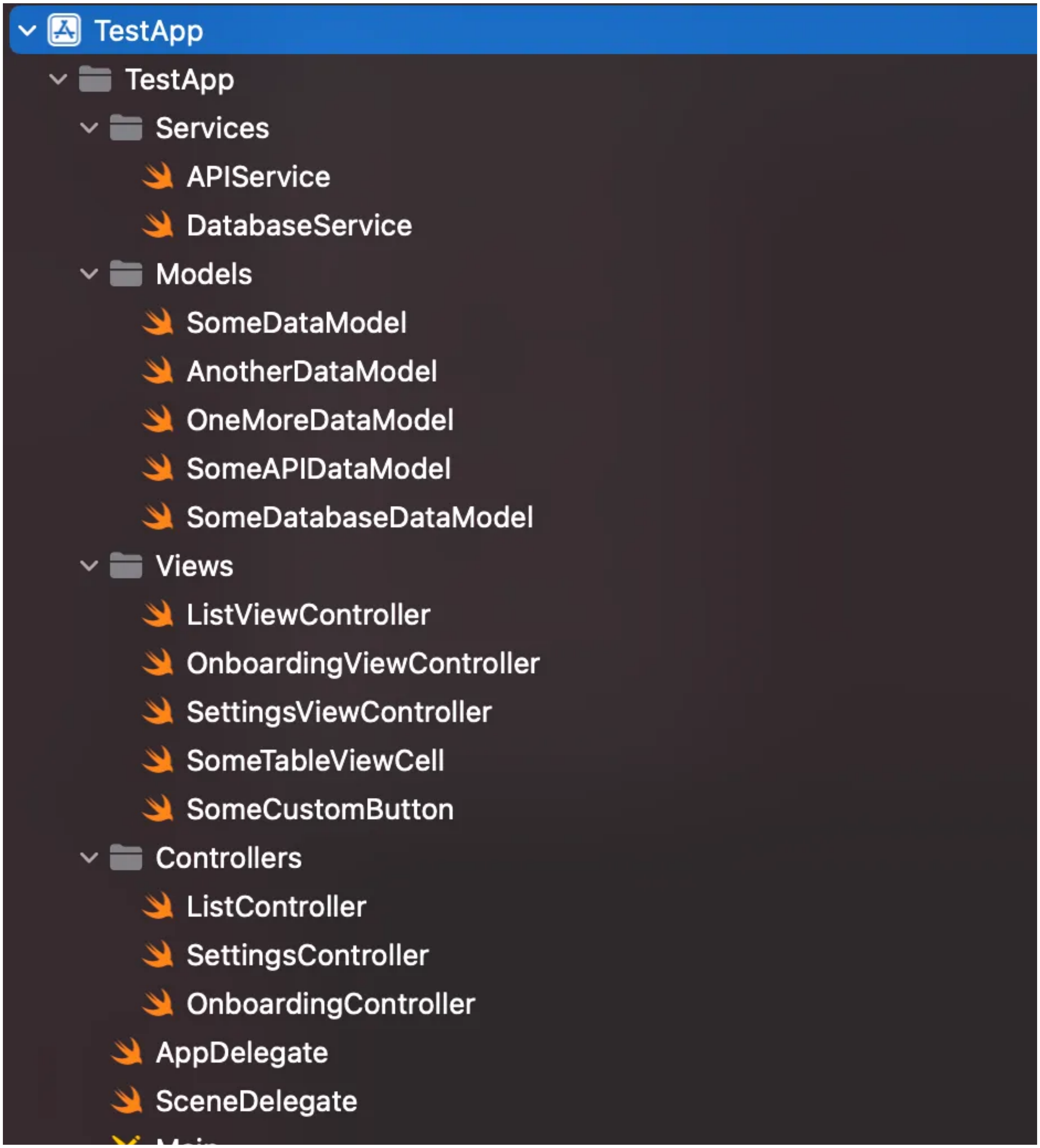
Photo by Viktor Talashuk on Unsplash

It's important to keep your project structure clean and organized. When you are working on a big project with hundreds of files in a team, you want you and your teammates to be able to find anything you want within a few seconds. The project needs to be organized from the very beginning and everyone in the team should follow the structure you have, because some devs may leave and new may join your team.

In this article, I'm going to show you some common mistakes that junior developers do and share the way I'm structuring every project I'm working on.

The most common mistake

Let’s take a look at this short example of a project with MVC architecture:



```
TestApp
  TestApp
    Services
      APIService
      DatabaseService
    Models
      SomeDataModel
      AnotherDataModel
      OneMoreDataModel
      SomeAPIDataModel
      SomeDatabaseDataModel
    Views
      ListViewController
      OnboardingViewController
      SettingsViewController
      SomeTableViewCell
      SomeCustomButton
    Controllers
      ListController
      SettingsController
      OnboardingController
    AppDelegate
    SceneDelegate
```

Everything seems to be sorted. All views are in the same folder, all controllers are together inside the `/Controllers` folder, and models are grouped in the `/Models` folder. With a project of this size, when you have 3–5 screens, it might work. But let's be realistic and imagine we work on a project with 20+ screens and have at least 20 view controllers. It will be a huge mess. You cannot just put files in the same folder only because they have something in common, e.g. inherited from the same class. Because in that case, you will end up with folders with 30+ files inside, and it's really tough to find something inside such folders.

The hallmark of a well-structured project

> Your project is well-structured if anybody who is not familiar with it fells comfortable with it.

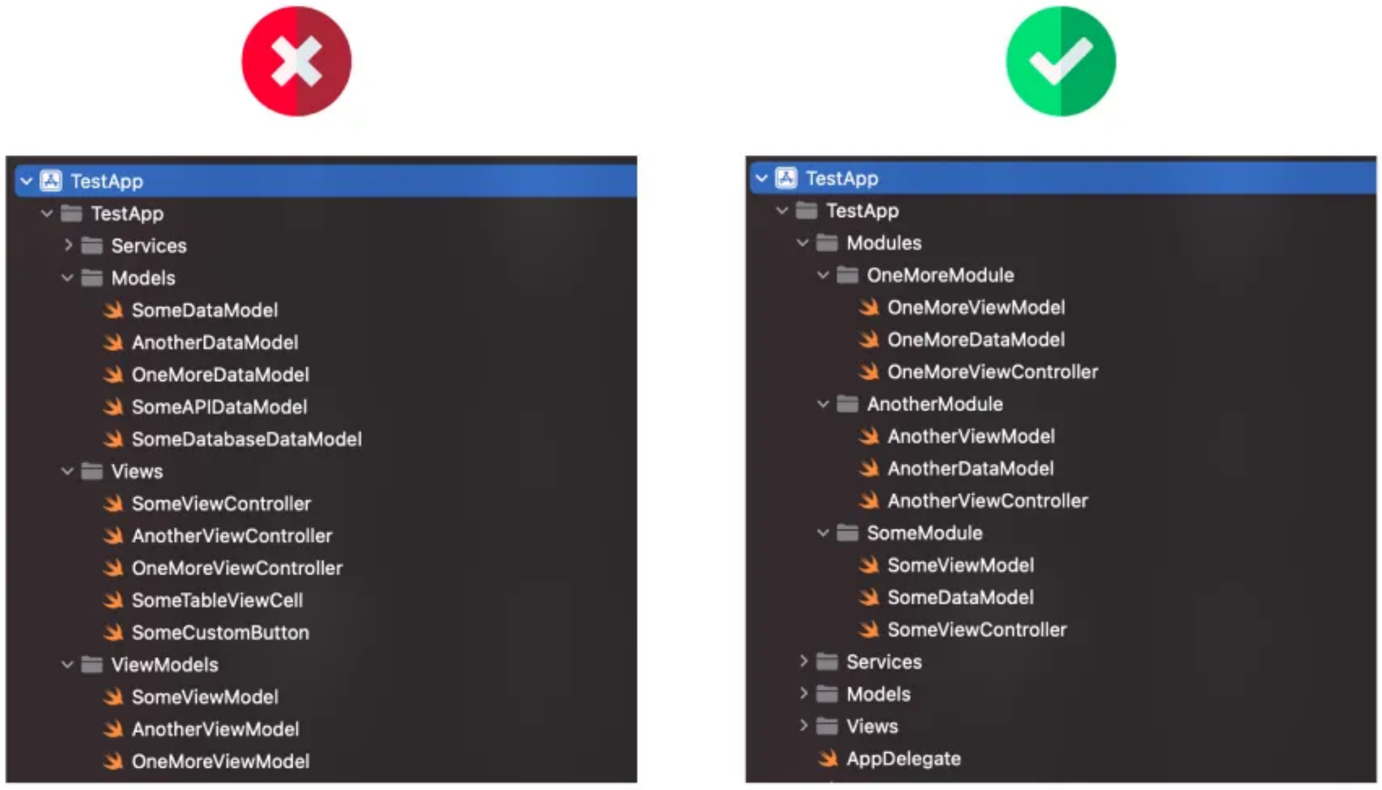
It means, that your folders and file should be organized and named, so anyone could find what they want within a few seconds. If I was fixing some bugs in a view, and then I need to do some fixes in a controller, I don't want to open a `/Controllers` folder with 30+ files inside and look for a controller I need.

Best practice

While grouping your files and folders, you should follow the following rule:

> *Keep together files that are related to each other in the first place or otherwise have something in common.*

What does it mean? Let me explain. Let's say you have a simple MVVM module, which contains view controller, view, view model, and some additional subviews. All these files should be putted in the same folder. That's because they all are related to each other, because they all are parts of a single module.



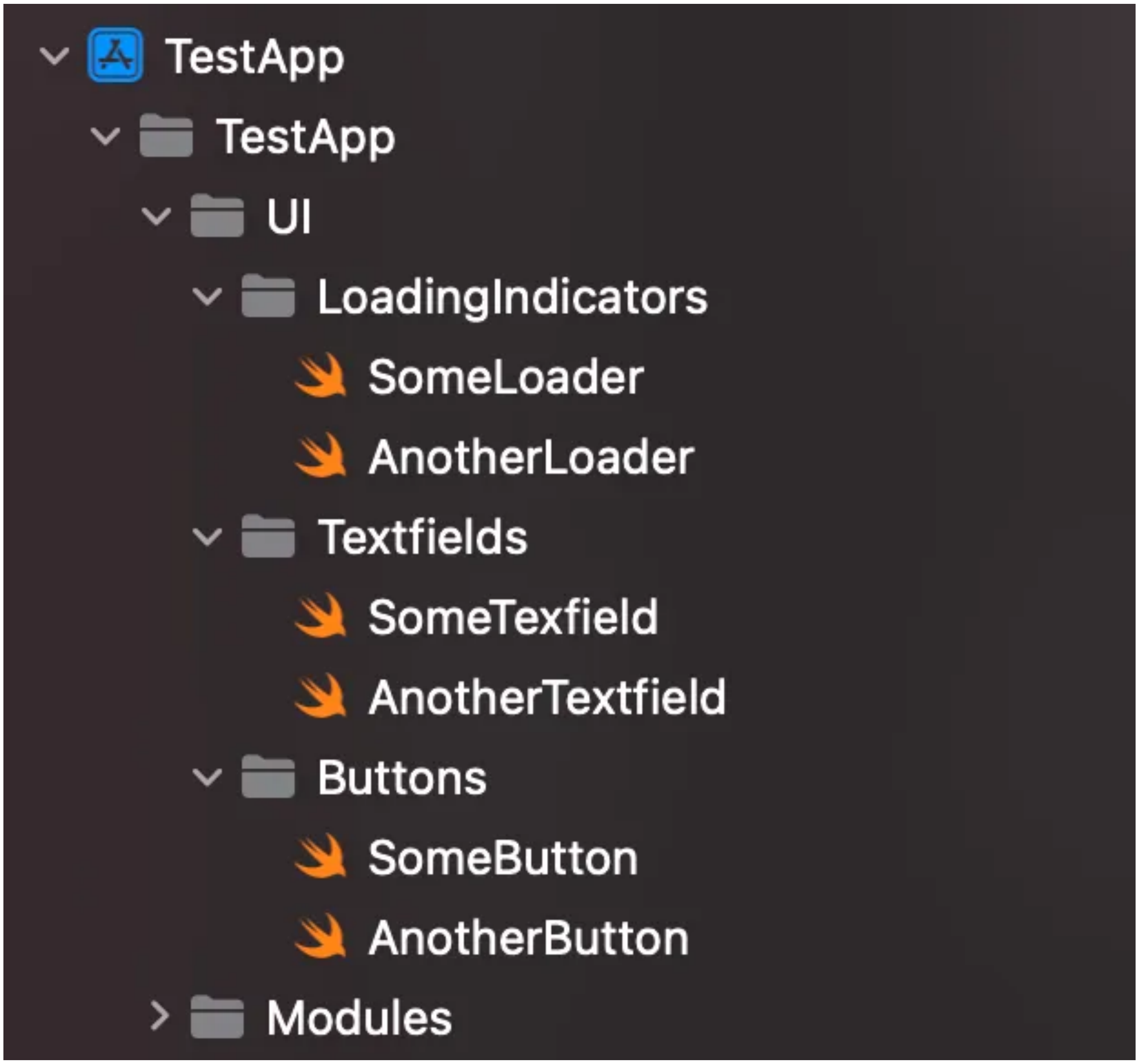
In the project at the left screenshot, all view models are in the same folder, all views are together, and the same for all models. We discussed it before, it's not the best decision to make. Now take a look at the screenshot at the right. It looks more structured. We have a folder for every module we have, those folders contain view controller, view model and a model for a particular module. Then all modules are grouped together inside the `/Modules` folder. With this organization, we know that all modules are inside the `/Modules` folder, and all files related to the particular module are inside the folder with the module name, e.g. `/SomeModule`.

Now we know how to group modules. But there are a few more things we may have in the project. They are services, custom reusable views, extensions and resources (colors, fonts, strings, urls and so on).

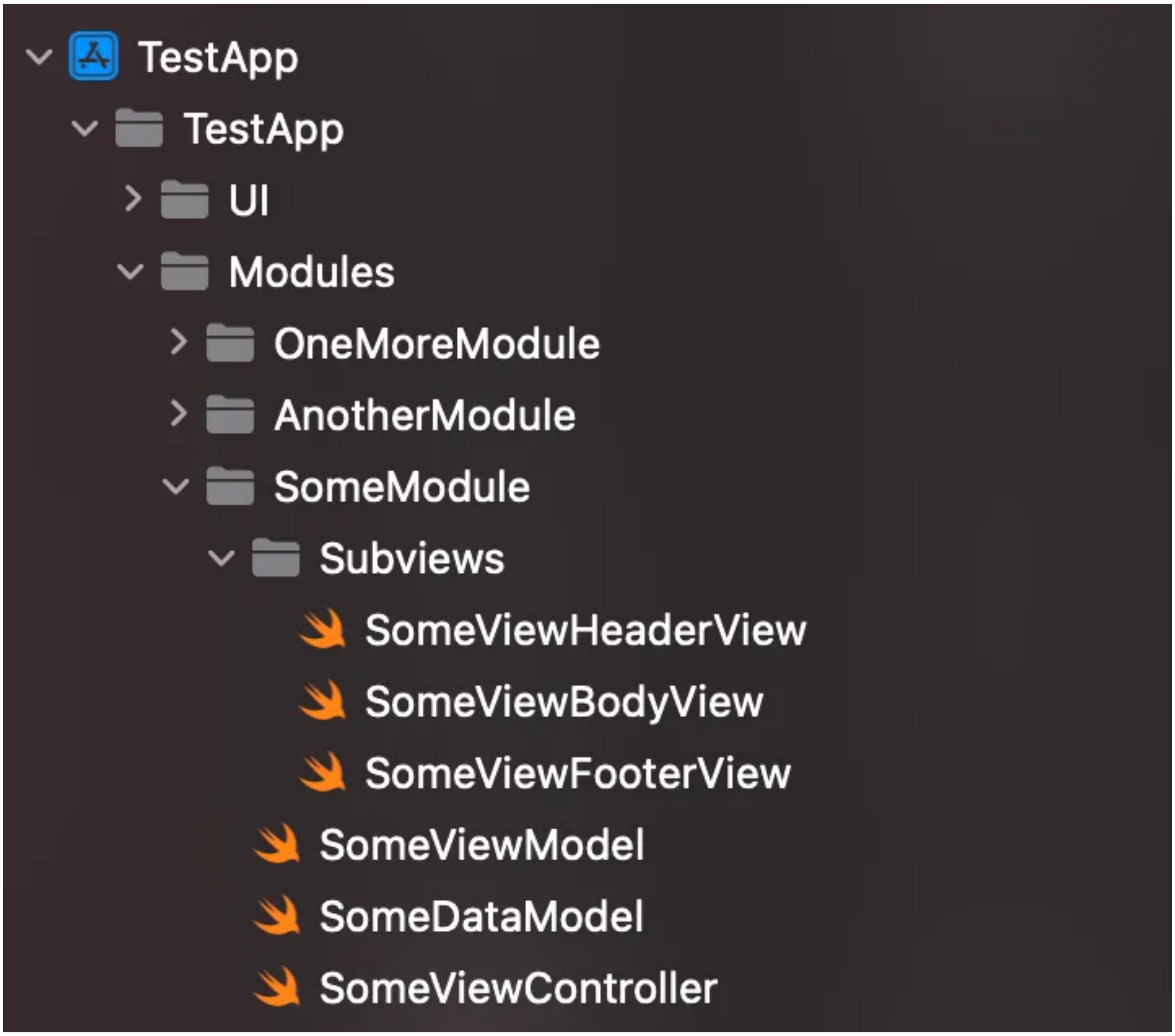
Reusable views

We may have 2 types of reusable views. Views we are going to use across the app, e.g. buttons and textfields. And views that are used only in one particular module, for example you decided to create a separate header view for one of your modules.

In the first case, all those views should be inside one folder, e.g. `/UIComponents`. And inside this folder you can group them depending on their superclasses, e.g. `/Buttons` and `/Textfields`.



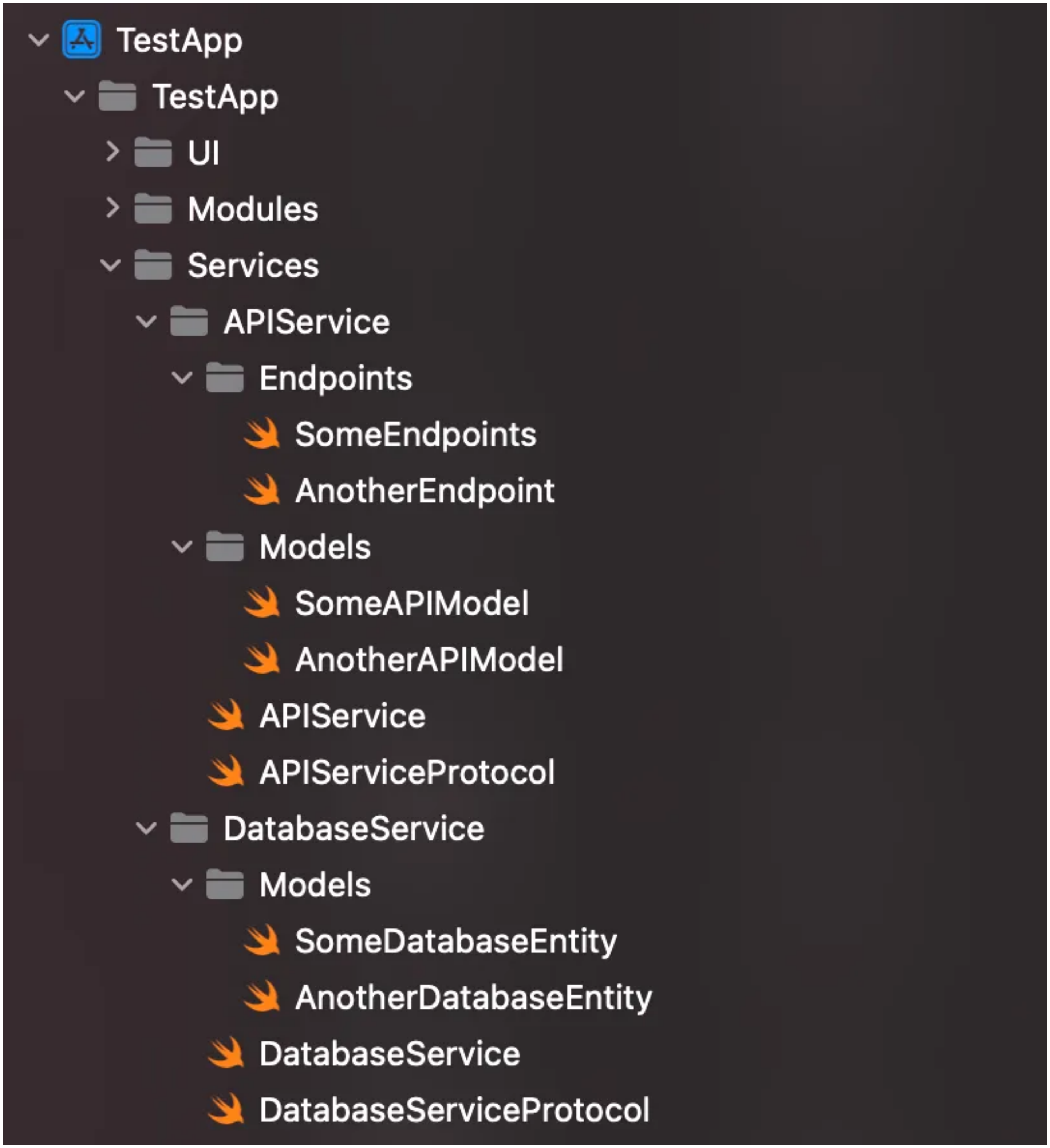
module folder in a separate folder for UI components, e.g. `/Subviews` or `/Components`.



```
TestApp
  TestApp
    UI
    Modules
      OneMoreModule
      AnotherModule
      SomeModule
        Subviews
          SomeViewHeaderView
          SomeViewBodyView
          SomeViewFooterView
        SomeViewModel
        SomeDataModel
        SomeViewController
```

Services

It’s pretty easy when it comes to services. You put all the services inside some `/Services` or `/Managers` folder. And inside the root service's folder, you create a new folder for each service you have. You create a folder for each service, since your service may have a few more files related to it. Imagine you are working on `APIService`. You need a file for its protocol, for the list of endpoints, for the DTOs (Data Transfer Objects) you are fetching and so on.

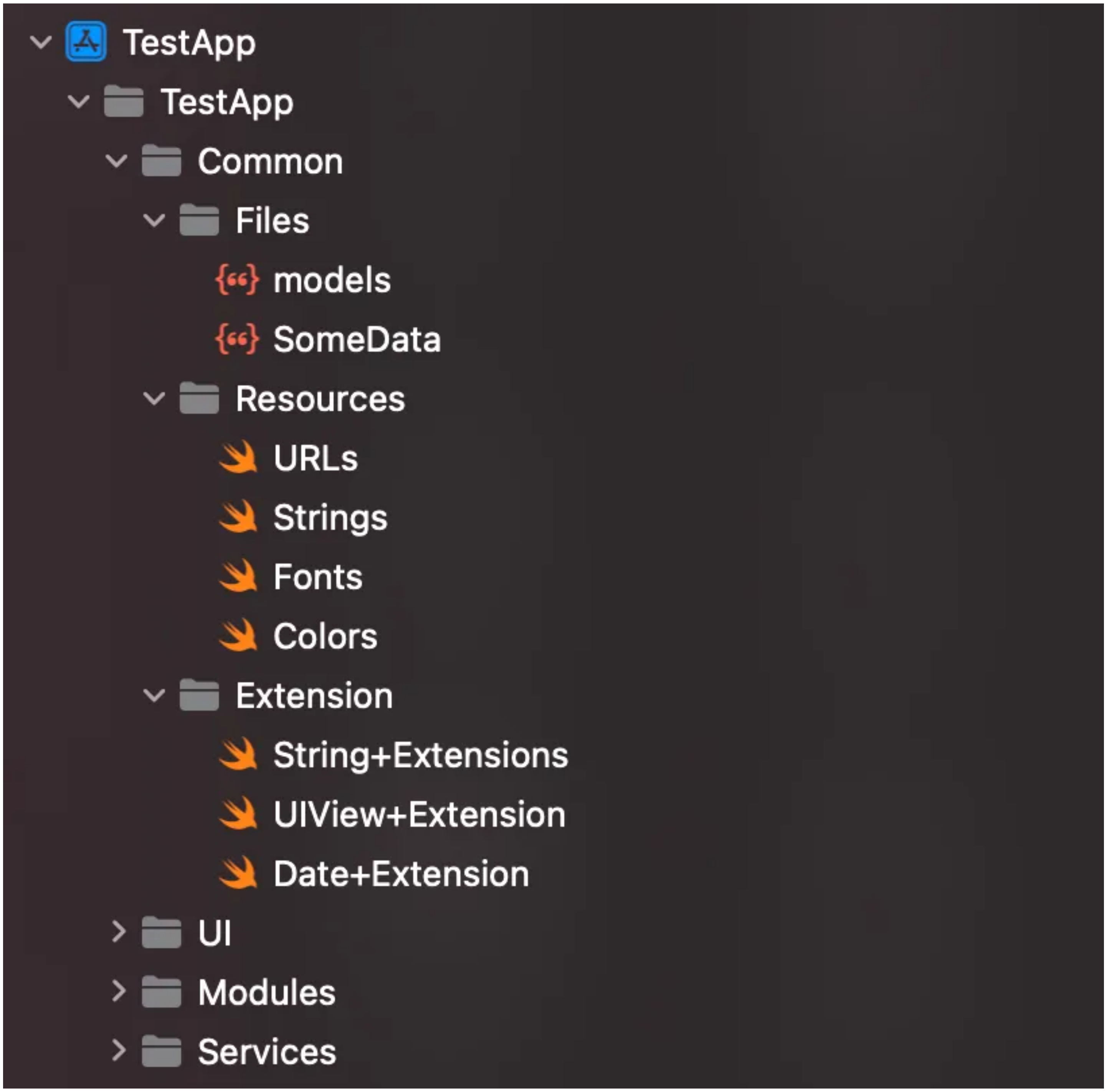


```
TestApp
  TestApp
    UI
    Modules
    Services
      APIService
        Endpoints
          SomeEndpoints
          AnotherEndpoint
        Models
          SomeAPIModel
          AnotherAPIModel
        APIService
        APIServiceProtocol
      DatabaseService
        Models
          SomeDatabaseEntity
          AnotherDatabaseEntity
        DatabaseService
        DatabaseServiceProtocol
```

Other files

I don’t like to have lots of folders and files inside the root project’s directory.

That's why I prefer to put all those files inside one folder named `/Common`, when it comes to the other stuff like resources, extensions, JSON/CSV files, etc. Inside the `/Common` folder, you create another folders for the files you have, e.g. `/Resources`, `/Extensions`, `/JsonFiles` and so on.



This approach helps to keep a root directory clean.

Conclusion

It's an important task to organize your project from the very beginning. All other devs who join your team will follow the structure you already have set up.

Thanks for your time. Hope you enjoyed this article and found it useful. Feel free to share your opinion on this topic in the comments.

. . .

```
Check out my other articles about iOS Development

https://medium.com/@artem.khalilyaev
```

Swift IOS App Development IOS Development Programming Mobile App Development

Written by Artiom Khalilyaev

Follow

102 Followers · Writer for Level Up Coding

iOS Developer, writing for BetterProgramming and LevelUpCoding

More from Artiom Khalilyaev and Level Up Coding

Artiom Khalilyaev in Level Up Coding

Separate View from the ViewController in Swift

It will improve any architecture, even MVC

5 min read · Feb 21

174 3

Sanjay Priyadarshi in Level Up Coding

I Spent 14 Days Studying A Programmer Who Built a $167 B...

Steal This Programmer Blueprint

· 11 min read · Apr 23

1.3K 16

Alexander Nguyen in Level Up Coding

Why I Keep Failing Candidates During Google Interviews…

They don't meet the bar.

✦ · 4 min read · Apr 13

3.6K 108

Artiom Khalilyaev in Level Up Coding

An Expandable UITableView Cells In Swift

Learn how to create expandable cells with ease

6 min read · Feb 28

105

See all from Artiom Khalilyaev

See all from Level Up Coding

Recommended from Medium

Imad Ali Mohammad

10 Swift Coding Tips—ONE Liner

Swift is an incredibly powerful and expressive programming language that...

3 min read · Apr 27

73

Garrett Barker

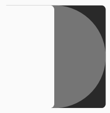
SwiftUI View Life Cycle

In this article, we will discuss the SwiftUI view life cycle and how it works.

4 min read · 4 days ago

85 2

Lists



Stories to Help You Grow as a Software Developer

19 stories · 22 saves

Leadership

30 stories · 9 saves

How to Run More Meaningful 1:1 Meetings

11 stories · 10 saves

Stories to Help You Level-Up at Work

19 stories · 19 saves

Vincenzo Pascarella

The Perfect Trio: MVVM, SwiftUI, and Combine

An Introduction to SwiftUI and Combine with the MVVM Design Pattern

7 min read · Apr 25

57 2

wordsmithwriter I Anshul

Notion's Dirty Little Secrets—Why I Kissed Notion Goodbye and...

What happened to Notion? Why didn't it live up to its hype?

· 6 min read · Apr 16

1K 22

Mammadowr

15 Interview-questions that you will be asked as an iOS Developer

Hi everybody! 😏

7 min read · Dec 21, 2022

31 5

Lee Kah Seng

How to Fetch and Show Remote Data on a Widget?

This article is originally published at swiftsenpai.com

· 5 min read · Jan 31

25

See more recommendations

Help Status Writers Blog Careers Privacy Terms About Text to speech