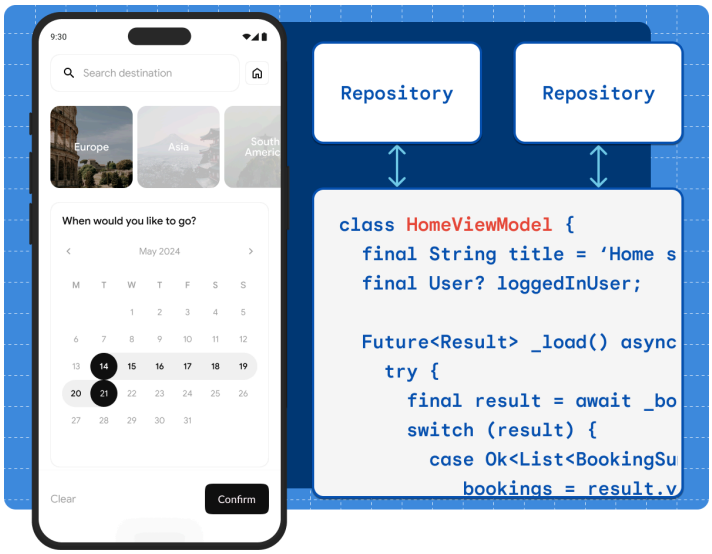
Architecting Flutter apps

Architecture

Architecture is an important part of building a maintainable, resilient, and scalable Flutter app. In this guide, you'll learn app architecture principles and best practices for building Flutter apps.

'Architecture' is a word that's hard to define. It's a broad term and can refer to any number of topics depending on the context. In this guide, 'architecture' refers to how to structure, organize, and design your Flutter app in order to scale as your project requirements and team grow.



What you'll learn

- Benefits of intentional architecture
- Common architectural principles
- The Flutter team's recommended app architecture
- MVVM and state management
- Dependency injection
- Common design patterns for writing robust Flutter applications

Benefits of intentional architecture

Good app architecture provides a number of benefits to engineering teams and their end users.

- Maintainability - App architecture makes it easier to modify, update, and fix issues over time.
- Scalability - A well thought out application allows more people to contribute to the same codebase concurrently, with minimal code conflicts.
- Testability - Applications with intentional architecture generally have simpler classes with well-defined inputs and outputs, which makes them easier to mock and test.
- Lower cognitive load - Developers who are new to the project will be more productive in a shorter amount of time, and code reviews are generally less time-consuming when code is easier to understand.
- A better user experience - Features can ship faster and with fewer bugs.

How to use this guide

This is a guide for building scalable Flutter applications and was written for teams that have multiple developers contributing to the same code base, who're building a feature-rich application. If you're writing a Flutter app that has a *growing team and codebase*, this guidance is for you.

Along with general architectural advice, this guide gives concrete examples of best practices and includes specific recommendations. Some libraries can be swapped out, and very large teams with unique complexity might find that some parts don't apply. In either case, the ideas remain sound. This is the recommended way to build a Flutter app.

In the first part of this guide, you'll learn about common architectural principles from a high level. In the second part, the guide walks through specific and concrete recommendations of architecting Flutter apps. Finally, at the end of the guide, you'll find a list of design patterns and sample code that shows the recommendations in action.

Feedback

As this section of the website is evolving, we welcome your feedback!