Stateful Widget

Situationserklärung

Die App enthält eine einfache Counter-App, die über die `setState`-Methode gesteuert wird. Wenn wir jedoch den Text, der den Wert des Counters anzeigt, und den Button, der den Counter steuert, in verschiedene Widgets extrahieren, um die Code-Klarheit zu verbessern, reagiert der Text nicht mehr auf den Button. Das liegt daran, dass `setState` nur das Widget und seine untergeordneten Widgets neu baut, aber nicht Widgets in anderen Zweigen des Widget-Baums.

Somit wird klar, dass ein `StatefulWidget` keinesfalls als Alternative zu einem vollständigen State-Management-System dienen kann. Es ist nicht in der Lage, Zustände über Widget-Grenzen hinweg zu verwalten, was bei komplexeren Anwendungen zu Problemen führen kann.

StatefulWidget eignet sich nicht als vollständige Lösung für das State-Management aus folgenden Gründen:

1. **Begrenzte Reichweite**: Der Zustand ist nur innerhalb des Widgets verfügbar, was den Austausch und die Wiederverwendbarkeit erschwert.
2. **Leistungsprobleme**: Jede Zustandsänderung führt zu einem vollständigen Neuaufbau des Widgets, was die Performance beeinträchtigen kann.
3. **Vermischung von Logik und UI**: Der Zustand und die Logik sind direkt im Widget integriert, was das Testen und Warten erschwert.
4. **Schwierig skalierbar**: Bei größeren Apps wird es schwierig, den Zustand effektiv zu verwalten, besonders bei mehreren Widgets, die denselben Zustand benötigen.
5. **Fehlende Persistenz**: Der Zustand geht verloren, wenn das Widget neu erstellt wird, z. B. beim Navigieren.

Für komplexe Anwendungen sind dedizierte State-Management-Lösungen wie Riverpod oder Bloc besser geeignet, da sie eine klarere Trennung von Logik und UI ermöglichen und den Zustand über mehrere Widgets hinweg effizient verwalten.