SEW Zusammenfassung

Flutter App Aufbau

Jede App startet in der `main()` Funktion. Flutter wird mittels `runApp(Widget w)` in einem Widget initialisiert.

Flutter besteht aus widgets. Jede Komponente ist ein Widget.

Widgets können ineinander verschachtelt werden mittels `child` oder `children`.

Widgets enthalten meist properties zur Konfiguration.

Diese Widgets bilden einen Baum welcher bis zum Root bzw bis zum `runApp` aufruf zurückverfolgt werden kann.

Es gibt grundsätzlich zwei Arten von Widgets:

- Stateless Widgets

  Immer Identisch / Extern bestimmt

- Stateful Widgets

  Enthalten einen Zustand, ändern sich von sich aus

Flutter bietet über Googles Material 3 Bibliothek verschiedenste fertige Widgets an. Meist wird das Scaffold als Grundgerüst einer App verwendet.

(Wichtige) Widgets

- Scaffold

  Grundgerüst für viele Apps, bietet Grundstruktur und enthält meist eine AppBar.

- AppBar

  Enthält Titel, Buttons, ...

- Layout

  Positioniert Widgets am Bildschirm außerhalb des existierenden Layoutsystems.

Assets

Müssen zuerst in der `pubspec.yaml` definiert werden.

Navigation

Der Navigator wird verwendet um zwischen Views zu wechseln. Der Navigator speichert die "History" als Stack. Mit push und pop kann hinzugefügt oder entfernt werden. Die AppBar enthält einen automatischen "Back" Button.

Flutter State

Damit sich in der App Daten oder Eigenschaften ändern können muss ein State bzw. Zustand definiert und gespeichert werden.

In Flutter gilt: `UI = f(state)` - UI ist Funktion aus State.

Dh. Gegeben den selben State ist die UI immer identisch.

Ändert sich der State, so wird die build() Funktion aufgerufen und das Widget neu gebaut / aktualisiert.

Unterteilung:

- Ephermal State
    - Nur in einem Widget
    - Zur Umsetzung: Stateful Widget
    - `setState()` aktualisiert den Zustand des Widgets und baut es neu
- App State
    - Öfters, Widget übergreifend benötigt
    - Pub/Sub pattern
    - Subscriber melden sich beim Publisher und werden über Änderungen informiert
    - Implementieren `update()` / `build()` (wird vom Publisher aufgerufen)
    - Publisher benachrichtigen Subscriber bei Stateänderungen

Provider Package

Möglichkeit Pub/Sub in Flutter umzusetzen.

Besteht aus:

- ChangeNotifier
  - Publisher
  - `notifyListeners()` wird nach Datenänderung aufgerufen (Aktualisiert Subscriber)
- ChangeNotifierProvider
  - Stellt children ein ChangeNotifier zur Verfügung
  - Sollten so wenig Widgets wie möglich umfassen (um rebuilds zu minimieren)
  - MultiProvider kann statt mehreren verwendet werden
- Consumer
  - Subscriber
  - Enthält Widgets, welche die Daten des Publishers benötigen
  - Enthält `builder` Funktion welche die zu bauenden Widgets enthält
  - Stellt Instanz des `ChangeNotifiers` als `state`
- Provider.of
  - Fragt State an, hört aber nicht aktive auf Änderungen
  - Zb. beim ändern von Werten, welche aber nicht vom Widget benötigt werden
    - Meist Buttons, zb. "Zum Warenkorb hinzufügen" → Ändert State vom Warenkorb, braucht den aber selber nicht, fügt nur hinzu