Dart & Flutter – Frontend

Lehrbuch Teil A: Dart, Flutter Grundlagen, Fortgeschritten & Profi

Lehrplan für Entwickler mit Vorkenntnissen in C++, JavaScript und Python

February 2026

Contents

Chapter 1

Block 1: Dart – Die Sprache

In diesem Block lernst du Dart als Sprache kennen. Da du bereits C++, JS und Python beherrschst, wirst du viele Konzepte wiedererkennen. Der Fokus liegt auf den Dart-spezifischen Eigenheiten.

1.1 Einheit 1.1: Dart Syntax & Typsystem

1.1.0.1 1.1 Dart im Überblick

Dart ist eine von Google entwickelte, objektorientierte Programmiersprache mit C-ähnlicher Syntax. Für Entwickler mit C++-, JavaScript- und Python-Erfahrung ist der Einstieg daher sehr intuitiv. Dart bringt jedoch einige einzigartige Eigenschaften mit:

Kompilierungsmodi

Dart unterstützt **zwei Kompilierungsmodi**, was es besonders vielseitig macht:

Modus	Beschreibung	Einsatz
JIT (Just-In-Time)	Kompiliert zur Laufzeit, ermöglicht Hot Reload	Entwicklungsphase (Flutter Debug)
AOT (Ahead-Of-Time)	Kompiliert zu nativem Maschinencode	Produktions-Builds (Flutter Release)

Vergleich: - **C++** ist rein AOT-kompiliert — Dart bietet zusätzlich JIT für schnellere Entwicklungszyklen. - **JavaScript** wird von der Engine zur Laufzeit interpretiert/JIT-kompiliert — Dart kann echten nativen Code erzeugen. - **Python** ist interpretiert — Dart ist in der Produktion deutlich performanter durch AOT-Kompilierung.

Typsystem

Dart verwendet ein **starkes, statisches Typsystem mit Typinferenz**. Das heißt: - Alle Typen werden zur Kompilierzeit geprüft (wie C++, anders als Python/JS). - Der Compiler kann Typen automatisch ableiten, sodass man nicht immer explizit annotieren muss (ähnlich `auto` in C++ oder TypeScript).

```
// Typinferenz – der Compiler erkennt den Typ automatisch
var name = 'Dart';       // Typ: String (inferiert)
var zahl = 42;            // Typ: int (inferiert)
var pi = 3.14;            // Typ: double (inferiert)

// Explizite Typangabe – gleichwertiger Code
String name2 = 'Dart';
int zahl2 = 42;
double pi2 = 3.14;
```

1.1.0.2 1.2 Variablen: var, final, const, late, dynamic

`var` — Typinferenz bei der Deklaration

```
var stadt = 'Berlin';   // Typ wird als String inferiert
stadt = 'München';      // OK – neuer Wert, gleicher Typ
// stadt = 42;           // FEHLER – Typ ist String, nicht int
```

Vergleich zu JS: In JavaScript ist `var` funktions-scoped und erlaubt beliebige Typwechsel. In Dart ist `var` block-scoped (wie JS `let`) und der inferierte Typ ist fix.

`final` — Einmalige Zuweisung (Laufzeit-Konstante)

```
final zeitpunkt = DateTime.now();  // Wird zur Laufzeit bestimmt
final String gruß = 'Hallo';      // Expliziter Typ möglich

// zeitpunkt = DateTime.now();     // FEHLER – final kann nicht neu zugewiesen werden
```

Vergleich: - Wie `const` in JavaScript (nicht zu verwechseln mit Dart's `const`!) - Wie `final` in Java - Ähnlich zu C++ `const` bei lokalen Variablen

`const` — Kompilierzeit-Konstante

```
const double pi = 3.14159265;     // Muss zur Kompilierzeit bekannt sein
const maxVersuche = 3;

// const jetzt = DateTime.now();   // FEHLER – DateTime.now() ist keine Kompilierzeit-Konstante
```

Wichtig: `const` in Dart ist strenger als in C++. Der Wert muss vollständig zur **Kompilierzeit** feststehen.

```
// const vs final – der entscheidende Unterschied:
final laufzeit = DateTime.now();       // OK – wird zur Laufzeit bestimmt
// const kompilierzeit = DateTime.now(); // FEHLER – nicht zur Kompilierzeit bekannt

const liste = [1, 2, 3];              // Die Liste selbst ist unveränderlich (deeply immutable)
final liste2 = [1, 2, 3];             // Die Referenz ist fix, aber die Liste kann verändert werden
// liste.add(4);                       // FEHLER – const-Liste ist unveränderlich
liste2.add(4);                         // OK – final schützt nur die Referenz
```

`late` — Verzögerte Initialisierung

```
late String beschreibung;

void initialisiere() {
  beschreibung = 'Wird später gesetzt';  // Initialisierung bei erster Zuweisung
}

// Lazy Initialization – wird erst beim ersten Zugriff berechnet
late final String teureBerechnung = _berechneAufwendig();
```

```
String _berechneAufwendig() {
  print('Berechnung läuft...');
  return 'Ergebnis';
}
```

Vergleich zu Python: Ähnlich wie eine Property mit `@cached_property`, aber als Sprachfeature.

`dynamic` — Opt-out aus dem Typsystem

```
dynamic irgendwas = 'Text';
irgendwas = 42;         // OK – Typ kann sich ändern
irgendwas = true;       // OK – beliebiger Typ

// VORSICHT: Fehler werden erst zur Laufzeit erkannt!
// irgendwas.nichtExistierendeMethode();  // Kein Kompilierfehler, aber
↪  Laufzeitfehler
```

Vergleich: `dynamic` verhält sich wie Variablen in Python oder JavaScript — kein statischer Typcheck. **Sollte sparsam eingesetzt werden!**

Übersicht

Schlüsselwort	Typ änderbar	Wert änderbar	Wann initialisiert
`var`	Nein (inferiert)	Ja	Bei Deklaration
`final`	Nein	Nein	Zur Laufzeit
`const`	Nein	Nein	Zur Kompilierzeit
`late`	Nein	Ja (oder final)	Verzögert
`dynamic`	Ja	Ja	Bei Deklaration

1.1.0.3 1.3 Typsystem: Grundtypen

Zahlen: `int`, `double`, `num`

```
int ganzzahl = 42;
double kommazahl = 3.14;
num beliebig = 42;          // num ist Obertyp von int und double
beliebig = 3.14;            // OK – num akzeptiert beides

// Nützliche Methoden
print(42.isEven);            // true
print(3.14.ceil());          // 4
print(3.14.toStringAsFixed(1)); // '3.1'

// Konvertierungen
int a = 3;
double b = a.toDouble();    // 3.0
int c = 3.7.toInt();        // 3 (abgeschnitten, nicht gerundet!)
int d = 3.7.round();        // 4

// Integer-Literale
```

```
var hex = 0xFF;              // 255
var binär = 0b1010;          // 10 (binär, wie in C++)
```

Vergleich zu C++: Kein `float` — Dart verwendet ausschließlich `double` (64-Bit IEEE 754). Es gibt keine implizite Konvertierung zwischen `int` und `double`.

Vergleich zu JS: JavaScript hat nur `number` (immer double). Dart unterscheidet echte Ganzzahlen (`int`) von Kommazahlen (`double`).

`String`

```
var einfach = 'Einfache Anführungszeichen';
var doppelt = "Doppelte Anführungszeichen";
var mehrzeilig = '''
Dies ist ein
mehrzeiliger String.
''';

// Raw Strings (kein Escaping)
var pfad = r'C:\Users\name\docs';   // Backslash wird nicht interpretiert

// Nützliche String-Methoden
var text = '  Hallo Welt  ';
print(text.trim());                  // 'Hallo Welt'
print(text.contains('Welt'));        // true
print('abc'.padLeft(6, '0'));        // '000abc'
print('Hallo'.replaceAll('l', 'r')); // 'Harro'
print('a,b,c'.split(','));          // ['a', 'b', 'c']
```

Vergleich zu Python: Mehrzeilige Strings verwenden `'''` wie Python. Raw Strings nutzen `r'...'` statt Python's `r"..."`.

`bool`

```
bool aktiv = true;
bool fertig = false;

// Dart ist streng – keine truthy/falsy Werte wie in JS/Python!
// if (1) { ... }     // FEHLER – int ist kein bool
// if ('text') { ... } // FEHLER – String ist kein bool
if (1 > 0) { }        // OK – Vergleichsoperator ergibt bool
```

Wichtiger Unterschied zu JS/Python: In Dart gibt es **keine implizite Konvertierung** zu `bool`. Bedingungen müssen explizit `bool`-Werte sein.

Typinferenz im Detail

```
var x = 42;           // Typ: int
var y = 42.0;         // Typ: double
var z = 'Hallo';      // Typ: String
var w = [1, 2, 3];    // Typ: List<int>
var m = {'a': 1};     // Typ: Map<String, int>

// runtimeType gibt den Laufzeittyp zurück
```

```
print(x.runtimeType);  // int
print(w.runtimeType);  // List<int>

// Typüberprüfung mit 'is'
if (x is int) {
  print('x ist ein int');
}
```

1.1.0.4 1.4 String-Interpolation

Dart bietet eine elegante Syntax für String-Interpolation, die einfacher ist als in den meisten anderen Sprachen:

```
var name = 'Dart';
var version = 3;

// Einfache Variable mit $
print('Willkommen bei $name!');              // Willkommen bei Dart!

// Ausdrücke mit ${}
print('$name Version ${version + 1}');       // Dart Version 4
print('Großbuchstaben: ${name.toUpperCase()}'); // Großbuchstaben: DART

// Verschachtelt und komplex
var liste = [1, 2, 3];
print('Summe: ${liste.reduce((a, b) => a + b)}'); // Summe: 6

// Vergleich mit anderen Sprachen:
// C++:    std::format("Hallo {}", name)  oder  "Hallo " + name   (C++20)
// JS:     `Hallo ${name}`                (Template Literals)
// Python: f'Hallo {name}'               (f-Strings)
// Dart:   'Hallo $name'                 (am kürzesten!)
```

Tipp: Verwende `$variable` für einfache Variablen und `${ausdruck}` nur wenn ein Ausdruck ausgewertet werden muss.

1.1.0.5 1.5 Enums (einfache Variante)

```
enum Wochentag { montag, dienstag, mittwoch, donnerstag, freitag, samstag,     ↪
 ↪  sonntag }

void main() {
  var heute = Wochentag.freitag;
  print(heute);            // Wochentag.freitag
  print(heute.name);       // 'freitag'
  print(heute.index);      // 4

  // In switch verwenden
  switch (heute) {
    case Wochentag.samstag:
    case Wochentag.sonntag:
```

```
      print('Wochenende!');
      break;
    default:
      print('Arbeitstag');
  }
}
```

1.1.0.6 1.6 Dein erstes Dart-Programm

```
// datei: mein_programm.dart

// Top-Level-Konstante
const String appName = 'Meine App';

// Top-Level-Funktion
String formatiereDatum(DateTime datum) {
  return '${datum.day}.${datum.month}.${datum.year}';
}

// Top-Level-Variable
var zähler = 0;

// Einstiegspunkt – jedes Dart-Programm benötigt eine main()-Funktion
void main() {
  print('$appName gestartet');
  print(formatiereDatum(DateTime.now()));
}
```

Programm ausführen

```
# Mit DartPad (online): https://dartpad.dev
# Oder lokal:
dart run mein_programm.dart
```

1.1.0.7 Zusammenfassung

Konzept	Dart	C++	JavaScript	Python
Typinferenz	`var x = 42;`	`auto x = 42;`	`let x = 42;`	`x = 42`
Konstante (Laufzeit)	`final`	`const`	`const`	-
Konstante (Kompilierzeit)	`const`	`constexpr`	-	-
String-Interpolation	`'$var'`	-	`` `${var}` ``	`f'{var}'`
Strikte Typen	Ja	Ja	Nein	Nein

1.1.1 Übung

Nutze DartPad für diese Übungen.

1.1.1.1 Aufgabe 1: Variablen-Deklaration (10 Min.)

Deklariere die folgenden Variablen mit dem **passenden Schlüsselwort** (`var`, `final`, `const`, `late`):

```
void main() {
  // 1. Eine Variable für den Namen eines Benutzers, der sich ändern kann
  // TODO: Deklariere 'benutzername'

  // 2. Die mathematische Konstante Pi (bekannt zur Kompilierzeit)
  // TODO: Deklariere 'pi'

  // 3. Der Zeitstempel, wann das Programm gestartet wurde
  // TODO: Deklariere 'startzeit'

  // 4. Eine Konfiguration, die erst später initialisiert wird
  // TODO: Deklariere 'konfiguration'

  // 5. Die maximale Anzahl an Login-Versuchen (immer 3)
  // TODO: Deklariere 'maxVersuche'

  // Gib alle Variablen aus
  print('Benutzer: $benutzername');
  print('Pi: $pi');
  print('Gestartet: $startzeit');
  print('Max Versuche: $maxVersuche');
}
```

1.1.1.2 Aufgabe 2: Typkonvertierungen (15 Min.)

Schreibe eine Funktion, die verschiedene Typkonvertierungen durchführt:

```
void main() {
  // Gegeben:
  String eingabe = '42.7';

  // TODO: Konvertiere 'eingabe' zu double
  // double alsDouble = ...

  // TODO: Konvertiere das double zu int (abgerundet)
  // int abgerundet = ...

  // TODO: Konvertiere das double zu int (gerundet)
  // int gerundet = ...

  // TODO: Konvertiere das double zu int (aufgerundet)
  // int aufgerundet = ...

  // TODO: Erstelle einen formatierten String mit 1 Nachkommastelle
  // String formatiert = ...

  print('Original: $eingabe');
  print('Als Double: $alsDouble');
```

```
  print('Abgerundet: $abgerundet');
  print('Gerundet: $gerundet');
  print('Aufgerundet: $aufgerundet');
  print('Formatiert: $formatiert');

  // Erwartete Ausgabe:
  // Original: 42.7
  // Als Double: 42.7
  // Abgerundet: 42
  // Gerundet: 43
  // Aufgerundet: 43
  // Formatiert: 42.7
}
```

1.1.1.3 Aufgabe 3: String-Interpolation (15 Min.)

Erstelle eine Funktion `erstelleVisitenkarte`, die eine formatierte Visitenkarte ausgibt:

```
void main() {
  erstelleVisitenkarte(
    name: 'Max Mustermann',
    firma: 'Tech GmbH',
    position: 'Senior Developer',
    email: 'max@tech.de',
    telefon: '+49 123 456789',
  );
}

void erstelleVisitenkarte({
  required String name,
  required String firma,
  required String position,
  required String email,
  required String telefon,
}) {
  // TODO: Erstelle eine formatierte Visitenkarte mit String-Interpolation
  // Verwende mehrzeilige Strings (''') und String-Interpolation ($)

  // Erwartete Ausgabe (ungefähr):
  // +------------------------------------+
  // |  MAX MUSTERMANN                    |
  // |  Senior Developer                  |
  // |                                    |
  // |  Tech GmbH                         |
  // |  max@tech.de                       |
  // |  +49 123 456789                    |
  // +------------------------------------+
}
```

Hinweise: - Verwende `name.toUpperCase()` für den Namen - Nutze `padRight()` zum Auffüllen mit Leerzeichen - Die Box sollte eine feste Breite von 38 Zeichen haben

1.1.1.4 Aufgabe 4: Typ-Checks und Typumwandlung (10 Min.)

Schreibe eine Funktion, die den Typ einer Variable erkennt und entsprechend verarbeitet:

```
void main() {
  verarbeite('Hallo');
  verarbeite(42);
  verarbeite(3.14);
  verarbeite(true);
  verarbeite([1, 2, 3]);
}

void verarbeite(dynamic wert) {
  // TODO: Prüfe den Typ mit 'is' und gib eine passende Beschreibung aus

  // Für String: "Text mit X Zeichen: ..."
  // Für int: "Ganzzahl: X (gerade/ungerade)"
  // Für double: "Kommazahl: X (gerundet: Y)"
  // Für bool: "Wahrheitswert: wahr/falsch"
  // Für List: "Liste mit X Elementen"
  // Sonst: "Unbekannter Typ: ..."

  // Erwartete Ausgabe:
  // Text mit 5 Zeichen: Hallo
  // Ganzzahl: 42 (gerade)
  // Kommazahl: 3.14 (gerundet: 3)
  // Wahrheitswert: wahr
  // Liste mit 3 Elementen
}
```

1.1.1.5 Aufgabe 5: Konstanten-Quiz (10 Min.)

Analysiere den folgenden Code und beantworte die Fragen:

```
void main() {
  // Welche der folgenden Zeilen kompilieren? Warum oder warum nicht?

  // A
  const a = DateTime.now();

  // B
  final b = DateTime.now();

  // C
  const c = 3.14 * 2;

  // D
  final d = [1, 2, 3];
  d.add(4);

  // E
  const e = [1, 2, 3];
  e.add(4);
```

```
  // F
  var f = 'test';
  f = 'neu';

  // G
  var g = 'test';
  g = 42;

  // H
  late String h;
  print(h);

  // I
  late String i;
  i = 'initialisiert';
  print(i);
}
```

Schreibe für jede Zeile (A-I): 1. Kompiliert sie? (Ja/Nein) 2. Wenn nein, warum nicht? 3. Wenn ja, aber Laufzeitfehler möglich, welcher?

1.1.1.6 Bonusaufgabe: Temperaturrechner (optional)

Erstelle einen Temperaturrechner, der zwischen Celsius, Fahrenheit und Kelvin umrechnet:

```
void main() {
  // Definiere Konstanten für absolute Nullpunkte
  // const kelvinOffset = ...

  var celsius = 25.0;

  // Berechne Fahrenheit und Kelvin
  // Formel C -> F: (C * 9/5) + 32
  // Formel C -> K: C + 273.15

  // Formatierte Ausgabe mit 2 Nachkommastellen:
  // 25.00 °C = 77.00 °F = 298.15 K
}
```

1.1.2 Lösung

1.1.2.1 Aufgabe 1: Variablen-Deklaration

```
void main() {
  // 1. var – der Name kann sich ändern
  var benutzername = 'Max';

  // 2. const – mathematische Konstante, zur Kompilierzeit bekannt
  const pi = 3.14159265;
```

```
  // 3. final – Zeitstempel wird zur Laufzeit bestimmt
  final startzeit = DateTime.now();

  // 4. late – wird später initialisiert
  late String konfiguration;
  konfiguration = 'debug'; // Muss vor erstem Zugriff initialisiert werden

  // 5. const – fester Wert, der sich nie ändert
  const maxVersuche = 3;

  print('Benutzer: $benutzername');
  print('Pi: $pi');
  print('Gestartet: $startzeit');
  print('Konfiguration: $konfiguration');
  print('Max Versuche: $maxVersuche');
}
```

1.1.2.2 Aufgabe 2: Typkonvertierungen

```
void main() {
  String eingabe = '42.7';

  // String zu double
  double alsDouble = double.parse(eingabe);

  // double zu int (abgerundet) – toInt() schneidet ab
  int abgerundet = alsDouble.toInt();

  // double zu int (gerundet)
  int gerundet = alsDouble.round();

  // double zu int (aufgerundet)
  int aufgerundet = alsDouble.ceil();

  // Formatierter String mit 1 Nachkommastelle
  String formatiert = alsDouble.toStringAsFixed(1);

  print('Original: $eingabe');
  print('Als Double: $alsDouble');
  print('Abgerundet: $abgerundet');
  print('Gerundet: $gerundet');
  print('Aufgerundet: $aufgerundet');
  print('Formatiert: $formatiert');
}
```

Erklärung: - `double.parse()` wandelt einen String in double um - `toInt()` schneidet die Nachkommastellen ab (floor für positive, ceil für negative) - `round()` rundet mathematisch korrekt - `ceil()` rundet immer auf - `toStringAsFixed(n)` formatiert mit n Nachkommastellen

1.1.2.3 Aufgabe 3: String-Interpolation

```
void main() {
  erstelleVisitenkarte(
    name: 'Max Mustermann',
    firma: 'Tech GmbH',
    position: 'Senior Developer',
    email: 'max@tech.de',
    telefon: '+49 123 456789',
  );
}

void erstelleVisitenkarte({
  required String name,
  required String firma,
  required String position,
  required String email,
  required String telefon,
}) {
  const breite = 36;

  var karte = '''
+${'-' * breite}+
|  ${name.toUpperCase().padRight(breite - 2)}|
|  ${position.padRight(breite - 2)}|
|${' ' * breite}|
|  ${firma.padRight(breite - 2)}|
|  ${email.padRight(breite - 2)}|
|  ${telefon.padRight(breite - 2)}|
+${'-' * breite}+''';

  print(karte);
}
```

Alternative kompaktere Version:

```
void erstelleVisitenkarte({
  required String name,
  required String firma,
  required String position,
  required String email,
  required String telefon,
}) {
  String zeile(String text) => '|  ${text.padRight(34)}|';
  String leer() => '|${' ' * 36}|';

  print('+${'-' * 36}+');
  print(zeile(name.toUpperCase()));
  print(zeile(position));
  print(leer());
  print(zeile(firma));
  print(zeile(email));
```

```
  print(zeile(telefon));
  print('+${'-' * 36}+');
}
```

1.1.2.4 Aufgabe 4: Typ-Checks und Typumwandlung

```
void main() {
  verarbeite('Hallo');
  verarbeite(42);
  verarbeite(3.14);
  verarbeite(true);
  verarbeite([1, 2, 3]);
}

void verarbeite(dynamic wert) {
  if (wert is String) {
    print('Text mit ${wert.length} Zeichen: $wert');
  } else if (wert is int) {
    var parität = wert.isEven ? 'gerade' : 'ungerade';
    print('Ganzzahl: $wert ($parität)');
  } else if (wert is double) {
    print('Kommazahl: $wert (gerundet: ${wert.round()})');
  } else if (wert is bool) {
    var bezeichnung = wert ? 'wahr' : 'falsch';
    print('Wahrheitswert: $bezeichnung');
  } else if (wert is List) {
    print('Liste mit ${wert.length} Elementen');
  } else {
    print('Unbekannter Typ: ${wert.runtimeType}');
  }
}
```

Mit switch (Dart 3 Pattern Matching):

```
void verarbeite(dynamic wert) {
  var beschreibung = switch (wert) {
    String s => 'Text mit ${s.length} Zeichen: $s',
    int i => 'Ganzzahl: $i (${i.isEven ? 'gerade' : 'ungerade'})',
    double d => 'Kommazahl: $d (gerundet: ${d.round()})',
    bool b => 'Wahrheitswert: ${b ? 'wahr' : 'falsch'}',
    List l => 'Liste mit ${l.length} Elementen',
    _ => 'Unbekannter Typ: ${wert.runtimeType}',
  };
  print(beschreibung);
}
```

1.1.2.5 Aufgabe 5: Konstanten-Quiz

Zeile	Kompiliert?	Erklärung
A	Nein	`DateTime.now()` ist keine Kompilierzeit-Konstante
B	Ja	`final` erlaubt Laufzeit-Initialisierung
C	Ja	Arithmetik mit Konstanten ergibt eine Konstante
D	Ja	`final` schützt nur die Referenz, nicht den Inhalt
E	Kompiliert, aber…	Laufzeitfehler! `const`-Listen sind unveränderlich
F	Ja	`var` erlaubt neue Werte des gleichen Typs
G	Nein	Typ wurde als String inferiert, 42 ist int
H	Kompiliert, aber…	Laufzeitfehler! `late` Variable wurde nicht initialisiert
I	Ja	`late` Variable wurde vor Zugriff initialisiert

1.1.2.6 Bonusaufgabe: Temperaturrechner

```
void main() {
  // Konstanten
  const double kelvinOffset = 273.15;
  const double fahrenheitFaktor = 9 / 5;
  const double fahrenheitOffset = 32;

  var celsius = 25.0;

  // Berechnungen
  double fahrenheit = (celsius * fahrenheitFaktor) + fahrenheitOffset;
  double kelvin = celsius + kelvinOffset;

  // Formatierte Ausgabe
  print('${celsius.toStringAsFixed(2)} °C = '
        '${fahrenheit.toStringAsFixed(2)} °F = '
        '${kelvin.toStringAsFixed(2)} K');

  // Erweitert: Als Funktion
  print(konvertiereTemperatur(0, 'C'));
  print(konvertiereTemperatur(100, 'C'));
  print(konvertiereTemperatur(-40, 'C'));  // Spaßfakt: -40°C = -40°F
}

String konvertiereTemperatur(double wert, String einheit) {
  const kelvinOffset = 273.15;

  double celsius;
```

```
  // Zuerst alles in Celsius umrechnen
  switch (einheit.toUpperCase()) {
    case 'C':
      celsius = wert;
      break;
    case 'F':
      celsius = (wert - 32) * 5 / 9;
      break;
    case 'K':
      celsius = wert - kelvinOffset;
      break;
    default:
      return 'Ungültige Einheit: $einheit';
  }

  // Von Celsius in alle Einheiten
  var fahrenheit = (celsius * 9 / 5) + 32;
  var kelvin = celsius + kelvinOffset;

  return '${celsius.toStringAsFixed(2)} °C = '
         '${fahrenheit.toStringAsFixed(2)} °F = '
         '${kelvin.toStringAsFixed(2)} K';
}
```

1.1.3 Ressourcen

1.1.3.1 Offizielle Dokumentation

- Dart Language Tour — Offizielle Sprachübersicht
- Dart Type System — Typsystem im Detail
- Effective Dart: Usage — Best Practices

1.1.3.2 Online-Tools

- DartPad — Online Dart/Flutter Editor (ideal für Übungen)
- Dart Cheatsheet — Interaktives Codelab

1.1.3.3 Vertiefende Artikel

var, final, const

- Understanding const in Dart — Wann welches verwenden

Typsystem

- Sound Null Safety — Dart's Null-Safety-System (Vorschau für spätere Einheiten)

1.1.3.4 Video-Tutorials

- Dart in 100 Seconds — Fireship (Schnellübersicht)
- Dart Tutorial for Beginners — Mitch Koko

1.1.3.5 Wichtige Konzepte zum Merken

```
// var -> Typ wird inferiert, Wert kann sich ändern
var name = 'Dart';

// final -> Wert wird einmal zur LAUFZEIT festgelegt
final startzeit = DateTime.now();

// const -> Wert muss zur KOMPILIERZEIT bekannt sein
const pi = 3.14159;

// late -> Initialisierung verzögert
late String config;

// String-Interpolation
print('Hallo $name!');           // Variable
print('2+2 = ${2 + 2}');         // Ausdruck
```

1.1.3.6 Nächste Einheit

Einheit 1.2: Funktionen & Kontrollstrukturen - Funktionsparameter (named, positional, optional) - Arrow-Syntax - if/else, switch, for, while

1.2 Einheit 1.2: Funktionen & Kontrollstrukturen

1.2.0.1 2.1 Funktionen in Dart

In Dart sind Funktionen **First-Class-Objekte** — sie können Variablen zugewiesen, als Parameter übergeben und von Funktionen zurückgegeben werden (wie in JS und Python).

Grundlegende Syntax

```
// Volle Syntax mit Rückgabetyp
int addiere(int a, int b) {
  return a + b;
}

// Arrow-Syntax für einzeilige Funktionen (wie JS Arrow Functions)
int addiere2(int a, int b) => a + b;

// Rückgabetyp kann inferiert werden (wird aber empfohlen, ihn anzugeben)
addiere3(int a, int b) => a + b;
```

Positionelle Parameter

```
// Erforderliche positionelle Parameter
String begrüße(String name, String grußwort) {
  return '$grußwort, $name!';
}

// Aufruf:
begrüße('Welt', 'Hallo');  // 'Hallo, Welt!'
```

1.2.0.2 2.2 Optionale Parameter

Optionale positionelle Parameter `[...]`

```
// Optionale Parameter stehen in eckigen Klammern
String begrüße(String name, [String grußwort = 'Hallo', String suffix = '']) {
  return '$grußwort, $name$suffix';
}

// Aufrufe:
begrüße('Welt');                    // 'Hallo, Welt'
begrüße('Welt', 'Hi');             // 'Hi, Welt'
begrüße('Welt', 'Hi', '!');       // 'Hi, Welt!'
```

Benannte Parameter `{...}`

```
// Benannte Parameter stehen in geschweiften Klammern
// 'required' markiert Pflichtparameter
void erstelleBenutzer({
  required String name,
  required String email,
  int alter = 0,            // Optional mit Standardwert
  String? telefon,           // Optional und nullable
}) {
  print('Name: $name, Email: $email, Alter: $alter');
  if (telefon != null) {
    print('Telefon: $telefon');
  }
}

// Aufrufe – Reihenfolge der benannten Parameter ist beliebig:
erstelleBenutzer(name: 'Max', email: 'max@mail.de');
erstelleBenutzer(email: 'anna@mail.de', name: 'Anna', alter: 25);
```

Vergleich: - **Python:** Ähnlich wie Python's Keyword-Argumente, aber expliziter durch `required`. - **C++:** C++ hat keine benannten Parameter — in Dart ist dies ein eingebautes Sprachfeature. - **JS:** JS-Objekt-Destructuring `({name, email})` ist konzeptionell ähnlich.

1.2.0.3 2.3 Arrow-Syntax =>

```
// Arrow-Syntax ist Kurzschreibweise für { return ausdruck; }
int quadrat(int n) => n * n;
String formatiere(double wert) => wert.toStringAsFixed(2);
bool istGerade(int n) => n % 2 == 0;

// Auch für void-Funktionen nutzbar
void logge(String nachricht) => print('[LOG] $nachricht');

// Arrow-Syntax mit mehreren Ausdrücken ist NICHT möglich
// Für mehrere Anweisungen den normalen Block { } verwenden
```

1.2.0.4 2.4 Funktionen als First-Class-Objekte

```
// Funktion in Variable speichern
var verdopple = (int n) => n * 2;
print(verdopple(5));  // 10

// Typ der Variable
int Function(int) dreifach = (n) => n * 3;

// Funktion als Parameter übergeben
void wendeAn(int wert, int Function(int) operation) {
  print('Ergebnis: ${operation(wert)}');
}
wendeAn(5, verdopple);   // Ergebnis: 10
wendeAn(5, (n) => n + 1); // Ergebnis: 6

// Funktion als Rückgabewert
int Function(int) multiplikator(int faktor) {
  return (int n) => n * faktor;
}
var mal5 = multiplikator(5);
print(mal5(3));  // 15
```

Typedefs

```
// Typedef für Funktionstypen (verbessert Lesbarkeit)
typedef Vergleich = int Function(String, String);

void sortiereMit(List<String> liste, Vergleich vergleiche) {
  liste.sort(vergleiche);
}

sortiereMit(['b', 'a', 'c'], (a, b) => a.compareTo(b));
```

1.2.0.5 2.5 Kontrollstrukturen

if / else

```
var alter = 20;

if (alter >= 18) {
  print('Volljährig');
} else if (alter >= 16) {
  print('Bedingt geschäftsfähig');
} else {
  print('Minderjährig');
}

// Ternärer Operator (wie C++/JS/Python's ... if ... else ...)
var status = alter >= 18 ? 'Erwachsen' : 'Minderjährig';
```

for und for-in

```
// Klassische for-Schleife (wie C++/JS)
for (var i = 0; i < 5; i++) {
  print(i);
}

// for-in (wie Python's for...in, JS's for...of)
var farben = ['Rot', 'Grün', 'Blau'];
for (var farbe in farben) {
  print(farbe);
}

// forEach mit Lambda
farben.forEach((farbe) => print(farbe));
// oder mit Methodenreferenz (tear-off):
farben.forEach(print);
```

while und do-while

```
var i = 0;
while (i < 5) {
  print(i);
  i++;
}

// do-while – Body wird mindestens einmal ausgeführt
var eingabe = '';
do {
  eingabe = 'simuliert'; // In der Praxis: Benutzereingabe lesen
} while (eingabe.isEmpty);
```

1.2.0.6 2.6 Switch-Statement

Klassisches switch

```
var befehl = 'start';
switch (befehl) {
  case 'start':
    print('Starte...');
    break;   // break ist erforderlich (kein Fall-Through wie in C++)
  case 'stop':
    print('Stoppe...');
    break;
  default:
    print('Unbekannter Befehl');
}
```

Dart 3: Switch als Ausdruck (Expression)

```
var statusCode = 404;
var nachricht = switch (statusCode) {
  200 => 'OK',
  301 => 'Umgeleitet',
```

```
  404 => 'Nicht gefunden',
  >= 500 && < 600 => 'Serverfehler',
  _ => 'Unbekannt',   // _ ist der Wildcard/Default
};
print(nachricht);  // 'Nicht gefunden'

// Switch Expression mit Guard-Klauseln
var wert = 42;
var beschreibung = switch (wert) {
  0 => 'Null',
  < 0 => 'Negativ',
  > 0 && < 100 => 'Klein und positiv',
  >= 100 when wert.isEven => 'Groß und gerade',
  _ => 'Sonstiges',
};
```

1.2.0.7 2.7 Exceptions und Fehlerbehandlung

```
// Dart kann jeden Typ werfen, empfohlen ist aber Exception/Error
void teile(int a, int b) {
  if (b == 0) {
    throw ArgumentError('Division durch Null');
  }
  print(a / b);
}

void main() {
  try {
    teile(10, 0);
  } on ArgumentError catch (e) {
    // Spezifischen Fehlertyp fangen
    print('Argument-Fehler: $e');
  } on FormatException {
    // Ohne catch-Variable
    print('Format-Fehler');
  } catch (e, stackTrace) {
    // Alle anderen Fehler fangen
    print('Unbekannter Fehler: $e');
    print('Stack Trace: $stackTrace');
  } finally {
    // Wird immer ausgeführt
    print('Aufräumarbeiten');
  }
}
```

Vergleich: - `on Type catch (e)` entspricht `catch (Type& e)` in C++ oder `except Type as e` in Python. - Das `finally`-Block existiert wie in Python und JS.

1.2.0.8 2.8 Assert

```
// assert wird nur im Debug-Modus ausgewertet (JIT/Debug-Build)
// In Produktions-Builds (AOT) werden asserts komplett ignoriert
var alter = 25;
assert(alter >= 0, 'Alter darf nicht negativ sein');

// Nützlich für Entwicklungszeit-Checks
void setzeProzentwert(double wert) {
  assert(wert >= 0 && wert <= 100, 'Wert muss zwischen 0 und 100 liegen');
  // ... Implementierung
}
```

1.2.0.9 2.9 Zusammenfassendes Beispiel

```
/// Ein vollständiges Beispiel mit Funktionen und Kontrollstrukturen.

typedef Bewertung = String Function(int);

// Benannte Parameter + Standardwerte
String erstelleProfil({
  required String name,
  required int alter,
  String beruf = 'Unbekannt',
}) {
  return 'Profil: $name, $alter Jahre, $beruf';
}

// Arrow-Funktion
bool istErwachsen(int alter) => alter >= 18;

// Funktion die eine Funktion zurückgibt
Bewertung bewertungsFunktion(String kategorie) {
  return (int punkte) => switch (punkte) {
    >= 90 => '$kategorie: Sehr gut ($punkte)',
    >= 70 => '$kategorie: Gut ($punkte)',
    >= 50 => '$kategorie: Befriedigend ($punkte)',
    _ => '$kategorie: Ungenügend ($punkte)',
  };
}

void main() {
  // Benannte Parameter
  print(erstelleProfil(name: 'Max', alter: 28, beruf: 'Entwickler'));

  // Arrow-Funktion + Ternärer Operator
  var alter = 17;
  print(istErwachsen(alter) ? 'Volljährig' : 'Minderjährig');

  // Higher-Order Function
  var matheBewertung = bewertungsFunktion('Mathe');
```

```
  print(matheBewertung(85));  // Mathe: Gut (85)

  // for-in + switch Expression
  var noten = [95, 72, 45, 88];
  for (var note in noten) {
    print(matheBewertung(note));
  }
}
```

1.2.1 Übung

1.2.1.1 Aufgabe 1: Funktionsparameter (15 Min.)

Implementiere eine Funktion `bestelleKaffee` mit verschiedenen Parametertypen:

```
void main() {
  // Alle diese Aufrufe sollten funktionieren:
  bestelleKaffee('Espresso');
  bestelleKaffee('Cappuccino', größe: 'groß');
  bestelleKaffee('Latte', größe: 'medium', extras: ['Vanille', 'Karamell']);
  bestelleKaffee('Americano', temperatur: 70);
}

// TODO: Implementiere bestelleKaffee
// - sorte: required (positionell)
// - größe: optional (named), Standard: 'klein'
// - temperatur: optional (named), Standard: 85
// - extras: optional (named), Standard: leere Liste

// Erwartete Ausgabe:
// Bestellung: klein Espresso (85°C)
// Bestellung: groß Cappuccino (85°C)
// Bestellung: medium Latte (85°C) mit Vanille, Karamell
// Bestellung: klein Americano (70°C)
```

1.2.1.2 Aufgabe 2: Higher-Order Functions (15 Min.)

Erstelle einen einfachen Taschenrechner mit Higher-Order Functions:

```
void main() {
  // TODO: Implementiere die Funktionen

  var addieren = operation('+');
  var subtrahieren = operation('-');
  var multiplizieren = operation('*');
  var dividieren = operation('/');

  print(addieren(10, 5));       // 15
  print(subtrahieren(10, 5));   // 5
  print(multiplizieren(10, 5)); // 50
  print(dividieren(10, 5));     // 2
```

```
  // Bonus: berechne sollte einen Operator-String akzeptieren
  print(berechne(10, 5, '+'));  // 15
  print(berechne(10, 5, '%'));  // 0 (Modulo)
}

// TODO: Definiere einen Typedef für Berechnungsfunktionen
// typedef Berechnung = ...

// TODO: Implementiere operation(String op) die eine Berechnung zurückgibt
// Hint: Verwende switch Expression

// TODO: Implementiere berechne(num a, num b, String op)
```

1.2.1.3 Aufgabe 3: Kontrollstrukturen (15 Min.)

Implementiere ein Textadventure-Fragment mit verschiedenen Kontrollstrukturen:

```
void main() {
  spiele();
}

void spiele() {
  var spieler = {
    'name': 'Held',
    'leben': 100,
    'gold': 50,
    'inventar': ['Schwert', 'Schild'],
  };

  var ereignisse = [
    {'typ': 'monster', 'name': 'Goblin', 'schaden': 20, 'beute': 30},
    {'typ': 'truhe', 'gold': 100},
    {'typ': 'falle', 'schaden': 15},
    {'typ': 'händler', 'item': 'Heiltrank', 'preis': 40},
    {'typ': 'monster', 'name': 'Drache', 'schaden': 50, 'beute': 200},
  ];

  // TODO: Iteriere über ereignisse mit for-in
  // Für jedes Ereignis:
  // - 'monster': Ziehe schaden von leben ab, addiere beute zu gold
  //              Ausgabe: "Kampf gegen [name]! -[schaden] Leben, +[beute] Gold"
  // - 'truhe': Addiere gold
  //            Ausgabe: "Truhe gefunden! +[gold] Gold"
  // - 'falle': Ziehe schaden ab
  //            Ausgabe: "In Falle getappt! -[schaden] Leben"
  // - 'händler': Wenn genug Gold, kaufe Item (ziehe preis ab, füge item zu
↪ inventar)
  //              Ausgabe: "Gekauft: [item]" oder "Nicht genug Gold für [item]"

  // Wenn Leben <= 0, breche die Schleife ab und gib "Game Over" aus
  // Sonst gib am Ende den Spielerstand aus
```

```
}
```

1.2.1.4 Aufgabe 4: Switch Expressions (10 Min.)

Verwende Dart 3 Switch Expressions für einen Notenrechner:

```
void main() {
  var punkte = [95, 82, 67, 54, 41, 38, 100, 0];

  for (var p in punkte) {
    var note = berechneNote(p);
    var beschreibung = beschreibeNote(note);
    print('$p Punkte = Note $note ($beschreibung)');
  }
}

// TODO: Implementiere berechneNote(int punkte) -> int
// 90-100: 1, 75-89: 2, 60-74: 3, 45-59: 4, 30-44: 5, <30: 6
// Verwende switch Expression mit Patterns

// TODO: Implementiere beschreibeNote(int note) -> String
// 1: "sehr gut", 2: "gut", 3: "befriedigend",
// 4: "ausreichend", 5: "mangelhaft", 6: "ungenügend"
// Verwende switch Expression
```

1.2.1.5 Aufgabe 5: Fehlerbehandlung (10 Min.)

Implementiere eine sichere Division mit Fehlerbehandlung:

```
void main() {
  print(sicheresDividieren(10, 2));   // 5.0
  print(sicheresDividieren(10, 0));   // Fehler: Division durch Null
  print(sicheresDividieren(10, -1));  // Fehler: Divisor darf nicht negativ sein

  // Mit benutzerdefinierter Exception
  try {
    var ergebnis = dividiereStrikt(10, 0);
    print(ergebnis);
  } on DivisionException catch (e) {
    print('Fehler: ${e.message}');
  }
}

// TODO: Implementiere sicheresDividieren(int a, int b) -> String
// - Bei b == 0: Gib Fehlermeldung zurück (kein throw)
// - Bei b < 0: Gib Fehlermeldung zurück
// - Sonst: Gib das Ergebnis als String zurück

// TODO: Definiere eine eigene Exception-Klasse DivisionException
// - Hat ein Feld 'message'
// - Hat einen Konstruktor mit required message
```

```
// TODO: Implementiere dividiereStrikt(int a, int b) -> double
// - Wirft DivisionException bei b == 0 oder b < 0
// - Gibt sonst das Ergebnis zurück
```

1.2.1.6 Bonusaufgabe: FizzBuzz funktional

Implementiere FizzBuzz auf funktionale Weise:

```
void main() {
  // FizzBuzz für 1-20 mit funktionalem Ansatz
  // Verwende: List.generate, map, switch expression

  // Regeln:
  // - Durch 3 und 5 teilbar: "FizzBuzz"
  // - Durch 3 teilbar: "Fizz"
  // - Durch 5 teilbar: "Buzz"
  // - Sonst: die Zahl

  // TODO: Implementiere mit einer einzigen Kette von Methodenaufrufen
  // List.generate(20, ...).map(...).forEach(print);
}
```

1.2.2 Lösung

1.2.2.1 Aufgabe 1: Funktionsparameter

```
void main() {
  bestelleKaffee('Espresso');
  bestelleKaffee('Cappuccino', größe: 'groß');
  bestelleKaffee('Latte', größe: 'medium', extras: ['Vanille', 'Karamell']);
  bestelleKaffee('Americano', temperatur: 70);
}

void bestelleKaffee(
  String sorte, {
  String größe = 'klein',
  int temperatur = 85,
  List<String> extras = const [],
}) {
  var bestellung = 'Bestellung: $größe $sorte (${temperatur}°C)';

  if (extras.isNotEmpty) {
    bestellung += ' mit ${extras.join(', ')}';
  }

  print(bestellung);
}
```

1.2.2.2 Aufgabe 2: Higher-Order Functions

```
typedef Berechnung = num Function(num, num);

Berechnung operation(String op) {
  return switch (op) {
    '+' => (a, b) => a + b,
    '-' => (a, b) => a - b,
    '*' => (a, b) => a * b,
    '/' => (a, b) => a / b,
    '%' => (a, b) => a % b,
    '^' => (a, b) => _potenz(a, b),
    _ => (a, b) => throw ArgumentError('Unbekannter Operator: $op'),
  };
}

num _potenz(num basis, num exponent) {
  num ergebnis = 1;
  for (var i = 0; i < exponent; i++) {
    ergebnis *= basis;
  }
  return ergebnis;
}

num berechne(num a, num b, String op) {
  var op_func = operation(op);
  return op_func(a, b);
}

void main() {
  var addieren = operation('+');
  var subtrahieren = operation('-');
  var multiplizieren = operation('*');
  var dividieren = operation('/');

  print(addieren(10, 5));       // 15
  print(subtrahieren(10, 5));   // 5
  print(multiplizieren(10, 5)); // 50
  print(dividieren(10, 5));     // 2.0

  print(berechne(10, 5, '+'));  // 15
  print(berechne(10, 5, '%'));  // 0
  print(berechne(2, 8, '^'));   // 256
}
```

1.2.2.3 Aufgabe 3: Kontrollstrukturen

```
void main() {
  spiele();
}
```

```
void spiele() {
  var spieler = {
    'name': 'Held',
    'leben': 100,
    'gold': 50,
    'inventar': <String>['Schwert', 'Schild'],
  };

  var ereignisse = [
    {'typ': 'monster', 'name': 'Goblin', 'schaden': 20, 'beute': 30},
    {'typ': 'truhe', 'gold': 100},
    {'typ': 'falle', 'schaden': 15},
    {'typ': 'händler', 'item': 'Heiltrank', 'preis': 40},
    {'typ': 'monster', 'name': 'Drache', 'schaden': 50, 'beute': 200},
  ];

  for (var ereignis in ereignisse) {
    var typ = ereignis['typ'] as String;

    switch (typ) {
      case 'monster':
        var name = ereignis['name'];
        var schaden = ereignis['schaden'] as int;
        var beute = ereignis['beute'] as int;
        spieler['leben'] = (spieler['leben'] as int) - schaden;
        spieler['gold'] = (spieler['gold'] as int) + beute;
        print('Kampf gegen $name! -$schaden Leben, +$beute Gold');

      case 'truhe':
        var gold = ereignis['gold'] as int;
        spieler['gold'] = (spieler['gold'] as int) + gold;
        print('Truhe gefunden! +$gold Gold');

      case 'falle':
        var schaden = ereignis['schaden'] as int;
        spieler['leben'] = (spieler['leben'] as int) - schaden;
        print('In Falle getappt! -$schaden Leben');

      case 'händler':
        var item = ereignis['item'] as String;
        var preis = ereignis['preis'] as int;
        var gold = spieler['gold'] as int;
        if (gold >= preis) {
          spieler['gold'] = gold - preis;
          (spieler['inventar'] as List<String>).add(item);
          print('Gekauft: $item');
        } else {
          print('Nicht genug Gold für $item');
        }
    }
```

```
    // Leben prüfen
    if ((spieler['leben'] as int) <= 0) {
      print('\n=== GAME OVER ===');
      return;
    }

    print('  -> Leben: ${spieler['leben']}, Gold: ${spieler['gold']}');
  }

  print('\n=== SIEG! ===');
  print('Endstand: Leben: ${spieler['leben']}, Gold: ${spieler['gold']}');
  print('Inventar: ${(spieler['inventar'] as List).join(', ')}');
}
```

1.2.2.4 Aufgabe 4: Switch Expressions

```
void main() {
  var punkte = [95, 82, 67, 54, 41, 38, 100, 0];

  for (var p in punkte) {
    var note = berechneNote(p);
    var beschreibung = beschreibeNote(note);
    print('$p Punkte = Note $note ($beschreibung)');
  }
}

int berechneNote(int punkte) => switch (punkte) {
  >= 90 => 1,
  >= 75 => 2,
  >= 60 => 3,
  >= 45 => 4,
  >= 30 => 5,
  _ => 6,
};

String beschreibeNote(int note) => switch (note) {
  1 => 'sehr gut',
  2 => 'gut',
  3 => 'befriedigend',
  4 => 'ausreichend',
  5 => 'mangelhaft',
  6 => 'ungenügend',
  _ => 'ungültig',
};
```

Alternative mit Records:

```
(int, String) bewerteVollständig(int punkte) {
  var note = berechneNote(punkte);
  return (note, beschreibeNote(note));
}
```

```
void main() {
  var (note, beschreibung) = bewerteVollständig(85);
  print('Note $note: $beschreibung');
}
```

1.2.2.5 Aufgabe 5: Fehlerbehandlung

```
class DivisionException implements Exception {
  final String message;
  DivisionException({required this.message});

  @override
  String toString() => 'DivisionException: $message';
}

String sicheresDividieren(int a, int b) {
  if (b == 0) {
    return 'Fehler: Division durch Null';
  }
  if (b < 0) {
    return 'Fehler: Divisor darf nicht negativ sein';
  }
  return (a / b).toString();
}

double dividiereStrikt(int a, int b) {
  if (b == 0) {
    throw DivisionException(message: 'Division durch Null');
  }
  if (b < 0) {
    throw DivisionException(message: 'Divisor darf nicht negativ sein');
  }
  return a / b;
}

void main() {
  print(sicheresDividieren(10, 2));   // 5.0
  print(sicheresDividieren(10, 0));   // Fehler: Division durch Null
  print(sicheresDividieren(10, -1));  // Fehler: Divisor darf nicht negativ sein

  try {
    var ergebnis = dividiereStrikt(10, 0);
    print(ergebnis);
  } on DivisionException catch (e) {
    print('Fehler: ${e.message}');
  }

  // Mehrere Tests
  for (var divisor in [2, 0, -1, 5]) {
```

```
    try {
      print('10 / $divisor = ${dividiereStrikt(10, divisor)}');
    } on DivisionException catch (e) {
      print('10 / $divisor -> ${e.message}');
    }
  }
}
```

1.2.2.6 Bonusaufgabe: FizzBuzz funktional

```
void main() {
  // Einzeiler-Version
  List.generate(20, (i) => i + 1)
      .map((n) => switch ((n % 3 == 0, n % 5 == 0)) {
            (true, true) => 'FizzBuzz',
            (true, false) => 'Fizz',
            (false, true) => 'Buzz',
            (false, false) => n.toString(),
          })
      .forEach(print);
}
```

Alternative mit Pattern Matching:

```
void main() {
  for (var i = 1; i <= 20; i++) {
    var result = switch (i) {
      _ when i % 15 == 0 => 'FizzBuzz',
      _ when i % 3 == 0 => 'Fizz',
      _ when i % 5 == 0 => 'Buzz',
      _ => i.toString(),
    };
    print(result);
  }
}
```

Als Higher-Order Function:

```
String Function(int) fizzBuzzFactory({
  Map<int, String> regeln = const {3: 'Fizz', 5: 'Buzz'},
}) {
  return (int n) {
    var ergebnis = StringBuffer();
    for (var eintrag in regeln.entries) {
      if (n % eintrag.key == 0) {
        ergebnis.write(eintrag.value);
      }
    }
    return ergebnis.isEmpty ? n.toString() : ergebnis.toString();
  };
}
```

```
void main() {
  var fizzBuzz = fizzBuzzFactory();
  var fizzBuzzBang = fizzBuzzFactory(regeln: {3: 'Fizz', 5: 'Buzz', 7: 'Bang'});

  for (var i = 1; i <= 21; i++) {
    print('$i: ${fizzBuzzBang(i)}');
  }
}
```

1.2.3 Ressourcen

1.2.3.1 Offizielle Dokumentation

- Functions — Dart-Funktionen im Detail
- Control Flow — Schleifen und Bedingungen
- Patterns — Dart 3 Pattern Matching

1.2.3.2 Best Practices

Parameter-Reihenfolge

```
// Gut: Required zuerst, dann optional
void funktion(
  String required1,
  int required2, {
  String optional1 = 'default',
  int? optional2,
}) { }

// Gut: Benannte Parameter bei mehr als 2-3 Parametern
void erstelleBenutzer({
  required String name,
  required String email,
  int? alter,
}) { }
```

Wann Arrow-Syntax verwenden?

```
// Gut: Einzeiler
int quadrat(int n) => n * n;

// Schlecht: Zu komplex für Arrow
int komplex(int n) => n > 0 ? n * n : n < 0 ? -n * n : 0;  // Schwer lesbar

// Besser: Normaler Body
int komplex(int n) {
  if (n > 0) return n * n;
  if (n < 0) return -n * n;
  return 0;
}
```

1.2.3.3 Wichtige Patterns

Callback-Pattern

```
void ladeDaten({
  required void Function(String) onErfolg,
  required void Function(Exception) onFehler,
}) {
  try {
    var daten = 'Geladene Daten';
    onErfolg(daten);
  } catch (e) {
    onFehler(e as Exception);
  }
}
```

Factory-Pattern mit Funktionen

```
typedef Logger = void Function(String);

Logger erstelleLogger(String prefix) {
  return (String nachricht) => print('[$prefix] $nachricht');
}
```

1.2.3.4 Nächste Einheit

Einheit 1.3: Klassen & Konstruktoren - Klassen und Properties - Verschiedene Konstruktortypen - Getter und Setter

1.3 Einheit 1.3: Klassen & Konstruktoren

1.3.0.1 3.1 Klassen und Objekte

Dart ist eine vollständig objektorientierte Sprache — **alles ist ein Objekt**, sogar Zahlen und Funktionen.

```
class Punkt {
  // Instanzvariablen (Felder)
  double x;
  double y;

  // Konstruktor (ausführlich)
  Punkt(double x, double y) {
    this.x = x;
    this.y = y;
  }
}

void main() {
  // Objekt erstellen – 'new' ist optional seit Dart 2
  var p = Punkt(3.0, 4.0);
  print('${p.x}, ${p.y}');  // 3.0, 4.0
}
```

Vergleich: - **C++:** Ähnlich, aber keine Header-Dateien nötig, kein manuelles Speichermanagement. - **Python:** Kein `self`-Parameter nötig, Felder werden direkt in der Klasse deklariert. - **JS:** Wie ES6-Klassen, aber mit echtem Typsystem.

1.3.0.2 3.2 Shorthand-Konstruktor mit `this.param`

Die häufigste und idiomatischste Form in Dart:

```
class Punkt {
  final double x;
  final double y;

  // this.x und this.y weisen die Parameter direkt den Feldern zu
  Punkt(this.x, this.y);
}

// Das ist äquivalent zu:
class PunktLang {
  final double x;
  final double y;

  PunktLang(double x, double y)
      : this.x = x,
        this.y = y;
}
```

1.3.0.3 3.3 Benannte Konstruktoren

Dart erlaubt mehrere Konstruktoren mit unterschiedlichen Namen:

```
class Punkt {
  final double x;
  final double y;

  Punkt(this.x, this.y);

  // Benannter Konstruktor – erstellt einen Punkt auf der X-Achse
  Punkt.aufXAchse(double x) : this(x, 0);

  // Benannter Konstruktor – Ursprung
  Punkt.ursprung() : this(0, 0);

  // Benannter Konstruktor – aus Map (z.B. JSON)
  Punkt.ausMap(Map<String, double> map)
      : x = map['x'] ?? 0,
        y = map['y'] ?? 0;

  @override
  String toString() => 'Punkt($x, $y)';
}

void main() {
```

```
  var p1 = Punkt(3, 4);
  var p2 = Punkt.aufXAchse(5);     // Punkt(5, 0)
  var p3 = Punkt.ursprung();        // Punkt(0, 0)
  var p4 = Punkt.ausMap({'x': 1, 'y': 2});  // Punkt(1, 2)
}
```

1.3.0.4 3.4 Initialisierungsliste

Code, der **vor** dem Konstruktor-Body ausgeführt wird. Wichtig für `final`-Felder:

```
class Rechteck {
  final double breite;
  final double höhe;
  final double fläche;

  // Initialisierungsliste – berechnet fläche vor dem Body
  Rechteck(this.breite, this.höhe)
      : fläche = breite * höhe,
        assert(breite > 0, 'Breite muss positiv sein'),
        assert(höhe > 0, 'Höhe muss positiv sein');
}
```

1.3.0.5 3.5 Factory-Konstruktor

Ein `factory`-Konstruktor muss nicht zwingend eine neue Instanz erstellen:

```
class Logger {
  final String name;

  // Cache für bereits erstellte Logger
  static final Map<String, Logger> _cache = {};

  // Privater Konstruktor (beginnt mit _)
  Logger._intern(this.name);

  // Factory – gibt gecachte Instanz zurück oder erstellt neue
  factory Logger(String name) {
    return _cache.putIfAbsent(name, () => Logger._intern(name));
  }
}

void main() {
  var a = Logger('UI');
  var b = Logger('UI');
  print(identical(a, b));  // true – selbe Instanz aus dem Cache!
}
```

1.3.0.6 3.6 Const-Konstruktor

Erzeugt kompilierzeitkonstante Objekte — wichtig in Flutter für Performance:

```
class Farbe {
  final int rot;
  final int grün;
  final int blau;

  // Alle Felder müssen final sein für einen const-Konstruktor
  const Farbe(this.rot, this.grün, this.blau);

  // Vordefinierte Konstanten
  static const Farbe weiß = Farbe(255, 255, 255);
  static const Farbe schwarz = Farbe(0, 0, 0);
}

void main() {
  // Zwei const-Objekte mit gleichen Werten sind identisch
  const a = Farbe(255, 0, 0);
  const b = Farbe(255, 0, 0);
  print(identical(a, b));  // true – nur EINE Instanz im Speicher!
}
```

1.3.0.7 3.7 Getters und Setters

Dart hat eingebaute Unterstützung für berechnete Properties:

```
class Kreis {
  double radius;

  Kreis(this.radius);

  // Getter – wird wie ein Feld zugegriffen: kreis.fläche
  double get fläche => 3.14159 * radius * radius;
  double get umfang => 2 * 3.14159 * radius;

  // Setter – mit Validierung
  set durchmesser(double d) {
    assert(d > 0, 'Durchmesser muss positiv sein');
    radius = d / 2;
  }

  double get durchmesser => radius * 2;
}

void main() {
  var k = Kreis(5);
  print(k.fläche);         // 78.53975
  print(k.umfang);         // 31.4159

  k.durchmesser = 20;      // Setter-Aufruf
  print(k.radius);         // 10.0
}
```

1.3.0.8 3.8 Sichtbarkeit / Private Member

Dart verwendet **keine Schlüsselwörter** wie `public`, `private`. Stattdessen:

```
class Beispiel {
  String öffentlich = 'sichtbar';       // Öffentlich (Standard)
  String _privat = 'nur in Library';    // Privat auf Library-Ebene (Unterstrich)
}
```

Wichtig: `_` macht ein Mitglied privat auf **Library-Ebene** (Datei-Ebene), nicht auf Klassen-Ebene.

1.3.0.9 3.9 Static Members

```
class Zähler {
  // Statische Variable – gehört zur Klasse, nicht zur Instanz
  static int _gesamtAnzahl = 0;

  final int id;
  final String name;

  Zähler(this.name) : id = ++_gesamtAnzahl;

  // Statische Methode
  static int get gesamtAnzahl => _gesamtAnzahl;

  @override
  String toString() => 'Zähler(id: $id, name: $name)';
}

void main() {
  var a = Zähler('Alpha');
  var b = Zähler('Beta');
  print('Gesamt: ${Zähler.gesamtAnzahl}');  // Gesamt: 2
}
```

1.3.0.10 3.10 Cascade Notation (..)

Mehrere Operationen auf demselben Objekt hintereinander:

```
class Anfrage {
  String url = '';
  String methode = 'GET';
  Map<String, String> header = {};
  String? body;

  void sende() => print('$methode $url');
}

void main() {
  // MIT Cascade – elegant und kompakt
  var req = Anfrage()
```

```
    ..url = 'https://api.example.com/daten'
    ..methode = 'POST'
    ..header['Content-Type'] = 'application/json'
    ..body = '{"key": "value"}'
    ..sende();
}
```

1.3.0.11 3.11 Zusammenfassendes Beispiel

```
class Bankkonto {
  final String inhaber;
  final String _kontonummer;
  double _saldo;

  static int _kontoZähler = 0;

  // Hauptkonstruktor mit Initialisierungsliste
  Bankkonto(this.inhaber, {double startguthaben = 0})
      : _kontonummer = 'DE${++_kontoZähler}'.padLeft(10, '0'),
        _saldo = startguthaben,
        assert(startguthaben >= 0);

  // Benannter Konstruktor
  Bankkonto.ohneGuthaben(String inhaber) : this(inhaber, startguthaben: 0);

  // Getter
  String get kontonummer => _kontonummer;
  double get saldo => _saldo;

  // Methoden
  void einzahlen(double betrag) {
    assert(betrag > 0);
    _saldo += betrag;
  }

  bool abheben(double betrag) {
    if (betrag > _saldo) return false;
    _saldo -= betrag;
    return true;
  }

  @override
  String toString() => 'Konto $_kontonummer ($inhaber):
 ↪  ${_saldo.toStringAsFixed(2)} EUR';
}

void main() {
  var konto = Bankkonto('Max Mustermann', startguthaben: 1000)
    ..einzahlen(500)
    ..abheben(200);
```

```
  print(konto);  // Konto DE000000001 (Max Mustermann): 1300.00 EUR
}
```

1.3.1 Übung

1.3.1.1 Aufgabe 1: Grundlegende Klasse (15 Min.)

Erstelle eine `Person`-Klasse mit verschiedenen Konstruktoren:

```
void main() {
  // Diese Aufrufe sollten funktionieren:
  var p1 = Person('Max', 'Mustermann', 30);
  var p2 = Person.nurVorname('Anna');
  var p3 = Person.ausMap({'vorname': 'Tom', 'nachname': 'Schmidt', 'alter': 25});

  print(p1.vollerName);  // Max Mustermann
  print(p1.istErwachsen);  // true
  print(p2);  // Person: Anna (Alter unbekannt)
  print(p3);  // Person: Tom Schmidt, 25 Jahre
}

// TODO: Implementiere die Person-Klasse
// - Felder: vorname, nachname (beide final), alter (nullable)
// - Hauptkonstruktor mit this.vorname, this.nachname, this.alter
// - Benannter Konstruktor: Person.nurVorname(String vorname)
// - Benannter Konstruktor: Person.ausMap(Map<String, dynamic> map)
// - Getter: vollerName, istErwachsen
// - toString() überschreiben
```

1.3.1.2 Aufgabe 2: Factory & Cache (15 Min.)

Implementiere eine `Datenbank`-Klasse mit Singleton-Pattern:

```
void main() {
  var db1 = Datenbank('produktions_db');
  var db2 = Datenbank('produktions_db');
  var db3 = Datenbank('test_db');

  print(identical(db1, db2));  // true (gleicher Name = gleiche Instanz)
  print(identical(db1, db3));  // false (unterschiedliche Namen)

  db1.verbinde();
  print(db1.istVerbunden);  // true
  print(db2.istVerbunden);  // true (selbe Instanz!)

  print(Datenbank.alleInstanzen);  // ['produktions_db', 'test_db']
}

// TODO: Implementiere die Datenbank-Klasse
// - Privater Konstruktor
// - Factory-Konstruktor mit Cache
```

```
// - Felder: name (final), _verbunden (privat)
// - Methode: verbinde()
// - Getter: istVerbunden
// - Statischer Getter: alleInstanzen (Liste aller gecachten Namen)
```

1.3.1.3 Aufgabe 3: Const-Konstruktor (10 Min.)

Erstelle eine immutable `Vektor2D`-Klasse:

```
void main() {
  const v1 = Vektor2D(3, 4);
  const v2 = Vektor2D(3, 4);
  const ursprung = Vektor2D.null_();

  print(identical(v1, v2));  // true (const-Objekte werden dedupliziert)
  print(v1.länge);           // 5.0
  print(v1 + Vektor2D(1, 1)); // Vektor2D(4, 5)
  print(v1 * 2);             // Vektor2D(6, 8)
  print(ursprung);           // Vektor2D(0, 0)
}

// TODO: Implementiere die Vektor2D-Klasse
// - Const-Konstruktor
// - Benannter Konstruktor: Vektor2D.null_() für (0, 0)
// - Getter: länge (Betrag des Vektors: sqrt(x² + y²))
// - Operator +, *, - überladen
// - toString() überschreiben

// Hinweis: import 'dart:math' für sqrt()
```

1.3.1.4 Aufgabe 4: Getters, Setters & Validierung (15 Min.)

Erstelle eine `Temperatur`-Klasse mit Umrechnung:

```
void main() {
  var t = Temperatur.celsius(25);

  print(t.celsius);     // 25.0
  print(t.fahrenheit);  // 77.0
  print(t.kelvin);      // 298.15

  t.fahrenheit = 32;
  print(t.celsius);     // 0.0

  t.kelvin = 0;
  print(t.celsius);     // -273.15

  // Das sollte einen Fehler werfen:
  try {
    t.kelvin = -10;  // Unter absolutem Nullpunkt!
  } catch (e) {
```

```
    print('Fehler: $e');
  }
}

// TODO: Implementiere die Temperatur-Klasse
// - Intern wird die Temperatur in Kelvin gespeichert (_kelvin)
// - Getter und Setter für: celsius, fahrenheit, kelvin
// - Setter sollten bei ungültigen Werten (< 0 Kelvin) eine Exception werfen
// - Benannte Konstruktoren: Temperatur.celsius(), .fahrenheit(), .kelvin()
//
// Formeln:
// K = C + 273.15
// F = C * 9/5 + 32
```

1.3.1.5 Aufgabe 5: Cascade Notation (10 Min.)

Refaktoriere den folgenden Code mit Cascade Notation:

```
// VORHER (ohne Cascade):
void main() {
  var builder = StringBuilder();
  builder.schreibeZeile('Überschrift');
  builder.schreibeZeile('============');
  builder.schreibeZeile('');
  builder.schreibe('Absatz 1: ');
  builder.schreibeZeile('Dies ist ein Text.');
  builder.schreibeZeile('');
  builder.schreibeZeile('Ende.');
  print(builder.build());
}

class StringBuilder {
  final _buffer = StringBuffer();

  void schreibe(String text) => _buffer.write(text);
  void schreibeZeile(String text) => _buffer.writeln(text);
  String build() => _buffer.toString();
}

// TODO: Schreibe main() mit Cascade Notation um
// Der StringBuilder-Code bleibt gleich
```

1.3.1.6 Bonusaufgabe: Builder-Pattern

Implementiere einen `EmailBuilder` mit dem Builder-Pattern:

```
void main() {
  var email = EmailBuilder()
    .von('sender@mail.de')
    .an('empfänger@mail.de')
    .cc(['kopie1@mail.de', 'kopie2@mail.de'])
```

```
    .betreff('Wichtige Nachricht')
    .text('Hallo,\n\ndies ist der Inhalt.\n\nGruß')
    .build();

  print(email);
  // Von: sender@mail.de
  // An: empfänger@mail.de
  // CC: kopie1@mail.de, kopie2@mail.de
  // Betreff: Wichtige Nachricht
  // ---
  // Hallo,
  //
  // dies ist der Inhalt.
  //
  // Gruß
}

// TODO: Implementiere Email und EmailBuilder
// EmailBuilder-Methoden geben 'this' zurück für Method Chaining
```

1.3.2 Lösung

1.3.2.1 Aufgabe 1: Grundlegende Klasse

```
class Person {
  final String vorname;
  final String nachname;
  final int? alter;

  // Hauptkonstruktor
  Person(this.vorname, this.nachname, this.alter);

  // Benannte Konstruktoren
  Person.nurVorname(this.vorname)
      : nachname = '',
        alter = null;

  Person.ausMap(Map<String, dynamic> map)
      : vorname = map['vorname'] as String,
        nachname = map['nachname'] as String? ?? '',
        alter = map['alter'] as int?;

  // Getter
  String get vollerName =>
      nachname.isEmpty ? vorname : '$vorname $nachname';

  bool get istErwachsen => (alter ?? 0) >= 18;

  @override
  String toString() {
```

```
    var altersInfo = alter != null ? '$alter Jahre' : 'Alter unbekannt';
    return nachname.isEmpty
        ? 'Person: $vorname ($altersInfo)'
        : 'Person: $vollerName, $altersInfo';
  }
}

void main() {
  var p1 = Person('Max', 'Mustermann', 30);
  var p2 = Person.nurVorname('Anna');
  var p3 = Person.ausMap({'vorname': 'Tom', 'nachname': 'Schmidt', 'alter': 25});

  print(p1.vollerName);    // Max Mustermann
  print(p1.istErwachsen);  // true
  print(p2);               // Person: Anna (Alter unbekannt)
  print(p3);               // Person: Tom Schmidt, 25 Jahre
}
```

1.3.2.2 Aufgabe 2: Factory & Cache

```
class Datenbank {
  final String name;
  bool _verbunden = false;

  // Cache für Instanzen
  static final Map<String, Datenbank> _cache = {};

  // Privater Konstruktor
  Datenbank._intern(this.name);

  // Factory-Konstruktor
  factory Datenbank(String name) {
    return _cache.putIfAbsent(name, () => Datenbank._intern(name));
  }

  // Getter
  bool get istVerbunden => _verbunden;

  // Statischer Getter für alle Instanznamen
  static List<String> get alleInstanzen => _cache.keys.toList();

  // Methoden
  void verbinde() {
    _verbunden = true;
    print('Verbunden mit $name');
  }

  void trenne() {
    _verbunden = false;
    print('Getrennt von $name');
```

```
  }
}

void main() {
  var db1 = Datenbank('produktions_db');
  var db2 = Datenbank('produktions_db');
  var db3 = Datenbank('test_db');

  print(identical(db1, db2));  // true
  print(identical(db1, db3));  // false

  db1.verbinde();
  print(db1.istVerbunden);  // true
  print(db2.istVerbunden);  // true

  print(Datenbank.alleInstanzen);  // [produktions_db, test_db]
}
```

1.3.2.3 Aufgabe 3: Const-Konstruktor

```
import 'dart:math';

class Vektor2D {
  final double x;
  final double y;

  // Const-Konstruktor
  const Vektor2D(this.x, this.y);

  // Benannter Konstruktor für Nullvektor
  const Vektor2D.null_() : x = 0, y = 0;

  // Getter für die Länge
  double get länge => sqrt(x * x + y * y);

  // Operator-Überladung
  Vektor2D operator +(Vektor2D other) => Vektor2D(x + other.x, y + other.y);
  Vektor2D operator -(Vektor2D other) => Vektor2D(x - other.x, y - other.y);
  Vektor2D operator *(num skalar) => Vektor2D(x * skalar, y * skalar);
  Vektor2D operator -() => Vektor2D(-x, -y);

  @override
  String toString() => 'Vektor2D($x, $y)';

  @override
  bool operator ==(Object other) =>
      other is Vektor2D && x == other.x && y == other.y;

  @override
  int get hashCode => Object.hash(x, y);
```

```
}

void main() {
  const v1 = Vektor2D(3, 4);
  const v2 = Vektor2D(3, 4);
  const ursprung = Vektor2D.null_();

  print(identical(v1, v2));     // true
  print(v1.länge);              // 5.0
  print(v1 + Vektor2D(1, 1));   // Vektor2D(4.0, 5.0)
  print(v1 * 2);                // Vektor2D(6.0, 8.0)
  print(ursprung);              // Vektor2D(0, 0)
}
```

1.3.2.4 Aufgabe 4: Getters, Setters & Validierung

```
class Temperatur {
  double _kelvin;

  // Private Konstruktor
  Temperatur._(this._kelvin) {
    _validiere(_kelvin);
  }

  // Benannte Konstruktoren
  Temperatur.kelvin(double k) : this._(k);
  Temperatur.celsius(double c) : this._(c + 273.15);
  Temperatur.fahrenheit(double f) : this._((f - 32) * 5 / 9 + 273.15);

  // Validierung
  void _validiere(double kelvin) {
    if (kelvin < 0) {
      throw ArgumentError('Temperatur kann nicht unter 0 Kelvin liegen');
    }
  }

  // Getter
  double get kelvin => _kelvin;
  double get celsius => _kelvin - 273.15;
  double get fahrenheit => celsius * 9 / 5 + 32;

  // Setter
  set kelvin(double k) {
    _validiere(k);
    _kelvin = k;
  }

  set celsius(double c) {
    kelvin = c + 273.15;
  }
```

```
  set fahrenheit(double f) {
    celsius = (f - 32) * 5 / 9;
  }

  @override
  String toString() =>
      '${celsius.toStringAsFixed(2)}°C / ${fahrenheit.toStringAsFixed(2)}°F /  ↪
↪  ${kelvin.toStringAsFixed(2)}K';
}

void main() {
  var t = Temperatur.celsius(25);

  print(t.celsius);     // 25.0
  print(t.fahrenheit);  // 77.0
  print(t.kelvin);      // 298.15

  t.fahrenheit = 32;
  print(t.celsius);     // 0.0

  t.kelvin = 0;
  print(t.celsius);     // -273.15

  try {
    t.kelvin = -10;
  } catch (e) {
    print('Fehler: $e');
  }
}
```

1.3.2.5 Aufgabe 5: Cascade Notation

```
class StringBuilder {
  final _buffer = StringBuffer();

  void schreibe(String text) => _buffer.write(text);
  void schreibeZeile(String text) => _buffer.writeln(text);
  String build() => _buffer.toString();
}

void main() {
  var text = (StringBuilder()
        ..schreibeZeile('Überschrift')
        ..schreibeZeile('============')
        ..schreibeZeile('')
        ..schreibe('Absatz 1: ')
        ..schreibeZeile('Dies ist ein Text.')
        ..schreibeZeile('')
        ..schreibeZeile('Ende.'))
```

```
      .build();

  print(text);
}
```

1.3.2.6 Bonusaufgabe: Builder-Pattern

```
class Email {
  final String von;
  final String an;
  final List<String> cc;
  final String betreff;
  final String text;

  Email._({
    required this.von,
    required this.an,
    required this.cc,
    required this.betreff,
    required this.text,
  });

  @override
  String toString() {
    var buffer = StringBuffer()
      ..writeln('Von: $von')
      ..writeln('An: $an');

    if (cc.isNotEmpty) {
      buffer.writeln('CC: ${cc.join(', ')}');
    }

    buffer
      ..writeln('Betreff: $betreff')
      ..writeln('---')
      ..write(text);

    return buffer.toString();
  }
}

class EmailBuilder {
  String _von = '';
  String _an = '';
  List<String> _cc = [];
  String _betreff = '';
  String _text = '';

  EmailBuilder von(String adresse) {
    _von = adresse;
```

```
    return this;
  }

  EmailBuilder an(String adresse) {
    _an = adresse;
    return this;
  }

  EmailBuilder cc(List<String> adressen) {
    _cc = adressen;
    return this;
  }

  EmailBuilder betreff(String betreff) {
    _betreff = betreff;
    return this;
  }

  EmailBuilder text(String text) {
    _text = text;
    return this;
  }

  Email build() {
    if (_von.isEmpty || _an.isEmpty) {
      throw StateError('Von und An müssen gesetzt sein');
    }
    return Email._(
      von: _von,
      an: _an,
      cc: _cc,
      betreff: _betreff,
      text: _text,
    );
  }
}

void main() {
  var email = EmailBuilder()
      .von('sender@mail.de')
      .an('empfänger@mail.de')
      .cc(['kopie1@mail.de', 'kopie2@mail.de'])
      .betreff('Wichtige Nachricht')
      .text('Hallo,\n\ndies ist der Inhalt.\n\nGruß')
      .build();

  print(email);
}
```

1.3.3 Ressourcen

1.3.3.1 Offizielle Dokumentation

- Classes — Klassen in Dart
- Constructors — Konstruktoren im Detail
- Methods — Methoden, Getter, Setter

1.3.3.2 Best Practices

Konstruktor-Wahl

Situation	Konstruktor-Typ
Einfache Initialisierung	`ClassName(this.field)`
Alternative Erstellung	Benannter Konstruktor
Caching/Singleton	Factory-Konstruktor
Immutable Objekte	Const-Konstruktor
Validierung vor Init	Initialisierungsliste

Wann `final` vs. `var` für Felder?

```
// Gut: final für Felder, die sich nicht ändern
class Person {
  final String name;  // Wird nie geändert
  int alter;          // Kann sich ändern
}
```

1.3.3.3 Nächste Einheit

Einheit 1.4: Vererbung & Interfaces - extends, super - Abstract Classes - implements

1.4 Einheit 1.4: Vererbung & Interfaces

1.4.0.1 4.1 Vererbung mit `extends`

```
class Fahrzeug {
  final String marke;
  int _kilometerstand = 0;

  Fahrzeug(this.marke);

  int get kilometerstand => _kilometerstand;

  void fahre(int km) {
    _kilometerstand += km;
    print('$marke fährt $km km. Stand: $_kilometerstand km');
  }
}

class Auto extends Fahrzeug {
  final int türen;
```

```
  // super. übergibt Parameter an den Eltern-Konstruktor
  Auto(super.marke, {this.türen = 4});

  // Methode überschreiben
  @override
  void fahre(int km) {
    print('Auto startet Motor...');
    super.fahre(km);  // Eltern-Methode aufrufen
  }
}

void main() {
  var auto = Auto('BMW', türen: 4);
  auto.fahre(100);
  // Auto startet Motor...
  // BMW fährt 100 km. Stand: 100 km
}
```

Vergleich zu C++: Dart hat nur Einfachvererbung. Kein `virtual` nötig — alle Methoden sind standardmäßig überschreibbar.

1.4.0.2 4.2 Abstrakte Klassen

```
// abstract-Klassen können nicht direkt instanziiert werden
abstract class Form {
  // Abstrakte Methoden – müssen von Unterklassen implementiert werden
  double berechneFlaeche();
  double berechneUmfang();

  // Konkrete Methode – wird vererbt
  void beschreibe() {
    print('${runtimeType}: Fläche=${berechneFlaeche().toStringAsFixed(2)}');
  }
}

class Quadrat extends Form {
  final double seite;
  Quadrat(this.seite);

  @override
  double berechneFlaeche() => seite * seite;

  @override
  double berechneUmfang() => 4 * seite;
}

class Kreis extends Form {
  final double radius;
  Kreis(this.radius);
```

```
  @override
  double berechneFlaeche() => 3.14159 * radius * radius;

  @override
  double berechneUmfang() => 2 * 3.14159 * radius;
}

void main() {
  // var f = Form();  // FEHLER – abstrakte Klasse
  List<Form> formen = [Quadrat(5), Kreis(3)];
  for (var form in formen) {
    form.beschreibe();
  }
}
```

1.4.0.3 4.3 Implizite Interfaces (`implements`)

In Dart ist **jede Klasse gleichzeitig ein Interface**. Mit `implements` muss man **alle** Methoden neu implementieren:

```
class Druckbar {
  void drucke() => print('Standard-Ausgabe');
}

class Speicherbar {
  void speichere() => print('In Datei gespeichert');
}

// implements – ALLE Methoden müssen neu implementiert werden
class Dokument implements Druckbar, Speicherbar {
  final String inhalt;
  Dokument(this.inhalt);

  @override
  void drucke() => print('Dokument drucken: $inhalt');

  @override
  void speichere() => print('Dokument speichern: $inhalt');
}
```

Unterschied `extends` vs. `implements`

	`extends`	`implements`
Anzahl	Nur eine Klasse	Mehrere Klassen
Erbt Code	Ja	Nein
Konstruktoren	Werden vererbt	Nein
Konzept	“ist ein” mit Code	“verhält sich wie”

1.4.0.4 4.4 Operator Overloading

```
class Vektor {
  final double x;
  final double y;

  const Vektor(this.x, this.y);

  Vektor operator +(Vektor other) => Vektor(x + other.x, y + other.y);
  Vektor operator -(Vektor other) => Vektor(x - other.x, y - other.y);
  Vektor operator *(double skalar) => Vektor(x * skalar, y * skalar);
  Vektor operator -() => Vektor(-x, -y);

  @override
  bool operator ==(Object other) =>
      other is Vektor && x == other.x && y == other.y;

  @override
  int get hashCode => Object.hash(x, y);

  // Index-Operator
  double operator [](int index) => switch (index) {
    0 => x,
    1 => y,
    _ => throw RangeError('Index muss 0 oder 1 sein'),
  };

  @override
  String toString() => 'Vektor($x, $y)';
}

void main() {
  var a = Vektor(1, 2);
  var b = Vektor(3, 4);

  print(a + b);      // Vektor(4.0, 6.0)
  print(a * 3);      // Vektor(3.0, 6.0)
  print(a == Vektor(1, 2));  // true
  print(a[0]);       // 1.0
}
```

Überladbare Operatoren: `+`, `-`, `*`, `/`, `~/`, `%`, `==`, `<`, `>`, `<=`, `>=`, `[]`, `[]=`, `~`, `<<`, `>>`, `^`, `|`, `&`

1.4.0.5 4.5 Polymorphismus

```
abstract class Tier {
  String get name;
  void sprich();
}

class Hund extends Tier {
```

```
  @override
  String get name => 'Hund';

  @override
  void sprich() => print('Wuff!');
}

class Katze extends Tier {
  @override
  String get name => 'Katze';

  @override
  void sprich() => print('Miau!');
}

void main() {
  List<Tier> tiere = [Hund(), Katze(), Hund()];

  for (var tier in tiere) {
    print('${tier.name} sagt:');
    tier.sprich();
  }
}
```

1.4.0.6 4.6 Typprüfung und Casting

```
void verarbeite(Object obj) {
  // Typprüfung mit 'is'
  if (obj is String) {
    // Smart Cast: obj wird automatisch als String behandelt
    print('String mit ${obj.length} Zeichen');
  } else if (obj is int) {
    print('Integer: $obj');
  } else if (obj is List) {
    print('Liste mit ${obj.length} Elementen');
  }

  // Explizites Casting mit 'as'
  // var text = obj as String;  // Wirft Exception wenn nicht String

  // Sicheres Casting
  var text = obj as String?;  // null wenn nicht String
}
```

1.4.0.7 4.7 Zusammenfassendes Beispiel

```
abstract class Zahlungsmethode {
  String get name;
  bool verarbeite(double betrag);
```

```
}

class Kreditkarte implements Zahlungsmethode {
  final String kartennummer;
  double _limit;

  Kreditkarte(this.kartennummer, this._limit);

  @override
  String get name => 'Kreditkarte
 ↪  (*${kartennummer.substring(kartennummer.length - 4)})';

  @override
  bool verarbeite(double betrag) {
    if (betrag > _limit) {
      print('$name: Limit überschritten!');
      return false;
    }
    _limit -= betrag;
    print('$name: ${betrag.toStringAsFixed(2)} EUR abgebucht');
    return true;
  }
}

class PayPal implements Zahlungsmethode {
  final String email;
  double guthaben;

  PayPal(this.email, this.guthaben);

  @override
  String get name => 'PayPal ($email)';

  @override
  bool verarbeite(double betrag) {
    if (betrag > guthaben) {
      print('$name: Nicht genug Guthaben!');
      return false;
    }
    guthaben -= betrag;
    print('$name: ${betrag.toStringAsFixed(2)} EUR bezahlt');
    return true;
  }
}

void bezahle(Zahlungsmethode methode, double betrag) {
  print('Zahlung über ${methode.name}...');
  methode.verarbeite(betrag);
}

void main() {
```

```
  var karte = Kreditkarte('1234567890123456', 1000);
  var paypal = PayPal('user@mail.de', 500);

  bezahle(karte, 150);
  bezahle(paypal, 75);
}
```

1.4.1 Übung

1.4.1.1 Aufgabe 1: Vererbungshierarchie (20 Min.)

Erstelle eine Klassenhierarchie für geometrische Formen:

```
void main() {
  List<Form> formen = [
    Rechteck(5, 3),
    Quadrat(4),
    Kreis(2.5),
    Dreieck(3, 4, 5),
  ];

  for (var form in formen) {
    print('${form.name}:');
    print('  Fläche: ${form.flaeche.toStringAsFixed(2)}');
    print('  Umfang: ${form.umfang.toStringAsFixed(2)}');
  }
}

// TODO: Implementiere:
// 1. Abstrakte Klasse 'Form' mit:
//    - Abstrakter Getter: name, flaeche, umfang
//
// 2. Klasse 'Rechteck' extends Form:
//    - Felder: breite, hoehe
//    - Fläche = breite * höhe
//    - Umfang = 2 * (breite + höhe)
//
// 3. Klasse 'Quadrat' extends Rechteck:
//    - Nur ein Parameter (seite)
//    - Nutzt den Rechteck-Konstruktor
//
// 4. Klasse 'Kreis' extends Form:
//    - Feld: radius
//    - Fläche = π * r²
//    - Umfang = 2 * π * r
//
// 5. Klasse 'Dreieck' extends Form:
//    - Felder: a, b, c (Seitenlängen)
//    - Umfang = a + b + c
//    - Fläche mit Heronformel: sqrt(s*(s-a)*(s-b)*(s-c)) wobei s = Umfang/2
```

1.4.1.2 Aufgabe 2: Interface-Implementierung (15 Min.)

Implementiere ein Plugin-System mit Interfaces:

```
void main() {
  var plugins = <Plugin>[
    LoggerPlugin('app.log'),
    CachePlugin(maxGröße: 100),
    AnalyticsPlugin('UA-12345'),
  ];

  var app = App(plugins);
  app.starte();
  app.beende();
}

// TODO: Implementiere:
// 1. Abstrakte Klasse/Interface 'Plugin' mit:
//    - String get name
//    - void initialisiere()
//    - void beende()
//
// 2. Klasse 'LoggerPlugin' implements Plugin
// 3. Klasse 'CachePlugin' implements Plugin
// 4. Klasse 'AnalyticsPlugin' implements Plugin
//
// 5. Klasse 'App':
//    - Nimmt Liste von Plugins im Konstruktor
//    - starte(): Ruft initialisiere() auf allen Plugins auf
//    - beende(): Ruft beende() auf allen Plugins auf
```

1.4.1.3 Aufgabe 3: Operator Overloading (15 Min.)

Erstelle eine `Bruch`-Klasse mit Operatoren:

```
void main() {
  var a = Bruch(1, 2);  // 1/2
  var b = Bruch(1, 3);  // 1/3

  print('$a + $b = ${a + b}');  // 1/2 + 1/3 = 5/6
  print('$a - $b = ${a - b}');  // 1/2 - 1/3 = 1/6
  print('$a * $b = ${a * b}');  // 1/2 * 1/3 = 1/6
  print('$a / $b = ${a / b}');  // 1/2 / 1/3 = 3/2

  print('$a == ${Bruch(2, 4)}: ${a == Bruch(2, 4)}');  // true (gekürzt gleich)
  print('$a > $b: ${a > b}');  // true
  print('$a < $b: ${a < b}');  // false

  print('${Bruch(6, 8).kuerze()}');  // 3/4
}

// TODO: Implementiere Bruch-Klasse mit:
// - Felder: zaehler, nenner
```

```
// - Operatoren: +, -, *, /, ==, <, >, <=, >=
// - Methode: kuerze() (gibt gekürzten Bruch zurück)
// - toString(): "zaehler/nenner"
//
// Formeln:
// a/b + c/d = (a*d + c*b) / (b*d)
// a/b - c/d = (a*d - c*b) / (b*d)
// a/b * c/d = (a*c) / (b*d)
// a/b / c/d = (a*d) / (b*c)
//
// Hinweis: Nutze den GGT (größter gemeinsamer Teiler) zum Kürzen
```

1.4.1.4 Aufgabe 4: Polymorphismus & Typprüfung (10 Min.)

Implementiere ein Nachrichtensystem:

```
void main() {
  List<Nachricht> nachrichten = [
    TextNachricht('Hallo Welt!'),
    BildNachricht('foto.jpg', 1024 * 500),
    AudioNachricht('sprachnachricht.mp3', Duration(seconds: 30)),
    TextNachricht('Wie gehts?'),
  ];

  var stats = analysiereNachrichten(nachrichten);
  print('Statistik: $stats');
  // {text: 2, bild: 1, audio: 1, gesamtGröße: 512000, gesamtDauer: 0:00:30}
}

// TODO: Implementiere:
// 1. Abstrakte Klasse 'Nachricht' mit:
//    - DateTime get zeitstempel
//    - String get vorschau
//
// 2. TextNachricht, BildNachricht, AudioNachricht
//
// 3. Funktion analysiereNachrichten:
//    - Zählt jeden Nachrichtentyp
//    - Summiert Größe aller Bilder
//    - Summiert Dauer aller Audio-Nachrichten
//    - Nutzt 'is' für Typprüfung
```

1.4.1.5 Bonusaufgabe: Mehrfache Interfaces

Implementiere ein Dateisystem mit mehreren Interfaces:

```
void main() {
  var dateien = <Datei>[
    TextDatei('readme.txt', 'Hallo Welt'),
    BildDatei('foto.png', 1920, 1080),
    Ordner('dokumente', [
```

```
      TextDatei('notes.txt', 'Notizen...'),
    ]),
  ];

  for (var datei in dateien) {
    print(datei.info);

    if (datei is Druckbar) {
      datei.drucke();
    }

    if (datei is Komprimierbar) {
      print('Komprimiert: ${datei.komprimiere()} Bytes');
    }
  }
}

// TODO: Implementiere:
// - Interface 'Druckbar' mit drucke()
// - Interface 'Komprimierbar' mit int komprimiere()
// - Abstrakte Klasse 'Datei' mit name, größe, info
// - TextDatei implements Druckbar, Komprimierbar
// - BildDatei implements Komprimierbar
// - Ordner (enthält Liste von Dateien, größe = Summe aller Inhalte)
```

1.4.2 Lösung

1.4.2.1 Aufgabe 1: Vererbungshierarchie

```
import 'dart:math';

abstract class Form {
  String get name;
  double get flaeche;
  double get umfang;
}

class Rechteck extends Form {
  final double breite;
  final double hoehe;

  Rechteck(this.breite, this.hoehe);

  @override
  String get name => 'Rechteck (${breite}x$hoehe)';

  @override
  double get flaeche => breite * hoehe;

  @override
```

```
  double get umfang => 2 * (breite + hoehe);
}

class Quadrat extends Rechteck {
  Quadrat(double seite) : super(seite, seite);

  @override
  String get name => 'Quadrat (${breite}x$breite)';
}

class Kreis extends Form {
  final double radius;

  Kreis(this.radius);

  @override
  String get name => 'Kreis (r=$radius)';

  @override
  double get flaeche => pi * radius * radius;

  @override
  double get umfang => 2 * pi * radius;
}

class Dreieck extends Form {
  final double a, b, c;

  Dreieck(this.a, this.b, this.c);

  @override
  String get name => 'Dreieck ($a, $b, $c)';

  @override
  double get umfang => a + b + c;

  @override
  double get flaeche {
    var s = umfang / 2;
    return sqrt(s * (s - a) * (s - b) * (s - c));
  }
}

void main() {
  List<Form> formen = [
    Rechteck(5, 3),
    Quadrat(4),
    Kreis(2.5),
    Dreieck(3, 4, 5),
  ];
```

```
  for (var form in formen) {
    print('${form.name}:');
    print('  Fläche: ${form.flaeche.toStringAsFixed(2)}');
    print('  Umfang: ${form.umfang.toStringAsFixed(2)}');
  }
}
```

1.4.2.2 Aufgabe 2: Interface-Implementierung

```
abstract class Plugin {
  String get name;
  void initialisiere();
  void beende();
}

class LoggerPlugin implements Plugin {
  final String dateiname;

  LoggerPlugin(this.dateiname);

  @override
  String get name => 'Logger';

  @override
  void initialisiere() => print('[$name] Öffne $dateiname');

  @override
  void beende() => print('[$name] Schließe $dateiname');
}

class CachePlugin implements Plugin {
  final int maxGröße;

  CachePlugin({required this.maxGröße});

  @override
  String get name => 'Cache';

  @override
  void initialisiere() => print('[$name] Cache initialisiert (max: $maxGröße
↪  MB)');

  @override
  void beende() => print('[$name] Cache geleert');
}

class AnalyticsPlugin implements Plugin {
  final String trackingId;

  AnalyticsPlugin(this.trackingId);
```

```
  @override
  String get name => 'Analytics';

  @override
  void initialisiere() => print('[$name] Tracking gestartet: $trackingId');

  @override
  void beende() => print('[$name] Session beendet');
}

class App {
  final List<Plugin> _plugins;

  App(this._plugins);

  void starte() {
    print('=== App startet ===');
    for (var plugin in _plugins) {
      plugin.initialisiere();
    }
    print('=== App bereit ===\n');
  }

  void beende() {
    print('\n=== App beendet ===');
    for (var plugin in _plugins.reversed) {
      plugin.beende();
    }
  }
}

void main() {
  var plugins = <Plugin>[
    LoggerPlugin('app.log'),
    CachePlugin(maxGröße: 100),
    AnalyticsPlugin('UA-12345'),
  ];

  var app = App(plugins);
  app.starte();
  app.beende();
}
```

1.4.2.3 Aufgabe 3: Operator Overloading

```
class Bruch implements Comparable<Bruch> {
  final int zaehler;
  final int nenner;
```

```
Bruch(this.zaehler, this.nenner) : assert(nenner != 0);

// GGT mit Euklids Algorithmus
static int _ggt(int a, int b) {
  a = a.abs();
  b = b.abs();
  while (b != 0) {
    var t = b;
    b = a % b;
    a = t;
  }
  return a;
}

Bruch kuerze() {
  var teiler = _ggt(zaehler, nenner);
  var z = zaehler ~/ teiler;
  var n = nenner ~/ teiler;
  // Vorzeichen normalisieren
  if (n < 0) {
    z = -z;
    n = -n;
  }
  return Bruch(z, n);
}

double get dezimal => zaehler / nenner;

Bruch operator +(Bruch other) =>
    Bruch(zaehler * other.nenner + other.zaehler * nenner,
          nenner * other.nenner).kuerze();

Bruch operator -(Bruch other) =>
    Bruch(zaehler * other.nenner - other.zaehler * nenner,
          nenner * other.nenner).kuerze();

Bruch operator *(Bruch other) =>
    Bruch(zaehler * other.zaehler, nenner * other.nenner).kuerze();

Bruch operator /(Bruch other) =>
    Bruch(zaehler * other.nenner, nenner * other.zaehler).kuerze();

@override
bool operator ==(Object other) {
  if (other is! Bruch) return false;
  var a = kuerze();
  var b = other.kuerze();
  return a.zaehler == b.zaehler && a.nenner == b.nenner;
}

@override
```

```
  int get hashCode => Object.hash(kuerze().zaehler, kuerze().nenner);

  @override
  int compareTo(Bruch other) => dezimal.compareTo(other.dezimal);

  bool operator <(Bruch other) => compareTo(other) < 0;
  bool operator >(Bruch other) => compareTo(other) > 0;
  bool operator <=(Bruch other) => compareTo(other) <= 0;
  bool operator >=(Bruch other) => compareTo(other) >= 0;

  @override
  String toString() => '$zaehler/$nenner';
}

void main() {
  var a = Bruch(1, 2);
  var b = Bruch(1, 3);

  print('$a + $b = ${a + b}');
  print('$a - $b = ${a - b}');
  print('$a * $b = ${a * b}');
  print('$a / $b = ${a / b}');
  print('$a == ${Bruch(2, 4)}: ${a == Bruch(2, 4)}');
  print('$a > $b: ${a > b}');
  print('${Bruch(6, 8).kuerze()}');
}
```

1.4.2.4 Aufgabe 4: Polymorphismus & Typprüfung

```
abstract class Nachricht {
  final DateTime zeitstempel = DateTime.now();
  String get vorschau;
}

class TextNachricht extends Nachricht {
  final String text;
  TextNachricht(this.text);

  @override
  String get vorschau => text.length > 20 ? '${text.substring(0, 20)}...' : text;
}

class BildNachricht extends Nachricht {
  final String dateiname;
  final int größeBytes;

  BildNachricht(this.dateiname, this.größeBytes);

  @override
  String get vorschau => '□ $dateiname';
```

```
}

class AudioNachricht extends Nachricht {
  final String dateiname;
  final Duration dauer;

  AudioNachricht(this.dateiname, this.dauer);

  @override
  String get vorschau => '□ ${dauer.inSeconds}s';
}

Map<String, dynamic> analysiereNachrichten(List<Nachricht> nachrichten) {
  var textCount = 0;
  var bildCount = 0;
  var audioCount = 0;
  var gesamtGröße = 0;
  var gesamtDauer = Duration.zero;

  for (var n in nachrichten) {
    if (n is TextNachricht) {
      textCount++;
    } else if (n is BildNachricht) {
      bildCount++;
      gesamtGröße += n.größeBytes;
    } else if (n is AudioNachricht) {
      audioCount++;
      gesamtDauer += n.dauer;
    }
  }

  return {
    'text': textCount,
    'bild': bildCount,
    'audio': audioCount,
    'gesamtGröße': gesamtGröße,
    'gesamtDauer': gesamtDauer,
  };
}

void main() {
  List<Nachricht> nachrichten = [
    TextNachricht('Hallo Welt!'),
    BildNachricht('foto.jpg', 1024 * 500),
    AudioNachricht('sprachnachricht.mp3', Duration(seconds: 30)),
    TextNachricht('Wie gehts?'),
  ];

  var stats = analysiereNachrichten(nachrichten);
  print('Statistik: $stats');
}
```

1.4.3 Ressourcen

1.4.3.1 Offizielle Dokumentation

- Extend a class
- Implicit interfaces

1.4.3.2 Nächste Einheit

Einheit 1.5: Mixins & Extensions

1.5 Einheit 1.5: Mixins & Extensions

1.5.0.1 5.1 Mixins — Code-Wiederverwendung ohne Vererbung

Mixins sind Darts Lösung für Code-Wiederverwendung ohne Mehrfachvererbung:

```
mixin Schwimmfähig {
  void schwimme() => print('$runtimeType schwimmt');
}

mixin Fliegbar {
  void fliege() => print('$runtimeType fliegt');
}

mixin Laufbar {
  void laufe() => print('$runtimeType läuft');
}

class Tier {
  final String name;
  Tier(this.name);
}

// Mixins mit 'with' einbinden
class Ente extends Tier with Schwimmfähig, Fliegbar, Laufbar {
  Ente(super.name);
}

class Fisch extends Tier with Schwimmfähig {
  Fisch(super.name);
}

void main() {
  var ente = Ente('Donald');
  ente.schwimme();  // Ente schwimmt
  ente.fliege();    // Ente fliegt
  ente.laufe();     // Ente läuft
}
```

1.5.0.2 5.2 Mixin mit `on`-Einschränkung

```
class Musiker {
  void spieleInstrument() => print('Spielt Instrument');
}

// Mixin kann NUR auf Musiker-Unterklassen angewendet werden
mixin Sänger on Musiker {
  void singe() {
    print('Singt');
    spieleInstrument();  // Kann Musiker-Methoden nutzen
  }
}

class Rockstar extends Musiker with Sänger {}

// class Fan with Sänger {}  // FEHLER — Fan ist kein Musiker
```

1.5.0.3 5.3 Extension Methods

Bestehenden Klassen neue Methoden hinzufügen, ohne sie zu verändern:

```
extension StringErweiterungen on String {
  bool get istEmail => contains('@') && contains('.');

  String wiederhole(int n) => List.filled(n, this).join(' ');

  String get großAnfang {
    if (isEmpty) return this;
    return '${this[0].toUpperCase()}${substring(1)}';
  }
}

extension IntErweiterungen on int {
  Duration get sekunden => Duration(seconds: this);
  Duration get minuten => Duration(minutes: this);

  bool get istPrimzahl {
    if (this < 2) return false;
    for (var i = 2; i * i <= this; i++) {
      if (this % i == 0) return false;
    }
    return true;
  }
}

void main() {
  print('test@mail.de'.istEmail);  // true
  print('ha'.wiederhole(3));       // ha ha ha
  print('dart'.großAnfang);        // Dart
  print(7.istPrimzahl);            // true
  print(5.sekunden);               // 0:00:05.000000
```

```
}
```

1.5.0.4 5.4 Enhanced Enums (Dart 2.17+)

Enums mit Feldern, Methoden und Interfaces:

```
enum Planet implements Comparable<Planet> {
  merkur(masseKg: 3.303e+23, radiusM: 2.4397e6),
  venus(masseKg: 4.869e+24, radiusM: 6.0518e6),
  erde(masseKg: 5.976e+24, radiusM: 6.37814e6),
  mars(masseKg: 6.421e+23, radiusM: 3.3972e6);

  final double masseKg;
  final double radiusM;

  const Planet({required this.masseKg, required this.radiusM});

  double get oberflächengravitation =>
      6.67300E-11 * masseKg / (radiusM * radiusM);

  @override
  int compareTo(Planet other) => masseKg.compareTo(other.masseKg);
}

enum HttpStatus {
  ok(200, 'OK'),
  notFound(404, 'Not Found'),
  serverError(500, 'Server Error');

  final int code;
  final String nachricht;

  const HttpStatus(this.code, this.nachricht);

  bool get istErfolgreich => code >= 200 && code < 300;

  @override
  String toString() => '$code $nachricht';
}

void main() {
  print(HttpStatus.notFound);           // 404 Not Found
  print(HttpStatus.ok.istErfolgreich);  // true
}
```

1.5.0.5 5.5 Zusammenfassendes Beispiel

```
mixin Serialisierbar {
  Map<String, dynamic> toJson();
  String toJsonString() => toJson().toString();
```

```
}

mixin Validierbar {
  List<String> validiere();
  bool get istGültig => validiere().isEmpty;
}

extension ListErweiterungen<T> on List<T> {
  T? get erstesOderNull => isEmpty ? null : first;
  List<T> ohneNull() => where((e) => e != null).toList();
}

class Benutzer with Serialisierbar, Validierbar {
  String name;
  String email;
  int? alter;

  Benutzer({required this.name, required this.email, this.alter});

  @override
  Map<String, dynamic> toJson() => {
    'name': name,
    'email': email,
    if (alter != null) 'alter': alter,
  };

  @override
  List<String> validiere() {
    var fehler = <String>[];
    if (name.isEmpty) fehler.add('Name fehlt');
    if (!email.contains('@')) fehler.add('Email ungültig');
    if (alter != null && alter! < 0) fehler.add('Alter ungültig');
    return fehler;
  }
}

void main() {
  var user = Benutzer(name: 'Max', email: 'max@mail.de', alter: 25);
  print(user.istGültig);      // true
  print(user.toJsonString()); // {name: Max, email: max@mail.de, alter: 25}
}
```

1.5.1 Übung

1.5.1.1 Aufgabe 1: Mixins für Spielcharaktere (20 Min.)

```
void main() {
  var krieger = Krieger('Conan');
  var magier = Magier('Gandalf');
  var paladin = Paladin('Arthas');
```

```
  krieger.angreifen();
  magier.zaubern();
  paladin.angreifen();
  paladin.heilen();
  paladin.zaubern();
}

// TODO: Implementiere Mixins:
// - Kämpfer: angreifen(), verteidigen()
// - Heiler: heilen()
// - Zauberer: zaubern(), manaRegenerieren()

// TODO: Implementiere Klassen:
// - Charakter (Basisklasse mit name)
// - Krieger extends Charakter with Kämpfer
// - Magier extends Charakter with Zauberer
// - Paladin extends Charakter with Kämpfer, Heiler, Zauberer
```

1.5.1.2 Aufgabe 2: Extension Methods (20 Min.)

```
void main() {
  // String Extensions
  print('hello world'.titleCase);        // Hello World
  print('hello'.reverse);                // olleh
  print('12345'.nurZiffern);             // true
  print('abc123'.nurZiffern);            // false

  // List Extensions
  var zahlen = [1, 2, 3, 4, 5];
  print(zahlen.summe);                   // 15
  print(zahlen.durchschnitt);            // 3.0
  print(zahlen.zweiteHälfte);            // [4, 5]

  // DateTime Extensions
  var heute = DateTime.now();
  print(heute.istWochenende);
  print(heute.deutschesDatum);           // 14.02.2026

  // int Extensions
  print(42.toRömisch);                   // XLII
}

// TODO: Implementiere alle Extensions
```

1.5.1.3 Aufgabe 3: Enhanced Enum (15 Min.)

```
void main() {
  var status = BestellStatus.versendet;
```

```
  print(status.label);                    // Versendet
  print(status.istAbgeschlossen);         // false
  print(status.nächsterStatus);           // BestellStatus.geliefert
  print(status.icon);                     // □

  // Alle Status durchgehen
  for (var s in BestellStatus.values) {
    print('${s.icon} ${s.label}');
  }
}

// TODO: Implementiere BestellStatus enum:
// - neu (icon: □, label: "Neu")
// - bezahlt (icon: □, label: "Bezahlt")
// - versendet (icon: □, label: "Versendet")
// - geliefert (icon: □, label: "Geliefert")
// - storniert (icon: □, label: "Storniert")
//
// - Getter: istAbgeschlossen (geliefert oder storniert)
// - Getter: nächsterStatus (null wenn kein nächster)
```

1.5.1.4 Bonusaufgabe: Mixin mit on-Einschränkung

```
void main() {
  var ordner = DateiOrdner('Dokumente', [
    TextDatei('readme.txt', 100),
    BildDatei('foto.jpg', 5000),
  ]);

  ordner.komprimiere();
  ordner.speichere();
  ordner.lade();
}

// TODO:
// - Abstrakte Klasse Datei mit name, größe
// - Mixin Komprimierbar on Datei mit komprimiere()
// - Mixin Speicherbar on Datei mit speichere(), lade()
// - TextDatei, BildDatei, DateiOrdner mit passenden Mixins
```

1.5.2 Lösung

1.5.2.1 Aufgabe 1: Mixins für Spielcharaktere

```
mixin Kämpfer {
  void angreifen() => print('$runtimeType greift an!');
  void verteidigen() => print('$runtimeType verteidigt sich!');
}
```

```
mixin Heiler {
  void heilen() => print('$runtimeType heilt!');
}

mixin Zauberer {
  void zaubern() => print('$runtimeType zaubert!');
  void manaRegenerieren() => print('$runtimeType regeneriert Mana');
}

class Charakter {
  final String name;
  Charakter(this.name);
}

class Krieger extends Charakter with Kämpfer {
  Krieger(super.name);
}

class Magier extends Charakter with Zauberer {
  Magier(super.name);
}

class Paladin extends Charakter with Kämpfer, Heiler, Zauberer {
  Paladin(super.name);
}

void main() {
  var krieger = Krieger('Conan');
  var magier = Magier('Gandalf');
  var paladin = Paladin('Arthas');

  krieger.angreifen();
  magier.zaubern();
  paladin.angreifen();
  paladin.heilen();
  paladin.zaubern();
}
```

1.5.2.2 Aufgabe 2: Extension Methods

```
extension StringExtensions on String {
  String get titleCase => split(' ')
      .map((w) => w.isEmpty ? '' : '${w[0].toUpperCase()}${w.substring(1)}')
      .join(' ');

  String get reverse => split('').reversed.join();

  bool get nurZiffern => isNotEmpty && split('').every((c) =>                    ↪
↪  '0123456789'.contains(c));
}
```

```
extension ListIntExtensions on List<int> {
  int get summe => fold(0, (a, b) => a + b);
  double get durchschnitt => isEmpty ? 0 : summe / length;
  List<int> get zweiteHälfte => sublist(length ~/ 2);
}

extension DateTimeExtensions on DateTime {
  bool get istWochenende => weekday == DateTime.saturday || weekday ==
 ↪  DateTime.sunday;
  String get deutschesDatum => '${day.toString().padLeft(2, '0')}.'
      '${month.toString().padLeft(2, '0')}.$year';
}

extension IntExtensions on int {
  String get toRömisch {
    if (this <= 0 || this > 3999) return toString();

    const römisch = [
      (1000, 'M'), (900, 'CM'), (500, 'D'), (400, 'CD'),
      (100, 'C'), (90, 'XC'), (50, 'L'), (40, 'XL'),
      (10, 'X'), (9, 'IX'), (5, 'V'), (4, 'IV'), (1, 'I'),
    ];

    var ergebnis = StringBuffer();
    var rest = this;

    for (var (wert, zeichen) in römisch) {
      while (rest >= wert) {
        ergebnis.write(zeichen);
        rest -= wert;
      }
    }
    return ergebnis.toString();
  }
}

void main() {
  print('hello world'.titleCase);
  print('hello'.reverse);
  print('12345'.nurZiffern);
  print([1, 2, 3, 4, 5].summe);
  print(DateTime.now().deutschesDatum);
  print(42.toRömisch);
}
```

1.5.2.3 Aufgabe 3: Enhanced Enum

```
enum BestellStatus {
  neu(icon: '', label: 'Neu'),
```

```
  bezahlt(icon: '', label: 'Bezahlt'),
  versendet(icon: '', label: 'Versendet'),
  geliefert(icon: '', label: 'Geliefert'),
  storniert(icon: '', label: 'Storniert');

  final String icon;
  final String label;

  const BestellStatus({required this.icon, required this.label});

  bool get istAbgeschlossen => this == geliefert || this == storniert;

  BestellStatus? get nächsterStatus => switch (this) {
    neu => bezahlt,
    bezahlt => versendet,
    versendet => geliefert,
    _ => null,
  };
}

void main() {
  var status = BestellStatus.versendet;
  print('${status.icon} ${status.label}');
  print('Abgeschlossen: ${status.istAbgeschlossen}');
  print('Nächster: ${status.nächsterStatus}');
}
```

1.5.3 Ressourcen

1.5.3.1 Offizielle Dokumentation

- Mixins
- Extension Methods
- Enums

1.5.3.2 Nächste Einheit

Einheit 1.6: Futures & async/await

1.6 Einheit 1.6: Futures & async/await

1.6.0.1 6.1 Das Event-Loop-Modell

Dart ist **single-threaded** mit einem **Event Loop**, ähnlich wie JavaScript:

```
void main() {
  print('1: Start');
  Future(() => print('3: Future'));
  print('2: Ende des synchronen Codes');
}
// Ausgabe: 1, 2, 3
```

1.6.0.2 6.2 Futures

Ein `Future<T>` repräsentiert einen Wert, der **in der Zukunft** verfügbar sein wird:

```
// Future erstellen
Future<String> holeDaten() {
  return Future.delayed(Duration(seconds: 1), () => 'Daten geladen');
}

// Future.value – sofort erfüllt
Future<int> sofort() => Future.value(42);

// Future.error – sofort fehlgeschlagen
Future<int> fehler() => Future.error('Fehler!');
```

Futures verketten

```
void main() {
  holeDaten()
      .then((daten) => print('Erhalten: $daten'))
      .catchError((e) => print('Fehler: $e'))
      .whenComplete(() => print('Fertig'));
}
```

1.6.0.3 6.3 async / await

Syntaktischer Zucker für lesbaren asynchronen Code:

```
Future<void> hauptprogramm() async {
  print('Start');

  try {
    final daten = await holeDaten();
    print('Daten: $daten');

    final verarbeitet = await verarbeite(daten);
    print('Verarbeitet: $verarbeitet');
  } catch (e) {
    print('Fehler: $e');
  }
}

Future<String> holeDaten() async {
  await Future.delayed(Duration(seconds: 1));
  return 'API-Antwort';
}

Future<String> verarbeite(String daten) async {
  await Future.delayed(Duration(milliseconds: 500));
  return daten.toUpperCase();
}
```

1.6.0.4 6.4 Parallele Futures mit `Future.wait`

```
Future<void> parallelLaden() async {
  // Alle Futures starten gleichzeitig
  var ergebnisse = await Future.wait([
    Future.delayed(Duration(seconds: 2), () => 'A'),
    Future.delayed(Duration(seconds: 1), () => 'B'),
    Future.delayed(Duration(seconds: 3), () => 'C'),
  ]);

  // Gesamtzeit: ~3 Sekunden (nicht 6!)
  print(ergebnisse);  // [A, B, C]
}
```

`Future.any` — Das schnellste gewinnt

```
var schnellstes = await Future.any([
  Future.delayed(Duration(seconds: 3), () => 'Langsam'),
  Future.delayed(Duration(seconds: 1), () => 'Schnell'),
]);
print(schnellstes);  // Schnell
```

1.6.0.5 6.5 Fehlerbehandlung

```
Future<void> mitFehler() async {
  try {
    var ergebnis = await riskanteOperation();
    print('OK: $ergebnis');
  } on FormatException catch (e) {
    print('Format-Fehler: $e');
  } on TimeoutException {
    print('Timeout!');
  } catch (e, stackTrace) {
    print('Fehler: $e');
    print('Stack: $stackTrace');
  } finally {
    print('Aufräumen...');
  }
}
```

1.6.0.6 6.6 Zusammenfassendes Beispiel

```
class ApiClient {
  Future<Map<String, dynamic>> get(String url) async {
    print('GET $url');
    await Future.delayed(Duration(milliseconds: 500));
    return {'status': 'ok', 'data': 'Antwort von $url'};
  }
}
```

```
Future<void> ladeAlles() async {
  var api = ApiClient();

  // Parallel laden
  var [benutzer, einstellungen] = await Future.wait([
    api.get('/user'),
    api.get('/settings'),
  ]);

  print('Benutzer: $benutzer');
  print('Einstellungen: $einstellungen');
}

void main() async {
  await ladeAlles();
}
```

1.6.1 Übung

1.6.1.1 Aufgabe 1: Basis-Futures (15 Min.)

```
void main() async {
  // TODO: Implementiere holeBenutzername(int id)
  // - Simuliere 500ms Verzögerung
  // - Gib "Benutzer_$id" zurück
  // - Werfe Exception wenn id < 0

  print(await holeBenutzername(1));   // Benutzer_1
  print(await holeBenutzername(42));  // Benutzer_42

  try {
    print(await holeBenutzername(-1));
  } catch (e) {
    print('Fehler: $e');
  }
}
```

1.6.1.2 Aufgabe 2: Verkettete Futures (15 Min.)

```
void main() async {
  // Simuliere: Login -> Profil laden -> Einstellungen laden
  // Jeder Schritt hängt vom vorherigen ab

  var benutzer = await login('max@mail.de', 'geheim');
  print('Eingeloggt: ${benutzer['name']}');

  var profil = await ladeProfil(benutzer['id'] as int);
  print('Profil: $profil');

  var settings = await ladeEinstellungen(benutzer['id'] as int);
```

```
  print('Einstellungen: $settings');
}

// TODO: Implementiere login(), ladeProfil(), ladeEinstellungen()
// Jede Funktion mit ~300ms Verzögerung
```

1.6.1.3 Aufgabe 3: Parallele Futures (15 Min.)

```
void main() async {
  var start = DateTime.now();

  // TODO: Lade alle drei Ressourcen PARALLEL mit Future.wait
  // Jede dauert 1 Sekunde
  // Gesamtzeit sollte ~1 Sekunde sein (nicht 3)

  var [wetter, nachrichten, aktien] = await Future.wait([
    ladeWetter(),
    ladeNachrichten(),
    ladeAktien(),
  ]);

  var dauer = DateTime.now().difference(start);
  print('Geladen in ${dauer.inMilliseconds}ms');
  print('Wetter: $wetter');
  print('Nachrichten: $nachrichten');
  print('Aktien: $aktien');
}

// TODO: Implementiere die drei Lade-Funktionen
```

1.6.1.4 Aufgabe 4: Timeout & Retry (15 Min.)

```
void main() async {
  // TODO: Implementiere mitTimeout<T>(Future<T> future, Duration timeout)
  // - Gibt das Ergebnis zurück wenn rechtzeitig
  // - Wirft TimeoutException wenn zu langsam

  try {
    var schnell = await mitTimeout(
      Future.delayed(Duration(milliseconds: 100), () => 'OK'),
      Duration(milliseconds: 500),
    );
    print('Schnell: $schnell');

    var langsam = await mitTimeout(
      Future.delayed(Duration(seconds: 2), () => 'Zu spät'),
      Duration(milliseconds: 500),
    );
    print('Langsam: $langsam');
```

```
  } on TimeoutException {
    print('Timeout!');
  }

  // BONUS: Implementiere retry<T>(Future<T> Function() fn, int versuche)
}
```

1.6.1.5 Bonusaufgabe: API-Client mit Cache

```
void main() async {
  var client = CachingApiClient();

  // Erster Aufruf: Lädt von "Server"
  print(await client.get('/users'));  // Lädt...

  // Zweiter Aufruf: Aus Cache
  print(await client.get('/users'));  // Sofort!

  // Cache leeren
  client.leereCache();
  print(await client.get('/users'));  // Lädt wieder...
}

// TODO: Implementiere CachingApiClient
// - get(String url) cached Ergebnisse
// - leereCache() löscht den Cache
```

1.6.2 Lösung

1.6.2.1 Aufgabe 1

```
Future<String> holeBenutzername(int id) async {
  if (id < 0) throw ArgumentError('ID muss positiv sein');
  await Future.delayed(Duration(milliseconds: 500));
  return 'Benutzer_$id';
}
```

1.6.2.2 Aufgabe 2

```
Future<Map<String, dynamic>> login(String email, String passwort) async {
  await Future.delayed(Duration(milliseconds: 300));
  return {'id': 1, 'name': 'Max', 'email': email};
}

Future<Map<String, dynamic>> ladeProfil(int userId) async {
  await Future.delayed(Duration(milliseconds: 300));
  return {'bio': 'Entwickler', 'avatar': 'avatar.png'};
}
```

```
Future<Map<String, dynamic>> ladeEinstellungen(int userId) async {
  await Future.delayed(Duration(milliseconds: 300));
  return {'theme': 'dark', 'notifications': true};
}
```

1.6.2.3 Aufgabe 3

```
Future<String> ladeWetter() async {
  await Future.delayed(Duration(seconds: 1));
  return 'Sonnig, 22°C';
}

Future<List<String>> ladeNachrichten() async {
  await Future.delayed(Duration(seconds: 1));
  return ['Nachricht 1', 'Nachricht 2'];
}

Future<Map<String, double>> ladeAktien() async {
  await Future.delayed(Duration(seconds: 1));
  return {'AAPL': 150.0, 'GOOGL': 2800.0};
}
```

1.6.2.4 Aufgabe 4

```
import 'dart:async';

Future<T> mitTimeout<T>(Future<T> future, Duration timeout) {
  return future.timeout(timeout);
}

// Alternativ manuell:
Future<T> mitTimeoutManuell<T>(Future<T> future, Duration timeout) async {
  var ergebnis = await Future.any([
    future,
    Future.delayed(timeout, () => throw TimeoutException('Timeout')),
  ]);
  return ergebnis as T;
}

// Bonus: Retry
Future<T> retry<T>(Future<T> Function() fn, int versuche) async {
  for (var i = 0; i < versuche; i++) {
    try {
      return await fn();
    } catch (e) {
      if (i == versuche - 1) rethrow;
      await Future.delayed(Duration(milliseconds: 100 * (i + 1)));
    }
  }
```

```
  throw StateError('Unreachable');
}
```

1.6.2.5 Bonusaufgabe

```
class CachingApiClient {
  final Map<String, dynamic> _cache = {};

  Future<dynamic> get(String url) async {
    if (_cache.containsKey(url)) {
      print('Cache-Hit: $url');
      return _cache[url];
    }

    print('Lade: $url');
    await Future.delayed(Duration(milliseconds: 500));
    var daten = {'url': url, 'data': 'Response'};
    _cache[url] = daten;
    return daten;
  }

  void leereCache() => _cache.clear();
}
```

1.6.3 Ressourcen

1.6.3.1 Offizielle Dokumentation

- Asynchronous programming
- Futures and error handling

1.6.3.2 Nächste Einheit

Einheit 1.7: Streams

1.7 Einheit 1.7: Streams

1.7.0.1 7.1 Was sind Streams?

- `Future<T>` = **ein** asynchroner Wert
- `Stream<T>` = **mehrere** asynchrone Werte über die Zeit

```
// Stream aus Liste
var stream = Stream.fromIterable([1, 2, 3, 4, 5]);

stream.listen(
  (wert) => print('Wert: $wert'),
  onDone: () => print('Fertig'),
  onError: (e) => print('Fehler: $e'),
);
```

1.7.0.2 7.2 Stream-Typen

Typ	Beschreibung	Beispiel
Single-Subscription	Nur ein Listener	Datei lesen
Broadcast	Mehrere Listener	UI-Events

```
// Broadcast Stream
var controller = StreamController<int>.broadcast();
controller.stream.listen((v) => print('A: $v'));
controller.stream.listen((v) => print('B: $v'));
controller.add(1);  // A: 1, B: 1
```

1.7.0.3 7.3 StreamController

```
import 'dart:async';

var controller = StreamController<String>();

// Stream abonnieren
controller.stream.listen((daten) => print('Empfangen: $daten'));

// Werte einspeisen
controller.add('Hallo');
controller.add('Welt');

// Stream schließen
controller.close();
```

1.7.0.4 7.4 async* und yield

```
// Stream-Generator
Stream<int> zähle(int bis) async* {
  for (var i = 1; i <= bis; i++) {
    await Future.delayed(Duration(milliseconds: 500));
    yield i;
  }
}

// yield* delegiert an anderen Stream
Stream<int> kombiniereStreams() async* {
  yield* zähle(3);
  yield 0;
  yield* zähle(3);
}

void main() async {
  await for (var zahl in zähle(5)) {
    print(zahl);
```

```
  }
}
```

1.7.0.5 7.5 Stream-Transformationen

```
void main() async {
  var zahlen = Stream.fromIterable([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]);

  var ergebnis = zahlen
      .where((n) => n.isEven)      // Filtern: 2, 4, 6, 8, 10
      .map((n) => n * n)            // Transformieren: 4, 16, 36, 64, 100
      .take(3);                     // Begrenzen: 4, 16, 36

  await for (var w in ergebnis) {
    print(w);
  }
}
```

1.7.0.6 7.6 Zusammenfassendes Beispiel

```
import 'dart:async';

class EventBus {
  final _controller = StreamController<Map<String, dynamic>>.broadcast();

  Stream<Map<String, dynamic>> get events => _controller.stream;

  void emit(String typ, dynamic daten) {
    _controller.add({'typ': typ, 'daten': daten, 'zeit': DateTime.now()});
  }

  Stream<Map<String, dynamic>> on(String typ) {
    return events.where((e) => e['typ'] == typ);
  }

  void dispose() => _controller.close();
}

void main() async {
  var bus = EventBus();

  bus.on('login').listen((e) => print('Login: ${e['daten']}'));
  bus.on('logout').listen((e) => print('Logout: ${e['daten']}'));

  bus.emit('login', 'Max');
  bus.emit('other', 'ignored');
  bus.emit('logout', 'Max');

  await Future.delayed(Duration(milliseconds: 100));
```

```
  bus.dispose();
}
```

1.7.1 Übung

1.7.1.1 Aufgabe 1: Stream-Generator (15 Min.)

```
void main() async {
  // TODO: Implementiere countdown(int von)
  // - Zählt von 'von' bis 0
  // - 1 Sekunde Pause zwischen Zahlen
  // - yield jede Zahl

  await for (var n in countdown(5)) {
    print(n);
  }
  print('Start!');
}
```

1.7.1.2 Aufgabe 2: Stream-Transformationen (15 Min.)

```
void main() async {
  var ereignisse = Stream.fromIterable([
    {'typ': 'klick', 'x': 100, 'y': 200},
    {'typ': 'move', 'x': 110, 'y': 200},
    {'typ': 'klick', 'x': 150, 'y': 250},
    {'typ': 'move', 'x': 160, 'y': 260},
    {'typ': 'klick', 'x': 200, 'y': 300},
  ]);

  // TODO: Filtere nur 'klick'-Events und extrahiere Koordinaten
  // Erwartete Ausgabe: (100, 200), (150, 250), (200, 300)
}
```

1.7.1.3 Aufgabe 3: StreamController (20 Min.)

```
void main() async {
  var chat = ChatRoom();

  chat.nachrichten.listen((n) => print('[${n['von']}]: ${n['text']}'));

  chat.sende('Max', 'Hallo!');
  chat.sende('Anna', 'Hi Max!');
  chat.sende('Max', 'Wie gehts?');

  await Future.delayed(Duration(milliseconds: 100));
  chat.schließe();
}
```

```
// TODO: Implementiere ChatRoom
// - StreamController<Map<String, String>> für Nachrichten
// - Stream<...> get nachrichten
// - void sende(String von, String text)
// - void schließe()
```

1.7.1.4 Bonusaufgabe: Debounce

```
void main() async {
  var eingaben = Stream.fromIterable(['H', 'Ha', 'Hal', 'Hall', 'Hallo']);

  // TODO: Implementiere debounce Extension
  // - Wartet 300ms nach letzter Eingabe
  // - Gibt nur die letzte Eingabe aus

  await for (var e in eingaben.debounce(Duration(milliseconds: 300))) {
    print('Suche: $e');
  }
  // Sollte nur "Hallo" ausgeben
}
```

1.7.2 Lösung

1.7.2.1 Aufgabe 1

```
Stream<int> countdown(int von) async* {
  for (var i = von; i >= 0; i--) {
    yield i;
    if (i > 0) await Future.delayed(Duration(seconds: 1));
  }
}
```

1.7.2.2 Aufgabe 2

```
void main() async {
  var ereignisse = Stream.fromIterable([
    {'typ': 'klick', 'x': 100, 'y': 200},
    {'typ': 'move', 'x': 110, 'y': 200},
    {'typ': 'klick', 'x': 150, 'y': 250},
    {'typ': 'move', 'x': 160, 'y': 260},
    {'typ': 'klick', 'x': 200, 'y': 300},
  ]);

  await for (var e in ereignisse.where((e) => e['typ'] == 'klick')) {
    print('(${e['x']}, ${e['y']})');
  }
}
```

1.7.2.3 Aufgabe 3

```
import 'dart:async';

class ChatRoom {
  final _controller = StreamController<Map<String, String>>.broadcast();

  Stream<Map<String, String>> get nachrichten => _controller.stream;

  void sende(String von, String text) {
    _controller.add({'von': von, 'text': text});
  }

  void schließe() => _controller.close();
}
```

1.7.2.4 Bonusaufgabe

```
extension StreamDebounce<T> on Stream<T> {
  Stream<T> debounce(Duration dauer) async* {
    Timer? timer;
    T? letzterWert;
    var hatWert = false;
    var completer = Completer<void>();

    listen(
      (wert) {
        letzterWert = wert;
        hatWert = true;
        timer?.cancel();
        timer = Timer(dauer, () => completer.complete());
      },
      onDone: () {
        timer?.cancel();
        completer.complete();
      },
    );

    await completer.future;
    if (hatWert) yield letzterWert as T;
  }
}
```

1.7.3 Ressourcen

1.7.3.1 Offizielle Dokumentation

- Streams
- Creating Streams

1.7.3.2 Nächste Einheit

Einheit 1.8: Collections

1.8 Einheit 1.8: Collections

1.8.0.1 8.1 List

```
// Erstellung
var zahlen = [1, 2, 3, 4, 5];
var leer = <int>[];
var fixiert = List.filled(5, 0);  // [0, 0, 0, 0, 0]
var generiert = List.generate(5, (i) => i * 2);  // [0, 2, 4, 6, 8]

// Zugriff
print(zahlen[0]);       // 1
print(zahlen.first);    // 1
print(zahlen.last);     // 5
print(zahlen.length);   // 5

// Modifikation
zahlen.add(6);
zahlen.addAll([7, 8]);
zahlen.insert(0, 0);
zahlen.removeAt(0);
zahlen.removeLast();
```

1.8.0.2 8.2 Map

```
var person = {
  'name': 'Max',
  'alter': 30,
  'stadt': 'Berlin',
};

// Zugriff
print(person['name']);           // Max
print(person['fehlt']);          // null
print(person.containsKey('name')); // true

// Modifikation
person['email'] = 'max@mail.de';
person.remove('stadt');

// Iteration
for (var entry in person.entries) {
  print('${entry.key}: ${entry.value}');
}
```

1.8.0.3 8.3 Set

```
var zahlen = {1, 2, 3, 3, 4};  // {1, 2, 3, 4} – keine Duplikate
var leer = <int>{};

zahlen.add(5);
zahlen.remove(1);
print(zahlen.contains(2));  // true

// Mengenoperationen
var a = {1, 2, 3};
var b = {2, 3, 4};
print(a.union(b));        // {1, 2, 3, 4}
print(a.intersection(b)); // {2, 3}
print(a.difference(b));   // {1}
```

1.8.0.4 8.4 Collection-Methoden

```
var zahlen = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

// map – transformieren
var verdoppelt = zahlen.map((n) => n * 2).toList();

// where – filtern
var gerade = zahlen.where((n) => n.isEven).toList();

// fold – reduzieren mit Startwert
var summe = zahlen.fold(0, (acc, n) => acc + n);

// reduce – reduzieren ohne Startwert
var produkt = zahlen.reduce((a, b) => a * b);

// any / every
print(zahlen.any((n) => n > 5));   // true
print(zahlen.every((n) => n > 0)); // true

// firstWhere / lastWhere
var erstesGerade = zahlen.firstWhere((n) => n.isEven);
```

1.8.0.5 8.5 Spread-Operator & Collection-if/for

```
// Spread
var a = [1, 2, 3];
var b = [0, ...a, 4];  // [0, 1, 2, 3, 4]

// Null-aware Spread
List<int>? vielleicht;
var c = [1, ...?vielleicht, 2];  // [1, 2]

// Collection-if
```

```
var istAdmin = true;
var menu = [
  'Start',
  'Profil',
  if (istAdmin) 'Admin',
];

// Collection-for
var quadrate = [
  for (var i = 1; i <= 5; i++) i * i,
];  // [1, 4, 9, 16, 25]
```

1.8.0.6 8.6 Zusammenfassendes Beispiel

```
void main() {
  var benutzer = [
    {'name': 'Max', 'alter': 30, 'aktiv': true},
    {'name': 'Anna', 'alter': 25, 'aktiv': false},
    {'name': 'Tom', 'alter': 35, 'aktiv': true},
    {'name': 'Lisa', 'alter': 28, 'aktiv': true},
  ];

  // Aktive Benutzer über 25, sortiert nach Alter
  var ergebnis = benutzer
      .where((u) => u['aktiv'] == true)
      .where((u) => (u['alter'] as int) > 25)
      .toList()
    ..sort((a, b) => (a['alter'] as int).compareTo(b['alter'] as int));

  for (var u in ergebnis) {
    print('${u['name']}: ${u['alter']} Jahre');
  }

  // Durchschnittsalter
  var durchschnitt = benutzer
      .map((u) => u['alter'] as int)
      .reduce((a, b) => a + b) / benutzer.length;
  print('Durchschnittsalter: $durchschnitt');
}
```

1.8.1 Übung

1.8.1.1 Aufgabe 1: List-Operationen (15 Min.)

```
void main() {
  var noten = [2, 1, 3, 1, 2, 4, 2, 1, 3, 2];

  // TODO:
  // 1. Berechne den Durchschnitt
  // 2. Finde die beste (kleinste) Note
```

```
  // 3. Zähle wie oft jede Note vorkommt (Map<int, int>)
  // 4. Sortiere die Noten aufsteigend (ohne Original zu ändern)
}
```

1.8.1.2 Aufgabe 2: Map-Verarbeitung (15 Min.)

```
void main() {
  var inventar = {
    'Apfel': 50,
    'Birne': 30,
    'Orange': 0,
    'Banane': 25,
    'Kiwi': 0,
  };

  // TODO:
  // 1. Finde alle Produkte mit Bestand > 0
  // 2. Berechne den Gesamtbestand
  // 3. Erstelle eine sortierte Liste der Produkte nach Bestand
  // 4. Erhöhe alle Bestände um 10
}
```

1.8.1.3 Aufgabe 3: Set-Operationen (10 Min.)

```
void main() {
  var teamA = {'Max', 'Anna', 'Tom', 'Lisa'};
  var teamB = {'Tom', 'Lisa', 'Paul', 'Marie'};

  // TODO:
  // 1. Wer ist in beiden Teams?
  // 2. Wer ist nur in Team A?
  // 3. Alle Personen (ohne Duplikate)
  // 4. Wer ist in genau einem Team?
}
```

1.8.1.4 Aufgabe 4: Komplexe Transformationen (20 Min.)

```
void main() {
  var verkäufe = [
    {'produkt': 'Laptop', 'preis': 999.0, 'menge': 5},
    {'produkt': 'Maus', 'preis': 29.0, 'menge': 50},
    {'produkt': 'Tastatur', 'preis': 79.0, 'menge': 30},
    {'produkt': 'Monitor', 'preis': 399.0, 'menge': 10},
    {'produkt': 'USB-Stick', 'preis': 15.0, 'menge': 100},
  ];

  // TODO:
  // 1. Gesamtumsatz berechnen (preis * menge)
  // 2. Top 3 Produkte nach Umsatz
```

```
  // 3. Produkte mit Stückpreis > 50€
  // 4. Erstelle Report-String mit Collection-for
}
```

1.8.1.5 Bonusaufgabe: Eigene Collection-Extension

```
void main() {
  var zahlen = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

  print(zahlen.gruppiereNach((n) => n.isEven ? 'gerade' : 'ungerade'));
  // {gerade: [2, 4, 6, 8, 10], ungerade: [1, 3, 5, 7, 9]}

  print(zahlen.chunks(3));
  // [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10]]
}

// TODO: Implementiere Extensions gruppiereNach und chunks
```

1.8.2 Lösung

1.8.2.1 Aufgabe 1

```
void main() {
  var noten = [2, 1, 3, 1, 2, 4, 2, 1, 3, 2];

  var durchschnitt = noten.reduce((a, b) => a + b) / noten.length;
  print('Durchschnitt: $durchschnitt');

  var beste = noten.reduce((a, b) => a < b ? a : b);
  print('Beste Note: $beste');

  var häufigkeit = <int, int>{};
  for (var note in noten) {
    häufigkeit[note] = (häufigkeit[note] ?? 0) + 1;
  }
  print('Häufigkeit: $häufigkeit');

  var sortiert = [...noten]..sort();
  print('Sortiert: $sortiert');
}
```

1.8.2.2 Aufgabe 2

```
void main() {
  var inventar = {'Apfel': 50, 'Birne': 30, 'Orange': 0, 'Banane': 25,        ↪
↪  'Kiwi': 0};

  var verfügbar = inventar.entries
      .where((e) => e.value > 0)
```

```
      .map((e) => e.key)
      .toList();
  print('Verfügbar: $verfügbar');

  var gesamt = inventar.values.reduce((a, b) => a + b);
  print('Gesamtbestand: $gesamt');

  var sortiert = inventar.entries.toList()
    ..sort((a, b) => b.value.compareTo(a.value));
  print('Nach Bestand: ${sortiert.map((e) => '${e.key}: ${e.value}')}');

  inventar.updateAll((key, value) => value + 10);
  print('Erhöht: $inventar');
}
```

1.8.2.3 Aufgabe 3

```
void main() {
  var teamA = {'Max', 'Anna', 'Tom', 'Lisa'};
  var teamB = {'Tom', 'Lisa', 'Paul', 'Marie'};

  print('Beide: ${teamA.intersection(teamB)}');
  print('Nur A: ${teamA.difference(teamB)}');
  print('Alle: ${teamA.union(teamB)}');
  print('Genau eins: ${teamA.difference(teamB).union(teamB.difference(teamA))}');
}
```

1.8.2.4 Aufgabe 4

```
void main() {
  var verkäufe = [
    {'produkt': 'Laptop', 'preis': 999.0, 'menge': 5},
    {'produkt': 'Maus', 'preis': 29.0, 'menge': 50},
    {'produkt': 'Tastatur', 'preis': 79.0, 'menge': 30},
    {'produkt': 'Monitor', 'preis': 399.0, 'menge': 10},
    {'produkt': 'USB-Stick', 'preis': 15.0, 'menge': 100},
  ];

  var gesamtumsatz = verkäufe
      .map((v) => (v['preis'] as double) * (v['menge'] as int))
      .reduce((a, b) => a + b);
  print('Gesamtumsatz: $gesamtumsatz €');

  var mitUmsatz = verkäufe.map((v) => {
    ...v,
    'umsatz': (v['preis'] as double) * (v['menge'] as int),
  }).toList()..sort((a, b) => (b['umsatz'] as double).compareTo(a['umsatz']
↪  as double));
```

```
  print('Top 3: ${mitUmsatz.take(3).map((v) => v['produkt'])}');

  var teuer = verkäufe.where((v) => (v['preis'] as double) > 50).toList();
  print('Teuer: ${teuer.map((v) => v['produkt'])}');

  var report = [
    'Verkaufsbericht',
    '===============',
    for (var v in mitUmsatz)
      '${v['produkt']}: ${v['umsatz']} €',
  ].join('\n');
  print(report);
}
```

1.8.2.5 Bonusaufgabe

```
extension ListExtensions<T> on List<T> {
  Map<K, List<T>> gruppiereNach<K>(K Function(T) keyFn) {
    var result = <K, List<T>>{};
    for (var item in this) {
      var key = keyFn(item);
      (result[key] ??= []).add(item);
    }
    return result;
  }

  List<List<T>> chunks(int größe) {
    var result = <List<T>>[];
    for (var i = 0; i < length; i += größe) {
      result.add(sublist(i, i + größe > length ? length : i + größe));
    }
    return result;
  }
}
```

1.8.3 Ressourcen

1.8.3.1 Offizielle Dokumentation

- Collections
- Iterable collections

1.8.3.2 Nächste Einheit

Einheit 1.9: Generics & Null Safety

1.9 Einheit 1.9: Generics & Null Safety

1.9.0.1 9.1 Generics Grundlagen

Generics ermöglichen typsichere, wiederverwendbare Komponenten:

```
class Box<T> {
  T inhalt;
  Box(this.inhalt);

  T hole() => inhalt;
  void setze(T wert) => inhalt = wert;
}

void main() {
  var intBox = Box<int>(42);
  var stringBox = Box<String>('Hallo');

  print(intBox.hole());     // 42
  print(stringBox.hole());  // Hallo
}
```

1.9.0.2 9.2 Generische Funktionen

```
T erstesElement<T>(List<T> liste) {
  if (liste.isEmpty) throw ArgumentError('Liste ist leer');
  return liste.first;
}

List<T> wiederhole<T>(T wert, int anzahl) {
  return List.filled(anzahl, wert);
}

void main() {
  print(erstesElement([1, 2, 3]));      // 1
  print(erstesElement(['a', 'b']));     // a
  print(wiederhole('x', 3));            // [x, x, x]
}
```

1.9.0.3 9.3 Type Constraints mit `extends`

```
// T muss num oder Unterklasse sein
T max<T extends num>(T a, T b) {
  return a > b ? a : b;
}

// T muss Comparable implementieren
T minimum<T extends Comparable<T>>(List<T> liste) {
  return liste.reduce((a, b) => a.compareTo(b) < 0 ? a : b);
}

void main() {
  print(max(5, 3));       // 5
  print(max(3.14, 2.71)); // 3.14
  // max('a', 'b');       // FEHLER: String ist kein num
```

```
}
```

1.9.0.4 9.4 Sound Null Safety

Seit Dart 2.12 sind Typen standardmäßig **non-nullable**:

```
String name = 'Dart';
// name = null;  // FEHLER!

// Mit ? wird der Typ nullable
String? vielleichtName = 'Dart';
vielleichtName = null;  // OK
```

Null-aware Operatoren

```
String? name;

// ?. – Null-safe Zugriff
var länge = name?.length;  // null wenn name null

// ?? – Null-Coalescing
var sicher = name ?? 'Standard';

// ??= – Zuweisen wenn null
name ??= 'Fallback';

// ! – Null-Assertion (mit Vorsicht!)
var garantiert = name!;  // Wirft wenn null
```

1.9.0.5 9.5 Null-Checks und Flow Analysis

```
void verarbeite(String? eingabe) {
  // Nach dem Check weiß Dart, dass eingabe nicht null ist
  if (eingabe == null) {
    print('Keine Eingabe');
    return;
  }

  // Hier ist eingabe automatisch non-null
  print('Länge: ${eingabe.length}');
}

// Auch mit is-Checks
void verarbeite2(Object? obj) {
  if (obj is String) {
    // obj ist hier automatisch String
    print(obj.toUpperCase());
  }
}
```

1.9.0.6 9.6 late Keyword

```
class Service {
  // Wird später initialisiert, aber garantiert vor Zugriff
  late final String apiKey;

  void konfiguriere(String key) {
    apiKey = key;
  }

  // Lazy Initialization
  late final String teureBerechnung = _berechne();

  String _berechne() {
    print('Wird nur bei Bedarf berechnet');
    return 'Ergebnis';
  }
}
```

1.9.0.7 9.7 Zusammenfassendes Beispiel

```
class Repository<T> {
  final Map<int, T> _speicher = {};
  int _nächsteId = 0;

  int speichere(T item) {
    var id = _nächsteId++;
    _speicher[id] = item;
    return id;
  }

  T? finde(int id) => _speicher[id];

  T findeOderFehler(int id) {
    var item = _speicher[id];
    if (item == null) throw ArgumentError('ID $id nicht gefunden');
    return item;
  }

  List<T> alle() => _speicher.values.toList();

  bool lösche(int id) => _speicher.remove(id) != null;
}

class Benutzer {
  final String name;
  Benutzer(this.name);
}

void main() {
  var repo = Repository<Benutzer>();
```

```
  var id1 = repo.speichere(Benutzer('Max'));
  var id2 = repo.speichere(Benutzer('Anna'));

  print(repo.finde(id1)?.name);  // Max
  print(repo.finde(999)?.name);  // null

  for (var u in repo.alle()) {
    print(u.name);
  }
}
```

1.9.1 Übung

1.9.1.1 Aufgabe 1: Generische Klasse (15 Min.)

```
void main() {
  var stack = Stack<int>();
  stack.push(1);
  stack.push(2);
  stack.push(3);

  print(stack.pop());   // 3
  print(stack.peek());  // 2
  print(stack.isEmpty); // false
  print(stack.größe);   // 2
}

// TODO: Implementiere Stack<T>
// - push(T): Element hinzufügen
// - pop(): Element entfernen und zurückgeben
// - peek(): Oberstes Element ohne entfernen
// - isEmpty, größe
```

1.9.1.2 Aufgabe 2: Generische Funktionen (15 Min.)

```
void main() {
  var zahlen = [3, 1, 4, 1, 5, 9, 2, 6];

  print(finde(zahlen, (n) => n > 5));  // 9
  print(finde(zahlen, (n) => n > 100)); // null

  print(transformiere<int, String>(zahlen, (n) => 'Zahl: $n'));
  // [Zahl: 3, Zahl: 1, ...]

  print(partitioniere(zahlen, (n) => n.isEven));
  // ([4, 2, 6], [3, 1, 1, 5, 9])
}

// TODO: Implementiere:
```

```
// - T? finde<T>(List<T>, bool Function(T))
// - List<R> transformiere<T, R>(List<T>, R Function(T))
// - (List<T>, List<T>) partitioniere<T>(List<T>, bool Function(T))
```

1.9.1.3 Aufgabe 3: Null Safety (15 Min.)

```
void main() {
  // Korrigiere die Null-Safety-Fehler:

  String? getName() => DateTime.now().second.isEven ? 'Max' : null;

  // TODO: Fix these
  var name = getName();
  print(name.length);  // Fehler!

  var länge = name.length ?? 0;  // Fehler!

  if (name != null) {
    // TODO: Warum funktioniert das hier?
  }
}

// TODO: Implementiere sicher
String grüße(String? name, String? grußwort) {
  // - Wenn beide null: "Hallo!"
  // - Wenn name null: "$grußwort!"
  // - Wenn grußwort null: "Hallo, $name!"
  // - Sonst: "$grußwort, $name!"
}
```

1.9.1.4 Aufgabe 4: Type Constraints (15 Min.)

```
void main() {
  var zahlen = [3, 1, 4, 1, 5];
  var texte = ['Birne', 'Apfel', 'Orange'];

  print(sortiert(zahlen));  // [1, 1, 3, 4, 5]
  print(sortiert(texte));   // [Apfel, Birne, Orange]

  print(inBereich(5, 1, 10));  // true
  print(inBereich(15, 1, 10)); // false
}

// TODO: Implementiere mit Type Constraints:
// - List<T> sortiert<T extends Comparable<T>>(List<T>)
// - bool inBereich<T extends num>(T wert, T min, T max)
```

1.9.1.5 Bonusaufgabe: Result-Typ

```
void main() {
  var r1 = teile(10, 2);
  var r2 = teile(10, 0);

  r1.wenn(
    erfolg: (wert) => print('Ergebnis: $wert'),
    fehler: (msg) => print('Fehler: $msg'),
  );

  r2.wenn(
    erfolg: (wert) => print('Ergebnis: $wert'),
    fehler: (msg) => print('Fehler: $msg'),
  );
}

Result<double> teile(int a, int b) {
  if (b == 0) return Fehler('Division durch Null');
  return Erfolg(a / b);
}

// TODO: Implementiere Result<T>, Erfolg<T>, Fehler<T>
// - sealed class Result<T>
// - wenn({void Function(T) erfolg, void Function(String) fehler})
```

1.9.2 Lösung

1.9.2.1 Aufgabe 1

```
class Stack<T> {
  final List<T> _elemente = [];

  void push(T element) => _elemente.add(element);

  T pop() {
    if (isEmpty) throw StateError('Stack ist leer');
    return _elemente.removeLast();
  }

  T peek() {
    if (isEmpty) throw StateError('Stack ist leer');
    return _elemente.last;
  }

  bool get isEmpty => _elemente.isEmpty;
  int get größe => _elemente.length;
}
```

1.9.2.2 Aufgabe 2

```
T? finde<T>(List<T> liste, bool Function(T) bedingung) {
  for (var element in liste) {
    if (bedingung(element)) return element;
  }
  return null;
}

List<R> transformiere<T, R>(List<T> liste, R Function(T) transform) {
  return liste.map(transform).toList();
}

(List<T>, List<T>) partitioniere<T>(List<T> liste, bool Function(T) bedingung) {
  var wahr = <T>[];
  var falsch = <T>[];
  for (var e in liste) {
    (bedingung(e) ? wahr : falsch).add(e);
  }
  return (wahr, falsch);
}
```

1.9.2.3 Aufgabe 3

```
void main() {
  String? getName() => DateTime.now().second.isEven ? 'Max' : null;

  var name = getName();

  // Fix 1: Null-Check
  print(name?.length);

  // Fix 2: Richtige Syntax
  var länge = name?.length ?? 0;

  // Flow Analysis: Nach dem Check ist name non-null
  if (name != null) {
    print(name.length);  // OK!
  }
}

String grüße(String? name, String? grußwort) {
  var g = grußwort ?? 'Hallo';
  var n = name;

  if (n == null) return '$g!';
  return '$g, $n!';
}
```

1.9.2.4 Aufgabe 4

```
List<T> sortiert<T extends Comparable<T>>(List<T> liste) {
  return [...liste]..sort();
}

bool inBereich<T extends num>(T wert, T min, T max) {
  return wert >= min && wert <= max;
}
```

1.9.2.5 Bonusaufgabe

```
sealed class Result<T> {
  void wenn({
    required void Function(T) erfolg,
    required void Function(String) fehler,
  });
}

class Erfolg<T> extends Result<T> {
  final T wert;
  Erfolg(this.wert);

  @override
  void wenn({
    required void Function(T) erfolg,
    required void Function(String) fehler,
  }) => erfolg(wert);
}

class Fehler<T> extends Result<T> {
  final String nachricht;
  Fehler(this.nachricht);

  @override
  void wenn({
    required void Function(T) erfolg,
    required void Function(String) fehler,
  }) => fehler(nachricht);
}

Result<double> teile(int a, int b) {
  if (b == 0) return Fehler('Division durch Null');
  return Erfolg(a / b);
}
```

1.9.3 Ressourcen

1.9.3.1 Offizielle Dokumentation

- Generics

- Sound Null Safety
- Understanding null safety

1.9.3.2 Nächste Einheit

Einheit 1.10: Pattern Matching & Records

1.10 Einheit 1.10: Pattern Matching & Records

1.10.0.1 10.1 Records

Records sind anonyme, immutable Datentypen:

```
// Positional Record
var punkt = (3, 4);
print(punkt.$1);  // 3
print(punkt.$2);  // 4

// Named Record
var person = (name: 'Max', alter: 30);
print(person.name);   // Max
print(person.alter);  // 30

// Gemischt
var gemischt = ('Wert', name: 'Max', 42);
print(gemischt.$1);    // Wert
print(gemischt.$2);    // 42
print(gemischt.name);  // Max
```

Records als Rückgabewert

```
(int, int) teileRest(int a, int b) {
  return (a ~/ b, a % b);
}

(String name, int alter) holePerson() {
  return ('Max', 30);
}

void main() {
  var (quotient, rest) = teileRest(10, 3);
  print('$quotient Rest $rest');  // 3 Rest 1

  var (name, alter) = holePerson();
  print('$name ist $alter');
}
```

1.10.0.2 10.2 Destructuring

```
// List Destructuring
var zahlen = [1, 2, 3, 4, 5];
var [erste, zweite, ...rest] = zahlen;
```

```
print(erste);  // 1
print(rest);   // [3, 4, 5]

// Map Destructuring
var map = {'name': 'Max', 'alter': 30};
var {'name': name, 'alter': alter} = map;

// Record Destructuring
var (x, y) = (3, 4);
var (:name, :alter) = (name: 'Max', alter: 30);
```

1.10.0.3 10.3 Pattern Matching mit switch

```
String beschreibe(Object? obj) {
  return switch (obj) {
    null => 'Nichts',
    int i when i < 0 => 'Negative Zahl: $i',
    int i => 'Zahl: $i',
    String s when s.isEmpty => 'Leerer String',
    String s => 'Text: $s',
    List l when l.isEmpty => 'Leere Liste',
    [var first, ...var rest] => 'Liste mit $first und ${rest.length} weiteren',
    (int x, int y) => 'Punkt ($x, $y)',
    {'typ': 'user', 'name': var name} => 'Benutzer: $name',
    _ => 'Unbekannt: $obj',
  };
}
```

1.10.0.4 10.4 Sealed Classes

```
sealed class Form {}

class Kreis extends Form {
  final double radius;
  Kreis(this.radius);
}

class Rechteck extends Form {
  final double breite, höhe;
  Rechteck(this.breite, this.höhe);
}

class Dreieck extends Form {
  final double a, b, c;
  Dreieck(this.a, this.b, this.c);
}

double berechneFlaeche(Form form) {
  return switch (form) {
```

```
    Kreis(:var radius) => 3.14159 * radius * radius,
    Rechteck(:var breite, :var höhe) => breite * höhe,
    Dreieck(:var a, :var b, :var c) => _heron(a, b, c),
  };
  // Kein default nötig – Compiler weiß, dass alle Fälle abgedeckt sind!
}

double _heron(double a, double b, double c) {
  var s = (a + b + c) / 2;
  return (s * (s - a) * (s - b) * (s - c)).abs();
}
```

1.10.0.5 10.5 If-Case

```
void main() {
  var daten = {'typ': 'benutzer', 'name': 'Max', 'alter': 30};

  // If-Case statt komplexer Typprüfungen
  if (daten case {'typ': 'benutzer', 'name': String name}) {
    print('Benutzer gefunden: $name');
  }

  var punkt = (3, 4);
  if (punkt case (int x, int y) when x == y) {
    print('Diagonaler Punkt');
  } else if (punkt case (int x, int y)) {
    print('Punkt bei $x, $y');
  }
}
```

1.10.0.6 10.6 Zusammenfassendes Beispiel

```
sealed class ApiAntwort<T> {}

class Erfolg<T> extends ApiAntwort<T> {
  final T daten;
  final int statusCode;
  Erfolg(this.daten, {this.statusCode = 200});
}

class Fehler<T> extends ApiAntwort<T> {
  final String nachricht;
  final int statusCode;
  Fehler(this.nachricht, {required this.statusCode});
}

class Laden<T> extends ApiAntwort<T> {}

void verarbeite<T>(ApiAntwort<T> antwort) {
```

```
  switch (antwort) {
    case Erfolg(:var daten, :var statusCode):
      print('Erfolg ($statusCode): $daten');
    case Fehler(:var nachricht, statusCode: var code) when code >= 500:
      print('Serverfehler: $nachricht');
    case Fehler(:var nachricht, :var statusCode):
      print('Fehler ($statusCode): $nachricht');
    case Laden():
      print('Lädt...');
  }
}

void main() {
  verarbeite(Erfolg({'name': 'Max'}));
  verarbeite(Fehler('Nicht gefunden', statusCode: 404));
  verarbeite(Fehler('Interner Fehler', statusCode: 500));
  verarbeite(Laden<String>());
}
```

1.10.1 Übung

1.10.1.1 Aufgabe 1: Records (15 Min.)

```
void main() {
  // TODO: Implementiere minMax
  var (min, max) = minMax([3, 1, 4, 1, 5, 9, 2, 6]);
  print('Min: $min, Max: $max');

  // TODO: Implementiere parseKoordinate
  var (x, y) = parseKoordinate('12.5,7.3');
  print('X: $x, Y: $y');

  // TODO: Implementiere erstellePerson
  var person = erstellePerson('Max Mustermann', 30);
  print('${person.vorname} ${person.nachname}, ${person.alter}');
}

// (int, int) minMax(List<int>)
// (double, double) parseKoordinate(String) – Format: "x,y"
// ({String vorname, String nachname, int alter}) erstellePerson(String name,  ↪
 ↪  int alter)
```

1.10.1.2 Aufgabe 2: Destructuring (15 Min.)

```
void main() {
  var daten = {
    'benutzer': {
      'name': 'Max',
      'adresse': {
        'stadt': 'Berlin',
```

```
        'plz': '10115',
      },
    },
    'bestellungen': [
      {'id': 1, 'betrag': 99.99},
      {'id': 2, 'betrag': 149.99},
    ],
  };

  // TODO: Extrahiere mit Destructuring:
  // - Benutzername
  // - Stadt
  // - Erste Bestellungs-ID
  // - Anzahl Bestellungen
}
```

1.10.1.3 Aufgabe 3: Pattern Matching (20 Min.)

```
void main() {
  var eingaben = [
    'exit',
    'help',
    'add 5 3',
    'mul 4 7',
    'div 10 2',
    'div 10 0',
    'unknown',
  ];

  for (var eingabe in eingaben) {
    print('> $eingabe');
    print(verarbeiteBefehl(eingabe));
    print('');
  }
}

// TODO: Implementiere verarbeiteBefehl(String eingabe) -> String
// - 'exit' -> 'Programm beendet'
// - 'help' -> 'Befehle: exit, help, add x y, mul x y, div x y'
// - 'add x y' -> Summe
// - 'mul x y' -> Produkt
// - 'div x y' -> Quotient (Fehler bei y=0)
// - sonst -> 'Unbekannter Befehl'
//
// Hint: Verwende switch mit List-Patterns für split(' ')
```

1.10.1.4 Aufgabe 4: Sealed Classes (20 Min.)

```
void main() {
  List<JsonWert> json = [
    JsonString('Hallo'),
    JsonNumber(42),
    JsonBool(true),
    JsonNull(),
    JsonArray([JsonNumber(1), JsonNumber(2)]),
    JsonObject({'name': JsonString('Max')}),
  ];

  for (var wert in json) {
    print(zuDartWert(wert));
  }
}

// TODO: Implementiere:
// - sealed class JsonWert
// - JsonString, JsonNumber, JsonBool, JsonNull, JsonArray, JsonObject
// - dynamic zuDartWert(JsonWert) mit exhaustive switch
```

1.10.1.5 Bonusaufgabe: Expression Parser

```
void main() {
  var ausdrücke = [
    Zahl(5),
    Addition(Zahl(3), Zahl(4)),
    Multiplikation(Addition(Zahl(2), Zahl(3)), Zahl(4)),
    Division(Zahl(10), Zahl(2)),
    Division(Zahl(10), Zahl(0)),
  ];

  for (var expr in ausdrücke) {
    print('${formatiere(expr)} = ${auswerten(expr)}');
  }
}

// TODO: Implementiere:
// - sealed class Ausdruck
// - Zahl, Addition, Subtraktion, Multiplikation, Division
// - double? auswerten(Ausdruck)
// - String formatiere(Ausdruck)
```

1.10.2 Lösung

1.10.2.1 Aufgabe 1

```
(int, int) minMax(List<int> zahlen) {
  if (zahlen.isEmpty) throw ArgumentError('Liste ist leer');
  var min = zahlen[0];
  var max = zahlen[0];
```

```
  for (var z in zahlen) {
    if (z < min) min = z;
    if (z > max) max = z;
  }
  return (min, max);
}

(double, double) parseKoordinate(String s) {
  var teile = s.split(',');
  return (double.parse(teile[0]), double.parse(teile[1]));
}

({String vorname, String nachname, int alter}) erstellePerson(String name,
 ↪  int alter) {
  var teile = name.split(' ');
  return (vorname: teile[0], nachname: teile.sublist(1).join(' '), alter: alter);
}
```

1.10.2.2 Aufgabe 2

```
void main() {
  var daten = {...};  // wie oben

  // Nested Destructuring
  if (daten case {
    'benutzer': {'name': String name, 'adresse': {'stadt': String stadt}},
    'bestellungen': [{'id': int ersteId}, ...var rest],
  }) {
    print('Name: $name');
    print('Stadt: $stadt');
    print('Erste Bestellungs-ID: $ersteId');
    print('Anzahl Bestellungen: ${rest.length + 1}');
  }
}
```

1.10.2.3 Aufgabe 3

```
String verarbeiteBefehl(String eingabe) {
  var teile = eingabe.split(' ');

  return switch (teile) {
    ['exit'] => 'Programm beendet',
    ['help'] => 'Befehle: exit, help, add x y, mul x y, div x y',
    ['add', var a, var b] => 'Ergebnis: ${int.parse(a) + int.parse(b)}',
    ['mul', var a, var b] => 'Ergebnis: ${int.parse(a) * int.parse(b)}',
    ['div', var a, '0'] => 'Fehler: Division durch Null',
    ['div', var a, var b] => 'Ergebnis: ${int.parse(a) / int.parse(b)}',
    _ => 'Unbekannter Befehl: $eingabe',
  };
```

```
}
```

1.10.2.4 Aufgabe 4

```
sealed class JsonWert {}

class JsonString extends JsonWert {
  final String wert;
  JsonString(this.wert);
}

class JsonNumber extends JsonWert {
  final num wert;
  JsonNumber(this.wert);
}

class JsonBool extends JsonWert {
  final bool wert;
  JsonBool(this.wert);
}

class JsonNull extends JsonWert {}

class JsonArray extends JsonWert {
  final List<JsonWert> elemente;
  JsonArray(this.elemente);
}

class JsonObject extends JsonWert {
  final Map<String, JsonWert> felder;
  JsonObject(this.felder);
}

dynamic zuDartWert(JsonWert json) => switch (json) {
  JsonString(:var wert) => wert,
  JsonNumber(:var wert) => wert,
  JsonBool(:var wert) => wert,
  JsonNull() => null,
  JsonArray(:var elemente) => elemente.map(zuDartWert).toList(),
  JsonObject(:var felder) => felder.map((k, v) => MapEntry(k, zuDartWert(v))),
};
```

1.10.2.5 Bonusaufgabe

```
sealed class Ausdruck {}

class Zahl extends Ausdruck {
  final double wert;
  Zahl(num wert) : wert = wert.toDouble();
```

```
}

class Addition extends Ausdruck {
  final Ausdruck links, rechts;
  Addition(this.links, this.rechts);
}

class Subtraktion extends Ausdruck {
  final Ausdruck links, rechts;
  Subtraktion(this.links, this.rechts);
}

class Multiplikation extends Ausdruck {
  final Ausdruck links, rechts;
  Multiplikation(this.links, this.rechts);
}

class Division extends Ausdruck {
  final Ausdruck links, rechts;
  Division(this.links, this.rechts);
}

double? auswerten(Ausdruck expr) => switch (expr) {
  Zahl(:var wert) => wert,
  Addition(:var links, :var rechts) =>
    (auswerten(links) ?? 0) + (auswerten(rechts) ?? 0),
  Subtraktion(:var links, :var rechts) =>
    (auswerten(links) ?? 0) - (auswerten(rechts) ?? 0),
  Multiplikation(:var links, :var rechts) =>
    (auswerten(links) ?? 0) * (auswerten(rechts) ?? 0),
  Division(:var links, :var rechts) =>
    auswerten(rechts) == 0 ? null : (auswerten(links) ?? 0) / auswerten(rechts)!,
};

String formatiere(Ausdruck expr) => switch (expr) {
  Zahl(:var wert) => wert.toString(),
  Addition(:var links, :var rechts) =>
    '(${formatiere(links)} + ${formatiere(rechts)})',
  Subtraktion(:var links, :var rechts) =>
    '(${formatiere(links)} - ${formatiere(rechts)})',
  Multiplikation(:var links, :var rechts) =>
    '(${formatiere(links)} * ${formatiere(rechts)})',
  Division(:var links, :var rechts) =>
    '(${formatiere(links)} / ${formatiere(rechts)})',
};
```

1.10.3 Ressourcen

1.10.3.1 Offizielle Dokumentation

- Records

- Patterns
- Branches (switch)

1.10.3.2 Block 1 abgeschlossen!

Du hast **Block 1: Dart — Die Sprache** erfolgreich abgeschlossen.

Nächster Schritt: Block 2 — Flutter Grundlagen

Chapter 2

Block 2: Flutter – Grundlagen

In diesem Block lernst du die Grundlagen von Flutter: Widgets, Layout, Navigation und Styling.

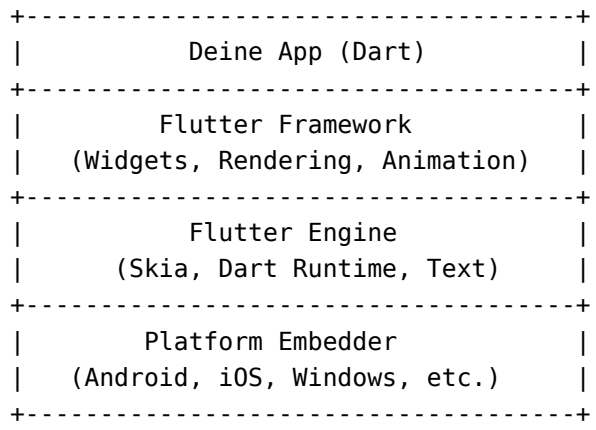
2.1 Einheit 2.1: Architektur & Setup

2.1.0.1 1.1 Was ist Flutter?

Flutter ist Googles UI-Toolkit für plattformübergreifende Apps aus einer Codebasis:

- **Mobile:** Android, iOS
- **Desktop:** Windows, macOS, Linux
- **Web:** Browser-Apps

Architektur



2.1.0.2 1.2 Installation prüfen

```
# Flutter-Installation prüfen
flutter doctor

# Erwartete Ausgabe:
# [[x]] Flutter (Channel stable, 3.x.x)
# [[x]] Android toolchain
# [[x]] Chrome - develop for the web
# [[x]] Android Studio
# [[x]] VS Code
```

2.1.0.3 1.3 Erstes Projekt erstellen

```
# Neues Projekt
flutter create meine_app
```

```
cd meine_app

# App starten
flutter run

# Oder spezifisches Gerät
flutter devices          # Verfügbare Geräte anzeigen
flutter run -d chrome    # Im Browser
flutter run -d android   # Android Emulator
```

2.1.0.4 1.4 Projektstruktur

```
meine_app/
+-- lib/
|   +-- main.dart         # Einstiegspunkt
+-- test/                  # Tests
+-- android/               # Android-spezifisch
+-- ios/                   # iOS-spezifisch
+-- web/                   # Web-spezifisch
+-- pubspec.yaml           # Dependencies & Assets
+-- pubspec.lock           # Lock-Datei
```

2.1.0.5 1.5 main.dart verstehen

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MeineApp());
}

class MeineApp extends StatelessWidget {
  const MeineApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Meine App',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
        useMaterial3: true,
      ),
      home: const StartSeite(),
    );
  }
}

class StartSeite extends StatelessWidget {
  const StartSeite({super.key});

  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Startseite'),
      ),
      body: const Center(
        child: Text('Hallo Flutter!'),
      ),
    );
  }
}
```

2.1.0.6 1.6 Widget-Tree

Alles in Flutter ist ein **Widget**. Widgets bilden einen Baum:

```
MaterialApp
+-- Scaffold
    +-- AppBar
    |   +-- Text
    +-- Center
        +-- Text
```

2.1.0.7 1.7 Hot Reload vs. Hot Restart

Feature	Hot Reload	Hot Restart
Shortcut	`r`	`R`
Geschwindigkeit	Sehr schnell	Schnell
State erhalten	Ja	Nein
Wann nutzen	UI-Änderungen	State-Änderungen

```
# In der Konsole während flutter run:
r  # Hot Reload
R  # Hot Restart
q  # Beenden
```

2.1.0.8 1.8 pubspec.yaml

```
name: meine_app
description: Meine erste Flutter-App
version: 1.0.0+1

environment:
  sdk: '>=3.0.0 <4.0.0'

dependencies:
  flutter:
    sdk: flutter
  # Externe Packages hier hinzufügen
```

```
  # http: ^1.1.0

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^3.0.0

flutter:
  uses-material-design: true
  # Assets hier deklarieren
  # assets:
  #   - assets/images/
```

2.1.0.9 1.9 Packages installieren

```
# Package hinzufügen
flutter pub add http

# Dependencies installieren
flutter pub get

# Outdated packages prüfen
flutter pub outdated

# Upgrade
flutter pub upgrade
```

2.1.0.10 1.10 DevTools

```
# DevTools öffnen
flutter pub global activate devtools
flutter pub global run devtools

# Oder über VS Code / Android Studio
```

DevTools bieten: - Widget Inspector - Performance Profiling - Memory Analysis - Network Monitoring

2.1.1 Übung

2.1.1.1 Aufgabe 1: Projekt erstellen (10 Min.)

1. Erstelle ein neues Flutter-Projekt namens `lernapp`
2. Starte die App im Browser oder Emulator
3. Ändere den Text “You have pushed…” zu “Willkommen bei Flutter!”
4. Nutze Hot Reload um die Änderung zu sehen

2.1.1.2 Aufgabe 2: App umbauen (20 Min.)

Ersetze den Inhalt von `lib/main.dart`:

```
import 'package:flutter/material.dart';

void main() {
  runApp(const LernApp());
}

class LernApp extends StatelessWidget {
  const LernApp({super.key});

  @override
  Widget build(BuildContext context) {
    // TODO: Implementiere MaterialApp mit:
    // - title: 'LernApp'
    // - theme mit colorScheme (Farbe deiner Wahl)
    // - home: ProfilSeite()
  }
}

class ProfilSeite extends StatelessWidget {
  const ProfilSeite({super.key});

  @override
  Widget build(BuildContext context) {
    // TODO: Implementiere Scaffold mit:
    // - AppBar mit Titel "Mein Profil"
    // - body: Column mit:
    //   - Icon (Icons.person, size: 100)
    //   - Text mit deinem Namen
    //   - Text mit "Flutter-Entwickler"
  }
}
```

2.1.1.3 Aufgabe 3: Package hinzufügen (15 Min.)

1. Füge das Package `google_fonts` hinzu
2. Ändere die Schriftart des Namens zu “Roboto Mono”
3. Teste mit Hot Reload

```
// Hint:
import 'package:google_fonts/google_fonts.dart';

Text(
  'Mein Name',
  style: GoogleFonts.robotoMono(fontSize: 24),
)
```

2.1.1.4 Aufgabe 4: Widget-Tree zeichnen (10 Min.)

Zeichne den Widget-Tree für folgende Struktur:

```
MaterialApp(
  home: Scaffold(
    appBar: AppBar(
      title: Text('Test'),
      actions: [
        IconButton(icon: Icon(Icons.settings), onPressed: () {}),
      ],
    ),
    body: Column(
      children: [
        Image.network('...'),
        Text('Titel'),
        Row(
          children: [
            ElevatedButton(child: Text('OK'), onPressed: () {}),
            ElevatedButton(child: Text('Abbrechen'), onPressed: () {}),
          ],
        ),
      ],
    ),
    floatingActionButton: FloatingActionButton(
      child: Icon(Icons.add),
      onPressed: () {},
    ),
  ),
)
```

2.1.1.5 Bonusaufgabe: Debug-Banner entfernen

Entferne das “DEBUG”-Banner in der oberen rechten Ecke.

Hint: MaterialApp hat eine Property `debugShowCheckedModeBanner`

2.1.2 Lösung

2.1.2.1 Aufgabe 2

```
import 'package:flutter/material.dart';

void main() {
  runApp(const LernApp());
}

class LernApp extends StatelessWidget {
  const LernApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'LernApp',
      debugShowCheckedModeBanner: false,
```

```
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.teal),
        useMaterial3: true,
      ),
      home: const ProfilSeite(),
    );
  }
}

class ProfilSeite extends StatelessWidget {
  const ProfilSeite({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Mein Profil'),
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      ),
      body: const Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Icon(Icons.person, size: 100, color: Colors.teal),
            SizedBox(height: 16),
            Text(
              'Max Mustermann',
              style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 8),
            Text(
              'Flutter-Entwickler',
              style: TextStyle(fontSize: 16, color: Colors.grey),
            ),
          ],
        ),
      ),
    );
  }
}
```

2.1.2.2 Aufgabe 3

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

// In ProfilSeite:
Text(
  'Max Mustermann',
  style: GoogleFonts.robotoMono(
```

```
    fontSize: 24,
    fontWeight: FontWeight.bold,
  ),
),
```

2.1.2.3 Aufgabe 4: Widget-Tree

```
MaterialApp
+-- Scaffold
    +-- AppBar
    |   +-- Text ("Test")
    |   +-- IconButton
    |       +-- Icon (settings)
    +-- Column
    |   +-- Image
    |   +-- Text ("Titel")
    |   +-- Row
    |       +-- ElevatedButton
    |       |   +-- Text ("OK")
    |       +-- ElevatedButton
    |           +-- Text ("Abbrechen")
    +-- FloatingActionButton
        +-- Icon (add)
```

2.1.2.4 Bonusaufgabe

```
MaterialApp(
  debugShowCheckedModeBanner: false,  // Diese Zeile hinzufügen
  // ...
)
```

2.1.3 Ressourcen

2.1.3.1 Offizielle Dokumentation

- Flutter Get Started
- Write your first Flutter app
- Flutter architectural overview

2.1.3.2 Tools

- DartPad mit Flutter
- pub.dev - Package Repository

2.1.3.3 Nächste Einheit

Einheit 2.2: StatelessWidget & Basis-Widgets

2.2 Einheit 2.2: StatelessWidget & Grundlagen

2.2.0.1 2.1 StatelessWidget

Ein StatelessWidget ist **unveränderlich** — es hat keinen internen State:

```
class Begrüßung extends StatelessWidget {
  final String name;

  const Begrüßung({super.key, required this.name});

  @override
  Widget build(BuildContext context) {
    return Text('Hallo, $name!');
  }
}

// Verwendung:
Begrüßung(name: 'Max')
```

2.2.0.2 2.2 BuildContext

`BuildContext` gibt Zugriff auf Position im Widget-Tree und Theme:

```
@override
Widget build(BuildContext context) {
  // Theme-Daten holen
  var theme = Theme.of(context);
  var screenSize = MediaQuery.of(context).size;

  return Text(
    'Bildschirmbreite: ${screenSize.width}',
    style: theme.textTheme.headlineMedium,
  );
}
```

2.2.0.3 2.3 Text Widget

```
Text('Einfacher Text')

Text(
  'Formatierter Text',
  style: TextStyle(
    fontSize: 24,
    fontWeight: FontWeight.bold,
    color: Colors.blue,
    letterSpacing: 2,
  ),
  textAlign: TextAlign.center,
  maxLines: 2,
  overflow: TextOverflow.ellipsis,
)
```

```
// Rich Text
Text.rich(
  TextSpan(
    text: 'Normal ',
    children: [
      TextSpan(
        text: 'Fett',
        style: TextStyle(fontWeight: FontWeight.bold),
      ),
      TextSpan(text: ' und wieder normal'),
    ],
  ),
)
```

2.2.0.4 2.4 Icon Widget

```
Icon(Icons.favorite)

Icon(
  Icons.star,
  size: 48,
  color: Colors.amber,
)

// Mit Semantik für Accessibility
Icon(
  Icons.home,
  semanticLabel: 'Startseite',
)
```

2.2.0.5 2.5 Image Widget

```
// Aus dem Netzwerk
Image.network(
  'https://picsum.photos/200',
  width: 200,
  height: 200,
  fit: BoxFit.cover,
  loadingBuilder: (context, child, progress) {
    if (progress == null) return child;
    return CircularProgressIndicator();
  },
)

// Aus Assets (pubspec.yaml konfigurieren!)
Image.asset('assets/images/logo.png')

// Aus Datei
```

```
Image.file(File('/path/to/image.png'))
```

2.2.0.6 2.6 Button Widgets

```
// ElevatedButton – hervorgehoben
ElevatedButton(
  onPressed: () => print('Gedrückt!'),
  child: Text('Klick mich'),
)

// TextButton – flach
TextButton(
  onPressed: () {},
  child: Text('Text Button'),
)

// OutlinedButton – mit Rahmen
OutlinedButton(
  onPressed: () {},
  child: Text('Outlined'),
)

// IconButton
IconButton(
  icon: Icon(Icons.favorite),
  onPressed: () {},
)

// FloatingActionButton
FloatingActionButton(
  onPressed: () {},
  child: Icon(Icons.add),
)

// Mit Icon
ElevatedButton.icon(
  onPressed: () {},
  icon: Icon(Icons.send),
  label: Text('Senden'),
)
```

2.2.0.7 2.7 Container Widget

```
Container(
  width: 200,
  height: 100,
  padding: EdgeInsets.all(16),
  margin: EdgeInsets.symmetric(vertical: 8),
  decoration: BoxDecoration(
```

```
    color: Colors.blue,
    borderRadius: BorderRadius.circular(12),
    boxShadow: [
      BoxShadow(
        color: Colors.black26,
        blurRadius: 10,
        offset: Offset(0, 4),
      ),
    ],
  ),
  child: Text('Container-Inhalt'),
)
```

2.2.0.8 2.8 Card Widget

```
Card(
  elevation: 4,
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(12),
  ),
  child: Padding(
    padding: EdgeInsets.all(16),
    child: Column(
      children: [
        Text('Titel', style: TextStyle(fontSize: 20)),
        SizedBox(height: 8),
        Text('Beschreibung hier...'),
      ],
    ),
  ),
)
```

2.2.0.9 2.9 Scaffold & AppBar

```
Scaffold(
  appBar: AppBar(
    title: Text('Meine App'),
    leading: IconButton(
      icon: Icon(Icons.menu),
      onPressed: () {},
    ),
    actions: [
      IconButton(icon: Icon(Icons.search), onPressed: () {}),
      IconButton(icon: Icon(Icons.more_vert), onPressed: () {}),
    ],
  ),
  body: Center(child: Text('Inhalt')),
  floatingActionButton: FloatingActionButton(
    onPressed: () {},
```

```
    child: Icon(Icons.add),
  ),
  bottomNavigationBar: BottomNavigationBar(
    items: [
      BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Start'),
      BottomNavigationBarItem(icon: Icon(Icons.person), label: 'Profil'),
    ],
  ),
)
```

2.2.0.10 2.10 Beispiel: Profilkarte

```
class ProfilKarte extends StatelessWidget {
  final String name;
  final String email;
  final String? avatarUrl;

  const ProfilKarte({
    super.key,
    required this.name,
    required this.email,
    this.avatarUrl,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: EdgeInsets.all(16),
      child: Padding(
        padding: EdgeInsets.all(16),
        child: Row(
          children: [
            CircleAvatar(
              radius: 30,
              backgroundImage: avatarUrl != null
                  ? NetworkImage(avatarUrl!)
                  : null,
              child: avatarUrl == null ? Icon(Icons.person) : null,
            ),
            SizedBox(width: 16),
            Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(name, style: TextStyle(
                  fontSize: 18,
                  fontWeight: FontWeight.bold,
                )),
                Text(email, style: TextStyle(color: Colors.grey)),
              ],
            ),
```

```
          ],
        ),
      ),
    );
  }
}
```

2.2.1 Übung

2.2.1.1 Aufgabe 1: Visitenkarte (20 Min.)

Erstelle ein `Visitenkarte`-Widget:

```
// Verwendung:
Visitenkarte(
  name: 'Max Mustermann',
  position: 'Senior Developer',
  firma: 'Tech GmbH',
  email: 'max@tech.de',
  telefon: '+49 123 456789',
)
```

Anforderungen: - Card mit abgerundeten Ecken - Icon für jeden Kontakttyp (Email, Telefon) - Name fett und größer - Firmenname in Grau

2.2.1.2 Aufgabe 2: Produktkarte (20 Min.)

Erstelle ein `ProduktKarte`-Widget für einen Shop:

```
ProduktKarte(
  name: 'Flutter T-Shirt',
  preis: 29.99,
  bildUrl: 'https://picsum.photos/200',
  aufLager: true,
)
```

Anforderungen: - Bild oben - Name und Preis darunter - “Auf Lager” / “Ausverkauft” Badge - “In den Warenkorb” Button

2.2.1.3 Aufgabe 3: Bewertungsanzeige (15 Min.)

Erstelle ein `Bewertung`-Widget:

```
Bewertung(sterne: 4, maxSterne: 5)
// Zeigt: ★★★★☆
```

Anforderungen: - Zeigt gefüllte und leere Sterne - Farbe: Amber für gefüllt, Grau für leer - Optional: Anzahl Bewertungen anzeigen

2.2.1.4 Bonusaufgabe: Wetterkarte

```
WetterKarte(
  stadt: 'Berlin',
```

```
  temperatur: 22,
  zustand: WetterZustand.sonnig,
)

enum WetterZustand { sonnig, bewölkt, regnerisch, schnee }
```

2.2.2 Lösung

2.2.2.1 Aufgabe 1: Visitenkarte

```
class Visitenkarte extends StatelessWidget {
  final String name;
  final String position;
  final String firma;
  final String email;
  final String telefon;

  const Visitenkarte({
    super.key,
    required this.name,
    required this.position,
    required this.firma,
    required this.email,
    required this.telefon,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: const EdgeInsets.all(16),
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
      elevation: 4,
      child: Padding(
        padding: const EdgeInsets.all(20),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          mainAxisSize: MainAxisSize.min,
          children: [
            Text(name, style: const TextStyle(
              fontSize: 24,
              fontWeight: FontWeight.bold,
            )),
            const SizedBox(height: 4),
            Text(position, style: const TextStyle(fontSize: 16)),
            Text(firma, style: TextStyle(color: Colors.grey[600])),
            const Divider(height: 24),
            _kontaktZeile(Icons.email, email),
            const SizedBox(height: 8),
            _kontaktZeile(Icons.phone, telefon),
          ],
```

```
        ),
      ),
    );
  }

  Widget _kontaktZeile(IconData icon, String text) {
    return Row(
      children: [
        Icon(icon, size: 20, color: Colors.blue),
        const SizedBox(width: 12),
        Text(text),
      ],
    );
  }
}
```

2.2.2.2 Aufgabe 2: Produktkarte

```
class ProduktKarte extends StatelessWidget {
  final String name;
  final double preis;
  final String bildUrl;
  final bool aufLager;

  const ProduktKarte({
    super.key,
    required this.name,
    required this.preis,
    required this.bildUrl,
    required this.aufLager,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      clipBehavior: Clip.antiAlias,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Stack(
            children: [
              Image.network(
                bildUrl,
                height: 150,
                width: double.infinity,
                fit: BoxFit.cover,
              ),
              Positioned(
                top: 8,
                right: 8,
```

```
              child: Container(
                padding: const EdgeInsets.symmetric(horizontal: 8,          ↪
↪  vertical: 4),
                decoration: BoxDecoration(
                  color: aufLager ? Colors.green : Colors.red,
                  borderRadius: BorderRadius.circular(4),
                ),
                child: Text(
                  aufLager ? 'Auf Lager' : 'Ausverkauft',
                  style: const TextStyle(color: Colors.white, fontSize: 12),
                ),
              ),
            ),
          ],
        ),
        Padding(
          padding: const EdgeInsets.all(12),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(name, style: const TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
              )),
              const SizedBox(height: 4),
              Text('${preis.toStringAsFixed(2)} €',
                style: const TextStyle(fontSize: 16, color: Colors.green)),
              const SizedBox(height: 12),
              SizedBox(
                width: double.infinity,
                child: ElevatedButton.icon(
                  onPressed: aufLager ? () {} : null,
                  icon: const Icon(Icons.shopping_cart),
                  label: const Text('In den Warenkorb'),
                ),
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
}
```

2.2.2.3 Aufgabe 3: Bewertung

```
class Bewertung extends StatelessWidget {
  final int sterne;
  final int maxSterne;
```

```
  final int? anzahlBewertungen;

  const Bewertung({
    super.key,
    required this.sterne,
    this.maxSterne = 5,
    this.anzahlBewertungen,
  });

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisSize: MainAxisSize.min,
      children: [
        ...List.generate(maxSterne, (i) => Icon(
          i < sterne ? Icons.star : Icons.star_border,
          color: i < sterne ? Colors.amber : Colors.grey,
          size: 24,
        )),
        if (anzahlBewertungen != null) ...[
          const SizedBox(width: 8),
          Text('($anzahlBewertungen)', style: TextStyle(color: Colors.grey)),
        ],
      ],
    );
  }
}
```

2.2.3 Ressourcen

2.2.3.1 Offizielle Dokumentation

- StatelessWidget
- Widget catalog

2.2.3.2 Nächste Einheit

Einheit 2.3: StatefulWidget Grundlagen

2.3 Einheit 2.3: StatefulWidget Grundlagen

2.3.0.1 3.1 StatefulWidget vs. StatelessWidget

StatelessWidget	StatefulWidget
Unveränderlich	Hat veränderlichen State
`build()` nur bei Parent-Änderung	`build()` auch bei State-Änderung
Für statische UI	Für interaktive UI

2.3.0.2 3.2 Struktur eines StatefulWidget

```
class Zähler extends StatefulWidget {
  const Zähler({super.key});

  @override
  State<Zähler> createState() => _ZählerState();
}

class _ZählerState extends State<Zähler> {
  int _anzahl = 0;

  void _erhöhen() {
    setState(() {
      _anzahl++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Text('Zähler: $_anzahl'),
        ElevatedButton(
          onPressed: _erhöhen,
          child: Text('Erhöhen'),
        ),
      ],
    );
  }
}
```

2.3.0.3 3.3 setState() verstehen

Wichtig: `setState()` signalisiert Flutter, dass sich der State geändert hat und `build()` neu aufgerufen werden soll.

```
// RICHTIG:
void _erhöhen() {
  setState(() {
    _anzahl++;
  });
}

// AUCH RICHTIG:
void _erhöhen() {
  _anzahl++;
  setState(() {});
}

// FALSCH – UI wird nicht aktualisiert:
void _erhöhen() {
```

```
  _anzahl++;  // Kein setState!
}
```

2.3.0.4 3.4 Wann StatefulWidget verwenden?

StatefulWidget wenn: - Benutzerinteraktion den UI ändert - Animationen - Timer / Streams - Formularfelder

StatelessWidget wenn: - UI hängt nur von Parametern ab - Keine Benutzerinteraktion

2.3.0.5 3.5 Beispiel: Toggle-Button

```
class ToggleButton extends StatefulWidget {
  final String label;
  final ValueChanged<bool>? onChanged;

  const ToggleButton({super.key, required this.label, this.onChanged});

  @override
  State<ToggleButton> createState() => _ToggleButtonState();
}

class _ToggleButtonState extends State<ToggleButton> {
  bool _aktiv = false;

  void _toggle() {
    setState(() {
      _aktiv = !_aktiv;
    });
    widget.onChanged?.call(_aktiv);
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: _toggle,
      child: Container(
        padding: EdgeInsets.symmetric(horizontal: 16, vertical: 8),
        decoration: BoxDecoration(
          color: _aktiv ? Colors.blue : Colors.grey[300],
          borderRadius: BorderRadius.circular(20),
        ),
        child: Text(
          widget.label,
          style: TextStyle(
            color: _aktiv ? Colors.white : Colors.black,
          ),
        ),
      ),
    );
  }
```

```
}
```

2.3.0.6 3.6 Zugriff auf Widget-Properties

In der State-Klasse über `widget.propertyName`:

```
class Begrüßung extends StatefulWidget {
  final String name;  // Widget-Property
  const Begrüßung({super.key, required this.name});

  @override
  State<Begrüßung> createState() => _BegrüßungState();
}

class _BegrüßungState extends State<Begrüßung> {
  int _klicks = 0;  // State-Variable

  @override
  Widget build(BuildContext context) {
    return Text('Hallo ${widget.name}! Klicks: $_klicks');
    //                  ^^^^^^^^^^^ Widget-Property
  }
}
```

2.3.0.7 3.7 Beispiel: Counter-App

```
class CounterApp extends StatefulWidget {
  const CounterApp({super.key});

  @override
  State<CounterApp> createState() => _CounterAppState();
}

class _CounterAppState extends State<CounterApp> {
  int _counter = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Counter')),
      body: Center(
        child: Text(
          '$_counter',
          style: Theme.of(context).textTheme.displayLarge,
        ),
      ),
      floatingActionButton: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          FloatingActionButton(
```

```
            onPressed: () => setState(() => _counter++),
            child: Icon(Icons.add),
          ),
          SizedBox(height: 8),
          FloatingActionButton(
            onPressed: () => setState(() => _counter--),
            child: Icon(Icons.remove),
          ),
        ],
      ),
    );
  }
}
```

2.3.1 Übung

2.3.1.1 Aufgabe 1: Einfacher Zähler (15 Min.)

Erstelle einen Zähler mit Plus/Minus-Buttons:

```
// Anforderungen:
// - Zähler startet bei 0
// - Plus-Button erhöht um 1
// - Minus-Button verringert um 1
// - Reset-Button setzt auf 0
// - Zähler kann nicht unter 0 fallen
```

2.3.1.2 Aufgabe 2: Farbwechsler (20 Min.)

Erstelle ein Widget, das bei Tap die Farbe wechselt:

```
// Anforderungen:
// - Container mit Farbe
// - Bei Tap: Wechsle zur nächsten Farbe aus einer Liste
// - Zeige den Farbnamen an
// - Bonus: Animation beim Wechsel
```

2.3.1.3 Aufgabe 3: Todo-Item (20 Min.)

Erstelle ein einzelnes Todo-Item Widget:

```
class TodoItem extends StatefulWidget {
  final String text;
  final ValueChanged<bool>? onChanged;

  const TodoItem({super.key, required this.text, this.onChanged});
  // ...
}

// Anforderungen:
// - Checkbox zum Abhaken
// - Text durchgestrichen wenn erledigt
```

```
// - Callback bei Änderung
```

2.3.1.4 Bonusaufgabe: Stoppuhr

```
// Anforderungen:
// - Start/Stop Button
// - Anzeige: MM:SS.ms
// - Reset Button
// - Hint: Timer.periodic()
```

2.3.2 Lösung

2.3.2.1 Aufgabe 1

```
class Zähler extends StatefulWidget {
  const Zähler({super.key});

  @override
  State<Zähler> createState() => _ZählerState();
}

class _ZählerState extends State<Zähler> {
  int _wert = 0;

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text('$_wert', style: TextStyle(fontSize: 48)),
        SizedBox(height: 20),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            IconButton(
              icon: Icon(Icons.remove),
              onPressed: _wert > 0 ? () => setState(() => _wert--) : null,
            ),
            IconButton(
              icon: Icon(Icons.add),
              onPressed: () => setState(() => _wert++),
            ),
            IconButton(
              icon: Icon(Icons.refresh),
              onPressed: () => setState(() => _wert = 0),
            ),
          ],
        ),
      ],
    );
```

```
  }
}
```

2.3.2.2 Aufgabe 2

```
class Farbwechsler extends StatefulWidget {
  const Farbwechsler({super.key});

  @override
  State<Farbwechsler> createState() => _FarbwechslerState();
}

class _FarbwechslerState extends State<Farbwechsler> {
  final _farben = [
    (Colors.red, 'Rot'),
    (Colors.green, 'Grün'),
    (Colors.blue, 'Blau'),
    (Colors.orange, 'Orange'),
  ];
  int _index = 0;

  void _wechsle() {
    setState(() {
      _index = (_index + 1) % _farben.length;
    });
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: _wechsle,
      child: AnimatedContainer(
        duration: Duration(milliseconds: 300),
        width: 200,
        height: 200,
        color: _farben[_index].$1,
        child: Center(
          child: Text(_farben[_index].$2,
            style: TextStyle(color: Colors.white, fontSize: 24)),
        ),
      ),
    );
  }
}
```

2.3.2.3 Aufgabe 3

```
class TodoItem extends StatefulWidget {
  final String text;
```

```
  final ValueChanged<bool>? onChanged;

  const TodoItem({super.key, required this.text, this.onChanged});

  @override
  State<TodoItem> createState() => _TodoItemState();
}

class _TodoItemState extends State<TodoItem> {
  bool _erledigt = false;

  @override
  Widget build(BuildContext context) {
    return ListTile(
      leading: Checkbox(
        value: _erledigt,
        onChanged: (value) {
          setState(() => _erledigt = value ?? false);
          widget.onChanged?.call(_erledigt);
        },
      ),
      title: Text(
        widget.text,
        style: TextStyle(
          decoration: _erledigt ? TextDecoration.lineThrough : null,
          color: _erledigt ? Colors.grey : null,
        ),
      ),
    );
  }
}
```

2.3.3 Ressourcen

2.3.3.1 Dokumentation

- StatefulWidget
- State

2.3.3.2 Nächste Einheit

Einheit 2.4: Widget-Lifecycle & Keys

2.4 Einheit 2.4: Lifecycle & Keys

2.4.0.1 4.1 State Lifecycle

```
class MeinWidget extends StatefulWidget {
  @override
  State<MeinWidget> createState() => _MeinWidgetState();
}
```

```
class _MeinWidgetState extends State<MeinWidget> {
  @override
  void initState() {
    super.initState();
    // 1. Einmalig beim Erstellen – Initialisierungen
    print('initState');
  }

  @override
  void didChangeDependencies() {
    super.didChangeDependencies();
    // 2. Nach initState und wenn InheritedWidget sich ändert
    print('didChangeDependencies');
  }

  @override
  void didUpdateWidget(MeinWidget oldWidget) {
    super.didUpdateWidget(oldWidget);
    // 3. Wenn Parent das Widget mit neuen Properties rebuildet
    print('didUpdateWidget');
  }

  @override
  Widget build(BuildContext context) {
    // 4. Bei jedem Rebuild
    print('build');
    return Container();
  }

  @override
  void dispose() {
    // 5. Beim Entfernen – Cleanup
    print('dispose');
    super.dispose();
  }
}
```

2.4.0.2 4.2 Typische Verwendung

```
class DatenLader extends StatefulWidget {
  final int userId;
  const DatenLader({super.key, required this.userId});

  @override
  State<DatenLader> createState() => _DatenLaderState();
}

class _DatenLaderState extends State<DatenLader> {
  String? _daten;
```

```
  late final StreamSubscription _subscription;

  @override
  void initState() {
    super.initState();
    _ladeDaten();
    _subscription = eventBus.listen(_onEvent);
  }

  @override
  void didUpdateWidget(DatenLader oldWidget) {
    super.didUpdateWidget(oldWidget);
    if (oldWidget.userId != widget.userId) {
      _ladeDaten();  // Neu laden wenn userId sich ändert
    }
  }

  @override
  void dispose() {
    _subscription.cancel();  // Wichtig: Subscription beenden!
    super.dispose();
  }

  void _ladeDaten() async {
    var daten = await api.lade(widget.userId);
    setState(() => _daten = daten);
  }

  @override
  Widget build(BuildContext context) {
    return Text(_daten ?? 'Lädt...');
  }
}
```

2.4.0.3 4.3 Keys verstehen

Keys helfen Flutter, Widgets zu identifizieren:

```
// OHNE Key – Flutter kann Widgets verwechseln
ListView(
  children: [
    TodoItem(text: 'A'),  // Position 0
    TodoItem(text: 'B'),  // Position 1
  ],
)

// MIT Key – Flutter erkennt welches Widget welches ist
ListView(
  children: [
    TodoItem(key: ValueKey('a'), text: 'A'),
    TodoItem(key: ValueKey('b'), text: 'B'),
```

```
  ],
)
```

2.4.0.4 4.4 Key-Typen

```
// ValueKey – basierend auf einem Wert
TodoItem(key: ValueKey(todo.id), ...)

// ObjectKey – basierend auf Objekt-Identität
TodoItem(key: ObjectKey(todo), ...)

// UniqueKey – immer einzigartig (neu bei jedem Build)
TodoItem(key: UniqueKey(), ...)

// GlobalKey – zugriff auf State von außen
final _formKey = GlobalKey<FormState>();
Form(key: _formKey, ...)
_formKey.currentState?.validate();
```

2.4.0.5 4.5 Wann Keys verwenden?

Keys sind wichtig bei: - Listen mit StatefulWidgets - Widgets die umsortiert werden - Widgets mit Animationen - Formularen (`GlobalKey<FormState>`)

```
// Beispiel: Sortierbare Liste
class TodoList extends StatefulWidget {
  @override
  State<TodoList> createState() => _TodoListState();
}

class _TodoListState extends State<TodoList> {
  var _todos = ['A', 'B', 'C'];

  void _shuffle() {
    setState(() => _todos.shuffle());
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        ElevatedButton(onPressed: _shuffle, child: Text('Mischen')),
        for (var todo in _todos)
          TodoItem(
            key: ValueKey(todo),  // Wichtig für korrektes Verhalten!
            text: todo,
          ),
      ],
    );
  }
```

```
}
```

2.4.0.6 4.6 GlobalKey Beispiel

```
class FormularSeite extends StatefulWidget {
  @override
  State<FormularSeite> createState() => _FormularSeiteState();
}

class _FormularSeiteState extends State<FormularSeite> {
  final _formKey = GlobalKey<FormState>();

  void _submit() {
    if (_formKey.currentState?.validate() ?? false) {
      _formKey.currentState?.save();
      print('Formular gültig!');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        children: [
          TextFormField(validator: (v) => v!.isEmpty ? 'Pflichtfeld' : null),
          ElevatedButton(onPressed: _submit, child: Text('Absenden')),
        ],
      ),
    );
  }
}
```

2.4.1 Übung

2.4.1.1 Aufgabe 1: Lifecycle-Logger (15 Min.)

Erstelle ein Widget das alle Lifecycle-Events loggt:

```
class LifecycleLogger extends StatefulWidget {
  final String name;
  const LifecycleLogger({super.key, required this.name});
  // TODO: Implementiere alle Lifecycle-Methoden mit print()
}

// Test: Füge mehrere LifecycleLogger in ein Widget ein
// und beobachte die Reihenfolge der Logs
```

2.4.1.2 Aufgabe 2: Timer mit Cleanup (20 Min.)

```
class CountdownTimer extends StatefulWidget {
  final int sekunden;
  final VoidCallback? onFinished;

  const CountdownTimer({super.key, required this.sekunden, this.onFinished});
  // TODO:
  // - Starte Timer in initState
  // - Cancele Timer in dispose
  // - Zeige verbleibende Zeit an
}
```

2.4.1.3 Aufgabe 3: Keys verstehen (20 Min.)

Erstelle zwei Listen — eine mit Keys, eine ohne:

```
// Jedes Item hat einen internen Counter
// Bei Shuffle: Beobachte den Unterschied!

class CounterItem extends StatefulWidget {
  final String label;
  const CounterItem({super.key, required this.label});
  // Interner State: _count
}
```

2.4.1.4 Bonusaufgabe: GlobalKey

```
// Erstelle ein Widget das von außen zurückgesetzt werden kann
// Hint: GlobalKey<_MeinWidgetState>
```

2.4.2 Lösung

2.4.2.1 Aufgabe 1

```
class LifecycleLogger extends StatefulWidget {
  final String name;
  const LifecycleLogger({super.key, required this.name});

  @override
  State<LifecycleLogger> createState() => _LifecycleLoggerState();
}

class _LifecycleLoggerState extends State<LifecycleLogger> {
  @override
  void initState() {
    super.initState();
    print('[${widget.name}] initState');
  }

  @override
```

```
  void didChangeDependencies() {
    super.didChangeDependencies();
    print('[${widget.name}] didChangeDependencies');
  }

  @override
  void didUpdateWidget(LifecycleLogger oldWidget) {
    super.didUpdateWidget(oldWidget);
    print('[${widget.name}] didUpdateWidget');
  }

  @override
  Widget build(BuildContext context) {
    print('[${widget.name}] build');
    return Text(widget.name);
  }

  @override
  void dispose() {
    print('[${widget.name}] dispose');
    super.dispose();
  }
}
```

2.4.2.2 Aufgabe 2

```
class CountdownTimer extends StatefulWidget {
  final int sekunden;
  final VoidCallback? onFinished;

  const CountdownTimer({super.key, required this.sekunden, this.onFinished});

  @override
  State<CountdownTimer> createState() => _CountdownTimerState();
}

class _CountdownTimerState extends State<CountdownTimer> {
  late int _verbleibend;
  Timer? _timer;

  @override
  void initState() {
    super.initState();
    _verbleibend = widget.sekunden;
    _starteTimer();
  }

  void _starteTimer() {
    _timer = Timer.periodic(Duration(seconds: 1), (timer) {
      if (_verbleibend > 0) {
```

```
        setState(() => _verbleibend--);
      } else {
        timer.cancel();
        widget.onFinished?.call();
      }
    });
  }

  @override
  void dispose() {
    _timer?.cancel();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Text(
      '$_verbleibend',
      style: TextStyle(fontSize: 48),
    );
  }
}
```

2.4.2.3 Aufgabe 3

```
class CounterItem extends StatefulWidget {
  final String label;
  const CounterItem({super.key, required this.label});

  @override
  State<CounterItem> createState() => _CounterItemState();
}

class _CounterItemState extends State<CounterItem> {
  int _count = 0;

  @override
  Widget build(BuildContext context) {
    return ListTile(
      title: Text('${widget.label}: $_count'),
      trailing: IconButton(
        icon: Icon(Icons.add),
        onPressed: () => setState(() => _count++),
      ),
    );
  }
}

// Vergleich:
// OHNE Key: State bleibt an Position
```

```
// MIT Key: State folgt dem Widget
```

2.4.3 Ressourcen

2.4.3.1 Dokumentation

- State lifecycle
- Keys
- When to use Keys

2.4.3.2 Nächste Einheit

Einheit 2.5: Layout Basics: Row, Column, Stack

2.5 Einheit 2.5: Layout Basics

2.5.0.1 5.1 Row — Horizontales Layout

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,  // Horizontal
  crossAxisAlignment: CrossAxisAlignment.start, // Vertikal
  children: [
    Icon(Icons.star),
    Text('Titel'),
    Icon(Icons.star),
  ],
)
```

MainAxisAlignment (Horizontal bei Row)

- `start` — Links
- `end` — Rechts
- `center` — Zentriert
- `spaceBetween` — Gleichmäßig, ohne Rand
- `spaceAround` — Gleichmäßig, halber Rand
- `spaceEvenly` — Gleichmäßig, gleicher Abstand

2.5.0.2 5.2 Column — Vertikales Layout

```
Column(
  mainAxisAlignment: MainAxisAlignment.start,   // Vertikal
  crossAxisAlignment: CrossAxisAlignment.center, // Horizontal
  children: [
    Text('Zeile 1'),
    Text('Zeile 2'),
    Text('Zeile 3'),
  ],
)
```

2.5.0.3 5.3 MainAxisSize

```
Column(
  mainAxisSize: MainAxisSize.min,  // Nur so groß wie nötig
  // mainAxisSize: MainAxisSize.max,  // Füllt verfügbaren Platz
  children: [...],
)
```

2.5.0.4 5.4 Stack — Überlappende Widgets

```
Stack(
  children: [
    Image.network('background.jpg'),
    Positioned(
      bottom: 16,
      left: 16,
      child: Text('Überschrift'),
    ),
    Positioned(
      top: 8,
      right: 8,
      child: Icon(Icons.favorite, color: Colors.red),
    ),
  ],
)
```

Positioned

```
Positioned(
  top: 10,      // Abstand von oben
  left: 10,     // Abstand von links
  right: 10,    // Abstand von rechts
  bottom: 10,   // Abstand von unten
  child: Widget(),
)

Positioned.fill(  // Füllt den gesamten Stack
  child: Widget(),
)
```

2.5.0.5 5.5 Stack Alignment

```
Stack(
  alignment: Alignment.center,  // Standard-Ausrichtung für nicht-positioned
  children: [
    Container(width: 200, height: 200, color: Colors.blue),
    Container(width: 100, height: 100, color: Colors.red),  // Zentriert
  ],
)
```

2.5.0.6 5.6 Nested Layouts

```
Column(
  children: [
    Text('Header'),
    Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Column(children: [Icon(Icons.home), Text('Home')]),
        Column(children: [Icon(Icons.search), Text('Suche')]),
        Column(children: [Icon(Icons.person), Text('Profil')]),
      ],
    ),
    Text('Footer'),
  ],
)
```

2.5.0.7 5.7 Beispiel: Profilkarte

```
class ProfilKarte extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Card(
      child: Stack(
        children: [
          // Hintergrundbild
          Positioned.fill(
            child: Image.network('banner.jpg', fit: BoxFit.cover),
          ),
          // Gradient Overlay
          Positioned.fill(
            child: Container(
              decoration: BoxDecoration(
                gradient: LinearGradient(
                  begin: Alignment.topCenter,
                  end: Alignment.bottomCenter,
                  colors: [Colors.transparent, Colors.black87],
                ),
              ),
            ),
          ),
          // Content
          Positioned(
            bottom: 16,
            left: 16,
            right: 16,
            child: Row(
              children: [
                CircleAvatar(radius: 30, child: Icon(Icons.person)),
                SizedBox(width: 16),
                Column(
```

```
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text('Max Mustermann',
                    style: TextStyle(color: Colors.white, fontSize: 18)),
                  Text('Entwickler',
                    style: TextStyle(color: Colors.white70)),
                ],
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
}
```

2.5.1 Übung

2.5.1.1 Aufgabe 1: Social Media Post (20 Min.)

Erstelle einen Social-Media-Post:

```
+-----------------------------+
| [Avatar] Username    [···]  |  <- Row
|                             |
| [  Großes Bild hier  ]      |  <- Image
|                             |
| ♡ ♡ ↗ □                     |  <- Row mit Icons
| 1.234 Likes                 |
| Beschreibungstext hier...   |
+-----------------------------+
```

2.5.1.2 Aufgabe 2: App-Drawer (20 Min.)

Erstelle einen Navigations-Drawer:

```
+-----------------------------+
| +-------------------------+ |
| |   Header mit Bild       | | <- Stack
| |   Name & Email          | |
| +-------------------------+ |
+-----------------------------+
| □ Home                      |  <- ListTile
| □ Profil                    |
| ⚙ Einstellungen             |
| -------------------------   |  <- Divider
| □ Logout                    |
+-----------------------------+
```

2.5.1.3 Aufgabe 3: Badge auf Icon (15 Min.)

Erstelle ein Icon mit Benachrichtigungs-Badge:

```
BadgeIcon(
  icon: Icons.notifications,
  count: 5,
)
// Zeigt das Icon mit rotem Kreis (Zahl) oben rechts
```

2.5.1.4 Bonusaufgabe: Responsive Grid

Erstelle ein Grid das sich an die Bildschirmbreite anpasst.

2.5.2 Lösung

2.5.2.1 Aufgabe 1

```
class SocialPost extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        // Header
        Padding(
          padding: EdgeInsets.all(12),
          child: Row(
            children: [
              CircleAvatar(child: Icon(Icons.person)),
              SizedBox(width: 12),
              Expanded(child: Text('username', style: TextStyle(fontWeight:    ↪
↪  FontWeight.bold))),
              IconButton(icon: Icon(Icons.more_horiz), onPressed: () {}),
            ],
          ),
        ),
        // Bild
        Image.network('https://picsum.photos/400/400', fit: BoxFit.cover),
        // Actions
        Row(
          children: [
            IconButton(icon: Icon(Icons.favorite_border), onPressed: () {}),
            IconButton(icon: Icon(Icons.chat_bubble_outline), onPressed: () {}),
            IconButton(icon: Icon(Icons.send), onPressed: () {}),
            Spacer(),
            IconButton(icon: Icon(Icons.bookmark_border), onPressed: () {}),
          ],
        ),
        // Likes & Beschreibung
        Padding(
          padding: EdgeInsets.symmetric(horizontal: 12),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
```

```
              Text('1.234 Likes', style: TextStyle(fontWeight: FontWeight.bold)),
              SizedBox(height: 4),
              Text('Beschreibungstext hier...'),
            ],
          ),
        ),
      ],
    );
  }
}
```

2.5.2.2 Aufgabe 3

```
class BadgeIcon extends StatelessWidget {
  final IconData icon;
  final int count;

  const BadgeIcon({super.key, required this.icon, required this.count});

  @override
  Widget build(BuildContext context) {
    return Stack(
      clipBehavior: Clip.none,
      children: [
        Icon(icon, size: 32),
        if (count > 0)
          Positioned(
            right: -8,
            top: -8,
            child: Container(
              padding: EdgeInsets.all(4),
              decoration: BoxDecoration(
                color: Colors.red,
                shape: BoxShape.circle,
              ),
              constraints: BoxConstraints(minWidth: 20, minHeight: 20),
              child: Text(
                count > 99 ? '99+' : '$count',
                style: TextStyle(color: Colors.white, fontSize: 12),
                textAlign: TextAlign.center,
              ),
            ),
          ),
      ],
    );
  }
}
```

2.5.3 Ressourcen

2.5.3.1 Dokumentation

- Layouts in Flutter
- Row
- Column
- Stack

2.5.3.2 Nächste Einheit

Einheit 2.6: Container, Sizing & Spacing

2.6 Einheit 2.6: Container & Sizing

2.6.0.1 6.1 Container

```
Container(
  width: 200,
  height: 100,
  padding: EdgeInsets.all(16),
  margin: EdgeInsets.symmetric(horizontal: 20, vertical: 10),
  alignment: Alignment.center,
  decoration: BoxDecoration(
    color: Colors.blue,
    borderRadius: BorderRadius.circular(12),
    border: Border.all(color: Colors.black, width: 2),
    boxShadow: [
      BoxShadow(color: Colors.black26, blurRadius: 10, offset: Offset(0, 4)),
    ],
  ),
  child: Text('Inhalt'),
)
```

2.6.0.2 6.2 SizedBox

```
// Feste Größe
SizedBox(
  width: 100,
  height: 50,
  child: ElevatedButton(...),
)

// Als Spacer
Column(children: [
  Text('Oben'),
  SizedBox(height: 20),  // Abstand
  Text('Unten'),
])

// Volle Breite
```

```
SizedBox(
  width: double.infinity,
  child: ElevatedButton(...),
)
```

2.6.0.3 6.3 Expanded & Flexible

```
Row(children: [
  Container(width: 50, color: Colors.red),
  Expanded(  // Füllt restlichen Platz
    child: Container(color: Colors.green),
  ),
  Container(width: 50, color: Colors.blue),
])

Row(children: [
  Expanded(flex: 2, child: Container(color: Colors.red)),   // 2/3
  Expanded(flex: 1, child: Container(color: Colors.blue)),  // 1/3
])

// Flexible – wie Expanded, aber kann kleiner sein
Row(children: [
  Flexible(
    fit: FlexFit.loose,  // Kann kleiner sein als verfügbar
    child: Text('Langer Text der abgeschnitten werden kann'),
  ),
])
```

2.6.0.4 6.4 Padding

```
Padding(
  padding: EdgeInsets.all(16),
  child: Text('Mit Padding'),
)

// EdgeInsets Varianten
EdgeInsets.all(16)                    // Alle Seiten gleich
EdgeInsets.symmetric(horizontal: 20, vertical: 10)
EdgeInsets.only(left: 10, top: 5)
EdgeInsets.fromLTRB(10, 20, 10, 20)  // Left, Top, Right, Bottom
```

2.6.0.5 6.5 Center & Align

```
Center(child: Text('Zentriert'))

Align(
  alignment: Alignment.topRight,
  child: Text('Oben rechts'),
)
```

```
// Alignment Werte
Alignment.topLeft      Alignment.topCenter      Alignment.topRight
Alignment.centerLeft   Alignment.center         Alignment.centerRight
Alignment.bottomLeft   Alignment.bottomCenter   Alignment.bottomRight
```

2.6.0.6 6.6 ConstrainedBox & FractionallySizedBox

```
// Einschränkungen
ConstrainedBox(
  constraints: BoxConstraints(
    minWidth: 100,
    maxWidth: 300,
    minHeight: 50,
    maxHeight: 200,
  ),
  child: Container(color: Colors.blue),
)

// Prozentuale Größe
FractionallySizedBox(
  widthFactor: 0.8,   // 80% der Breite
  heightFactor: 0.5,  // 50% der Höhe
  child: Container(color: Colors.red),
)
```

2.6.0.7 6.7 AspectRatio

```
AspectRatio(
  aspectRatio: 16 / 9,  // Breite / Höhe
  child: Container(color: Colors.blue),
)
```

2.6.0.8 6.8 Spacer

```
Row(children: [
  Text('Links'),
  Spacer(),  // Füllt den Platz dazwischen
  Text('Rechts'),
])

Row(children: [
  Text('1'),
  Spacer(flex: 1),
  Text('2'),
  Spacer(flex: 2),  // Doppelt so viel Platz
  Text('3'),
])
```

2.6.0.9 6.9 IntrinsicWidth & IntrinsicHeight

```
// Alle Kinder gleich breit wie das breiteste
IntrinsicWidth(
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.stretch,
    children: [
      ElevatedButton(onPressed: () {}, child: Text('Kurz')),
      ElevatedButton(onPressed: () {}, child: Text('Mittellanger Text')),
      ElevatedButton(onPressed: () {}, child: Text('OK')),
    ],
  ),
)
```

2.6.0.10 6.10 Beispiel: Login-Formular Layout

```
class LoginLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Padding(
          padding: EdgeInsets.all(24),
          child: Column(
            children: [
              Spacer(flex: 2),
              Icon(Icons.lock, size: 80, color: Colors.blue),
              SizedBox(height: 48),
              TextField(decoration: InputDecoration(labelText: 'Email')),
              SizedBox(height: 16),
              TextField(decoration: InputDecoration(labelText: 'Passwort')),
              SizedBox(height: 24),
              SizedBox(
                width: double.infinity,
                height: 48,
                child: ElevatedButton(
                  onPressed: () {},
                  child: Text('Anmelden'),
                ),
              ),
              Spacer(flex: 3),
              TextButton(onPressed: () {}, child: Text('Registrieren')),
            ],
          ),
        ),
      ),
    );
  }
}
```

2.6.1 Übung

2.6.1.1 Aufgabe 1: Responsive Buttons (15 Min.)

Erstelle eine Reihe von Buttons die sich den Platz teilen:

```
// 3 Buttons nebeneinander
// Gleicher Abstand dazwischen
// Alle gleich breit
```

2.6.1.2 Aufgabe 2: Pricing Card (25 Min.)

Erstelle eine Preiskarte:

```
+---------------------+
|       BASIC         |  <- Header (farbig)
|      €9.99/mo       |
+---------------------+
|  [x] Feature 1        |
|  [x] Feature 2        |
|  [x] Feature 3        |
|  [ ] Feature 4        |
|  [ ] Feature 5        |
|                     |
|   [ Auswählen ]     |  <- Button unten
+---------------------+
```

2.6.1.3 Aufgabe 3: Aspect Ratio Grid (15 Min.)

Erstelle ein 2x2 Grid mit quadratischen Bildern:

```
// 4 Bilder
// Immer quadratisch (1:1)
// Responsive (passt sich an)
```

2.6.1.4 Bonusaufgabe: Responsive Layout

Erstelle ein Layout das sich ändert: - Schmal: Column - Breit: Row

2.6.2 Lösung

2.6.2.1 Aufgabe 1

```
Row(
  children: [
    Expanded(child: ElevatedButton(onPressed: () {}, child: Text('Eins'))),
    SizedBox(width: 8),
    Expanded(child: ElevatedButton(onPressed: () {}, child: Text('Zwei'))),
    SizedBox(width: 8),
    Expanded(child: ElevatedButton(onPressed: () {}, child: Text('Drei'))),
  ],
)
```

2.6.2.2 Aufgabe 2

```
class PricingCard extends StatelessWidget {
  final String titel;
  final double preis;
  final List<(String, bool)> features;
  final Color farbe;

  const PricingCard({
    super.key,
    required this.titel,
    required this.preis,
    required this.features,
    this.farbe = Colors.blue,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      clipBehavior: Clip.antiAlias,
      child: Column(
        children: [
          Container(
            width: double.infinity,
            padding: EdgeInsets.all(24),
            color: farbe,
            child: Column(
              children: [
                Text(titel, style: TextStyle(
                  color: Colors.white, fontSize: 20, fontWeight:
↪  FontWeight.bold)),
                SizedBox(height: 8),
                Text('€${preis.toStringAsFixed(2)}/mo',
                  style: TextStyle(color: Colors.white, fontSize: 28)),
              ],
            ),
          ),
          Expanded(
            child: Padding(
              padding: EdgeInsets.all(16),
              child: Column(
                children: [
                  for (var (text, aktiv) in features)
                    Padding(
                      padding: EdgeInsets.symmetric(vertical: 4),
                      child: Row(
                        children: [
                          Icon(aktiv ? Icons.check : Icons.close,
                            color: aktiv ? Colors.green : Colors.grey),
                          SizedBox(width: 8),
                          Text(text, style: TextStyle(
```

```
                          color: aktiv ? null : Colors.grey)),
                    ],
                  ),
                ),
                Spacer(),
                SizedBox(
                  width: double.infinity,
                  child: ElevatedButton(
                    onPressed: () {},
                    style: ElevatedButton.styleFrom(backgroundColor: farbe),
                    child: Text('Auswählen'),
                  ),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

2.6.2.3 Aufgabe 3

```
GridView.count(
  crossAxisCount: 2,
  mainAxisSpacing: 8,
  crossAxisSpacing: 8,
  children: List.generate(4, (i) => AspectRatio(
    aspectRatio: 1,
    child: Image.network('https://picsum.photos/200?$i', fit: BoxFit.cover),
  )),
)
```

2.6.3 Ressourcen

2.6.3.1 Dokumentation

- Container
- BoxConstraints

2.6.3.2 Nächste Einheit

Einheit 2.7: Listen & Scrolling

2.7 Einheit 2.7: Listen & Scrolling

2.7.0.1 7.1 ListView

```
// Einfache ListView
ListView(
  children: [
    ListTile(title: Text('Eintrag 1')),
    ListTile(title: Text('Eintrag 2')),
    ListTile(title: Text('Eintrag 3')),
  ],
)

// ListView.builder – für lange Listen (lazy loading)
ListView.builder(
  itemCount: 100,
  itemBuilder: (context, index) {
    return ListTile(
      leading: CircleAvatar(child: Text('$index')),
      title: Text('Element $index'),
      subtitle: Text('Beschreibung'),
      trailing: Icon(Icons.arrow_forward_ios),
      onTap: () => print('Tapped $index'),
    );
  },
)
```

2.7.0.2 7.2 ListView.separated

```
ListView.separated(
  itemCount: 20,
  itemBuilder: (context, index) {
    return ListTile(title: Text('Eintrag $index'));
  },
  separatorBuilder: (context, index) {
    return Divider(height: 1);
  },
)
```

2.7.0.3 7.3 GridView

```
// GridView.count – feste Spaltenanzahl
GridView.count(
  crossAxisCount: 2,
  mainAxisSpacing: 8,
  crossAxisSpacing: 8,
  padding: EdgeInsets.all(8),
  children: List.generate(10, (i) => Card(
    child: Center(child: Text('Item $i')),
  )),
```

```
)

// GridView.builder – lazy loading
GridView.builder(
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 3,
    mainAxisSpacing: 4,
    crossAxisSpacing: 4,
    childAspectRatio: 1.0,
  ),
  itemCount: 50,
  itemBuilder: (context, index) {
    return Container(
      color: Colors.primaries[index % Colors.primaries.length],
      child: Center(child: Text('$index')),
    );
  },
)

// GridView.extent – maximale Breite pro Item
GridView.extent(
  maxCrossAxisExtent: 150,
  children: [...],
)
```

2.7.0.4 7.4 SingleChildScrollView

```
// Für einzelne scrollbare Inhalte
SingleChildScrollView(
  child: Column(
    children: [
      Container(height: 200, color: Colors.red),
      Container(height: 200, color: Colors.green),
      Container(height: 200, color: Colors.blue),
      // ... mehr Inhalte
    ],
  ),
)

// Mit horizontalem Scrolling
SingleChildScrollView(
  scrollDirection: Axis.horizontal,
  child: Row(
    children: List.generate(10, (i) => Container(
      width: 150,
      height: 100,
      margin: EdgeInsets.all(8),
      color: Colors.primaries[i],
    )),
  ),
```

```
)
```

2.7.0.5 7.5 ScrollController

```
class ScrollExample extends StatefulWidget {
  @override
  State<ScrollExample> createState() => _ScrollExampleState();
}

class _ScrollExampleState extends State<ScrollExample> {
  final _controller = ScrollController();

  @override
  void initState() {
    super.initState();
    _controller.addListener(_onScroll);
  }

  void _onScroll() {
    print('Position: ${_controller.offset}');
    if (_controller.position.atEdge) {
      if (_controller.position.pixels == 0) {
        print('Am Anfang');
      } else {
        print('Am Ende');
      }
    }
  }

  void _scrollToTop() {
    _controller.animateTo(
      0,
      duration: Duration(milliseconds: 500),
      curve: Curves.easeOut,
    );
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      controller: _controller,
      itemCount: 50,
      itemBuilder: (context, index) => ListTile(title: Text('Item $index')),
    );
```

```
  }
}
```

2.7.0.6 7.6 RefreshIndicator (Pull-to-Refresh)

```
class RefreshExample extends StatefulWidget {
  @override
  State<RefreshExample> createState() => _RefreshExampleState();
}

class _RefreshExampleState extends State<RefreshExample> {
  List<String> items = List.generate(10, (i) => 'Item $i');

  Future<void> _refresh() async {
    await Future.delayed(Duration(seconds: 2));
    setState(() {
      items = List.generate(10, (i) => 'Neu geladen $i');
    });
  }

  @override
  Widget build(BuildContext context) {
    return RefreshIndicator(
      onRefresh: _refresh,
      child: ListView.builder(
        itemCount: items.length,
        itemBuilder: (context, index) {
          return ListTile(title: Text(items[index]));
        },
      ),
    );
  }
}
```

2.7.0.7 7.7 CustomScrollView & Slivers

```
CustomScrollView(
  slivers: [
    // Collapsing Header
    SliverAppBar(
      expandedHeight: 200,
      floating: false,
      pinned: true,
      flexibleSpace: FlexibleSpaceBar(
        title: Text('Titel'),
        background: Image.network(
          'https://picsum.photos/400/200',
          fit: BoxFit.cover,
        ),
```

```
      ),
    ),
    // Liste
    SliverList(
      delegate: SliverChildBuilderDelegate(
        (context, index) => ListTile(title: Text('Item $index')),
        childCount: 20,
      ),
    ),
    // Grid
    SliverGrid(
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
        crossAxisCount: 2,
      ),
      delegate: SliverChildBuilderDelegate(
        (context, index) => Card(child: Center(child: Text('Grid $index'))),
        childCount: 10,
      ),
    ),
  ],
)
```

2.7.0.8 7.8 Beispiel: Kontaktliste mit Suche

```
class ContactList extends StatefulWidget {
  @override
  State<ContactList> createState() => _ContactListState();
}

class _ContactListState extends State<ContactList> {
  final allContacts = [
    'Anna', 'Bob', 'Clara', 'David', 'Eva',
    'Frank', 'Greta', 'Hans', 'Iris', 'Jan',
  ];

  List<String> filteredContacts = [];
  final searchController = TextEditingController();

  @override
  void initState() {
    super.initState();
    filteredContacts = allContacts;
  }

  void _filter(String query) {
    setState(() {
      filteredContacts = allContacts
          .where((c) => c.toLowerCase().contains(query.toLowerCase()))
          .toList();
    });
```

```
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Padding(
          padding: EdgeInsets.all(8),
          child: TextField(
            controller: searchController,
            decoration: InputDecoration(
              hintText: 'Suchen...',
              prefixIcon: Icon(Icons.search),
              border: OutlineInputBorder(),
            ),
            onChanged: _filter,
          ),
        ),
        Expanded(
          child: ListView.builder(
            itemCount: filteredContacts.length,
            itemBuilder: (context, index) {
              final contact = filteredContacts[index];
              return ListTile(
                leading: CircleAvatar(child: Text(contact[0])),
                title: Text(contact),
              );
            },
          ),
        ),
      ],
    );
  }
}
```

2.7.1 Übung

2.7.1.1 Aufgabe 1: Chat-Verlauf (20 Min.)

Erstelle eine Chat-Ansicht:

```
// Liste von Nachrichten
// Neueste unten (reversed)
// Scroll zum Ende bei neuer Nachricht
// Unterscheide eigene/fremde Nachrichten
```

2.7.1.2 Aufgabe 2: Foto-Galerie (20 Min.)

Erstelle eine Foto-Galerie:

```
+-----+-----+-----+
|     |     |     |
```

```
| Img | Img | Img |
|     |     |     |
+-----+-----+-----+
|     |     |     |
| Img | Img | Img |
|     |     |     |
+-----+-----+-----+
```

- 3 Spalten
- Quadratische Bilder
- Pull-to-Refresh zum "Neuladen"

2.7.1.3 Aufgabe 3: Infinite Scroll (15 Min.)

Erstelle eine Liste die beim Scrollen mehr lädt:

```
// Starte mit 20 Items
// Lade 10 mehr wenn am Ende
// Zeige Ladeindikator
```

2.7.1.4 Bonusaufgabe: Alphabet-Index

Erstelle eine Kontaktliste mit Buchstaben-Index am Rand (A-Z).

2.7.2 Lösung

2.7.2.1 Aufgabe 1

```
class ChatView extends StatefulWidget {
  @override
  State<ChatView> createState() => _ChatViewState();
}

class _ChatViewState extends State<ChatView> {
  final _controller = ScrollController();
  final _textController = TextEditingController();
  final List<({String text, bool isMe})> messages = [
    (text: 'Hallo!', isMe: false),
    (text: 'Hi, wie gehts?', isMe: true),
    (text: 'Gut, danke!', isMe: false),
  ];

  void _sendMessage() {
    if (_textController.text.isEmpty) return;
    setState(() {
      messages.add((text: _textController.text, isMe: true));
      _textController.clear();
    });
    Future.delayed(Duration(milliseconds: 100), () {
      _controller.animateTo(
        _controller.position.maxScrollExtent,
        duration: Duration(milliseconds: 300),
        curve: Curves.easeOut,
```

```
    );
  });
}

@override
Widget build(BuildContext context) {
  return Column(
    children: [
      Expanded(
        child: ListView.builder(
          controller: _controller,
          itemCount: messages.length,
          itemBuilder: (context, index) {
            final msg = messages[index];
            return Align(
              alignment: msg.isMe ? Alignment.centerRight :
↪ Alignment.centerLeft,
              child: Container(
                margin: EdgeInsets.all(8),
                padding: EdgeInsets.symmetric(horizontal: 16, vertical: 10),
                decoration: BoxDecoration(
                  color: msg.isMe ? Colors.blue : Colors.grey[300],
                  borderRadius: BorderRadius.circular(20),
                ),
                child: Text(
                  msg.text,
                  style: TextStyle(
                    color: msg.isMe ? Colors.white : Colors.black,
                  ),
                ),
              ),
            );
          },
        ),
      ),
      Padding(
        padding: EdgeInsets.all(8),
        child: Row(
          children: [
            Expanded(
              child: TextField(
                controller: _textController,
                decoration: InputDecoration(
                  hintText: 'Nachricht...',
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(25),
                  ),
                ),
                onSubmitted: (_) => _sendMessage(),
              ),
            ),
```

```
              SizedBox(width: 8),
              IconButton(
                icon: Icon(Icons.send),
                onPressed: _sendMessage,
              ),
            ],
          ),
        ),
      ],
    );
  }
}
```

2.7.2.2 Aufgabe 2

```
class PhotoGallery extends StatefulWidget {
  @override
  State<PhotoGallery> createState() => _PhotoGalleryState();
}

class _PhotoGalleryState extends State<PhotoGallery> {
  int _seed = 0;

  Future<void> _refresh() async {
    await Future.delayed(Duration(seconds: 1));
    setState(() {
      _seed = DateTime.now().millisecondsSinceEpoch;
    });
  }

  @override
  Widget build(BuildContext context) {
    return RefreshIndicator(
      onRefresh: _refresh,
      child: GridView.builder(
        padding: EdgeInsets.all(4),
        gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
          crossAxisCount: 3,
          mainAxisSpacing: 4,
          crossAxisSpacing: 4,
          childAspectRatio: 1,
        ),
        itemCount: 30,
        itemBuilder: (context, index) {
          return Image.network(
            'https://picsum.photos/200?random=$_seed$index',
            fit: BoxFit.cover,
          );
        },
      ),
```

```
    );
  }
}
```

2.7.2.3 Aufgabe 3

```
class InfiniteScrollList extends StatefulWidget {
  @override
  State<InfiniteScrollList> createState() => _InfiniteScrollListState();
}

class _InfiniteScrollListState extends State<InfiniteScrollList> {
  final _controller = ScrollController();
  List<int> items = List.generate(20, (i) => i);
  bool isLoading = false;

  @override
  void initState() {
    super.initState();
    _controller.addListener(_onScroll);
  }

  void _onScroll() {
    if (_controller.position.pixels >= _controller.position.maxScrollExtent -
↪ 200) {
      _loadMore();
    }
  }

  Future<void> _loadMore() async {
    if (isLoading) return;
    setState(() => isLoading = true);

    await Future.delayed(Duration(seconds: 1));

    setState(() {
      final start = items.length;
      items.addAll(List.generate(10, (i) => start + i));
      isLoading = false;
    });
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
```

```
    return ListView.builder(
      controller: _controller,
      itemCount: items.length + (isLoading ? 1 : 0),
      itemBuilder: (context, index) {
        if (index == items.length) {
          return Center(
            child: Padding(
              padding: EdgeInsets.all(16),
              child: CircularProgressIndicator(),
            ),
          );
        }
        return ListTile(
          leading: CircleAvatar(child: Text('${items[index]}')),
          title: Text('Element ${items[index]}'),
        );
      },
    );
  }
}
```

2.7.3 Ressourcen

2.7.3.1 Dokumentation

- ListView
- GridView
- CustomScrollView
- Slivers

2.7.3.2 Nächste Einheit

Einheit 2.8: Styling & Themes

2.8 Einheit 2.8: Styling & Themes

2.8.0.1 8.1 ThemeData

```
MaterialApp(
  theme: ThemeData(
    primarySwatch: Colors.blue,
    brightness: Brightness.light,
    scaffoldBackgroundColor: Colors.grey[100],
    appBarTheme: AppBarTheme(
      backgroundColor: Colors.blue,
      foregroundColor: Colors.white,
      elevation: 0,
    ),
  ),
  darkTheme: ThemeData(
    brightness: Brightness.dark,
```

```
    primarySwatch: Colors.blue,
  ),
  themeMode: ThemeMode.system,  // system, light, dark
  home: MyApp(),
)
```

2.8.0.2 8.2 ColorScheme

```
ThemeData(
  colorScheme: ColorScheme.fromSeed(
    seedColor: Colors.deepPurple,
    brightness: Brightness.light,
  ),
)

// Verwendung
Container(
  color: Theme.of(context).colorScheme.primary,
)

// ColorScheme Farben
colorScheme.primary        // Hauptfarbe
colorScheme.onPrimary      // Text auf primary
colorScheme.secondary      // Sekundärfarbe
colorScheme.onSecondary    // Text auf secondary
colorScheme.surface        // Oberfläche (Cards, etc.)
colorScheme.onSurface      // Text auf surface
colorScheme.error          // Fehlerfarbe
colorScheme.onError        // Text auf error
```

2.8.0.3 8.3 TextTheme

```
ThemeData(
  textTheme: TextTheme(
    displayLarge: TextStyle(fontSize: 57, fontWeight: FontWeight.bold),
    displayMedium: TextStyle(fontSize: 45),
    displaySmall: TextStyle(fontSize: 36),
    headlineLarge: TextStyle(fontSize: 32),
    headlineMedium: TextStyle(fontSize: 28),
    headlineSmall: TextStyle(fontSize: 24),
    titleLarge: TextStyle(fontSize: 22),
    titleMedium: TextStyle(fontSize: 16, fontWeight: FontWeight.w500),
    titleSmall: TextStyle(fontSize: 14, fontWeight: FontWeight.w500),
    bodyLarge: TextStyle(fontSize: 16),
    bodyMedium: TextStyle(fontSize: 14),
    bodySmall: TextStyle(fontSize: 12),
    labelLarge: TextStyle(fontSize: 14, fontWeight: FontWeight.w500),
    labelMedium: TextStyle(fontSize: 12),
    labelSmall: TextStyle(fontSize: 11),
```

```
  ),
)

// Verwendung
Text(
  'Überschrift',
  style: Theme.of(context).textTheme.headlineMedium,
)
```

2.8.0.4 8.4 Button Styles

```
ThemeData(
  elevatedButtonTheme: ElevatedButtonThemeData(
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.blue,
      foregroundColor: Colors.white,
      padding: EdgeInsets.symmetric(horizontal: 24, vertical: 12),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(8),
      ),
    ),
  ),
  outlinedButtonTheme: OutlinedButtonThemeData(
    style: OutlinedButton.styleFrom(
      foregroundColor: Colors.blue,
      side: BorderSide(color: Colors.blue),
    ),
  ),
  textButtonTheme: TextButtonThemeData(
    style: TextButton.styleFrom(
      foregroundColor: Colors.blue,
    ),
  ),
)

// Individuelles Styling
ElevatedButton(
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.red,
    minimumSize: Size(200, 50),
  ),
  onPressed: () {},
  child: Text('Button'),
)
```

2.8.0.5 8.5 Input Decoration Theme

```
ThemeData(
  inputDecorationTheme: InputDecorationTheme(
```

```
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    filled: true,
    fillColor: Colors.grey[100],
    contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),
    prefixIconColor: Colors.grey,
    suffixIconColor: Colors.grey,
    labelStyle: TextStyle(color: Colors.grey[700]),
    hintStyle: TextStyle(color: Colors.grey[400]),
    errorStyle: TextStyle(color: Colors.red),
    focusedBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.blue, width: 2),
    ),
    errorBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.red),
    ),
  ),
)
```

2.8.0.6 8.6 Card & Dialog Theme

```
ThemeData(
  cardTheme: CardTheme(
    elevation: 2,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    margin: EdgeInsets.all(8),
  ),
  dialogTheme: DialogTheme(
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(16),
    ),
    titleTextStyle: TextStyle(
      fontSize: 20,
      fontWeight: FontWeight.bold,
      color: Colors.black,
    ),
  ),
)
```

2.8.0.7 8.7 Theme Extensions

```
// Eigene Theme-Erweiterung definieren
class CustomColors extends ThemeExtension<CustomColors> {
  final Color? success;
```

```
  final Color? warning;

  CustomColors({this.success, this.warning});

  @override
  CustomColors copyWith({Color? success, Color? warning}) {
    return CustomColors(
      success: success ?? this.success,
      warning: warning ?? this.warning,
    );
  }

  @override
  CustomColors lerp(ThemeExtension<CustomColors>? other, double t) {
    if (other is! CustomColors) return this;
    return CustomColors(
      success: Color.lerp(success, other.success, t),
      warning: Color.lerp(warning, other.warning, t),
    );
  }
}

// Im Theme registrieren
ThemeData(
  extensions: [
    CustomColors(
      success: Colors.green,
      warning: Colors.orange,
    ),
  ],
)

// Verwenden
final customColors = Theme.of(context).extension<CustomColors>()!;
Container(color: customColors.success)
```

2.8.0.8 8.8 Google Fonts

```
# pubspec.yaml
dependencies:
  google_fonts: ^6.1.0
```

```
import 'package:google_fonts/google_fonts.dart';

// Einzelner Text
Text(
  'Hello',
  style: GoogleFonts.roboto(fontSize: 24),
)
```

```
// Im Theme
ThemeData(
  textTheme: GoogleFonts.latoTextTheme(),
)

// Angepasstes TextTheme
ThemeData(
  textTheme: GoogleFonts.latoTextTheme().copyWith(
    headlineMedium: GoogleFonts.oswald(fontSize: 28),
  ),
)
```

2.8.0.9 8.9 Beispiel: Komplettes Theme

```
class AppTheme {
  static ThemeData get lightTheme {
    return ThemeData(
      useMaterial3: true,
      colorScheme: ColorScheme.fromSeed(
        seedColor: Color(0xFF6750A4),
        brightness: Brightness.light,
      ),
      appBarTheme: AppBarTheme(
        centerTitle: true,
        elevation: 0,
      ),
      cardTheme: CardTheme(
        elevation: 2,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
      ),
      elevatedButtonTheme: ElevatedButtonThemeData(
        style: ElevatedButton.styleFrom(
          padding: EdgeInsets.symmetric(horizontal: 24, vertical: 12),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(8),
          ),
        ),
      ),
      inputDecorationTheme: InputDecorationTheme(
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(12),
        ),
        filled: true,
      ),
    );
  }

  static ThemeData get darkTheme {
```

```
    return ThemeData(
      useMaterial3: true,
      colorScheme: ColorScheme.fromSeed(
        seedColor: Color(0xFF6750A4),
        brightness: Brightness.dark,
      ),
      appBarTheme: AppBarTheme(
        centerTitle: true,
        elevation: 0,
      ),
      cardTheme: CardTheme(
        elevation: 2,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
      ),
    );
  }
}

// Verwendung
MaterialApp(
  theme: AppTheme.lightTheme,
  darkTheme: AppTheme.darkTheme,
  themeMode: ThemeMode.system,
  home: MyApp(),
)
```

2.8.1 Übung

2.8.1.1 Aufgabe 1: Custom Theme (20 Min.)

Erstelle ein eigenes App-Theme:

```
// Primärfarbe: Orange (#FF6B00)
// Abgerundete Buttons (radius: 20)
// Gefüllte TextFields
// Cards mit Schatten
```

2.8.1.2 Aufgabe 2: Dark Mode Toggle (20 Min.)

Erstelle eine App mit Theme-Umschalter:

```
// Switch in der AppBar
// Light/Dark Mode wechseln
// Zustand merken
```

2.8.1.3 Aufgabe 3: Branded Components (15 Min.)

Style diese Komponenten passend:

```
+-------------------------+
```

```
|   Mein Shop           |  <- AppBar
+-------------------------+
|                         |
|  +-------------------+  |
|  | Produktkarte      |  |  <- Styled Card
|  | mit eigenem Look  |  |
|  +-------------------+  |
|                         |
|  [ In den Warenkorb ]   |  <- Branded Button
|                         |
+-------------------------+
```

2.8.1.4 Bonusaufgabe: Theme Extension

Erstelle eine ThemeExtension für Spacing-Werte (small, medium, large).

2.8.2 Lösung

2.8.2.1 Aufgabe 1

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        useMaterial3: true,
        colorScheme: ColorScheme.fromSeed(
          seedColor: Color(0xFFFF6B00),
        ),
        elevatedButtonTheme: ElevatedButtonThemeData(
          style: ElevatedButton.styleFrom(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(20),
            ),
            padding: EdgeInsets.symmetric(horizontal: 24, vertical: 12),
          ),
        ),
        outlinedButtonTheme: OutlinedButtonThemeData(
          style: OutlinedButton.styleFrom(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(20),
            ),
          ),
        ),
        inputDecorationTheme: InputDecorationTheme(
          filled: true,
          fillColor: Colors.grey[100],
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
            borderSide: BorderSide.none,
          ),
          focusedBorder: OutlineInputBorder(
```

```
            borderRadius: BorderRadius.circular(12),
            borderSide: BorderSide(color: Color(0xFFFF6B00), width: 2),
          ),
        ),
        cardTheme: CardTheme(
          elevation: 4,
          shadowColor: Colors.black26,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(16),
          ),
        ),
      ),
      home: Scaffold(
        appBar: AppBar(title: Text('Custom Theme')),
        body: Padding(
          padding: EdgeInsets.all(16),
          child: Column(
            children: [
              Card(
                child: Padding(
                  padding: EdgeInsets.all(16),
                  child: Text('Styled Card'),
                ),
              ),
              SizedBox(height: 16),
              TextField(decoration: InputDecoration(hintText: 'TextField')),
              SizedBox(height: 16),
              ElevatedButton(onPressed: () {}, child: Text('Button')),
            ],
          ),
        ),
      ),
    );
  }
}
```

2.8.2.2 Aufgabe 2

```
class ThemeToggleApp extends StatefulWidget {
  @override
  State<ThemeToggleApp> createState() => _ThemeToggleAppState();
}

class _ThemeToggleAppState extends State<ThemeToggleApp> {
  ThemeMode _themeMode = ThemeMode.light;

  void _toggleTheme(bool isDark) {
    setState(() {
      _themeMode = isDark ? ThemeMode.dark : ThemeMode.light;
    });
```

```
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        useMaterial3: true,
        colorScheme: ColorScheme.fromSeed(
          seedColor: Colors.blue,
          brightness: Brightness.light,
        ),
      ),
      darkTheme: ThemeData(
        useMaterial3: true,
        colorScheme: ColorScheme.fromSeed(
          seedColor: Colors.blue,
          brightness: Brightness.dark,
        ),
      ),
      themeMode: _themeMode,
      home: Scaffold(
        appBar: AppBar(
          title: Text('Theme Toggle'),
          actions: [
            Icon(_themeMode == ThemeMode.dark ? Icons.dark_mode :
↪ Icons.light_mode),
            Switch(
              value: _themeMode == ThemeMode.dark,
              onChanged: _toggleTheme,
            ),
          ],
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Icon(Icons.palette, size: 80),
              SizedBox(height: 16),
              Text(
                _themeMode == ThemeMode.dark ? 'Dark Mode' : 'Light Mode',
                style: Theme.of(context).textTheme.headlineMedium,
              ),
              SizedBox(height: 24),
              ElevatedButton(
                onPressed: () {},
                child: Text('Beispiel Button'),
              ),
            ],
          ),
        ),
      ),
```

```
    );
  }
}
```

2.8.2.3 Aufgabe 3

```
class ShopTheme {
  static ThemeData get theme {
    return ThemeData(
      useMaterial3: true,
      colorScheme: ColorScheme.fromSeed(
        seedColor: Color(0xFF2E7D32),  // Grün
      ),
      appBarTheme: AppBarTheme(
        backgroundColor: Color(0xFF2E7D32),
        foregroundColor: Colors.white,
        centerTitle: false,
        titleTextStyle: TextStyle(
          fontSize: 20,
          fontWeight: FontWeight.bold,
        ),
      ),
      cardTheme: CardTheme(
        elevation: 3,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
      ),
      elevatedButtonTheme: ElevatedButtonThemeData(
        style: ElevatedButton.styleFrom(
          backgroundColor: Color(0xFF2E7D32),
          foregroundColor: Colors.white,
          padding: EdgeInsets.symmetric(horizontal: 32, vertical: 16),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(8),
          ),
          textStyle: TextStyle(
            fontSize: 16,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
    );
  }
}

class ShopPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
```

```
      theme: ShopTheme.theme,
      home: Scaffold(
        appBar: AppBar(
          leading: Icon(Icons.home),
          title: Text('Mein Shop'),
        ),
        body: Padding(
          padding: EdgeInsets.all(16),
          child: Column(
            children: [
              Card(
                child: Padding(
                  padding: EdgeInsets.all(16),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      AspectRatio(
                        aspectRatio: 16 / 9,
                        child: Container(
                          color: Colors.grey[200],
                          child: Icon(Icons.image, size: 60, color: Colors.grey),
                        ),
                      ),
                      SizedBox(height: 12),
                      Text('Produktname', style: TextStyle(
                        fontSize: 18, fontWeight: FontWeight.bold)),
                      SizedBox(height: 4),
                      Text('€ 29,99', style: TextStyle(
                        fontSize: 16, color: Colors.green[700])),
                    ],
                  ),
                ),
              ),
              SizedBox(height: 16),
              SizedBox(
                width: double.infinity,
                child: ElevatedButton.icon(
                  onPressed: () {},
                  icon: Icon(Icons.shopping_cart),
                  label: Text('In den Warenkorb'),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

2.8.2.4 Bonusaufgabe

```
class AppSpacing extends ThemeExtension<AppSpacing> {
  final double small;
  final double medium;
  final double large;

  const AppSpacing({
    required this.small,
    required this.medium,
    required this.large,
  });

  @override
  AppSpacing copyWith({double? small, double? medium, double? large}) {
    return AppSpacing(
      small: small ?? this.small,
      medium: medium ?? this.medium,
      large: large ?? this.large,
    );
  }

  @override
  AppSpacing lerp(ThemeExtension<AppSpacing>? other, double t) {
    if (other is! AppSpacing) return this;
    return AppSpacing(
      small: lerpDouble(small, other.small, t) ?? small,
      medium: lerpDouble(medium, other.medium, t) ?? medium,
      large: lerpDouble(large, other.large, t) ?? large,
    );
  }
}

// Verwendung
ThemeData(
  extensions: [
    AppSpacing(small: 8, medium: 16, large: 24),
  ],
)

// Im Widget
final spacing = Theme.of(context).extension<AppSpacing>()!;
Padding(padding: EdgeInsets.all(spacing.medium), ...)
```

2.8.3 Ressourcen

2.8.3.1 Dokumentation

- ThemeData
- ColorScheme
- Material 3 Design
- Google Fonts Package

2.8.3.2 Nächste Einheit

Einheit 2.9: Navigation Basics

2.9 Einheit 2.9: Navigation Basics

2.9.0.1 9.1 Navigator.push & pop

```
// Zu neuer Seite navigieren
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => DetailPage()),
);

// Zurück zur vorherigen Seite
Navigator.pop(context);

// Mit Kurzform
Navigator.of(context).push(
  MaterialPageRoute(builder: (context) => DetailPage()),
);
```

2.9.0.2 9.2 Daten übergeben

```
// Daten an neue Seite übergeben
class DetailPage extends StatelessWidget {
  final String titel;
  final int id;

  const DetailPage({required this.titel, required this.id});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(titel)),
      body: Center(child: Text('ID: $id')),
    );
  }
}

// Navigation mit Daten
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => DetailPage(titel: 'Produkt', id: 42),
  ),
);
```

2.9.0.3 9.3 Daten zurückgeben

```
// Seite die Daten zurückgibt
class SelectionPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Auswahl')),
      body: ListView(
        children: [
          ListTile(
            title: Text('Option A'),
            onTap: () => Navigator.pop(context, 'A'),
          ),
          ListTile(
            title: Text('Option B'),
            onTap: () => Navigator.pop(context, 'B'),
          ),
        ],
      ),
    );
  }
}

// Auf Ergebnis warten
void _selectOption() async {
  final result = await Navigator.push<String>(
    context,
    MaterialPageRoute(builder: (context) => SelectionPage()),
  );

  if (result != null) {
    print('Ausgewählt: $result');
  }
}
```

2.9.0.4 9.4 pushReplacement & pushAndRemoveUntil

```
// Aktuelle Seite ersetzen (kein Zurück möglich)
Navigator.pushReplacement(
  context,
  MaterialPageRoute(builder: (context) => HomePage()),
);

// Alle Seiten entfernen und neue hinzufügen
Navigator.pushAndRemoveUntil(
  context,
  MaterialPageRoute(builder: (context) => LoginPage()),
  (route) => false,  // Entfernt alle vorherigen Routes
);
```

```
// Bis zu bestimmter Route entfernen
Navigator.pushAndRemoveUntil(
  context,
  MaterialPageRoute(builder: (context) => HomePage()),
  ModalRoute.withName('/'),  // Behält Root-Route
);
```

2.9.0.5 9.5 WillPopScope / PopScope

```
// Zurück-Navigation abfangen (Flutter 3.16+)
PopScope(
  canPop: false,
  onPopInvokedWithResult: (didPop, result) {
    if (didPop) return;
    _showExitDialog();
  },
  child: Scaffold(...),
)

// Ältere Version (deprecated)
WillPopScope(
  onWillPop: () async {
    final shouldPop = await _showExitDialog();
    return shouldPop ?? false;
  },
  child: Scaffold(...),
)

Future<bool?> _showExitDialog() {
  return showDialog<bool>(
    context: context,
    builder: (context) => AlertDialog(
      title: Text('Beenden?'),
      content: Text('Möchtest du wirklich zurück?'),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context, false),
          child: Text('Nein'),
        ),
        TextButton(
          onPressed: () => Navigator.pop(context, true),
          child: Text('Ja'),
        ),
      ],
    ),
  );
}
```

2.9.0.6 9.6 Page Transitions

```
// Custom PageRoute
class FadeRoute<T> extends PageRouteBuilder<T> {
  final Widget page;

  FadeRoute({required this.page})
      : super(
          pageBuilder: (context, animation, secondaryAnimation) => page,
          transitionsBuilder: (context, animation, secondaryAnimation, child) {
            return FadeTransition(opacity: animation, child: child);
          },
          transitionDuration: Duration(milliseconds: 300),
        );
}

// Verwendung
Navigator.push(context, FadeRoute(page: DetailPage()));

// Slide Transition
class SlideRoute<T> extends PageRouteBuilder<T> {
  final Widget page;

  SlideRoute({required this.page})
      : super(
          pageBuilder: (context, animation, secondaryAnimation) => page,
          transitionsBuilder: (context, animation, secondaryAnimation, child) {
            final tween = Tween(
              begin: Offset(1.0, 0.0),
              end: Offset.zero,
            ).chain(CurveTween(curve: Curves.easeInOut));

            return SlideTransition(
              position: animation.drive(tween),
              child: child,
            );
          },
        );
}
```

2.9.0.7 9.7 BottomNavigationBar

```
class MainScreen extends StatefulWidget {
  @override
  State<MainScreen> createState() => _MainScreenState();
}

class _MainScreenState extends State<MainScreen> {
  int _currentIndex = 0;

  final _pages = [
```

```
    HomePage(),
    SearchPage(),
    ProfilePage(),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: _pages[_currentIndex],
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _currentIndex,
        onTap: (index) => setState(() => _currentIndex = index),
        items: [
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.search),
            label: 'Suche',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Profil',
          ),
        ],
      ),
    );
  }
}
```

2.9.0.8 9.8 Drawer Navigation

```
Scaffold(
  appBar: AppBar(title: Text('App')),
  drawer: Drawer(
    child: ListView(
      padding: EdgeInsets.zero,
      children: [
        DrawerHeader(
          decoration: BoxDecoration(color: Colors.blue),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            mainAxisAlignment: MainAxisAlignment.end,
            children: [
              CircleAvatar(radius: 30, child: Icon(Icons.person)),
              SizedBox(height: 8),
              Text('Benutzer Name', style: TextStyle(
                color: Colors.white, fontSize: 18)),
            ],
```

```
          ),
        ),
        ListTile(
          leading: Icon(Icons.home),
          title: Text('Home'),
          onTap: () {
            Navigator.pop(context);  // Drawer schließen
            // Navigation...
          },
        ),
        ListTile(
          leading: Icon(Icons.settings),
          title: Text('Einstellungen'),
          onTap: () {
            Navigator.pop(context);
            Navigator.push(context,
              MaterialPageRoute(builder: (_) => SettingsPage()));
          },
        ),
        Divider(),
        ListTile(
          leading: Icon(Icons.logout),
          title: Text('Abmelden'),
          onTap: () => _logout(),
        ),
      ],
    ),
  ),
  body: HomePage(),
)
```

2.9.0.9 9.9 TabBar Navigation

```
class TabExample extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return DefaultTabController(
      length: 3,
      child: Scaffold(
        appBar: AppBar(
          title: Text('Tabs'),
          bottom: TabBar(
            tabs: [
              Tab(icon: Icon(Icons.home), text: 'Home'),
              Tab(icon: Icon(Icons.star), text: 'Favoriten'),
              Tab(icon: Icon(Icons.person), text: 'Profil'),
            ],
          ),
        ),
        body: TabBarView(
```

```
          children: [
            Center(child: Text('Home Tab')),
            Center(child: Text('Favoriten Tab')),
            Center(child: Text('Profil Tab')),
          ],
        ),
      ),
    );
  }
}
```

2.9.1 Übung

2.9.1.1 Aufgabe 1: Master-Detail Navigation (20 Min.)

Erstelle eine Produktliste mit Detail-Ansicht:

```
// Liste mit Produkten
// Tap öffnet Detail-Seite
// Detail zeigt alle Produkt-Infos
// Zurück-Button in AppBar
```

2.9.1.2 Aufgabe 2: Formular mit Bestätigung (20 Min.)

Erstelle ein Formular das bei Zurück warnt:

```
// Eingabeformular
// Bei Zurück: "Änderungen verwerfen?" Dialog
// Speichern-Button gibt Daten zurück
```

2.9.1.3 Aufgabe 3: Tab-basierte App (15 Min.)

Erstelle eine App mit 3 Tabs:

```
+-------------------------+
|     Tab Navigation      |
+--------+--------+-------+
|  Feed  | Suche  | Profil|
+--------+--------+-------+
|                         |
|    [Tab Content]        |
|                         |
+-------------------------+
```

2.9.1.4 Bonusaufgabe: Nested Navigation

Erstelle eine App wo jeder Tab seinen eigenen Navigator hat.

2.9.2 Lösung

2.9.2.1 Aufgabe 1

```
class Product {
  final int id;
  final String name;
  final String description;
  final double price;

  Product({
    required this.id,
    required this.name,
    required this.description,
    required this.price,
  });
}

class ProductListPage extends StatelessWidget {
  final products = [
    Product(id: 1, name: 'Laptop', description: 'Schneller Laptop', price:
↪  999.99),
    Product(id: 2, name: 'Maus', description: 'Wireless Maus', price: 49.99),
    Product(id: 3, name: 'Tastatur', description: 'Mechanisch', price: 129.99),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Produkte')),
      body: ListView.builder(
        itemCount: products.length,
        itemBuilder: (context, index) {
          final product = products[index];
          return ListTile(
            leading: CircleAvatar(child: Text('${product.id}')),
            title: Text(product.name),
            subtitle: Text('€${product.price.toStringAsFixed(2)}'),
            trailing: Icon(Icons.arrow_forward_ios),
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => ProductDetailPage(product: product),
                ),
              );
            },
          );
        },
      ),
    );
  }
```

```
}

class ProductDetailPage extends StatelessWidget {
  final Product product;

  const ProductDetailPage({required this.product});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(product.name)),
      body: Padding(
        padding: EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Container(
              height: 200,
              width: double.infinity,
              color: Colors.grey[200],
              child: Icon(Icons.image, size: 80, color: Colors.grey),
            ),
            SizedBox(height: 16),
            Text(product.name, style: TextStyle(
              fontSize: 24, fontWeight: FontWeight.bold)),
            SizedBox(height: 8),
            Text('€${product.price.toStringAsFixed(2)}',
              style: TextStyle(fontSize: 20, color: Colors.green)),
            SizedBox(height: 16),
            Text(product.description, style: TextStyle(fontSize: 16)),
            Spacer(),
            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: () {},
                child: Text('In den Warenkorb'),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

2.9.2.2 Aufgabe 2

```
class EditFormPage extends StatefulWidget {
  @override
  State<EditFormPage> createState() => _EditFormPageState();
```

```
}

class _EditFormPageState extends State<EditFormPage> {
  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
  bool _hasChanges = false;

  @override
  void initState() {
    super.initState();
    _nameController.addListener(_onChanged);
    _emailController.addListener(_onChanged);
  }

  void _onChanged() {
    setState(() => _hasChanges = true);
  }

  Future<bool> _confirmDiscard() async {
    if (!_hasChanges) return true;

    final result = await showDialog<bool>(
      context: context,
      builder: (context) => AlertDialog(
        title: Text('Änderungen verwerfen?'),
        content: Text('Nicht gespeicherte Änderungen gehen verloren.'),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context, false),
            child: Text('Abbrechen'),
          ),
          TextButton(
            onPressed: () => Navigator.pop(context, true),
            child: Text('Verwerfen'),
          ),
        ],
      ),
    );
    return result ?? false;
  }

  void _save() {
    final data = {
      'name': _nameController.text,
      'email': _emailController.text,
    };
    Navigator.pop(context, data);
  }

  @override
  Widget build(BuildContext context) {
```

```
    return PopScope(
      canPop: !_hasChanges,
      onPopInvokedWithResult: (didPop, result) async {
        if (didPop) return;
        final shouldPop = await _confirmDiscard();
        if (shouldPop && context.mounted) {
          Navigator.pop(context);
        }
      },
      child: Scaffold(
        appBar: AppBar(
          title: Text('Bearbeiten'),
          actions: [
            TextButton(
              onPressed: _save,
              child: Text('Speichern'),
            ),
          ],
        ),
        body: Padding(
          padding: EdgeInsets.all(16),
          child: Column(
            children: [
              TextField(
                controller: _nameController,
                decoration: InputDecoration(labelText: 'Name'),
              ),
              SizedBox(height: 16),
              TextField(
                controller: _emailController,
                decoration: InputDecoration(labelText: 'Email'),
              ),
            ],
          ),
        ),
      ),
    );
  }

  @override
  void dispose() {
    _nameController.dispose();
    _emailController.dispose();
    super.dispose();
  }
}
```

2.9.2.3 Aufgabe 3

```
class TabApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: DefaultTabController(
        length: 3,
        child: Scaffold(
          appBar: AppBar(
            title: Text('Tab Navigation'),
            bottom: TabBar(
              tabs: [
                Tab(icon: Icon(Icons.feed), text: 'Feed'),
                Tab(icon: Icon(Icons.search), text: 'Suche'),
                Tab(icon: Icon(Icons.person), text: 'Profil'),
              ],
            ),
          ),
          body: TabBarView(
            children: [
              FeedTab(),
              SearchTab(),
              ProfileTab(),
            ],
          ),
        ),
      ),
    );
  }
}

class FeedTab extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: 20,
      itemBuilder: (context, index) {
        return Card(
          margin: EdgeInsets.all(8),
          child: ListTile(
            leading: CircleAvatar(child: Text('${index + 1}')),
            title: Text('Post $index'),
            subtitle: Text('Beschreibung des Posts'),
          ),
        );
      },
    );
  }
}

class SearchTab extends StatelessWidget {
  @override
```

```
  Widget build(BuildContext context) {
    return Padding(
      padding: EdgeInsets.all(16),
      child: Column(
        children: [
          TextField(
            decoration: InputDecoration(
              hintText: 'Suchen...',
              prefixIcon: Icon(Icons.search),
              border: OutlineInputBorder(),
            ),
          ),
          SizedBox(height: 16),
          Expanded(
            child: Center(child: Text('Suchergebnisse erscheinen hier')),
          ),
        ],
      ),
    );
  }
}

class ProfileTab extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          CircleAvatar(radius: 50, child: Icon(Icons.person, size: 50)),
          SizedBox(height: 16),
          Text('Benutzername', style: TextStyle(fontSize: 24)),
          SizedBox(height: 8),
          Text('user@example.com'),
          SizedBox(height: 24),
          ElevatedButton(
            onPressed: () {},
            child: Text('Profil bearbeiten'),
          ),
        ],
      ),
    );
  }
}
```

2.9.3 Ressourcen

2.9.3.1 Dokumentation

- Navigation and routing
- Navigator

- BottomNavigationBar
- TabBar

2.9.3.2 Nächste Einheit

Einheit 2.10: Named Routes & go_router

2.10 Einheit 2.10: Named Routes & go_router

2.10.0.1 10.1 Named Routes (Flutter Standard)

```
MaterialApp(
  initialRoute: '/',
  routes: {
    '/': (context) => HomePage(),
    '/login': (context) => LoginPage(),
    '/settings': (context) => SettingsPage(),
    '/profile': (context) => ProfilePage(),
  },
)

// Navigation
Navigator.pushNamed(context, '/settings');

// Mit Argumenten
Navigator.pushNamed(
  context,
  '/profile',
  arguments: {'userId': 42},
);

// Argumente empfangen
class ProfilePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final args = ModalRoute.of(context)!.settings.arguments as Map;
    final userId = args['userId'];
    return Scaffold(...);
  }
}
```

2.10.0.2 10.2 onGenerateRoute

```
MaterialApp(
  initialRoute: '/',
  onGenerateRoute: (settings) {
    // Dynamisches Routing
    if (settings.name == '/') {
      return MaterialPageRoute(builder: (_) => HomePage());
    }
```

```
    // Route mit Parameter: /user/123
    final uri = Uri.parse(settings.name!);
    if (uri.pathSegments.length == 2 && uri.pathSegments[0] == 'user') {
      final userId = int.parse(uri.pathSegments[1]);
      return MaterialPageRoute(
        builder: (_) => UserPage(userId: userId),
      );
    }

    // 404 Seite
    return MaterialPageRoute(builder: (_) => NotFoundPage());
  },
)

// Navigation
Navigator.pushNamed(context, '/user/42');
```

2.10.0.3 10.3 go_router Setup

```
# pubspec.yaml
dependencies:
  go_router: ^13.0.0
```

```
import 'package:go_router/go_router.dart';

final router = GoRouter(
  initialLocation: '/',
  routes: [
    GoRoute(
      path: '/',
      builder: (context, state) => HomePage(),
    ),
    GoRoute(
      path: '/login',
      builder: (context, state) => LoginPage(),
    ),
    GoRoute(
      path: '/settings',
      builder: (context, state) => SettingsPage(),
    ),
  ],
);

// App
MaterialApp.router(
  routerConfig: router,
)
```

2.10.0.4 10.4 go_router Navigation

```
// Navigation
context.go('/settings');        // Ersetzt Stack
context.push('/settings');      // Fügt hinzu
context.pop();                  // Zurück
context.replace('/home');       // Ersetzt aktuelle Route

// Mit GoRouter-Instanz
GoRouter.of(context).go('/settings');
```

2.10.0.5 10.5 Path Parameters

```
GoRoute(
  path: '/user/:id',
  builder: (context, state) {
    final userId = state.pathParameters['id']!;
    return UserPage(userId: int.parse(userId));
  },
),

GoRoute(
  path: '/product/:category/:id',
  builder: (context, state) {
    final category = state.pathParameters['category']!;
    final id = state.pathParameters['id']!;
    return ProductPage(category: category, id: id);
  },
),

// Navigation
context.go('/user/42');
context.go('/product/electronics/123');
```

2.10.0.6 10.6 Query Parameters

```
GoRoute(
  path: '/search',
  builder: (context, state) {
    final query = state.uri.queryParameters['q'] ?? '';
    final page = int.tryParse(state.uri.queryParameters['page'] ?? '1') ?? 1;
    return SearchPage(query: query, page: page);
  },
),

// Navigation
context.go('/search?q=flutter&page=2');

// Mit Uri
context.go(Uri(
```

```
  path: '/search',
  queryParameters: {'q': 'flutter', 'page': '2'},
).toString());
```

2.10.0.7 10.7 Nested Routes (ShellRoute)

```
final router = GoRouter(
  initialLocation: '/',
  routes: [
    ShellRoute(
      builder: (context, state, child) {
        return MainShell(child: child);
      },
      routes: [
        GoRoute(
          path: '/',
          builder: (context, state) => HomePage(),
        ),
        GoRoute(
          path: '/search',
          builder: (context, state) => SearchPage(),
        ),
        GoRoute(
          path: '/profile',
          builder: (context, state) => ProfilePage(),
        ),
      ],
    ),
    GoRoute(
      path: '/login',
      builder: (context, state) => LoginPage(),
    ),
  ],
);

class MainShell extends StatelessWidget {
  final Widget child;
  const MainShell({required this.child});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: child,
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _calculateIndex(GoRouterState.of(context).uri.path),
        onTap: (index) {
          switch (index) {
            case 0: context.go('/');
            case 1: context.go('/search');
            case 2: context.go('/profile');
```

```
          }
        },
        items: [
          BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Home'),
          BottomNavigationBarItem(icon: Icon(Icons.search), label: 'Suche'),
          BottomNavigationBarItem(icon: Icon(Icons.person), label: 'Profil'),
        ],
      ),
    );
  }

  int _calculateIndex(String path) {
    if (path.startsWith('/search')) return 1;
    if (path.startsWith('/profile')) return 2;
    return 0;
  }
}
```

2.10.0.8 10.8 Redirect & Guards

```
final router = GoRouter(
  initialLocation: '/',
  redirect: (context, state) {
    final isLoggedIn = AuthService.isLoggedIn;
    final isLoginRoute = state.matchedLocation == '/login';

    // Nicht eingeloggt und nicht auf Login-Seite
    if (!isLoggedIn && !isLoginRoute) {
      return '/login';
    }

    // Eingeloggt aber auf Login-Seite
    if (isLoggedIn && isLoginRoute) {
      return '/';
    }

    return null;  // Keine Umleitung
  },
  routes: [...],
);

// Pro Route
GoRoute(
  path: '/admin',
  redirect: (context, state) {
    if (!AuthService.isAdmin) {
      return '/unauthorized';
    }
    return null;
  },
```

```
  builder: (context, state) => AdminPage(),
),
```

2.10.0.9 10.9 Extra Data

```
// Komplexe Objekte übergeben
class Product {
  final int id;
  final String name;
  Product({required this.id, required this.name});
}

GoRoute(
  path: '/product/:id',
  builder: (context, state) {
    final product = state.extra as Product?;
    if (product != null) {
      return ProductPage(product: product);
    }
    // Fallback: ID aus URL laden
    final id = state.pathParameters['id']!;
    return ProductPage(productId: int.parse(id));
  },
),

// Navigation mit extra
context.go(
  '/product/42',
  extra: Product(id: 42, name: 'Laptop'),
);
```

2.10.0.10 10.10 Beispiel: Komplette App-Navigation

```
final router = GoRouter(
  initialLocation: '/',
  debugLogDiagnostics: true,
  redirect: (context, state) {
    // Auth-Check hier
    return null;
  },
  routes: [
    ShellRoute(
      builder: (context, state, child) => AppShell(child: child),
      routes: [
        GoRoute(
          path: '/',
          builder: (context, state) => HomePage(),
          routes: [
            GoRoute(
```

```
              path: 'product/:id',
              builder: (context, state) {
                final id = state.pathParameters['id']!;
                return ProductDetailPage(id: id);
              },
            ),
          ],
        ),
        GoRoute(
          path: '/cart',
          builder: (context, state) => CartPage(),
        ),
        GoRoute(
          path: '/profile',
          builder: (context, state) => ProfilePage(),
          routes: [
            GoRoute(
              path: 'settings',
              builder: (context, state) => SettingsPage(),
            ),
            GoRoute(
              path: 'orders',
              builder: (context, state) => OrdersPage(),
            ),
          ],
        ),
      ],
    ),
    GoRoute(
      path: '/login',
      builder: (context, state) => LoginPage(),
    ),
    GoRoute(
      path: '/onboarding',
      builder: (context, state) => OnboardingPage(),
    ),
  ],
  errorBuilder: (context, state) => ErrorPage(error: state.error),
);
```

2.10.1 Übung

2.10.1.1 Aufgabe 1: Named Routes Setup (15 Min.)

Erstelle eine App mit Named Routes:

```
// Routes:
// '/' -> HomePage
// '/about' -> AboutPage
// '/contact' -> ContactPage
```

```
// Navigation zwischen den Seiten
```

2.10.1.2 Aufgabe 2: go_router mit Parametern (25 Min.)

Erstelle einen Blog mit go_router:

```
// Routes:
// '/' -> Liste aller Posts
// '/post/:id' -> Einzelner Post
// '/category/:name' -> Posts einer Kategorie
// '/search?q=...' -> Suche mit Query
```

2.10.1.3 Aufgabe 3: ShellRoute mit Navigation (15 Min.)

Erstelle eine App mit BottomNavigationBar:

```
+-------------------------+
|                         |
|     [Page Content]      |
|                         |
+-------------------------+
|  Home | Shop | Account  |
+-------------------------+

// Jeder Tab hat eigene Sub-Routes
// Home: /
// Shop: /shop, /shop/product/:id
// Account: /account, /account/settings
```

2.10.1.4 Bonusaufgabe: Auth Guard

Implementiere einen Login-Redirect: - Geschützte Routen leiten zu /login um - Nach Login zurück zur ursprünglichen Route

2.10.2 Lösung

2.10.2.1 Aufgabe 1

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => HomePage(),
        '/about': (context) => AboutPage(),
        '/contact': (context) => ContactPage(),
      },
    );
  }
}
```

```
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Home')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text('Willkommen!', style: TextStyle(fontSize: 24)),
            SizedBox(height: 24),
            ElevatedButton(
              onPressed: () => Navigator.pushNamed(context, '/about'),
              child: Text('Über uns'),
            ),
            SizedBox(height: 8),
            ElevatedButton(
              onPressed: () => Navigator.pushNamed(context, '/contact'),
              child: Text('Kontakt'),
            ),
          ],
        ),
      ),
    );
  }
}

class AboutPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Über uns')),
      body: Center(child: Text('Über uns Seite')),
    );
  }
}

class ContactPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Kontakt')),
      body: Center(child: Text('Kontakt Seite')),
    );
  }
}
```

2.10.2.2 Aufgabe 2

```
import 'package:go_router/go_router.dart';

class Post {
  final int id;
  final String title;
  final String category;
  final String content;

  Post({
    required this.id,
    required this.title,
    required this.category,
    required this.content,
  });
}

final posts = [
  Post(id: 1, title: 'Flutter Basics', category: 'flutter', content: '...'),
  Post(id: 2, title: 'Dart Tips', category: 'dart', content: '...'),
  Post(id: 3, title: 'State Management', category: 'flutter', content: '...'),
];

final router = GoRouter(
  initialLocation: '/',
  routes: [
    GoRoute(
      path: '/',
      builder: (context, state) => PostListPage(),
    ),
    GoRoute(
      path: '/post/:id',
      builder: (context, state) {
        final id = int.parse(state.pathParameters['id']!);
        return PostDetailPage(postId: id);
      },
    ),
    GoRoute(
      path: '/category/:name',
      builder: (context, state) {
        final name = state.pathParameters['name']!;
        return CategoryPage(category: name);
      },
    ),
    GoRoute(
      path: '/search',
      builder: (context, state) {
        final query = state.uri.queryParameters['q'] ?? '';
        return SearchPage(query: query);
      },
    ),
  ],
```

```
);

class BlogApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp.router(routerConfig: router);
  }
}

class PostListPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Blog'),
        actions: [
          IconButton(
            icon: Icon(Icons.search),
            onPressed: () => context.go('/search?q='),
          ),
        ],
      ),
      body: ListView.builder(
        itemCount: posts.length,
        itemBuilder: (context, index) {
          final post = posts[index];
          return ListTile(
            title: Text(post.title),
            subtitle: GestureDetector(
              onTap: () => context.go('/category/${post.category}'),
              child: Text(post.category, style: TextStyle(color: Colors.blue)),
            ),
            onTap: () => context.go('/post/${post.id}'),
          );
        },
      ),
    );
  }
}

class PostDetailPage extends StatelessWidget {
  final int postId;
  const PostDetailPage({required this.postId});

  @override
  Widget build(BuildContext context) {
    final post = posts.firstWhere((p) => p.id == postId);
    return Scaffold(
      appBar: AppBar(title: Text(post.title)),
      body: Padding(
        padding: EdgeInsets.all(16),
```

```
        child: Text(post.content),
      ),
    );
  }
}

class CategoryPage extends StatelessWidget {
  final String category;
  const CategoryPage({required this.category});

  @override
  Widget build(BuildContext context) {
    final categoryPosts = posts.where((p) => p.category == category).toList();
    return Scaffold(
      appBar: AppBar(title: Text('Kategorie: $category')),
      body: ListView.builder(
        itemCount: categoryPosts.length,
        itemBuilder: (context, index) {
          final post = categoryPosts[index];
          return ListTile(
            title: Text(post.title),
            onTap: () => context.go('/post/${post.id}'),
          );
        },
      ),
    );
  }
}

class SearchPage extends StatefulWidget {
  final String query;
  const SearchPage({required this.query});

  @override
  State<SearchPage> createState() => _SearchPageState();
}

class _SearchPageState extends State<SearchPage> {
  late TextEditingController _controller;

  @override
  void initState() {
    super.initState();
    _controller = TextEditingController(text: widget.query);
  }

  @override
  Widget build(BuildContext context) {
    final results = posts.where(
      (p) => p.title.toLowerCase().contains(widget.query.toLowerCase())
    ).toList();
```

```
    return Scaffold(
      appBar: AppBar(title: Text('Suche')),
      body: Column(
        children: [
          Padding(
            padding: EdgeInsets.all(8),
            child: TextField(
              controller: _controller,
              decoration: InputDecoration(
                hintText: 'Suchen...',
                suffixIcon: IconButton(
                  icon: Icon(Icons.search),
                  onPressed: () => context.go('/search?q=${_controller.text}'),
                ),
              ),
              onSubmitted: (q) => context.go('/search?q=$q'),
            ),
          ),
          Expanded(
            child: ListView.builder(
              itemCount: results.length,
              itemBuilder: (context, index) {
                final post = results[index];
                return ListTile(
                  title: Text(post.title),
                  onTap: () => context.go('/post/${post.id}'),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

2.10.2.3 Aufgabe 3

```
import 'package:go_router/go_router.dart';

final router = GoRouter(
  initialLocation: '/',
  routes: [
    ShellRoute(
      builder: (context, state, child) => AppShell(child: child),
      routes: [
        GoRoute(
          path: '/',
          builder: (context, state) => HomePage(),
```

```
        ),
        GoRoute(
          path: '/shop',
          builder: (context, state) => ShopPage(),
          routes: [
            GoRoute(
              path: 'product/:id',
              builder: (context, state) {
                final id = state.pathParameters['id']!;
                return ProductPage(id: id);
              },
            ),
          ],
        ),
        GoRoute(
          path: '/account',
          builder: (context, state) => AccountPage(),
          routes: [
            GoRoute(
              path: 'settings',
              builder: (context, state) => SettingsPage(),
            ),
          ],
        ),
      ],
    ),
  ],
);

class AppShell extends StatelessWidget {
  final Widget child;
  const AppShell({required this.child});

  @override
  Widget build(BuildContext context) {
    final location = GoRouterState.of(context).uri.path;

    return Scaffold(
      body: child,
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _getIndex(location),
        onTap: (index) {
          switch (index) {
            case 0: context.go('/');
            case 1: context.go('/shop');
            case 2: context.go('/account');
          }
        },
        items: [
          BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Home'),
          BottomNavigationBarItem(icon: Icon(Icons.shop), label: 'Shop'),
```

```
          BottomNavigationBarItem(icon: Icon(Icons.person), label: 'Account'),
        ],
      ),
    );
  }

  int _getIndex(String location) {
    if (location.startsWith('/shop')) return 1;
    if (location.startsWith('/account')) return 2;
    return 0;
  }
}

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Home')),
      body: Center(child: Text('Home Page')),
    );
  }
}

class ShopPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Shop')),
      body: ListView.builder(
        itemCount: 10,
        itemBuilder: (context, index) {
          return ListTile(
            title: Text('Produkt $index'),
            onTap: () => context.go('/shop/product/$index'),
          );
        },
      ),
    );
  }
}

class ProductPage extends StatelessWidget {
  final String id;
  const ProductPage({required this.id});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Produkt $id')),
      body: Center(child: Text('Produkt Details für ID: $id')),
    );
```

```
  }
}

class AccountPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Account')),
      body: ListView(
        children: [
          ListTile(
            leading: Icon(Icons.settings),
            title: Text('Einstellungen'),
            onTap: () => context.go('/account/settings'),
          ),
        ],
      ),
    );
  }
}

class SettingsPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Einstellungen')),
      body: Center(child: Text('Einstellungen')),
    );
  }
}
```

2.10.3 Ressourcen

2.10.3.1 Dokumentation

- Named Routes
- go_router Package
- go_router Dokumentation

2.10.3.2 Nächste Einheit

Block 3: State Management - Einheit 3.1: setState & Lifting State Up

Chapter 3

Block 3: Flutter – Fortgeschritten

State Management, HTTP, lokale Datenspeicherung und Formulare.

3.1 Einheit 3.1: State Management Konzepte

3.1.0.1 Lernziele

Nach dieser Einheit kannst du: - Erklären, warum State Management in Flutter wichtig ist - Die Grenzen von `setState()` verstehen - Das "Lifting State Up"-Pattern anwenden - `InheritedWidget` konzeptionell verstehen - Verschiedene State Management Lösungen einordnen

3.1.0.2 1. Was ist "State" in Flutter?

State ist jede Information, die sich während der Laufzeit einer App ändern kann und die UI beeinflusst:

```
// Beispiele für State:
int counter = 0;              // UI-State: Zählerstand
bool isLoggedIn = false;      // App-State: Authentifizierung
List<Todo> todos = [];        // Daten-State: Liste von Items
String searchQuery = '';      // Formular-State: Eingaben
ThemeMode themeMode;          // Einstellungen: Dark/Light Mode
```

Arten von State

Art	Beschreibung	Beispiel
Ephemeral State	Lokal, kurzlebig, nur in einem Widget	Animation, Scroll-Position, Tab-Index
App State	Global, langlebig, widget-übergreifend	User-Session, Warenkorb, Einstellungen

Faustregel: Wenn mehrere Widgets denselben State brauchen, ist es App State.

3.1.0.3 2. Das Problem mit `setState()`

`setState()` funktioniert gut für einfachen, lokalen State:

```
class CounterWidget extends StatefulWidget {
  @override
  State<CounterWidget> createState() => _CounterWidgetState();
}

class _CounterWidgetState extends State<CounterWidget> {
  int _count = 0;
```

```
  void _increment() {
    setState(() {
      _count++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Text('Count: $_count'),
        ElevatedButton(
          onPressed: _increment,
          child: Text('Increment'),
        ),
      ],
    );
  }
}
```

Probleme bei wachsender Komplexität

```
        App
         |
    +----+----+
    |         |
  Home     Settings
    |
 +--+--+
 |     |
List  Detail
 |
Item <- Braucht State von App!
```

Problem: Wenn `Item` einen State braucht, der in `App` definiert ist, müsste der State durch `Home`, `List` und `Item` durchgereicht werden.

```
// □ "Prop Drilling" - State durch alle Ebenen durchreichen
class App extends StatefulWidget {
  @override
  State<App> createState() => _AppState();
}

class _AppState extends State<App> {
  List<Todo> todos = [];

  void addTodo(Todo todo) {
    setState(() => todos.add(todo));
  }

  @override
```

```
  Widget build(BuildContext context) {
    return Home(
      todos: todos,           // Durchreichen
      onAddTodo: addTodo,     // Durchreichen
    );
  }
}

class Home extends StatelessWidget {
  final List<Todo> todos;
  final void Function(Todo) onAddTodo;

  Home({required this.todos, required this.onAddTodo});

  @override
  Widget build(BuildContext context) {
    return TodoList(
      todos: todos,           // Wieder durchreichen
      onAddTodo: onAddTodo,   // Wieder durchreichen
    );
  }
}

// ... und so weiter durch jede Ebene
```

Nachteile: 1. **Boilerplate:** Jedes Widget dazwischen braucht die Parameter 2. **Wartbarkeit:** Änderungen erfordern Anpassungen in vielen Dateien 3. **Performance:** Alle Widgets werden neu gebaut, auch wenn sie den State nicht nutzen

3.1.0.4 3. Lifting State Up

Das einfachste Pattern für geteilten State: Den State zum nächsten gemeinsamen Vorfahren "hochheben".

```
// Vorher: Jedes Widget hat eigenen State
class WidgetA extends StatefulWidget { /* eigener counter */ }
class WidgetB extends StatefulWidget { /* eigener counter */ }

// Nachher: Gemeinsamer State im Parent
class ParentWidget extends StatefulWidget {
  @override
  State<ParentWidget> createState() => _ParentWidgetState();
}

class _ParentWidgetState extends State<ParentWidget> {
  int _counter = 0;

  void _increment() {
    setState(() => _counter++);
  }

  @override
```

```
  Widget build(BuildContext context) {
    return Column(
      children: [
        // State und Callback werden als Props übergeben
        DisplayWidget(counter: _counter),
        ButtonWidget(onIncrement: _increment),
      ],
    );
  }
}

class DisplayWidget extends StatelessWidget {
  final int counter;

  DisplayWidget({required this.counter});

  @override
  Widget build(BuildContext context) {
    return Text('Count: $counter');
  }
}

class ButtonWidget extends StatelessWidget {
  final VoidCallback onIncrement;

  ButtonWidget({required this.onIncrement});

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: onIncrement,
      child: Text('Increment'),
    );
  }
}
```

Wann Lifting State Up verwenden: - Wenige Ebenen zwischen State-Owner und State-Consumer - Einfache State-Strukturen - Kleine Widget-Bäume

3.1.0.5 4. InheritedWidget: Flutters eingebauter Mechanismus

`InheritedWidget` ermöglicht es, Daten effizient im Widget-Tree bereitzustellen, ohne sie durch jeden Konstruktor zu reichen.

Das Konzept

```
        InheritedWidget (State lebt hier)
              |
         +----+----+
         |         |
       Child     Child
         |
```

```
       Child
         |
    Nachfahre <- Direkter Zugriff auf State!
```

Einfaches Beispiel

```
// 1. InheritedWidget definieren
class CounterInherited extends InheritedWidget {
  final int count;
  final void Function() increment;

  CounterInherited({
    required this.count,
    required this.increment,
    required Widget child,
  }) : super(child: child);

  // Statische Methode für einfachen Zugriff
  static CounterInherited of(BuildContext context) {
    return context.dependOnInheritedWidgetOfExactType<CounterInherited>()!;
  }

  @override
  bool updateShouldNotify(CounterInherited oldWidget) {
    return count != oldWidget.count;
  }
}

// 2. StatefulWidget als "Owner" des State
class CounterProvider extends StatefulWidget {
  final Widget child;

  CounterProvider({required this.child});

  @override
  State<CounterProvider> createState() => _CounterProviderState();
}

class _CounterProviderState extends State<CounterProvider> {
  int _count = 0;

  void _increment() {
    setState(() => _count++);
  }

  @override
  Widget build(BuildContext context) {
    return CounterInherited(
      count: _count,
      increment: _increment,
      child: widget.child,
    );
```

```
  }
}

// 3. Verwendung in der App
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return CounterProvider(
      child: MaterialApp(
        home: CounterPage(),
      ),
    );
  }
}

// 4. Zugriff von überall im Tree
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Direkter Zugriff - kein Prop Drilling!
    final counter = CounterInherited.of(context);

    return Scaffold(
      body: Center(
        child: Text('Count: ${counter.count}'),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: counter.increment,
        child: Icon(Icons.add),
      ),
    );
  }
}
```

Wie `context.dependOnInheritedWidgetOfExactType` funktioniert

Flutter durchsucht den Widget-Tree nach oben, bis es das passende `InheritedWidget` findet. Zusätzlich registriert sich das Widget als “Dependent” - wenn sich der State ändert, wird nur dieses Widget neu gebaut.

```
// Kurzer Zugriff über dependOnInheritedWidgetOfExactType
final counter = context.dependOnInheritedWidgetOfExactType<CounterInherited>()!;

// Dasselbe mit der statischen Hilfsmethode
final counter = CounterInherited.of(context);
```

3.1.0.6 5. Warum externe State Management Lösungen?

`InheritedWidget` ist mächtig, aber erfordert viel Boilerplate. Packages wie Provider, Riverpod oder Bloc bauen darauf auf und vereinfachen die Verwendung.

Vergleich der Ansätze

Lösung	Komplexität	Lernkurve	Best Use Case
`setState()`	Niedrig	Flach	Lokaler UI-State
Lifting State Up	Niedrig	Flach	Wenige Widgets teilen State
`InheritedWidget`	Mittel	Mittel	Verstehen, wie Flutter intern funktioniert
Provider	Niedrig	Flach	Empfohlen für die meisten Apps
Riverpod	Mittel	Mittel	Compile-Time Safety, testbar
Bloc	Hoch	Steil	Große Teams, komplexe Business Logic
GetX	Niedrig	Flach	Schnelle Prototypen (weniger empfohlen)

Provider: Der Quasi-Standard

```
// Mit Provider - viel weniger Boilerplate
class Counter extends ChangeNotifier {
  int _count = 0;
  int get count => _count;

  void increment() {
    _count++;
    notifyListeners();  // Benachrichtigt alle Listener
  }
}

// In der App
ChangeNotifierProvider(
  create: (_) => Counter(),
  child: MyApp(),
);

// Zugriff
final counter = context.watch<Counter>();
Text('Count: ${counter.count}');
```

3.1.0.7 6. State Management Entscheidungsbaum

```
Braucht nur ein Widget den State?
+-- Ja -> setState()
+-- Nein -> Brauchen mehrere Widgets den State?
    +-- Ja, aber nur 1-2 Ebenen entfernt -> Lifting State Up
    +-- Ja, weit verteilt -> State Management Solution
        +-- Einsteiger / kleine App -> Provider
        +-- Testbarkeit wichtig -> Riverpod
        +-- Große App / Team -> Bloc
```

3.1.0.8 7. Praktisches Beispiel: Todo-App State

```
// Model
class Todo {
  final String id;
  final String title;
  final bool completed;

  Todo({
    required this.id,
    required this.title,
    this.completed = false,
  });

  Todo copyWith({String? title, bool? completed}) {
    return Todo(
      id: id,
      title: title ?? this.title,
      completed: completed ?? this.completed,
    );
  }
}

// State Management mit ChangeNotifier (für Provider)
class TodosNotifier extends ChangeNotifier {
  final List<Todo> _todos = [];

  List<Todo> get todos => List.unmodifiable(_todos);

  List<Todo> get completedTodos =>
      _todos.where((t) => t.completed).toList();

  List<Todo> get pendingTodos =>
      _todos.where((t) => !t.completed).toList();

  void add(String title) {
    _todos.add(Todo(
      id: DateTime.now().toString(),
      title: title,
    ));
    notifyListeners();
  }

  void toggle(String id) {
    final index = _todos.indexWhere((t) => t.id == id);
    if (index != -1) {
      _todos[index] = _todos[index].copyWith(
        completed: !_todos[index].completed,
      );
      notifyListeners();
    }
```

```
  }

  void remove(String id) {
    _todos.removeWhere((t) => t.id == id);
    notifyListeners();
  }
}
```

3.1.0.9 Zusammenfassung

Konzept	Wann verwenden
`setState()`	Lokaler State in einem Widget
Lifting State Up	State wird von wenigen, nahen Widgets geteilt
`InheritedWidget`	Grundverständnis, eigene Lösungen bauen
Provider/Riverpod	Empfohlen für App State

Wichtig: Es gibt keine "beste" Lösung - wähle basierend auf: - Größe und Komplexität der App - Teamgröße und -erfahrung - Testbarkeit-Anforderungen

In den nächsten Einheiten lernst du Provider im Detail kennen.

3.1.1 Übung

3.1.1.1 Ziel

Eine Einkaufslisten-App implementieren, die State Management Probleme demonstriert und mit "Lifting State Up" löst.

3.1.1.2 Aufgabe 1: Prop Drilling Problem erkennen (20 min)

Analysiere folgende Widget-Struktur:

```
ShoppingApp
+-- AppBar (zeigt Anzahl der Items)
+-- ShoppingList
|   +-- ShoppingItem (mehrere)
|       +-- ItemName
|       +-- ItemQuantity
|       +-- DeleteButton
+-- AddItemButton (fügt neues Item hinzu)
```

Fragen: 1. Wo sollte der State (Liste der Items) leben? 2. Welche Widgets brauchen Zugriff auf den State? 3. Welche Widgets brauchen nur Callbacks? 4. Durch welche Widgets müsste man Daten/Callbacks durchreichen?

3.1.1.3 Aufgabe 2: Lifting State Up implementieren (40 min)

Erstelle eine einfache Einkaufslisten-App mit folgenden Features:

Datenmodell

```
class ShoppingItem {
  final String id;
  final String name;
  final int quantity;
  final bool purchased;

  ShoppingItem({
    required this.id,
    required this.name,
    this.quantity = 1,
    this.purchased = false,
  });

  // copyWith implementieren
}
```

Anforderungen

1. **ShoppingApp (StatefulWidget)**
   - Hält die Liste aller `ShoppingItem`s
   - Stellt Methoden bereit: `addItem`, `togglePurchased`, `removeItem`, `updateQuantity`
2. **ShoppingAppBar (StatelessWidget)**
   - Zeigt: "Einkaufsliste (3 von 5 erledigt)"
   - Erhält die Item-Liste als Parameter
3. **ShoppingListView (StatelessWidget)**
   - Zeigt alle Items als Liste
   - Erhält Items und Callbacks als Parameter
4. **ShoppingItemTile (StatelessWidget)**
   - Zeigt Name und Menge
   - Checkbox für "purchased"
   - Buttons für +/- Menge
   - Delete-Button
   - Durchgestrichener Text wenn purchased
5. **AddItemDialog**
   - TextField für Name
   - Ruft `addItem` Callback auf

Beispiel-UI

```
+---------------------------------+
| Einkaufsliste (2 von 4 erledigt)|
+---------------------------------+
| ☑ Milch            [-] 2 [+] 🗑 |
| ☑ Brot             [-] 1 [+] 🗑 |
| ☐ Eier             [-] 6 [+] 🗑 |
| ☐ Butter           [-] 1 [+] 🗑 |
+---------------------------------+
|           [+ Hinzufügen]        |
+---------------------------------+
```

3.1.1.4 Aufgabe 3: InheritedWidget verstehen (Theorie)

Ohne Code zu schreiben, beantworte:

1. Was ist der Unterschied zwischen `dependOnInheritedWidgetOfExactType` und `getInheritedWidgetOfExactType`?
2. Was bewirkt `updateShouldNotify`? Wann gibt man `true` zurück, wann `false`?
3. Warum braucht man neben dem `InheritedWidget` auch ein `StatefulWidget`?
4. Was passiert, wenn ein Widget `InheritedWidget.of(context)` aufruft, aber kein passendes `InheritedWidget` im Tree existiert?

3.1.1.5 Aufgabe 4: State-Kategorisierung

Kategorisiere folgenden State als "Ephemeral" oder "App State":

State	Kategorie	Begründung
Aktueller Tab-Index in einer TabBar		
Eingeloggter User		
Scroll-Position in einer Liste		
Dark/Light Mode Einstellung		
Formular-Eingaben während der Eingabe		
Warenkorb-Inhalt		
Animation-Progress		
Ausgewähltes Element in einer Liste		
Offline-Cache von API-Daten		
Ob ein Dropdown geöffnet ist		

3.1.1.6 Bonus: ChangeNotifier vorbereiten

Refaktoriere den State aus Aufgabe 2 in einen `ChangeNotifier`:

```
class ShoppingListNotifier extends ChangeNotifier {
  final List<ShoppingItem> _items = [];

  // Implementiere:
  // - items getter (unmodifizierbare Liste)
  // - purchasedCount getter
  // - totalCount getter
  // - addItem(String name)
  // - togglePurchased(String id)
  // - updateQuantity(String id, int delta)
  // - removeItem(String id)
}
```

Dies bereitet auf die nächste Einheit (Provider Basics) vor.

3.1.1.7 Abgabe-Checkliste

- ☐ Prop Drilling Analyse dokumentiert
- ☐ Einkaufslisten-App funktioniert
- ☐ State lebt im richtigen Widget
- ☐ Keine unnötigen StatefulWidgets
- ☐ Callbacks werden korrekt durchgereicht
- ☐ Theorie-Fragen beantwortet
- ☐ State-Kategorisierung ausgefüllt

3.1.2 Lösung

3.1.2.1 Aufgabe 1: Prop Drilling Analyse

1. **Wo lebt der State?**
   - In `ShoppingApp` (oberste Ebene), da mehrere Kinder Zugriff brauchen
2. **Welche Widgets brauchen State-Zugriff?**
   - `AppBar`: Lesend (Anzahl Items)
   - `ShoppingList`: Lesend (alle Items)
   - `ShoppingItem`: Lesend (einzelnes Item)
3. **Welche Widgets brauchen Callbacks?**
   - `DeleteButton`: `onDelete`
   - `AddItemButton`: `onAdd`
   - `ItemQuantity`: `onQuantityChanged`
   - Checkbox: `onToggle`
4. **Prop Drilling Pfade:**
   - `ShoppingApp` -> `ShoppingList` -> `ShoppingItem` -> `DeleteButton`
   - Jedes Widget dazwischen muss Parameter durchreichen

3.1.2.2 Aufgabe 2: Einkaufslisten-App

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Einkaufsliste',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.green),
        useMaterial3: true,
      ),
      home: const ShoppingApp(),
    );
  }
}

// Datenmodell
class ShoppingItem {
  final String id;
  final String name;
  final int quantity;
  final bool purchased;

  ShoppingItem({
    required this.id,
```

```
    required this.name,
    this.quantity = 1,
    this.purchased = false,
  });

  ShoppingItem copyWith({
    String? name,
    int? quantity,
    bool? purchased,
  }) {
    return ShoppingItem(
      id: id,
      name: name ?? this.name,
      quantity: quantity ?? this.quantity,
      purchased: purchased ?? this.purchased,
    );
  }
}

// Haupt-Widget mit State
class ShoppingApp extends StatefulWidget {
  const ShoppingApp({super.key});

  @override
  State<ShoppingApp> createState() => _ShoppingAppState();
}

class _ShoppingAppState extends State<ShoppingApp> {
  final List<ShoppingItem> _items = [
    ShoppingItem(id: '1', name: 'Milch', quantity: 2, purchased: true),
    ShoppingItem(id: '2', name: 'Brot', quantity: 1, purchased: true),
    ShoppingItem(id: '3', name: 'Eier', quantity: 6),
    ShoppingItem(id: '4', name: 'Butter', quantity: 1),
  ];

  void _addItem(String name) {
    setState(() {
      _items.add(ShoppingItem(
        id: DateTime.now().millisecondsSinceEpoch.toString(),
        name: name,
      ));
    });
  }

  void _togglePurchased(String id) {
    setState(() {
      final index = _items.indexWhere((item) => item.id == id);
      if (index != -1) {
        _items[index] = _items[index].copyWith(
          purchased: !_items[index].purchased,
        );
```

```
      }
    });
  }

  void _updateQuantity(String id, int delta) {
    setState(() {
      final index = _items.indexWhere((item) => item.id == id);
      if (index != -1) {
        final newQuantity = _items[index].quantity + delta;
        if (newQuantity > 0) {
          _items[index] = _items[index].copyWith(quantity: newQuantity);
        }
      }
    });
  }

  void _removeItem(String id) {
    setState(() {
      _items.removeWhere((item) => item.id == id);
    });
  }

  void _showAddDialog() {
    showDialog(
      context: context,
      builder: (context) => AddItemDialog(onAdd: _addItem),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: ShoppingAppBar(items: _items),
      body: ShoppingListView(
        items: _items,
        onToggle: _togglePurchased,
        onQuantityChanged: _updateQuantity,
        onRemove: _removeItem,
      ),
      floatingActionButton: FloatingActionButton.extended(
        onPressed: _showAddDialog,
        icon: const Icon(Icons.add),
        label: const Text('Hinzufügen'),
      ),
    );
  }
}

// AppBar mit Item-Zähler
class ShoppingAppBar extends StatelessWidget implements PreferredSizeWidget {
  final List<ShoppingItem> items;
```

```
  const ShoppingAppBar({super.key, required this.items});

  @override
  Size get preferredSize => const Size.fromHeight(kToolbarHeight);

  @override
  Widget build(BuildContext context) {
    final purchasedCount = items.where((item) => item.purchased).length;
    final totalCount = items.length;

    return AppBar(
      title: Text('Einkaufsliste ($purchasedCount von $totalCount erledigt)'),
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
    );
  }
}

// Liste der Items
class ShoppingListView extends StatelessWidget {
  final List<ShoppingItem> items;
  final void Function(String id) onToggle;
  final void Function(String id, int delta) onQuantityChanged;
  final void Function(String id) onRemove;

  const ShoppingListView({
    super.key,
    required this.items,
    required this.onToggle,
    required this.onQuantityChanged,
    required this.onRemove,
  });

  @override
  Widget build(BuildContext context) {
    if (items.isEmpty) {
      return const Center(
        child: Text('Keine Items vorhanden'),
      );
    }

    return ListView.builder(
      padding: const EdgeInsets.only(bottom: 80),
      itemCount: items.length,
      itemBuilder: (context, index) {
        final item = items[index];
        return ShoppingItemTile(
          item: item,
          onToggle: () => onToggle(item.id),
          onQuantityChanged: (delta) => onQuantityChanged(item.id, delta),
          onRemove: () => onRemove(item.id),
```

```
        );
      },
    );
  }
}

// Einzelnes Item
class ShoppingItemTile extends StatelessWidget {
  final ShoppingItem item;
  final VoidCallback onToggle;
  final void Function(int delta) onQuantityChanged;
  final VoidCallback onRemove;

  const ShoppingItemTile({
    super.key,
    required this.item,
    required this.onToggle,
    required this.onQuantityChanged,
    required this.onRemove,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: const EdgeInsets.symmetric(horizontal: 8, vertical: 4),
      child: ListTile(
        leading: Checkbox(
          value: item.purchased,
          onChanged: (_) => onToggle(),
        ),
        title: Text(
          item.name,
          style: TextStyle(
            decoration: item.purchased
                ? TextDecoration.lineThrough
                : TextDecoration.none,
            color: item.purchased
                ? Colors.grey
                : null,
          ),
        ),
        trailing: Row(
          mainAxisSize: MainAxisSize.min,
          children: [
            IconButton(
              icon: const Icon(Icons.remove),
              onPressed: item.quantity > 1
                  ? () => onQuantityChanged(-1)
                  : null,
            ),
            SizedBox(
```

```
              width: 30,
              child: Text(
                '${item.quantity}',
                textAlign: TextAlign.center,
                style: const TextStyle(fontSize: 16),
              ),
            ),
            IconButton(
              icon: const Icon(Icons.add),
              onPressed: () => onQuantityChanged(1),
            ),
            IconButton(
              icon: const Icon(Icons.delete_outline),
              color: Colors.red,
              onPressed: onRemove,
            ),
          ],
        ),
      ),
    );
  }
}

// Dialog zum Hinzufügen
class AddItemDialog extends StatefulWidget {
  final void Function(String name) onAdd;

  const AddItemDialog({super.key, required this.onAdd});

  @override
  State<AddItemDialog> createState() => _AddItemDialogState();
}

class _AddItemDialogState extends State<AddItemDialog> {
  final _controller = TextEditingController();

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  void _submit() {
    final name = _controller.text.trim();
    if (name.isNotEmpty) {
      widget.onAdd(name);
      Navigator.of(context).pop();
    }
  }

  @override
```

```
  Widget build(BuildContext context) {
    return AlertDialog(
      title: const Text('Neues Item'),
      content: TextField(
        controller: _controller,
        autofocus: true,
        decoration: const InputDecoration(
          hintText: 'Name eingeben',
        ),
        onSubmitted: (_) => _submit(),
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.of(context).pop(),
          child: const Text('Abbrechen'),
        ),
        ElevatedButton(
          onPressed: _submit,
          child: const Text('Hinzufügen'),
        ),
      ],
    );
  }
}
```

3.1.2.3 Aufgabe 3: InheritedWidget Theorie

1. **dependOnInheritedWidgetOfExactType vs. getInheritedWidgetOfExactType:**
   - `dependOn...`: Registriert das Widget als Dependent. Wenn sich `InheritedWidget` ändert, wird das Widget neu gebaut.
   - `get...`: Nur Lesezugriff ohne Subscription. Widget wird NICHT neu gebaut bei Änderungen.
2. **updateShouldNotify:**
   - Bestimmt, ob Dependents benachrichtigt werden sollen
   - `true`: Wenn sich relevante Daten geändert haben -> Rebuild der Dependents
   - `false`: Keine Änderung -> kein Rebuild
   - Typisch: `return oldWidget.value != value;`
3. **Warum StatefulWidget zusätzlich?**
   - `InheritedWidget` selbst ist immutable
   - Das `StatefulWidget` hält den mutablen State und ruft `setState()` auf
   - Bei `setState()` wird ein neues `InheritedWidget` mit neuen Werten erstellt
4. **Wenn kein InheritedWidget im Tree:**
   - `dependOnInheritedWidgetOfExactType` gibt `null` zurück
   - Typischerweise wirft man dann eine Exception oder verwendet einen Default
   - Darum: `context.dependOn...()!` oder null-check

3.1.2.4 Aufgabe 4: State-Kategorisierung

State	Kategorie	Begründung
Aktueller Tab-Index	**Ephemeral**	Lokal in TabBar, kein anderes Widget braucht ihn
Eingeloggter User	**App State**	Global, viele Widgets brauchen die Info
Scroll-Position	**Ephemeral**	Lokal im ScrollController
Dark/Light Mode	**App State**	Global, beeinflusst gesamte App
Formular-Eingaben	**Ephemeral**	Lokal während Eingabe (wird App State bei Submit)
Warenkorb-Inhalt	**App State**	Wird auf mehreren Screens gebraucht
Animation-Progress	**Ephemeral**	Lokal im AnimationController
Ausgewähltes Element	**Kommt drauf an**	Ephemeral wenn lokal, App State wenn andere Screens es brauchen
Offline-Cache	**App State**	Global, persistiert
Dropdown offen/zu	**Ephemeral**	Lokaler UI-State

3.1.2.5 Bonus: ChangeNotifier

```
class ShoppingListNotifier extends ChangeNotifier {
  final List<ShoppingItem> _items = [];

  // Unmodifizierbare Kopie nach außen geben
  List<ShoppingItem> get items => List.unmodifiable(_items);

  int get purchasedCount => _items.where((i) => i.purchased).length;
  int get totalCount => _items.length;

  // Computed Property für Fortschritt
  double get progress =>
      totalCount == 0 ? 0 : purchasedCount / totalCount;

  void addItem(String name) {
    _items.add(ShoppingItem(
      id: DateTime.now().millisecondsSinceEpoch.toString(),
      name: name,
    ));
    notifyListeners();
  }

  void togglePurchased(String id) {
    final index = _items.indexWhere((item) => item.id == id);
    if (index != -1) {
      _items[index] = _items[index].copyWith(
        purchased: !_items[index].purchased,
      );
      notifyListeners();
    }
  }
```

```
  void updateQuantity(String id, int delta) {
    final index = _items.indexWhere((item) => item.id == id);
    if (index != -1) {
      final newQuantity = _items[index].quantity + delta;
      if (newQuantity > 0) {
        _items[index] = _items[index].copyWith(quantity: newQuantity);
        notifyListeners();
      }
    }
  }

  void removeItem(String id) {
    _items.removeWhere((item) => item.id == id);
    notifyListeners();
  }

  // Optional: Alle als erledigt markieren
  void markAllPurchased() {
    for (int i = 0; i < _items.length; i++) {
      if (!_items[i].purchased) {
        _items[i] = _items[i].copyWith(purchased: true);
      }
    }
    notifyListeners();
  }

  // Optional: Erledigte entfernen
  void clearPurchased() {
    _items.removeWhere((item) => item.purchased);
    notifyListeners();
  }
}
```

Dieser `ChangeNotifier` kann in der nächsten Einheit direkt mit Provider verwendet werden.

3.1.3 Ressourcen

3.1.3.1 Offizielle Dokumentation

- State Management - Flutter Docs
- Differentiate between ephemeral state and app state
- Simple app state management
- List of state management approaches
- InheritedWidget API

3.1.3.2 Empfohlene Artikel

- Flutter State Management: The Grand Tour
- Pragmatic State Management in Flutter - von der Flutter Team
- InheritedWidget explained

3.1.3.3 Videos

- State Management Explained - Flutter Official
- Flutter State Management for Beginners - Reso Coder
- InheritedWidget - Widget of the Week

3.1.3.4 Vergleiche und Übersichten

- Flutter State Management in 2024
- Provider vs Riverpod vs Bloc

3.1.3.5 Zum Vertiefen

- ChangeNotifier explained
- ValueNotifier für einfachen State
- Flutter Architecture Samples - Gleiche App mit verschiedenen State Management Lösungen

3.2 Einheit 3.2: Provider Basics

3.2.0.1 Lernziele

Nach dieser Einheit kannst du: - Das `provider` Package einrichten und verwenden - `ChangeNotifier` für State-Klassen implementieren - `ChangeNotifierProvider` korrekt einsetzen - Zwischen `context.watch`, `context.read` und `Consumer` unterscheiden - Häufige Fehler vermeiden

3.2.0.2 1. Provider einrichten

Provider ist das empfohlene State Management Package für Flutter (von Google selbst empfohlen).

Installation

```
# pubspec.yaml
dependencies:
  flutter:
    sdk: flutter
  provider: ^6.1.1
```

```
flutter pub get
```

Import

```
import 'package:provider/provider.dart';
```

3.2.0.3 2. ChangeNotifier: Die Basis

`ChangeNotifier` ist eine Klasse aus dem Flutter Framework, die das Observer-Pattern implementiert.

```
import 'package:flutter/foundation.dart';

class Counter extends ChangeNotifier {
  int _count = 0;

  int get count => _count;
```

```
  void increment() {
    _count++;
    notifyListeners();  // Benachrichtigt alle Listener!
  }

  void decrement() {
    _count--;
    notifyListeners();
  }

  void reset() {
    _count = 0;
    notifyListeners();
  }
}
```

Wichtige Regeln

```
class GoodNotifier extends ChangeNotifier {
  // ✅ Private Felder mit Gettern
  int _value = 0;
  int get value => _value;

  // ✅ notifyListeners() nur wenn sich wirklich etwas ändert
  void setValue(int newValue) {
    if (_value != newValue) {
      _value = newValue;
      notifyListeners();
    }
  }

  // ✅ Immutable Listen nach außen geben
  final List<String> _items = [];
  List<String> get items => List.unmodifiable(_items);

  void addItem(String item) {
    _items.add(item);
    notifyListeners();
  }
}

class BadNotifier extends ChangeNotifier {
  // ❌ Öffentliche Felder - können von außen verändert werden
  int value = 0;

  // ❌ Mutable Liste nach außen geben
  List<String> items = [];

  // ❌ notifyListeners() vergessen
  void increment() {
```

```
    value++;
    // Forgot notifyListeners()!
  }
}
```

3.2.0.4 3. ChangeNotifierProvider

`ChangeNotifierProvider` stellt eine `ChangeNotifier`-Instanz im Widget-Tree bereit.

Einfaches Setup

```
void main() {
  runApp(
    ChangeNotifierProvider(
      create: (context) => Counter(),
      child: const MyApp(),
    ),
  );
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: CounterPage(),
    );
  }
}
```

Provider um einen Teil der App

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: ChangeNotifierProvider(
        // Provider nur für diesen Subtree
        create: (_) => ShoppingCart(),
        child: ShoppingPage(),
      ),
    );
  }
}
```

Lazy vs. Non-Lazy

```
// Standard: Lazy (Instanz wird erst bei erstem Zugriff erstellt)
ChangeNotifierProvider(
  create: (_) => ExpensiveNotifier(),
  child: MyApp(),
);
```

```
// Non-lazy: Sofort erstellen
ChangeNotifierProvider(
  create: (_) => ExpensiveNotifier(),
  lazy: false,  // Wird sofort instanziiert
  child: MyApp(),
);
```

3.2.0.5 4. Auf Provider zugreifen

Es gibt drei Hauptwege, um auf einen Provider zuzugreifen:

4.1 context.watch()

Subscribed zum Provider und **rebuildet** das Widget bei Änderungen.

```
class CounterDisplay extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Widget wird bei jeder Änderung neu gebaut
    final counter = context.watch<Counter>();

    return Text('Count: ${counter.count}');
  }
}
```

4.2 context.read()

Einmaliger Zugriff **ohne Subscription**. Widget wird **nicht** neu gebaut.

```
class IncrementButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: () {
        // Nur Methode aufrufen, kein Rebuild nötig
        context.read<Counter>().increment();
      },
      child: Text('Increment'),
    );
  }
}
```

4.3 Consumer

Widget-basierte Alternative zu `context.watch`. Nützlich wenn nur ein Teil des Widgets rebuilden soll.

```
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Counter')),
      body: Column(
```

```
        children: [
          // Nur dieser Teil rebuildet
          Consumer<Counter>(
            builder: (context, counter, child) {
              return Text('Count: ${counter.count}');
            },
          ),

          // Dieser Teil rebuildet NICHT
          Text('Static content'),
        ],
      ),
    );
  }
}
```

Vergleich

Methode	Rebuild bei Änderung	Use Case
`context.watch<T>()`	Ja (gesamtes Widget)	Daten anzeigen
`context.read<T>()`	Nein	Methoden aufrufen (onPressed, etc.)
`Consumer<T>`	Ja (nur builder)	Teilbereich eines Widgets

3.2.0.6 5. Wichtige Regeln

watch() nur in build()

```
class MyWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // ✅ Korrekt: watch() in build()
    final counter = context.watch<Counter>();

    return ElevatedButton(
      onPressed: () {
        // ✅ Korrekt: read() in Callback
        context.read<Counter>().increment();
      },
      child: Text('${counter.count}'),
    );
  }
}
```

Niemals watch() in Callbacks

```
// ❌ FALSCH: watch() in onPressed
onPressed: () {
  final counter = context.watch<Counter>();  // FEHLER!
  counter.increment();
}
```

```
// ✅ RICHTIG: read() in onPressed
onPressed: () {
  context.read<Counter>().increment();
}
```

Provider-Scope beachten

```
// ❌ Fehler: Provider existiert nicht über diesem Widget
class MyWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (_) => Counter(),
      child: ElevatedButton(
        onPressed: () {
          // FEHLER: Counter ist im child, nicht darüber!
          context.read<Counter>().increment();
        },
        child: Text('Click'),
      ),
    );
  }
}

// ✅ Korrekt: Separates Widget
class MyWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (_) => Counter(),
      child: CounterButton(),  // Separates Widget
    );
  }
}

class CounterButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Jetzt ist Counter verfügbar
    return ElevatedButton(
      onPressed: () => context.read<Counter>().increment(),
      child: Text('Click'),
    );
  }
}
```

3.2.0.7 6. Vollständiges Beispiel

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
```

```
void main() {
  runApp(
    ChangeNotifierProvider(
      create: (_) => TodoList(),
      child: const MyApp(),
    ),
  );
}

// Model
class Todo {
  final String id;
  final String title;
  bool completed;

  Todo({required this.id, required this.title, this.completed = false});
}

// State/Notifier
class TodoList extends ChangeNotifier {
  final List<Todo> _todos = [];

  List<Todo> get todos => List.unmodifiable(_todos);
  int get completedCount => _todos.where((t) => t.completed).length;
  int get totalCount => _todos.length;

  void add(String title) {
    _todos.add(Todo(
      id: DateTime.now().toString(),
      title: title,
    ));
    notifyListeners();
  }

  void toggle(String id) {
    final todo = _todos.firstWhere((t) => t.id == id);
    todo.completed = !todo.completed;
    notifyListeners();
  }

  void remove(String id) {
    _todos.removeWhere((t) => t.id == id);
    notifyListeners();
  }
}

// App
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
```

```
  Widget build(BuildContext context) {
    return MaterialApp(
      home: TodoPage(),
    );
  }
}

// Page
class TodoPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Consumer<TodoList>(
          builder: (_, todoList, __) {
            return Text(
              'Todos (${todoList.completedCount}/${todoList.totalCount})',
            );
          },
        ),
      ),
      body: Consumer<TodoList>(
        builder: (_, todoList, __) {
          if (todoList.todos.isEmpty) {
            return const Center(child: Text('Keine Todos'));
          }

          return ListView.builder(
            itemCount: todoList.todos.length,
            itemBuilder: (context, index) {
              final todo = todoList.todos[index];
              return TodoTile(todo: todo);
            },
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => _showAddDialog(context),
        child: const Icon(Icons.add),
      ),
    );
  }

  void _showAddDialog(BuildContext context) {
    final controller = TextEditingController();

    showDialog(
      context: context,
      builder: (dialogContext) => AlertDialog(
        title: const Text('Neues Todo'),
        content: TextField(
```

```
          controller: controller,
          autofocus: true,
        ),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(dialogContext),
            child: const Text('Abbrechen'),
          ),
          ElevatedButton(
            onPressed: () {
              if (controller.text.isNotEmpty) {
                // read() in Callback!
                context.read<TodoList>().add(controller.text);
                Navigator.pop(dialogContext);
              }
            },
            child: const Text('Hinzufügen'),
          ),
        ],
      ),
    );
  }
}

// Todo Item
class TodoTile extends StatelessWidget {
  final Todo todo;

  const TodoTile({super.key, required this.todo});

  @override
  Widget build(BuildContext context) {
    return ListTile(
      leading: Checkbox(
        value: todo.completed,
        onChanged: (_) => context.read<TodoList>().toggle(todo.id),
      ),
      title: Text(
        todo.title,
        style: TextStyle(
          decoration: todo.completed
              ? TextDecoration.lineThrough
              : null,
        ),
      ),
      trailing: IconButton(
        icon: const Icon(Icons.delete),
        onPressed: () => context.read<TodoList>().remove(todo.id),
      ),
    );
  }
```

```
}
```

3.2.0.8 7. dispose() nicht vergessen

`ChangeNotifierProvider` ruft automatisch `dispose()` auf dem `ChangeNotifier` auf, wenn der Provider aus dem Widget-Tree entfernt wird.

```
class MyNotifier extends ChangeNotifier {
  final StreamSubscription _subscription;

  MyNotifier(Stream stream)
      : _subscription = stream.listen((_) {});

  @override
  void dispose() {
    _subscription.cancel();  // Ressourcen freigeben!
    super.dispose();
  }
}
```

3.2.0.9 Zusammenfassung

Konzept	Verwendung
`ChangeNotifier`	State-Klasse, ruft `notifyListeners()` bei Änderungen
`ChangeNotifierProvider`	Stellt ChangeNotifier im Widget-Tree bereit
`context.watch<T>()`	Lesen + Subscription (in `build()`)
`context.read<T>()`	Nur lesen, keine Subscription (in Callbacks)
`Consumer<T>`	Widget-basierte Alternative zu `watch()`

Goldene Regeln: 1. `watch()` nur in `build()` 2. `read()` für Callbacks (onPressed, etc.) 3. Private Felder + Getter in ChangeNotifier 4. `notifyListeners()` nur bei echten Änderungen

3.2.1 Übung

3.2.1.1 Ziel

Die Einkaufslisten-App aus Einheit 3.1 auf Provider umstellen.

3.2.1.2 Aufgabe 1: Setup (10 min)

1. Erstelle ein neues Flutter-Projekt oder verwende das bestehende
2. Füge `provider: ^6.1.1` zur `pubspec.yaml` hinzu
3. Führe `flutter pub get` aus
4. Importiere Provider in deiner `main.dart`

3.2.1.3 Aufgabe 2: ChangeNotifier erstellen (20 min)

Erstelle eine `ShoppingListNotifier`-Klasse:

```
class ShoppingItem {
  final String id;
  final String name;
  final int quantity;
  final bool purchased;

  ShoppingItem({
    required this.id,
    required this.name,
    this.quantity = 1,
    this.purchased = false,
  });

  ShoppingItem copyWith({String? name, int? quantity, bool? purchased}) {
    return ShoppingItem(
      id: id,
      name: name ?? this.name,
      quantity: quantity ?? this.quantity,
      purchased: purchased ?? this.purchased,
    );
  }
}
```

Der `ShoppingListNotifier` soll implementieren:

- `List<ShoppingItem> get items` - Unmodifizierbare Liste
- `int get totalCount` - Gesamtzahl
- `int get purchasedCount` - Anzahl erledigter Items
- `void addItem(String name)` - Neues Item hinzufügen
- `void togglePurchased(String id)` - Purchased-Status umschalten
- `void updateQuantity(String id, int delta)` - Menge ändern (+1 oder -1)
- `void removeItem(String id)` - Item entfernen

3.2.1.4 Aufgabe 3: Provider einrichten (10 min)

1. Wrape deine `MaterialApp` mit einem `ChangeNotifierProvider`
2. Stelle sicher, dass der Provider über der gesamten App liegt

```
void main() {
  runApp(
    ChangeNotifierProvider(
      create: (_) => ShoppingListNotifier(),
      child: const MyApp(),
    ),
  );
}
```

3.2.1.5 Aufgabe 4: Widgets umbauen (30 min)

Baue die Widgets um, sodass sie Provider verwenden:

4.1 ShoppingAppBar

- Verwende `Consumer<ShoppingListNotifier>` oder `context.watch`
- Zeige "Einkaufsliste (X von Y erledigt)"

4.2 ShoppingListView

- Entferne alle Callback-Parameter
- Greife direkt auf den Provider zu

4.3 ShoppingItemTile

- Verwende `context.read` für die Callbacks (onPressed)
- Das Item selbst kommt weiterhin als Parameter

Beispiel für ein Item:

```
class ShoppingItemTile extends StatelessWidget {
  final ShoppingItem item;

  const ShoppingItemTile({super.key, required this.item});

  @override
  Widget build(BuildContext context) {
    return ListTile(
      leading: Checkbox(
        value: item.purchased,
        onChanged: (_) {
          // read() für Callbacks!
          context.read<ShoppingListNotifier>().togglePurchased(item.id);
        },
      ),
      // ... rest des Widgets
    );
  }
}
```

3.2.1.6 Aufgabe 5: Verständnisfragen

Beantworte folgende Fragen:

1. Warum verwenden wir `context.read()` in `onPressed` und nicht `context.watch()`?
2. Was passiert, wenn du `context.watch()` in einem `onPressed`-Handler verwendest?
3. Wo genau wird `dispose()` auf dem `ChangeNotifier` aufgerufen?
4. Warum geben wir `List.unmodifiable(_items)` zurück statt `_items` direkt?
5. Was ist der Unterschied zwischen:

```
// Variante A
final notifier = context.watch<ShoppingListNotifier>();
return Text('${notifier.totalCount}');

// Variante B
return Consumer<ShoppingListNotifier>(
  builder: (_, notifier, __) => Text('${notifier.totalCount}'),
);
```

3.2.1.7 Aufgabe 6: Bonus - Computed Properties

Füge dem `ShoppingListNotifier` hinzu:

1. `double get progress` - Fortschritt als Wert zwischen 0.0 und 1.0
2. `List<ShoppingItem> get pendingItems` - Nur nicht-erledigte Items
3. `List<ShoppingItem> get purchasedItems` - Nur erledigte Items
4. `int get totalQuantity` - Summe aller Mengen

Zeige den Fortschritt als `LinearProgressIndicator` in der AppBar an:

```
+---------------------------------+
| Einkaufsliste (2 von 4)         |
| ████████▒▒▒▒▒▒▒▒ 50%            |
+---------------------------------+
```

3.2.1.8 Abgabe-Checkliste

- ☐ Provider Package installiert
- ☐ `ShoppingListNotifier` implementiert mit allen Methoden
- ☐ Provider korrekt in `main()` eingerichtet
- ☐ `context.watch()` nur in `build()` verwendet
- ☐ `context.read()` in Callbacks verwendet
- ☐ App funktioniert wie zuvor
- ☐ Kein Prop Drilling mehr nötig
- ☐ Verständnisfragen beantwortet

3.2.1.9 Erwartete Projektstruktur

```
lib/
+-- main.dart
+-- models/
|   +-- shopping_item.dart
+-- providers/
|   +-- shopping_list_notifier.dart
+-- widgets/
    +-- shopping_app_bar.dart
    +-- shopping_list_view.dart
    +-- shopping_item_tile.dart
    +-- add_item_dialog.dart
```

3.2.2 Lösung

3.2.2.1 Projektstruktur

```
lib/
+-- main.dart
+-- models/
|   +-- shopping_item.dart
+-- providers/
|   +-- shopping_list_notifier.dart
+-- widgets/
    +-- shopping_app_bar.dart
    +-- shopping_list_view.dart
    +-- shopping_item_tile.dart
```

```
    +-- add_item_dialog.dart
```

3.2.2.2 models/shopping_item.dart

```
class ShoppingItem {
  final String id;
  final String name;
  final int quantity;
  final bool purchased;

  ShoppingItem({
    required this.id,
    required this.name,
    this.quantity = 1,
    this.purchased = false,
  });

  ShoppingItem copyWith({
    String? name,
    int? quantity,
    bool? purchased,
  }) {
    return ShoppingItem(
      id: id,
      name: name ?? this.name,
      quantity: quantity ?? this.quantity,
      purchased: purchased ?? this.purchased,
    );
  }
}
```

3.2.2.3 providers/shopping_list_notifier.dart

```
import 'package:flutter/foundation.dart';
import '../models/shopping_item.dart';

class ShoppingListNotifier extends ChangeNotifier {
  final List<ShoppingItem> _items = [
    // Initiale Testdaten
    ShoppingItem(id: '1', name: 'Milch', quantity: 2, purchased: true),
    ShoppingItem(id: '2', name: 'Brot', quantity: 1),
    ShoppingItem(id: '3', name: 'Eier', quantity: 6),
  ];

  // Unmodifizierbare Liste nach außen
  List<ShoppingItem> get items => List.unmodifiable(_items);

  // Computed Properties
  int get totalCount => _items.length;
```

```
int get purchasedCount => _items.where((i) => i.purchased).length;

double get progress =>
    totalCount == 0 ? 0.0 : purchasedCount / totalCount;

List<ShoppingItem> get pendingItems =>
    _items.where((i) => !i.purchased).toList();

List<ShoppingItem> get purchasedItems =>
    _items.where((i) => i.purchased).toList();

int get totalQuantity =>
    _items.fold(0, (sum, item) => sum + item.quantity);

// Mutationen
void addItem(String name) {
  if (name.trim().isEmpty) return;

  _items.add(ShoppingItem(
    id: DateTime.now().millisecondsSinceEpoch.toString(),
    name: name.trim(),
  ));
  notifyListeners();
}

void togglePurchased(String id) {
  final index = _items.indexWhere((item) => item.id == id);
  if (index == -1) return;

  _items[index] = _items[index].copyWith(
    purchased: !_items[index].purchased,
  );
  notifyListeners();
}

void updateQuantity(String id, int delta) {
  final index = _items.indexWhere((item) => item.id == id);
  if (index == -1) return;

  final newQuantity = _items[index].quantity + delta;
  if (newQuantity < 1) return;

  _items[index] = _items[index].copyWith(quantity: newQuantity);
  notifyListeners();
}

void removeItem(String id) {
  final removed = _items.removeWhere((item) => item.id == id);
  // Nur benachrichtigen wenn etwas entfernt wurde
  notifyListeners();
}
```

```
  void clearPurchased() {
    _items.removeWhere((item) => item.purchased);
    notifyListeners();
  }
}
```

3.2.2.4 main.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'providers/shopping_list_notifier.dart';
import 'widgets/shopping_app_bar.dart';
import 'widgets/shopping_list_view.dart';
import 'widgets/add_item_dialog.dart';

void main() {
  runApp(
    ChangeNotifierProvider(
      create: (_) => ShoppingListNotifier(),
      child: const MyApp(),
    ),
  );
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Einkaufsliste',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.green),
        useMaterial3: true,
      ),
      home: const ShoppingPage(),
    );
  }
}

class ShoppingPage extends StatelessWidget {
  const ShoppingPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: const ShoppingAppBar(),
      body: const ShoppingListView(),
      floatingActionButton: FloatingActionButton.extended(
```

```
        onPressed: () => _showAddDialog(context),
        icon: const Icon(Icons.add),
        label: const Text('Hinzufügen'),
      ),
    );
  }

  void _showAddDialog(BuildContext context) {
    showDialog(
      context: context,
      builder: (_) => const AddItemDialog(),
    );
  }
}
```

3.2.2.5 widgets/shopping_app_bar.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/shopping_list_notifier.dart';

class ShoppingAppBar extends StatelessWidget implements PreferredSizeWidget {
  const ShoppingAppBar({super.key});

  @override
  Size get preferredSize => const Size.fromHeight(kToolbarHeight + 4);

  @override
  Widget build(BuildContext context) {
    // Consumer für feingranulare Rebuilds
    return Consumer<ShoppingListNotifier>(
      builder: (context, notifier, _) {
        return AppBar(
          title: Text(
            'Einkaufsliste (${notifier.purchasedCount} von
↪  ${notifier.totalCount})',
          ),
          backgroundColor: Theme.of(context).colorScheme.inversePrimary,
          bottom: PreferredSize(
            preferredSize: const Size.fromHeight(4),
            child: LinearProgressIndicator(
              value: notifier.progress,
              backgroundColor: Colors.white24,
            ),
          ),
          actions: [
            if (notifier.purchasedCount > 0)
              IconButton(
                icon: const Icon(Icons.delete_sweep),
                tooltip: 'Erledigte löschen',
```

```
                onPressed: () {
                  context.read<ShoppingListNotifier>().clearPurchased();
                },
              ),
          ],
        );
      },
    );
  }
}
```

3.2.2.6 widgets/shopping_list_view.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/shopping_list_notifier.dart';
import 'shopping_item_tile.dart';

class ShoppingListView extends StatelessWidget {
  const ShoppingListView({super.key});

  @override
  Widget build(BuildContext context) {
    // watch() subscribes to changes
    final notifier = context.watch<ShoppingListNotifier>();

    if (notifier.items.isEmpty) {
      return const Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Icon(Icons.shopping_cart_outlined, size: 64, color: Colors.grey),
            SizedBox(height: 16),
            Text(
              'Keine Items vorhanden',
              style: TextStyle(color: Colors.grey, fontSize: 18),
            ),
          ],
        ),
      );
    }

    return ListView.builder(
      padding: const EdgeInsets.only(bottom: 80),
      itemCount: notifier.items.length,
      itemBuilder: (context, index) {
        final item = notifier.items[index];
        return ShoppingItemTile(
          key: ValueKey(item.id),
          item: item,
```

```
        );
      },
    );
  }
}
```

3.2.2.7 widgets/shopping_item_tile.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/shopping_item.dart';
import '../providers/shopping_list_notifier.dart';

class ShoppingItemTile extends StatelessWidget {
  final ShoppingItem item;

  const ShoppingItemTile({super.key, required this.item});

  @override
  Widget build(BuildContext context) {
    return Dismissible(
      key: ValueKey(item.id),
      direction: DismissDirection.endToStart,
      background: Container(
        color: Colors.red,
        alignment: Alignment.centerRight,
        padding: const EdgeInsets.only(right: 16),
        child: const Icon(Icons.delete, color: Colors.white),
      ),
      onDismissed: (_) {
        // read() in Callback!
        context.read<ShoppingListNotifier>().removeItem(item.id);
      },
      child: Card(
        margin: const EdgeInsets.symmetric(horizontal: 8, vertical: 4),
        child: ListTile(
          leading: Checkbox(
            value: item.purchased,
            onChanged: (_) {
              // read() in Callback!
              context.read<ShoppingListNotifier>().togglePurchased(item.id);
            },
          ),
          title: Text(
            item.name,
            style: TextStyle(
              decoration: item.purchased
                  ? TextDecoration.lineThrough
                  : TextDecoration.none,
              color: item.purchased ? Colors.grey : null,
```

```
            ),
          ),
          trailing: Row(
            mainAxisSize: MainAxisSize.min,
            children: [
              IconButton(
                icon: const Icon(Icons.remove),
                onPressed: item.quantity > 1
                    ? () {
                        context
                            .read<ShoppingListNotifier>()
                            .updateQuantity(item.id, -1);
                      }
                    : null,
              ),
              SizedBox(
                width: 30,
                child: Text(
                  '${item.quantity}',
                  textAlign: TextAlign.center,
                  style: const TextStyle(fontSize: 16),
                ),
              ),
              IconButton(
                icon: const Icon(Icons.add),
                onPressed: () {
                  context
                      .read<ShoppingListNotifier>()
                      .updateQuantity(item.id, 1);
                },
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

3.2.2.8 widgets/add_item_dialog.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/shopping_list_notifier.dart';

class AddItemDialog extends StatefulWidget {
  const AddItemDialog({super.key});

  @override
  State<AddItemDialog> createState() => _AddItemDialogState();
```

```
}

class _AddItemDialogState extends State<AddItemDialog> {
  final _controller = TextEditingController();

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  void _submit() {
    if (_controller.text.trim().isNotEmpty) {
      // read() in Callback!
      context.read<ShoppingListNotifier>().addItem(_controller.text);
      Navigator.of(context).pop();
    }
  }

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: const Text('Neues Item'),
      content: TextField(
        controller: _controller,
        autofocus: true,
        decoration: const InputDecoration(
          hintText: 'Name eingeben',
          border: OutlineInputBorder(),
        ),
        textCapitalization: TextCapitalization.sentences,
        onSubmitted: (_) => _submit(),
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.of(context).pop(),
          child: const Text('Abbrechen'),
        ),
        ElevatedButton(
          onPressed: _submit,
          child: const Text('Hinzufügen'),
        ),
      ],
    );
  }
}
```

3.2.2.9 Verständnisfragen - Antworten

1. Warum `read()` in `onPressed`?

`watch()` registriert das Widget als Listener und triggert Rebuilds bei Änderungen. In einem `onPressed`-Handler wollen wir nur eine Methode aufrufen, ohne das Widget zu subscriben. `read()` gibt einmaligen Zugriff ohne Subscription.

2. Was passiert bei `watch()` in `onPressed`?

Flutter wirft einen Fehler: "Tried to listen to a value exposed with provider, from outside of the widget tree." Das liegt daran, dass `watch()` im Kontext einer Build-Phase aufgerufen werden muss, nicht in einem Callback.

3. Wo wird `dispose()` aufgerufen?

`ChangeNotifierProvider` ruft automatisch `dispose()` auf dem `ChangeNotifier` auf, wenn der Provider aus dem Widget-Tree entfernt wird. Das passiert z.B. wenn die Seite geschlossen wird oder die App beendet wird.

4. Warum `List.unmodifiable()`?

Ohne `List.unmodifiable()` könnte ein Consumer die Liste direkt modifizieren:

```
notifier.items.add(newItem);  // Funktioniert ohne unmodifiable!
```

Das würde den State ändern, ohne `notifyListeners()` aufzurufen. Mit `List.unmodifiable()` wirft die Zeile einen Fehler.

5. Unterschied `watch()` vs `Consumer`

Variante A mit `watch()`: - Das gesamte Widget subscribed zum Provider - Bei Änderung wird das komplette Widget neu gebaut

Variante B mit `Consumer`: - Nur der `builder`-Teil subscribed - Bei Änderung wird nur der `builder` neu gebaut - Der Rest des Widgets bleibt unverändert

`Consumer` ist effizienter wenn nur ein Teil des Widgets die Daten braucht. In diesem einfachen Beispiel macht es keinen großen Unterschied, aber bei komplexeren Widgets mit teuren Berechnungen oder Animationen ist `Consumer` vorzuziehen.

3.2.3 Ressourcen

3.2.3.1 Offizielle Dokumentation

- Provider Package auf pub.dev
- Provider API Documentation
- Simple app state management (Flutter Docs)
- ChangeNotifier API

3.2.3.2 Tutorials

- Provider Tutorial - Flutter Official
- Getting Started with Provider
- Provider Cheat Sheet

3.2.3.3 Videos

- Flutter Provider - Complete Tutorial - Reso Coder
- Provider State Management - Flutter Official
- Provider in 10 Minutes - Fireship

3.2.3.4 Häufige Fehler und Lösungen

- Provider FAQ
- Common Provider Mistakes

3.2.3.5 Zum Vertiefen

- Provider Architecture Best Practices
- Testing with Provider
- Provider vs Riverpod

3.2.3.6 Debugging

- Flutter DevTools - Provider Tab
- Provider Debugging Tips

3.3 Einheit 3.3: Provider Advanced

3.3.0.1 Lernziele

Nach dieser Einheit kannst du: - `Selector` für granulare Rebuilds verwenden - Mehrere Provider mit `MultiProvider` kombinieren - Abhängigkeiten zwischen Providern mit `ProxyProvider` modellieren - `FutureProvider` für asynchrone Initialisierung nutzen - Riverpod als moderne Alternative einordnen

3.3.0.2 1. Selector: Granulare Rebuilds

`Selector` rebuildet nur wenn sich ein bestimmter Wert ändert.

Problem ohne Selector

```
class UserProfile extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Rebuildet bei JEDER Änderung am User (Name, Email, Avatar, etc.)
    final user = context.watch<UserNotifier>();

    return Text(user.name);  // Braucht nur den Namen!
  }
}
```

Lösung mit Selector

```
class UserProfile extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Rebuildet NUR wenn sich der Name ändert
    final name = context.select<UserNotifier, String>(
      (notifier) => notifier.name,
    );

    return Text(name);
  }
}
```

Selector Widget-Variante

```
Selector<UserNotifier, String>(
  selector: (_, notifier) => notifier.name,
  builder: (context, name, child) {
    return Text(name);
  },
  // child wird NICHT rebuilt
  child: const Icon(Icons.person),
);
```

Mehrere Werte selektieren

```
// Mit einem Record
final (name, email) = context.select<UserNotifier, (String, String)>(
  (notifier) => (notifier.name, notifier.email),
);

// Oder ein eigenes Objekt
class UserDisplayData {
  final String name;
  final String email;
  UserDisplayData(this.name, this.email);

  @override
  bool operator ==(Object other) =>
      other is UserDisplayData &&
      name == other.name &&
      email == other.email;

  @override
  int get hashCode => Object.hash(name, email);
}

final displayData = context.select<UserNotifier, UserDisplayData>(
  (n) => UserDisplayData(n.name, n.email),
);
```

3.3.0.3 2. MultiProvider

Kombiniert mehrere Provider sauber ohne Verschachtelung.

Ohne MultiProvider (tief verschachtelt)

```
// □ Schwer lesbar
ChangeNotifierProvider(
  create: (_) => AuthNotifier(),
  child: ChangeNotifierProvider(
    create: (_) => CartNotifier(),
    child: ChangeNotifierProvider(
      create: (_) => SettingsNotifier(),
      child: MyApp(),
    ),
```

```
  ),
);
```

Mit MultiProvider

```
// ✅ Flache, lesbare Struktur
MultiProvider(
  providers: [
    ChangeNotifierProvider(create: (_) => AuthNotifier()),
    ChangeNotifierProvider(create: (_) => CartNotifier()),
    ChangeNotifierProvider(create: (_) => SettingsNotifier()),
  ],
  child: MyApp(),
);
```

Verschiedene Provider-Typen mischen

```
MultiProvider(
  providers: [
    // ChangeNotifier
    ChangeNotifierProvider(create: (_) => AuthNotifier()),

    // Einfacher Wert
    Provider(create: (_) => ApiService()),

    // Stream
    StreamProvider(
      create: (_) => FirebaseAuth.instance.authStateChanges(),
      initialData: null,
    ),

    // Future
    FutureProvider(
      create: (_) => loadConfig(),
      initialData: Config.defaultConfig,
    ),
  ],
  child: MyApp(),
);
```

3.3.0.4 3. ProxyProvider: Abhängigkeiten zwischen Providern

`ProxyProvider` erstellt einen Provider, der von anderen Providern abhängt.

Szenario

```
AuthNotifier (User-Session)
       ↓
   UserService (API-Calls mit Auth-Token)
```

Implementierung

```
MultiProvider(
  providers: [
```

```
    // 1. Auth Provider zuerst
    ChangeNotifierProvider(create: (_) => AuthNotifier()),

    // 2. UserService hängt von Auth ab
    ProxyProvider<AuthNotifier, UserService>(
      update: (_, auth, previousService) {
        return UserService(authToken: auth.token);
      },
    ),
  ],
  child: MyApp(),
);

// UserService
class UserService {
  final String? authToken;

  UserService({this.authToken});

  Future<User> fetchUser(String id) async {
    final response = await http.get(
      Uri.parse('https://api.example.com/users/$id'),
      headers: {'Authorization': 'Bearer $authToken'},
    );
    return User.fromJson(jsonDecode(response.body));
  }
}
```

ProxyProvider mit mehreren Abhängigkeiten

```
// ProxyProvider2 für 2 Abhängigkeiten
ProxyProvider2<AuthNotifier, SettingsNotifier, ApiService>(
  update: (_, auth, settings, __) {
    return ApiService(
      token: auth.token,
      baseUrl: settings.apiBaseUrl,
    );
  },
);

// Es gibt ProxyProvider0 bis ProxyProvider6
```

ChangeNotifierProxyProvider

Wenn der abhängige Provider selbst ein `ChangeNotifier` ist:

```
ChangeNotifierProxyProvider<AuthNotifier, UserProfileNotifier>(
  create: (_) => UserProfileNotifier(),
  update: (_, auth, profileNotifier) {
    // Update den bestehenden Notifier
    return profileNotifier!..updateAuth(auth.token);
  },
```

```
);
```

3.3.0.5 4. FutureProvider

Für asynchrone Initialisierung.

```
FutureProvider<Config>(
  create: (_) async {
    // Lade Konfiguration beim App-Start
    final response = await http.get(Uri.parse('https://api.example.com/config'));
    return Config.fromJson(jsonDecode(response.body));
  },
  initialData: Config.defaultConfig,  // Während des Ladens
  child: MyApp(),
);

// Verwendung
class MyWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final config = context.watch<Config>();
    return Text('API: ${config.apiUrl}');
  }
}
```

Mit Loading/Error Handling

```
FutureProvider<AsyncValue<Config>>(
  create: (_) async {
    try {
      final config = await loadConfig();
      return AsyncValue.data(config);
    } catch (e) {
      return AsyncValue.error(e);
    }
  },
  initialData: AsyncValue.loading(),
  child: MyApp(),
);

// AsyncValue selbst definieren (oder Riverpod verwenden)
sealed class AsyncValue<T> {}
class AsyncLoading<T> extends AsyncValue<T> {}
class AsyncData<T> extends AsyncValue<T> {
  final T data;
  AsyncData(this.data);
}
class AsyncError<T> extends AsyncValue<T> {
  final Object error;
  AsyncError(this.error);
}
```

3.3.0.6 5. StreamProvider

Für reaktive Datenquellen.

```
StreamProvider<User?>(
  create: (_) => FirebaseAuth.instance.authStateChanges(),
  initialData: null,
  child: MyApp(),
);

// Verwendung
class AuthWrapper extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final user = context.watch<User?>();

    if (user == null) {
      return LoginPage();
    }
    return HomePage();
  }
}
```

3.3.0.7 6. Provider.value

Für bereits existierende Instanzen (nicht create).

```
// ⚠ Achtung: dispose() wird NICHT aufgerufen!
Provider.value(
  value: existingService,
  child: MyWidget(),
);

// Sinnvoll z.B. für Route-Parameter
class DetailPage extends StatelessWidget {
  final Product product;

  DetailPage({required this.product});

  @override
  Widget build(BuildContext context) {
    return Provider.value(
      value: product,
      child: ProductDetails(),
    );
  }
}
```

3.3.0.8 7. Einführung in Riverpod

Riverpod ist die "nächste Generation" von Provider, vom selben Autor.

Hauptunterschiede

Feature	Provider	Riverpod
BuildContext nötig	Ja	Nein
Compile-Time Safety	Teilweise	Voll
Testing	Umständlich	Einfach
Auto-dispose	Manuell	Automatisch
Provider außerhalb von Widgets	Schwierig	Einfach

Riverpod Grundkonzept

```
// Provider-Definition (global)
final counterProvider = StateNotifierProvider<CounterNotifier, int>((ref) {
  return CounterNotifier();
});

class CounterNotifier extends StateNotifier<int> {
  CounterNotifier() : super(0);

  void increment() => state++;
}

// Verwendung
class CounterWidget extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final count = ref.watch(counterProvider);

    return Column(
      children: [
        Text('Count: $count'),
        ElevatedButton(
          onPressed: () => ref.read(counterProvider.notifier).increment(),
          child: Text('Increment'),
        ),
      ],
    );
  }
}
```

Wann Riverpod statt Provider?

- **Provider:** Einfachere Apps, schneller Einstieg
- **Riverpod:** Komplexere Apps, bessere Testbarkeit, keine BuildContext-Abhängigkeit

3.3.0.9 8. Best Practices

Provider-Organisation

```
// providers.dart - Zentrale Provider-Definition
class Providers {
  static List<SingleChildWidget> get all => [
    ChangeNotifierProvider(create: (_) => AuthNotifier()),
    ChangeNotifierProvider(create: (_) => ThemeNotifier()),
```

```
    ProxyProvider<AuthNotifier, ApiService>(
      update: (_, auth, __) => ApiService(auth.token),
    ),
  ];
}

// main.dart
void main() {
  runApp(
    MultiProvider(
      providers: Providers.all,
      child: MyApp(),
    ),
  );
}
```

Scope richtig setzen

```
// Global (App-weiter State)
MultiProvider(
  providers: [
    ChangeNotifierProvider(create: (_) => AuthNotifier()),
  ],
  child: MaterialApp(...),
);

// Lokal (Seiten-spezifischer State)
class ProductPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (_) => ProductDetailNotifier(),
      child: ProductContent(),
    );
  }
}
```

Selector vs Watch Entscheidung

```
// Verwende watch() wenn:
// - Du alle/die meisten Properties brauchst
// - Der Notifier selten Updates hat

// Verwende select() wenn:
// - Du nur 1-2 Properties brauchst
// - Der Notifier häufig Updates hat
// - Performance kritisch ist
```

3.3.0.10 Zusammenfassung

Konzept	Verwendung
`Selector`	Rebuild nur bei bestimmten Änderungen
`MultiProvider`	Mehrere Provider kombinieren
`ProxyProvider`	Provider mit Abhängigkeiten
`FutureProvider`	Asynchrone Initialisierung
`StreamProvider`	Reaktive Datenquellen
`Provider.value`	Bestehende Instanzen bereitstellen

Entscheidungshilfe: - Einfache App -> Provider reicht - Komplexe Dependencies -> ProxyProvider - Compile-Time Safety wichtig -> Riverpod - Team-Projekt -> Riverpod oder Bloc

3.3.1 Übung

3.3.1.1 Ziel

Eine App mit mehreren Providern und Abhängigkeiten zwischen ihnen erstellen.

3.3.1.2 Aufgabe 1: MultiProvider Setup (15 min)

Erstelle eine App mit folgenden Providern:

1. **ThemeNotifier** - Verwaltet Dark/Light Mode
2. **AuthNotifier** - Verwaltet User-Login-Status
3. **SettingsNotifier** - Verwaltet App-Einstellungen

```
// ThemeNotifier
class ThemeNotifier extends ChangeNotifier {
  ThemeMode _mode = ThemeMode.system;

  ThemeMode get mode => _mode;

  void setTheme(ThemeMode mode) {
    _mode = mode;
    notifyListeners();
  }

  void toggleDarkMode() {
    _mode = _mode == ThemeMode.dark ? ThemeMode.light : ThemeMode.dark;
    notifyListeners();
  }
}

// AuthNotifier
class AuthNotifier extends ChangeNotifier {
  String? _userId;
  String? _token;

  bool get isLoggedIn => _userId != null;
  String? get userId => _userId;
  String? get token => _token;

  void login(String userId, String token) {
```

```
    _userId = userId;
    _token = token;
    notifyListeners();
  }

  void logout() {
    _userId = null;
    _token = null;
    notifyListeners();
  }
}

// SettingsNotifier
class SettingsNotifier extends ChangeNotifier {
  bool _notificationsEnabled = true;
  String _language = 'de';

  bool get notificationsEnabled => _notificationsEnabled;
  String get language => _language;

  void setNotifications(bool enabled) {
    _notificationsEnabled = enabled;
    notifyListeners();
  }

  void setLanguage(String lang) {
    _language = lang;
    notifyListeners();
  }
}
```

Registriere alle drei mit `MultiProvider` in `main()`.

3.3.1.3 Aufgabe 2: ProxyProvider für API-Service (20 min)

Erstelle einen `ApiService`, der vom `AuthNotifier` abhängt:

```
class ApiService {
  final String? authToken;

  ApiService({this.authToken});

  // Simulierte API-Calls
  Future<Map<String, dynamic>> fetchUserData() async {
    if (authToken == null) {
      throw Exception('Not authenticated');
    }

    // Simuliere Netzwerk-Delay
    await Future.delayed(Duration(seconds: 1));

    return {
```

```
      'name': 'Max Mustermann',
      'email': 'max@example.com',
      'premium': true,
    };
  }

  Future<List<String>> fetchItems() async {
    await Future.delayed(Duration(milliseconds: 500));
    return ['Item 1', 'Item 2', 'Item 3'];
  }
}
```

Füge den `ApiService` als `ProxyProvider` hinzu, der bei Änderungen am Auth-Token aktualisiert wird.

3.3.1.4 Aufgabe 3: Selector für Performance (20 min)

Erstelle ein `UserStatsNotifier` mit vielen Properties:

```
class UserStatsNotifier extends ChangeNotifier {
  int _totalItems = 0;
  int _completedItems = 0;
  int _points = 0;
  String _level = 'Beginner';
  DateTime _lastActive = DateTime.now();

  int get totalItems => _totalItems;
  int get completedItems => _completedItems;
  int get points => _points;
  String get level => _level;
  DateTime get lastActive => _lastActive;

  // Computed
  double get completionRate =>
      _totalItems == 0 ? 0 : _completedItems / _totalItems;

  void addItem() {
    _totalItems++;
    notifyListeners();
  }

  void completeItem() {
    _completedItems++;
    _points += 10;
    _checkLevelUp();
    notifyListeners();
  }

  void _checkLevelUp() {
    if (_points >= 100) _level = 'Pro';
    if (_points >= 500) _level = 'Expert';
    if (_points >= 1000) _level = 'Master';
```

```
  }

  void updateLastActive() {
    _lastActive = DateTime.now();
    notifyListeners();
  }
}
```

Erstelle drei separate Widgets, die jeweils `Selector` verwenden:

1. **PointsDisplay** - Zeigt nur `points` und `level`
2. **ProgressDisplay** - Zeigt nur `completedItems` und `totalItems`
3. **LastActiveDisplay** - Zeigt nur `lastActive`

Füge Debug-Prints in jeden `builder` ein, um zu sehen, wann sie rebuilden:

```
Selector<UserStatsNotifier, int>(
  selector: (_, stats) => stats.points,
  builder: (_, points, __) {
    print('PointsDisplay rebuild');
    return Text('Points: $points');
  },
);
```

3.3.1.5 Aufgabe 4: Kombination - Settings-Seite (30 min)

Baue eine Settings-Seite, die alle Provider verwendet:

```
+---------------------------------+
| Einstellungen                   |
+---------------------------------+
|                                 |
| Konto                           |
| +-----------------------------+ |
| | Max Mustermann              | |
| | max@example.com             | |
| | [Abmelden]                  | |
| +-----------------------------+ |
|                                 |
| Darstellung                     |
| +-----------------------------+ |
| | Dark Mode          [Toggle] | |
| +-----------------------------+ |
|                                 |
| Benachrichtigungen              |
| +-----------------------------+ |
| | Push-Benachrichtigungen [[x]] | |
| +-----------------------------+ |
|                                 |
| Sprache                         |
| +-----------------------------+ |
| | Deutsch              [▼]   | |
| +-----------------------------+ |
```

```
|                                 |
| Statistiken                     |
| +-----------------------------+ |
| | Level: Pro                  | |
| | Punkte: 150                 | |
| | Fortschritt: 8/10 (80%)     | |
| +-----------------------------+ |
|                                 |
+---------------------------------+
```

Anforderungen: - Nutze `Consumer` oder `Selector` je nach Bedarf - “Abmelden” Button ruft `authNotifier.logout()` auf - Theme Toggle ändert zwischen Dark/Light Mode - Statistiken verwenden `Selector` für optimale Performance

3.3.1.6 Aufgabe 5: FutureProvider für Konfiguration (15 min)

Simuliere das Laden einer App-Konfiguration:

```
class AppConfig {
  final String apiBaseUrl;
  final int maxUploadSize;
  final bool maintenanceMode;

  AppConfig({
    required this.apiBaseUrl,
    required this.maxUploadSize,
    required this.maintenanceMode,
  });

  static Future<AppConfig> load() async {
    // Simuliere Laden vom Server
    await Future.delayed(Duration(seconds: 2));

    return AppConfig(
      apiBaseUrl: 'https://api.example.com',
      maxUploadSize: 10 * 1024 * 1024, // 10 MB
      maintenanceMode: false,
    );
  }

  static AppConfig get defaultConfig => AppConfig(
    apiBaseUrl: 'https://api.example.com',
    maxUploadSize: 5 * 1024 * 1024,
    maintenanceMode: false,
  );
}
```

1. Füge einen `FutureProvider<AppConfig>` hinzu
2. Zeige einen Loading-Spinner während des Ladens
3. Zeige die Config-Werte nach dem Laden

3.3.1.7 Bonus: Riverpod Vergleich

Implementiere einen einfachen Counter mit Riverpod und vergleiche die Syntax:

```
// pubspec.yaml
// flutter_riverpod: ^2.4.9

import 'package:flutter_riverpod/flutter_riverpod.dart';

// 1. Provider definieren
final counterProvider = StateProvider<int>((ref) => 0);

// 2. App mit ProviderScope wrappen
void main() {
  runApp(ProviderScope(child: MyApp()));
}

// 3. ConsumerWidget statt StatelessWidget
class CounterPage extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final count = ref.watch(counterProvider);

    return Scaffold(
      body: Center(child: Text('Count: $count')),
      floatingActionButton: FloatingActionButton(
        onPressed: () => ref.read(counterProvider.notifier).state++,
        child: Icon(Icons.add),
      ),
    );
  }
}
```

Notiere Unterschiede zu Provider.

3.3.1.8 Abgabe-Checkliste

- ☐ MultiProvider mit 3+ Providern konfiguriert
- ☐ ProxyProvider für ApiService implementiert
- ☐ Selector in mindestens 3 Widgets verwendet
- ☐ Debug-Prints zeigen korrektes Rebuild-Verhalten
- ☐ Settings-Seite funktioniert vollständig
- ☐ FutureProvider mit Loading-State
- ☐ (Bonus) Riverpod-Vergleich dokumentiert

3.3.2 Lösung

3.3.2.1 Projektstruktur

```
lib/
+-- main.dart
+-- models/
|   +-- app_config.dart
+-- providers/
```

```
|   +-- auth_notifier.dart
|   +-- theme_notifier.dart
|   +-- settings_notifier.dart
|   +-- user_stats_notifier.dart
|   +-- providers.dart
+-- services/
|   +-- api_service.dart
+-- pages/
    +-- settings_page.dart
```

3.3.2.2 models/app_config.dart

```
class AppConfig {
  final String apiBaseUrl;
  final int maxUploadSize;
  final bool maintenanceMode;

  AppConfig({
    required this.apiBaseUrl,
    required this.maxUploadSize,
    required this.maintenanceMode,
  });

  static Future<AppConfig> load() async {
    await Future.delayed(const Duration(seconds: 2));
    return AppConfig(
      apiBaseUrl: 'https://api.example.com',
      maxUploadSize: 10 * 1024 * 1024,
      maintenanceMode: false,
    );
  }

  static AppConfig get defaultConfig => AppConfig(
        apiBaseUrl: 'https://api.example.com',
        maxUploadSize: 5 * 1024 * 1024,
        maintenanceMode: false,
      );
}
```

3.3.2.3 providers/auth_notifier.dart

```
import 'package:flutter/foundation.dart';

class AuthNotifier extends ChangeNotifier {
  String? _userId;
  String? _token;
  String? _userName;
  String? _userEmail;

  bool get isLoggedIn => _userId != null;
```

```
  String? get userId => _userId;
  String? get token => _token;
  String? get userName => _userName;
  String? get userEmail => _userEmail;

  void login(String userId, String token,
      {String? name, String? email}) {
    _userId = userId;
    _token = token;
    _userName = name ?? 'Max Mustermann';
    _userEmail = email ?? 'max@example.com';
    notifyListeners();
  }

  void logout() {
    _userId = null;
    _token = null;
    _userName = null;
    _userEmail = null;
    notifyListeners();
  }
}
```

3.3.2.4 providers/theme_notifier.dart

```
import 'package:flutter/material.dart';

class ThemeNotifier extends ChangeNotifier {
  ThemeMode _mode = ThemeMode.system;

  ThemeMode get mode => _mode;
  bool get isDarkMode => _mode == ThemeMode.dark;

  void setTheme(ThemeMode mode) {
    if (_mode != mode) {
      _mode = mode;
      notifyListeners();
    }
  }

  void toggleDarkMode() {
    _mode = _mode == ThemeMode.dark ? ThemeMode.light : ThemeMode.dark;
    notifyListeners();
  }
}
```

3.3.2.5 providers/settings_notifier.dart

```
import 'package:flutter/foundation.dart';

class SettingsNotifier extends ChangeNotifier {
  bool _notificationsEnabled = true;
  String _language = 'de';

  bool get notificationsEnabled => _notificationsEnabled;
  String get language => _language;

  void setNotifications(bool enabled) {
    if (_notificationsEnabled != enabled) {
      _notificationsEnabled = enabled;
      notifyListeners();
    }
  }

  void setLanguage(String lang) {
    if (_language != lang) {
      _language = lang;
      notifyListeners();
    }
  }
}
```

3.3.2.6 providers/user_stats_notifier.dart

```
import 'package:flutter/foundation.dart';

class UserStatsNotifier extends ChangeNotifier {
  int _totalItems = 10;
  int _completedItems = 8;
  int _points = 150;
  String _level = 'Pro';
  DateTime _lastActive = DateTime.now();

  int get totalItems => _totalItems;
  int get completedItems => _completedItems;
  int get points => _points;
  String get level => _level;
  DateTime get lastActive => _lastActive;

  double get completionRate =>
      _totalItems == 0 ? 0 : _completedItems / _totalItems;

  void addItem() {
    _totalItems++;
    notifyListeners();
  }

  void completeItem() {
```

```
    if (_completedItems < _totalItems) {
      _completedItems++;
      _points += 10;
      _checkLevelUp();
      notifyListeners();
    }
  }

  void _checkLevelUp() {
    if (_points >= 1000) {
      _level = 'Master';
    } else if (_points >= 500) {
      _level = 'Expert';
    } else if (_points >= 100) {
      _level = 'Pro';
    } else {
      _level = 'Beginner';
    }
  }

  void updateLastActive() {
    _lastActive = DateTime.now();
    notifyListeners();
  }
}
```

3.3.2.7 services/api_service.dart

```
class ApiService {
  final String? authToken;

  ApiService({this.authToken});

  bool get isAuthenticated => authToken != null;

  Future<Map<String, dynamic>> fetchUserData() async {
    if (authToken == null) {
      throw Exception('Not authenticated');
    }

    await Future.delayed(const Duration(seconds: 1));

    return {
      'name': 'Max Mustermann',
      'email': 'max@example.com',
      'premium': true,
    };
  }

  Future<List<String>> fetchItems() async {
```

```
    await Future.delayed(const Duration(milliseconds: 500));
    return ['Item 1', 'Item 2', 'Item 3'];
  }
}
```

3.3.2.8 providers/providers.dart

```
import 'package:provider/provider.dart';
import 'package:provider/single_child_widget.dart';
import '../models/app_config.dart';
import '../services/api_service.dart';
import 'auth_notifier.dart';
import 'theme_notifier.dart';
import 'settings_notifier.dart';
import 'user_stats_notifier.dart';

class Providers {
  static List<SingleChildWidget> get all => [
        // 1. Theme Notifier
        ChangeNotifierProvider(create: (_) => ThemeNotifier()),

        // 2. Auth Notifier
        ChangeNotifierProvider(create: (_) {
          final auth = AuthNotifier();
          // Simuliere eingeloggten User
          auth.login('user123', 'token_abc',
              name: 'Max Mustermann', email: 'max@example.com');
          return auth;
        }),

        // 3. Settings Notifier
        ChangeNotifierProvider(create: (_) => SettingsNotifier()),

        // 4. User Stats Notifier
        ChangeNotifierProvider(create: (_) => UserStatsNotifier()),

        // 5. API Service (abhängig von Auth)
        ProxyProvider<AuthNotifier, ApiService>(
          update: (_, auth, __) => ApiService(authToken: auth.token),
        ),

        // 6. App Config (async laden)
        FutureProvider<AppConfig>(
          create: (_) => AppConfig.load(),
          initialData: AppConfig.defaultConfig,
        ),
      ];
}
```

3.3.2.9 main.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'providers/providers.dart';
import 'providers/theme_notifier.dart';
import 'pages/settings_page.dart';

void main() {
  runApp(
    MultiProvider(
      providers: Providers.all,
      child: const MyApp(),
    ),
  );
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return Consumer<ThemeNotifier>(
      builder: (context, themeNotifier, _) {
        return MaterialApp(
          title: 'Provider Advanced',
          themeMode: themeNotifier.mode,
          theme: ThemeData(
            colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
            useMaterial3: true,
          ),
          darkTheme: ThemeData(
            colorScheme: ColorScheme.fromSeed(
              seedColor: Colors.blue,
              brightness: Brightness.dark,
            ),
            useMaterial3: true,
          ),
          home: const SettingsPage(),
        );
      },
    );
  }
}
```

3.3.2.10 pages/settings_page.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/app_config.dart';
import '../providers/auth_notifier.dart';
```

```
import '../providers/theme_notifier.dart';
import '../providers/settings_notifier.dart';
import '../providers/user_stats_notifier.dart';

class SettingsPage extends StatelessWidget {
  const SettingsPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Einstellungen'),
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      ),
      body: ListView(
        padding: const EdgeInsets.all(16),
        children: const [
          _AccountSection(),
          SizedBox(height: 24),
          _AppearanceSection(),
          SizedBox(height: 24),
          _NotificationSection(),
          SizedBox(height: 24),
          _LanguageSection(),
          SizedBox(height: 24),
          _StatsSection(),
          SizedBox(height: 24),
          _ConfigSection(),
        ],
      ),
    );
  }
}

// Account Section
class _AccountSection extends StatelessWidget {
  const _AccountSection();

  @override
  Widget build(BuildContext context) {
    return Consumer<AuthNotifier>(
      builder: (context, auth, _) {
        print('AccountSection rebuild');

        if (!auth.isLoggedIn) {
          return Card(
            child: ListTile(
              title: const Text('Nicht angemeldet'),
              trailing: ElevatedButton(
                onPressed: () {
                  auth.login('user123', 'token_abc');
```

```
                },
                child: const Text('Anmelden'),
              ),
            ),
          );
        }

        return Card(
          child: Padding(
            padding: const EdgeInsets.all(16),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  'Konto',
                  style: Theme.of(context).textTheme.titleMedium,
                ),
                const SizedBox(height: 8),
                ListTile(
                  leading: const CircleAvatar(child: Icon(Icons.person)),
                  title: Text(auth.userName ?? 'Unbekannt'),
                  subtitle: Text(auth.userEmail ?? ''),
                ),
                Align(
                  alignment: Alignment.centerRight,
                  child: TextButton.icon(
                    onPressed: () => auth.logout(),
                    icon: const Icon(Icons.logout),
                    label: const Text('Abmelden'),
                  ),
                ),
              ],
            ),
          ),
        );
      },
    );
  }
}

// Appearance Section mit Selector
class _AppearanceSection extends StatelessWidget {
  const _AppearanceSection();

  @override
  Widget build(BuildContext context) {
    // Selector: Nur bei Mode-Änderung rebuilden
    final isDarkMode = context.select<ThemeNotifier, bool>(
      (notifier) => notifier.isDarkMode,
    );
```

```
    print('AppearanceSection rebuild');

    return Card(
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Darstellung',
              style: Theme.of(context).textTheme.titleMedium,
            ),
            SwitchListTile(
              title: const Text('Dark Mode'),
              value: isDarkMode,
              onChanged: (_) {
                context.read<ThemeNotifier>().toggleDarkMode();
              },
            ),
          ],
        ),
      ),
    );
  }
}

// Notification Section
class _NotificationSection extends StatelessWidget {
  const _NotificationSection();

  @override
  Widget build(BuildContext context) {
    final enabled = context.select<SettingsNotifier, bool>(
      (s) => s.notificationsEnabled,
    );

    print('NotificationSection rebuild');

    return Card(
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Benachrichtigungen',
              style: Theme.of(context).textTheme.titleMedium,
            ),
            SwitchListTile(
              title: const Text('Push-Benachrichtigungen'),
              value: enabled,
```

```
              onChanged: (value) {
                context.read<SettingsNotifier>().setNotifications(value);
              },
            ),
          ],
        ),
      ),
    );
  }
}

// Language Section
class _LanguageSection extends StatelessWidget {
  const _LanguageSection();

  @override
  Widget build(BuildContext context) {
    final language = context.select<SettingsNotifier, String>(
      (s) => s.language,
    );

    print('LanguageSection rebuild');

    return Card(
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Sprache',
              style: Theme.of(context).textTheme.titleMedium,
            ),
            const SizedBox(height: 8),
            DropdownButtonFormField<String>(
              value: language,
              decoration: const InputDecoration(
                border: OutlineInputBorder(),
              ),
              items: const [
                DropdownMenuItem(value: 'de', child: Text('Deutsch')),
                DropdownMenuItem(value: 'en', child: Text('English')),
                DropdownMenuItem(value: 'fr', child: Text('Français')),
              ],
              onChanged: (value) {
                if (value != null) {
                  context.read<SettingsNotifier>().setLanguage(value);
                }
              },
            ),
          ],
```

```
        ),
      ),
    );
  }
}

// Stats Section mit mehreren Selectors
class _StatsSection extends StatelessWidget {
  const _StatsSection();

  @override
  Widget build(BuildContext context) {
    print('StatsSection container rebuild');

    return Card(
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Statistiken',
              style: Theme.of(context).textTheme.titleMedium,
            ),
            const SizedBox(height: 8),
            const _PointsDisplay(),
            const _ProgressDisplay(),
            const SizedBox(height: 16),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                ElevatedButton(
                  onPressed: () {
                    context.read<UserStatsNotifier>().addItem();
                  },
                  child: const Text('+ Item'),
                ),
                ElevatedButton(
                  onPressed: () {
                    context.read<UserStatsNotifier>().completeItem();
                  },
                  child: const Text('+ Erledigt'),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

```
class _PointsDisplay extends StatelessWidget {
  const _PointsDisplay();

  @override
  Widget build(BuildContext context) {
    // Nur Points und Level selektieren
    return Selector<UserStatsNotifier, (int, String)>(
      selector: (_, stats) => (stats.points, stats.level),
      builder: (_, data, __) {
        final (points, level) = data;
        print('PointsDisplay rebuild: $points points, $level');

        return ListTile(
          leading: const Icon(Icons.star),
          title: Text('Level: $level'),
          subtitle: Text('Punkte: $points'),
        );
      },
    );
  }
}

class _ProgressDisplay extends StatelessWidget {
  const _ProgressDisplay();

  @override
  Widget build(BuildContext context) {
    // Nur Progress-relevante Daten selektieren
    return Selector<UserStatsNotifier, (int, int)>(
      selector: (_, stats) => (stats.completedItems, stats.totalItems),
      builder: (_, data, __) {
        final (completed, total) = data;
        final percentage = total == 0 ? 0 : (completed / total * 100).round();
        print('ProgressDisplay rebuild: $completed/$total');

        return Column(
          children: [
            ListTile(
              leading: const Icon(Icons.check_circle),
              title: Text('Fortschritt: $completed/$total ($percentage%)'),
            ),
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 16),
              child: LinearProgressIndicator(
                value: total == 0 ? 0 : completed / total,
              ),
            ),
          ],
        );
      },
```

```
    );
  }
}

// Config Section mit FutureProvider
class _ConfigSection extends StatelessWidget {
  const _ConfigSection();

  @override
  Widget build(BuildContext context) {
    final config = context.watch<AppConfig>();

    print('ConfigSection rebuild');

    return Card(
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Konfiguration',
              style: Theme.of(context).textTheme.titleMedium,
            ),
            const SizedBox(height: 8),
            ListTile(
              title: const Text('API URL'),
              subtitle: Text(config.apiBaseUrl),
            ),
            ListTile(
              title: const Text('Max Upload'),
              subtitle: Text(
                '${(config.maxUploadSize / 1024 / 1024).toStringAsFixed(0)} MB',
              ),
            ),
            ListTile(
              title: const Text('Wartungsmodus'),
              subtitle: Text(config.maintenanceMode ? 'Aktiv' : 'Inaktiv'),
            ),
          ],
        ),
      ),
    );
  }
}
```

3.3.2.11 Debug-Output Analyse

Beim Ändern einzelner Werte werden nur die relevanten Widgets rebuilt:

```
// Bei Dark Mode Toggle:
```

```
AppearanceSection rebuild

// Bei Notification Toggle:
NotificationSection rebuild

// Bei completeItem():
PointsDisplay rebuild: 160 points, Pro
ProgressDisplay rebuild: 9/10

// Bei addItem():
ProgressDisplay rebuild: 9/11
// PointsDisplay wird NICHT rebuilt!
```

3.3.2.12 Bonus: Riverpod Vergleich

```
// Riverpod Version - providers.dart
import 'package:flutter_riverpod/flutter_riverpod.dart';

// Einfacher State
final counterProvider = StateProvider<int>((ref) => 0);

// Notifier (wie ChangeNotifier)
final userStatsProvider =
    StateNotifierProvider<UserStatsNotifier, UserStats>((ref) {
  return UserStatsNotifier();
});

// Abhängiger Provider
final apiServiceProvider = Provider<ApiService>((ref) {
  final auth = ref.watch(authProvider);
  return ApiService(authToken: auth.token);
});

// Usage in Widget
class MyWidget extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    // watch = context.watch
    final count = ref.watch(counterProvider);

    return ElevatedButton(
      // read = context.read
      onPressed: () => ref.read(counterProvider.notifier).state++,
      child: Text('$count'),
    );
  }
}
```

Hauptunterschiede: 1. Kein `BuildContext` nötig 2. Provider sind global definiert 3. `ref.watch` statt `context.watch` 4. Automatisches Dispose 5. Bessere Compile-Time Checks

3.3.3 Ressourcen

3.3.3.1 Offizielle Dokumentation

- Provider Package - Advanced Usage
- ProxyProvider
- Selector
- MultiProvider

3.3.3.2 Riverpod

- Riverpod Package
- Riverpod Dokumentation
- Provider to Riverpod Migration Guide
- Riverpod Generator

3.3.3.3 Tutorials

- Advanced Provider Patterns
- ProxyProvider Deep Dive
- Selector Performance Optimization

3.3.3.4 Videos

- MultiProvider & ProxyProvider
- Riverpod Complete Course
- Provider vs Riverpod

3.3.3.5 Best Practices

- Provider Architecture
- Testing with Provider
- Provider Anti-Patterns

3.3.3.6 Zum Vertiefen

- Freezed Package - Immutable State Classes
- State Notifier - Separates State vom Notifier
- Flutter Hooks - React-artige Hooks für Flutter

3.4 Einheit 3.4: HTTP Requests

3.4.0.1 Lernziele

Nach dieser Einheit kannst du: - Das `http` Package für Netzwerk-Requests verwenden - GET, POST, PUT, DELETE Requests durchführen - Headers und Query-Parameter setzen - Timeouts und Fehlerbehandlung implementieren - Response-Daten verarbeiten

3.4.0.2 1. Setup

Installation

```
# pubspec.yaml
dependencies:
  http: ^1.2.0
```

```
flutter pub get
```

Import

```
import 'package:http/http.dart' as http;
```

Der `as http` Alias verhindert Namenskonflikte.

3.4.0.3 2. GET Requests

Einfacher GET Request

```
Future<void> fetchData() async {
  final url = Uri.parse('https://jsonplaceholder.typicode.com/posts/1');

  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      print('Success: ${response.body}');
    } else {
      print('Error: ${response.statusCode}');
    }
  } catch (e) {
    print('Network error: $e');
  }
}
```

Response-Objekt

```
final response = await http.get(url);

// Wichtige Properties
response.statusCode;     // 200, 404, 500, etc.
response.body;           // Response als String
response.bodyBytes;      // Response als Bytes (für Bilder, etc.)
response.headers;        // Map<String, String>
response.contentLength;  // Länge in Bytes
response.isRedirect;     // true wenn Redirect
response.reasonPhrase;   // "OK", "Not Found", etc.
```

Mit Query-Parametern

```
// Option 1: In URL einbauen
final url = Uri.parse(
  'https://api.example.com/search?query=flutter&page=1',
);

// Option 2: Uri.https mit queryParameters (empfohlen)
final url = Uri.https(
  'api.example.com',      // Host (ohne https://)
  '/search',              // Path
  {                       // Query Parameters
```

```
    'query': 'flutter',
    'page': '1',
    'sort': 'date',
  },
);

// Für HTTP (nicht HTTPS):
final url = Uri.http('localhost:8080', '/api/items', {'limit': '10'});
```

Mit Headers

```
final response = await http.get(
  url,
  headers: {
    'Authorization': 'Bearer $token',
    'Accept': 'application/json',
    'X-Custom-Header': 'value',
  },
);
```

3.4.0.4 3. POST Requests

JSON senden

```
import 'dart:convert';

Future<void> createPost() async {
  final url = Uri.parse('https://jsonplaceholder.typicode.com/posts');

  final response = await http.post(
    url,
    headers: {
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode({
      'title': 'Mein Post',
      'body': 'Inhalt des Posts',
      'userId': 1,
    }),
  );

  if (response.statusCode == 201) {
    print('Created: ${response.body}');
  } else {
    throw Exception('Failed to create post: ${response.statusCode}');
  }
}
```

Form Data senden

```
final response = await http.post(
  url,
```

```
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded',
  },
  body: {
    'username': 'max',
    'password': 'secret',
  },
  // body wird automatisch URL-encoded
);
```

3.4.0.5 4. PUT und DELETE

PUT (Update)

```
Future<void> updatePost(int id, String title) async {
  final url = Uri.parse('https://jsonplaceholder.typicode.com/posts/$id');

  final response = await http.put(
    url,
    headers: {'Content-Type': 'application/json'},
    body: jsonEncode({
      'id': id,
      'title': title,
      'body': 'Updated content',
      'userId': 1,
    }),
  );

  if (response.statusCode == 200) {
    print('Updated successfully');
  }
}
```

PATCH (Partial Update)

```
final response = await http.patch(
  url,
  headers: {'Content-Type': 'application/json'},
  body: jsonEncode({
    'title': 'Nur Titel ändern',
  }),
);
```

DELETE

```
Future<void> deletePost(int id) async {
  final url = Uri.parse('https://jsonplaceholder.typicode.com/posts/$id');

  final response = await http.delete(url);

  if (response.statusCode == 200 || response.statusCode == 204) {
    print('Deleted successfully');
```

```
  }
}
```

3.4.0.6 5. Timeouts

Request-Timeout

```
try {
  final response = await http
      .get(url)
      .timeout(const Duration(seconds: 10));

  print(response.body);
} on TimeoutException {
  print('Request timed out');
}
```

Mit http.Client für mehr Kontrolle

```
final client = http.Client();

try {
  final response = await client
      .get(url)
      .timeout(const Duration(seconds: 10));

  print(response.body);
} finally {
  client.close();  // Wichtig: Ressourcen freigeben!
}
```

3.4.0.7 6. Fehlerbehandlung

Umfassende Error-Handling

```
import 'dart:async';
import 'dart:io';

Future<String> fetchData(String endpoint) async {
  final url = Uri.parse('https://api.example.com/$endpoint');

  try {
    final response = await http
        .get(url)
        .timeout(const Duration(seconds: 10));

    switch (response.statusCode) {
      case 200:
        return response.body;
      case 400:
        throw BadRequestException(response.body);
      case 401:
```

```
        throw UnauthorizedException();
      case 403:
        throw ForbiddenException();
      case 404:
        throw NotFoundException(endpoint);
      case 500:
        throw ServerException();
      default:
        throw HttpException(
          'Unexpected status code: ${response.statusCode}',
        );
    }
  } on SocketException {
    throw NoInternetException();
  } on TimeoutException {
    throw RequestTimeoutException();
  } on FormatException {
    throw InvalidResponseException();
  }
}

// Custom Exceptions
class BadRequestException implements Exception {
  final String message;
  BadRequestException(this.message);
}

class UnauthorizedException implements Exception {}

class ForbiddenException implements Exception {}

class NotFoundException implements Exception {
  final String resource;
  NotFoundException(this.resource);
}

class ServerException implements Exception {}

class NoInternetException implements Exception {}

class RequestTimeoutException implements Exception {}

class InvalidResponseException implements Exception {}
```

3.4.0.8 7. API Service Klasse

Strukturierter Ansatz

```
import 'dart:convert';
import 'dart:io';
import 'package:http/http.dart' as http;
```

```
class ApiService {
  final String baseUrl;
  final http.Client _client;
  String? _authToken;

  ApiService({
    required this.baseUrl,
    http.Client? client,
  }) : _client = client ?? http.Client();

  void setAuthToken(String token) {
    _authToken = token;
  }

  void clearAuthToken() {
    _authToken = null;
  }

  Map<String, String> get _headers => {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
        if (_authToken != null) 'Authorization': 'Bearer $_authToken',
      };

  Future<dynamic> get(String endpoint,
      {Map<String, String>? queryParams}) async {
    final uri = Uri.parse('$baseUrl$endpoint').replace(
      queryParameters: queryParams,
    );

    final response = await _request(() => _client.get(uri, headers: _headers));
    return jsonDecode(response.body);
  }

  Future<dynamic> post(String endpoint, Map<String, dynamic> data) async {
    final uri = Uri.parse('$baseUrl$endpoint');

    final response = await _request(() => _client.post(
          uri,
          headers: _headers,
          body: jsonEncode(data),
        ));

    return jsonDecode(response.body);
  }

  Future<dynamic> put(String endpoint, Map<String, dynamic> data) async {
    final uri = Uri.parse('$baseUrl$endpoint');

    final response = await _request(() => _client.put(
```

```
          uri,
          headers: _headers,
          body: jsonEncode(data),
        ));

    return jsonDecode(response.body);
  }

  Future<void> delete(String endpoint) async {
    final uri = Uri.parse('$baseUrl$endpoint');
    await _request(() => _client.delete(uri, headers: _headers));
  }

  Future<http.Response> _request(
    Future<http.Response> Function() request,
  ) async {
    try {
      final response = await request().timeout(
        const Duration(seconds: 30),
      );

      _checkResponse(response);
      return response;
    } on SocketException {
      throw ApiException('No internet connection');
    } on TimeoutException {
      throw ApiException('Request timed out');
    }
  }

  void _checkResponse(http.Response response) {
    if (response.statusCode >= 200 && response.statusCode < 300) {
      return;
    }

    final body = response.body.isNotEmpty
        ? jsonDecode(response.body)
        : {'message': 'Unknown error'};

    throw ApiException(
      body['message'] ?? 'Request failed',
      statusCode: response.statusCode,
    );
  }

  void dispose() {
    _client.close();
  }
}

class ApiException implements Exception {
```

```
  final String message;
  final int? statusCode;

  ApiException(this.message, {this.statusCode});

  @override
  String toString() => 'ApiException: $message (Status: $statusCode)';
}
```

Verwendung

```
final api = ApiService(baseUrl: 'https://api.example.com');

// Auth setzen
api.setAuthToken('my-token');

// GET
final posts = await api.get('/posts', queryParams: {'limit': '10'});

// POST
final newPost = await api.post('/posts', {
  'title': 'Hello',
  'body': 'World',
});

// PUT
await api.put('/posts/1', {'title': 'Updated'});

// DELETE
await api.delete('/posts/1');

// Cleanup
api.dispose();
```

3.4.0.9 8. Internet-Verbindung prüfen

Mit connectivity_plus Package

```
dependencies:
  connectivity_plus: ^5.0.2
```

```
import 'package:connectivity_plus/connectivity_plus.dart';

Future<bool> hasInternetConnection() async {
  final connectivityResult = await Connectivity().checkConnectivity();
  return connectivityResult != ConnectivityResult.none;
}

// Stream für Verbindungsänderungen
Connectivity().onConnectivityChanged.listen((result) {
  if (result == ConnectivityResult.none) {
    print('Offline');
```

```
  } else {
    print('Online: $result');
  }
});
```

3.4.0.10 9. Android/iOS Konfiguration

Android: Internet-Permission

```
<!-- android/app/src/main/AndroidManifest.xml -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.INTERNET"/>
    <!-- ... -->
</manifest>
```

iOS: HTTP erlauben (falls nötig)

```
<!-- ios/Runner/Info.plist -->
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

Hinweis: Für Produktion besser nur spezifische Domains erlauben.

3.4.0.11 Zusammenfassung

Methode	HTTP Verb	Verwendung
`http.get()`	GET	Daten abrufen
`http.post()`	POST	Neue Ressource erstellen
`http.put()`	PUT	Ressource komplett ersetzen
`http.patch()`	PATCH	Ressource teilweise aktualisieren
`http.delete()`	DELETE	Ressource löschen

Best Practices: 1. Immer Timeouts setzen 2. Fehler sauber behandeln 3. `http.Client` für mehrere Requests verwenden 4. API-Logik in Service-Klasse kapseln 5. Internet-Verbindung prüfen vor Requests

3.4.1 Übung

3.4.1.1 Ziel

Eine App erstellen, die Daten von einer REST API abruft und anzeigt.

3.4.1.2 Aufgabe 1: Setup & GET Request (20 min)

1. Erstelle ein neues Flutter-Projekt
2. Füge das `http` Package hinzu
3. Erstelle eine einfache Funktion, die Posts von JSONPlaceholder abruft:

```
// https://jsonplaceholder.typicode.com/posts
```

4. Zeige die Response im Terminal an
5. Behandle mögliche Fehler

Tipp: Vergiss nicht die Internet-Permission in `AndroidManifest.xml`!

3.4.1.3 Aufgabe 2: API Service Klasse (25 min)

Erstelle eine `PostService`-Klasse mit folgenden Methoden:

```
class PostService {
  static const baseUrl = 'https://jsonplaceholder.typicode.com';

  // Alle Posts abrufen
  Future<List<Post>> fetchPosts() async { ... }

  // Einzelnen Post abrufen
  Future<Post> fetchPost(int id) async { ... }

  // Neuen Post erstellen
  Future<Post> createPost(String title, String body, int userId) async { ... }

  // Post aktualisieren
  Future<Post> updatePost(int id, String title, String body) async { ... }

  // Post löschen
  Future<void> deletePost(int id) async { ... }
}
```

Das `Post`-Modell:

```
class Post {
  final int id;
  final int userId;
  final String title;
  final String body;

  Post({
    required this.id,
    required this.userId,
    required this.title,
    required this.body,
  });

  // fromJson und toJson implementieren
}
```

3.4.1.4 Aufgabe 3: Posts anzeigen (25 min)

Erstelle eine `PostsPage`, die alle Posts anzeigt:

1. Zeige einen Loading-Indicator während des Ladens

2. Zeige eine Fehlermeldung bei Problemen
3. Zeige die Posts in einer ListView
4. Implementiere Pull-to-Refresh

```
+---------------------------------+
| Posts               [Refresh]  |
+---------------------------------+
| +-----------------------------+ |
| | Post Title 1                | |
| | Post body excerpt...        | |
| +-----------------------------+ |
| +-----------------------------+ |
| | Post Title 2                | |
| | Post body excerpt...        | |
| +-----------------------------+ |
| ...                             |
+---------------------------------+
```

3.4.1.5 Aufgabe 4: Post-Details & CRUD (30 min)

Erweitere die App:

1. **Detail-Ansicht:** Tippe auf einen Post -> zeige Details
2. **Erstellen:** FAB -> Dialog zum Erstellen eines neuen Posts
3. **Löschen:** Swipe-to-Delete auf Posts
4. **Aktualisieren:** Edit-Button in der Detail-Ansicht

Hinweis: JSONPlaceholder simuliert nur die Operationen - die Daten werden nicht wirklich gespeichert.

3.4.1.6 Aufgabe 5: Error Handling (20 min)

Implementiere robuste Fehlerbehandlung:

1. **Timeout:** Request bricht nach 10 Sekunden ab
2. **Keine Verbindung:** Zeige “Keine Internetverbindung”
3. **Server-Fehler:** Zeige “Server nicht erreichbar”
4. **Retry-Button:** Ermögliche erneuten Versuch

```
// Custom Exceptions
class NetworkException implements Exception {
  final String message;
  NetworkException(this.message);
}

class ServerException implements Exception {
  final int statusCode;
  ServerException(this.statusCode);
}
```

3.4.1.7 Aufgabe 6: Verständnisfragen

1. Was ist der Unterschied zwischen `Uri.parse()` und `Uri.https()`?
2. Warum sollte man `http.Client()` verwenden statt direkt `http.get()`?

3. Was bedeutet der Statuscode 201 vs 200?
4. Warum wird `as http` beim Import verwendet?
5. Was passiert, wenn man `client.close()` nicht aufruft?

3.4.1.8 Bonus: User-Posts laden

Erweitere die App um eine User-Ansicht:

1. Lade User von `https://jsonplaceholder.typicode.com/users`
2. Zeige eine User-Liste
3. Tippe auf User -> zeige nur Posts dieses Users
4. Nutze Query-Parameter: `/posts?userId=1`

```
class User {
  final int id;
  final String name;
  final String email;
  final String username;

  // ...
}
```

3.4.1.9 Abgabe-Checkliste

- ☐ http Package installiert
- ☐ GET Request funktioniert
- ☐ PostService mit allen CRUD-Methoden
- ☐ Posts werden in ListView angezeigt
- ☐ Loading-State wird angezeigt
- ☐ Error-State wird angezeigt
- ☐ Pull-to-Refresh funktioniert
- ☐ Timeout implementiert
- ☐ Verständnisfragen beantwortet

3.4.2 Lösung

3.4.2.1 Projektstruktur

```
lib/
+-- main.dart
+-- models/
|   +-- post.dart
+-- services/
|   +-- post_service.dart
+-- pages/
    +-- posts_page.dart
    +-- post_detail_page.dart
```

3.4.2.2 models/post.dart

```
class Post {
  final int id;
```

```
  final int userId;
  final String title;
  final String body;

  Post({
    required this.id,
    required this.userId,
    required this.title,
    required this.body,
  });

  factory Post.fromJson(Map<String, dynamic> json) {
    return Post(
      id: json['id'] as int,
      userId: json['userId'] as int,
      title: json['title'] as String,
      body: json['body'] as String,
    );
  }

  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'userId': userId,
      'title': title,
      'body': body,
    };
  }

  Post copyWith({
    int? id,
    int? userId,
    String? title,
    String? body,
  }) {
    return Post(
      id: id ?? this.id,
      userId: userId ?? this.userId,
      title: title ?? this.title,
      body: body ?? this.body,
    );
  }
}
```

3.4.2.3 services/post_service.dart

```
import 'dart:async';
import 'dart:convert';
import 'dart:io';
import 'package:http/http.dart' as http;
```

```
import '../models/post.dart';

class PostService {
  static const String baseUrl = 'https://jsonplaceholder.typicode.com';
  static const Duration timeout = Duration(seconds: 10);

  final http.Client _client;

  PostService({http.Client? client}) : _client = client ?? http.Client();

  Future<List<Post>> fetchPosts({int? userId}) async {
    try {
      final uri = userId != null
          ? Uri.parse('$baseUrl/posts?userId=$userId')
          : Uri.parse('$baseUrl/posts');

      final response = await _client.get(uri).timeout(timeout);

      _checkResponse(response);

      final List<dynamic> jsonList = jsonDecode(response.body);
      return jsonList.map((json) => Post.fromJson(json)).toList();
    } on SocketException {
      throw NetworkException('Keine Internetverbindung');
    } on TimeoutException {
      throw NetworkException('Zeitüberschreitung - Server antwortet nicht');
    }
  }

  Future<Post> fetchPost(int id) async {
    try {
      final uri = Uri.parse('$baseUrl/posts/$id');
      final response = await _client.get(uri).timeout(timeout);

      _checkResponse(response);

      return Post.fromJson(jsonDecode(response.body));
    } on SocketException {
      throw NetworkException('Keine Internetverbindung');
    } on TimeoutException {
      throw NetworkException('Zeitüberschreitung');
    }
  }

  Future<Post> createPost({
    required String title,
    required String body,
    required int userId,
  }) async {
    try {
      final uri = Uri.parse('$baseUrl/posts');
```

```
    final response = await _client
        .post(
          uri,
          headers: {'Content-Type': 'application/json; charset=UTF-8'},
          body: jsonEncode({
            'title': title,
            'body': body,
            'userId': userId,
          }),
        )
        .timeout(timeout);

    if (response.statusCode != 201) {
      throw ServerException(response.statusCode);
    }

    return Post.fromJson(jsonDecode(response.body));
  } on SocketException {
    throw NetworkException('Keine Internetverbindung');
  } on TimeoutException {
    throw NetworkException('Zeitüberschreitung');
  }
}

Future<Post> updatePost(int id, {String? title, String? body}) async {
  try {
    final uri = Uri.parse('$baseUrl/posts/$id');
    final response = await _client
        .patch(
          uri,
          headers: {'Content-Type': 'application/json; charset=UTF-8'},
          body: jsonEncode({
            if (title != null) 'title': title,
            if (body != null) 'body': body,
          }),
        )
        .timeout(timeout);

    _checkResponse(response);

    return Post.fromJson(jsonDecode(response.body));
  } on SocketException {
    throw NetworkException('Keine Internetverbindung');
  } on TimeoutException {
    throw NetworkException('Zeitüberschreitung');
  }
}

Future<void> deletePost(int id) async {
  try {
    final uri = Uri.parse('$baseUrl/posts/$id');
```

```
      final response = await _client.delete(uri).timeout(timeout);

      if (response.statusCode != 200 && response.statusCode != 204) {
        throw ServerException(response.statusCode);
      }
    } on SocketException {
      throw NetworkException('Keine Internetverbindung');
    } on TimeoutException {
      throw NetworkException('Zeitüberschreitung');
    }
  }

  void _checkResponse(http.Response response) {
    if (response.statusCode == 200) return;

    switch (response.statusCode) {
      case 400:
        throw ServerException(400, message: 'Ungültige Anfrage');
      case 401:
        throw ServerException(401, message: 'Nicht autorisiert');
      case 404:
        throw ServerException(404, message: 'Nicht gefunden');
      case 500:
        throw ServerException(500, message: 'Server-Fehler');
      default:
        throw ServerException(response.statusCode);
    }
  }

  void dispose() {
    _client.close();
  }
}

// Custom Exceptions
class NetworkException implements Exception {
  final String message;
  NetworkException(this.message);

  @override
  String toString() => message;
}

class ServerException implements Exception {
  final int statusCode;
  final String? message;

  ServerException(this.statusCode, {this.message});

  @override
  String toString() =>
```

```
      message ?? 'Server-Fehler (Status: $statusCode)';
}
```

3.4.2.4 pages/posts_page.dart

```
import 'package:flutter/material.dart';
import '../models/post.dart';
import '../services/post_service.dart';
import 'post_detail_page.dart';

class PostsPage extends StatefulWidget {
  const PostsPage({super.key});

  @override
  State<PostsPage> createState() => _PostsPageState();
}

class _PostsPageState extends State<PostsPage> {
  final PostService _postService = PostService();

  List<Post>? _posts;
  String? _error;
  bool _isLoading = true;

  @override
  void initState() {
    super.initState();
    _loadPosts();
  }

  @override
  void dispose() {
    _postService.dispose();
    super.dispose();
  }

  Future<void> _loadPosts() async {
    setState(() {
      _isLoading = true;
      _error = null;
    });

    try {
      final posts = await _postService.fetchPosts();
      setState(() {
        _posts = posts;
        _isLoading = false;
      });
    } on NetworkException catch (e) {
      setState(() {
```

```
      _error = e.message;
      _isLoading = false;
    });
  } on ServerException catch (e) {
    setState(() {
      _error = e.toString();
      _isLoading = false;
    });
  } catch (e) {
    setState(() {
      _error = 'Unbekannter Fehler: $e';
      _isLoading = false;
    });
  }
}

Future<void> _deletePost(Post post) async {
  try {
    await _postService.deletePost(post.id);

    setState(() {
      _posts?.removeWhere((p) => p.id == post.id);
    });

    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Post "${post.title}" gelöscht')),
      );
    }
  } catch (e) {
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Fehler beim Löschen: $e'),
          backgroundColor: Colors.red,
        ),
      );
    }
  }
}

Future<void> _showCreateDialog() async {
  final titleController = TextEditingController();
  final bodyController = TextEditingController();

  final result = await showDialog<bool>(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text('Neuer Post'),
      content: Column(
        mainAxisSize: MainAxisSize.min,
```

```
        children: [
          TextField(
            controller: titleController,
            decoration: const InputDecoration(labelText: 'Titel'),
          ),
          const SizedBox(height: 16),
          TextField(
            controller: bodyController,
            decoration: const InputDecoration(labelText: 'Inhalt'),
            maxLines: 3,
          ),
        ],
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context, false),
          child: const Text('Abbrechen'),
        ),
        ElevatedButton(
          onPressed: () => Navigator.pop(context, true),
          child: const Text('Erstellen'),
        ),
      ],
    ),
  );

  if (result == true && titleController.text.isNotEmpty) {
    try {
      final newPost = await _postService.createPost(
        title: titleController.text,
        body: bodyController.text,
        userId: 1,
      );

      setState(() {
        _posts?.insert(0, newPost);
      });

      if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Post erstellt')),
        );
      }
    } catch (e) {
      if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text('Fehler: $e'),
            backgroundColor: Colors.red,
          ),
        );
```

```
      }
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Posts'),
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      actions: [
        IconButton(
          icon: const Icon(Icons.refresh),
          onPressed: _loadPosts,
        ),
      ],
    ),
    body: _buildBody(),
    floatingActionButton: FloatingActionButton(
      onPressed: _showCreateDialog,
      child: const Icon(Icons.add),
    ),
  );
}

Widget _buildBody() {
  if (_isLoading) {
    return const Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          CircularProgressIndicator(),
          SizedBox(height: 16),
          Text('Lade Posts...'),
        ],
      ),
    );
  }

  if (_error != null) {
    return Center(
      child: Padding(
        padding: const EdgeInsets.all(24),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Icon(
              Icons.error_outline,
              size: 64,
              color: Colors.red,
```

```
            ),
            const SizedBox(height: 16),
            Text(
              _error!,
              textAlign: TextAlign.center,
              style: const TextStyle(fontSize: 16),
            ),
            const SizedBox(height: 24),
            ElevatedButton.icon(
              onPressed: _loadPosts,
              icon: const Icon(Icons.refresh),
              label: const Text('Erneut versuchen'),
            ),
          ],
        ),
      ),
    );
  }

  if (_posts == null || _posts!.isEmpty) {
    return const Center(
      child: Text('Keine Posts vorhanden'),
    );
  }

  return RefreshIndicator(
    onRefresh: _loadPosts,
    child: ListView.builder(
      padding: const EdgeInsets.only(bottom: 80),
      itemCount: _posts!.length,
      itemBuilder: (context, index) {
        final post = _posts![index];
        return _PostCard(
          post: post,
          onTap: () => _navigateToDetail(post),
          onDelete: () => _deletePost(post),
        );
      },
    ),
  );
}

void _navigateToDetail(Post post) {
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (_) => PostDetailPage(
        post: post,
        onUpdate: (updatedPost) {
          setState(() {
            final index = _posts?.indexWhere((p) => p.id == updatedPost.id);
```

```
              if (index != null && index != -1) {
                _posts![index] = updatedPost;
              }
            });
          },
        ),
      ),
    );
  }
}

class _PostCard extends StatelessWidget {
  final Post post;
  final VoidCallback onTap;
  final VoidCallback onDelete;

  const _PostCard({
    required this.post,
    required this.onTap,
    required this.onDelete,
  });

  @override
  Widget build(BuildContext context) {
    return Dismissible(
      key: ValueKey(post.id),
      direction: DismissDirection.endToStart,
      background: Container(
        color: Colors.red,
        alignment: Alignment.centerRight,
        padding: const EdgeInsets.only(right: 16),
        child: const Icon(Icons.delete, color: Colors.white),
      ),
      confirmDismiss: (_) async {
        return await showDialog<bool>(
          context: context,
          builder: (context) => AlertDialog(
            title: const Text('Post löschen?'),
            content: Text('Möchtest du "${post.title}" wirklich löschen?'),
            actions: [
              TextButton(
                onPressed: () => Navigator.pop(context, false),
                child: const Text('Abbrechen'),
              ),
              TextButton(
                onPressed: () => Navigator.pop(context, true),
                style: TextButton.styleFrom(foregroundColor: Colors.red),
                child: const Text('Löschen'),
              ),
            ],
          ),
```

```
        );
      },
      onDismissed: (_) => onDelete(),
      child: Card(
        margin: const EdgeInsets.symmetric(horizontal: 8, vertical: 4),
        child: ListTile(
          title: Text(
            post.title,
            maxLines: 1,
            overflow: TextOverflow.ellipsis,
          ),
          subtitle: Text(
            post.body,
            maxLines: 2,
            overflow: TextOverflow.ellipsis,
          ),
          trailing: const Icon(Icons.chevron_right),
          onTap: onTap,
        ),
      ),
    );
  }
}
```

3.4.2.5 pages/post_detail_page.dart

```
import 'package:flutter/material.dart';
import '../models/post.dart';
import '../services/post_service.dart';

class PostDetailPage extends StatefulWidget {
  final Post post;
  final void Function(Post)? onUpdate;

  const PostDetailPage({
    super.key,
    required this.post,
    this.onUpdate,
  });

  @override
  State<PostDetailPage> createState() => _PostDetailPageState();
}

class _PostDetailPageState extends State<PostDetailPage> {
  final PostService _postService = PostService();
  late Post _post;
  bool _isEditing = false;

  late TextEditingController _titleController;
```

```
late TextEditingController _bodyController;

@override
void initState() {
  super.initState();
  _post = widget.post;
  _titleController = TextEditingController(text: _post.title);
  _bodyController = TextEditingController(text: _post.body);
}

@override
void dispose() {
  _titleController.dispose();
  _bodyController.dispose();
  _postService.dispose();
  super.dispose();
}

Future<void> _saveChanges() async {
  try {
    final updatedPost = await _postService.updatePost(
      _post.id,
      title: _titleController.text,
      body: _bodyController.text,
    );

    setState(() {
      _post = _post.copyWith(
        title: _titleController.text,
        body: _bodyController.text,
      );
      _isEditing = false;
    });

    widget.onUpdate?.call(_post);

    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Post aktualisiert')),
      );
    }
  } catch (e) {
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Fehler: $e'),
          backgroundColor: Colors.red,
        ),
      );
    }
  }
```

```
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(_isEditing ? 'Bearbeiten' : 'Post Details'),
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        actions: [
          if (_isEditing)
            IconButton(
              icon: const Icon(Icons.close),
              onPressed: () {
                _titleController.text = _post.title;
                _bodyController.text = _post.body;
                setState(() => _isEditing = false);
              },
            )
          else
            IconButton(
              icon: const Icon(Icons.edit),
              onPressed: () => setState(() => _isEditing = true),
            ),
        ],
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Post #${_post.id}',
              style: Theme.of(context).textTheme.bodySmall,
            ),
            const SizedBox(height: 8),
            if (_isEditing) ...[
              TextField(
                controller: _titleController,
                decoration: const InputDecoration(
                  labelText: 'Titel',
                  border: OutlineInputBorder(),
                ),
              ),
              const SizedBox(height: 16),
              TextField(
                controller: _bodyController,
                decoration: const InputDecoration(
                  labelText: 'Inhalt',
                  border: OutlineInputBorder(),
                ),
                maxLines: 10,
```

```
              ),
              const SizedBox(height: 24),
              SizedBox(
                width: double.infinity,
                child: ElevatedButton(
                  onPressed: _saveChanges,
                  child: const Text('Speichern'),
                ),
              ),
            ] else ...[
              Text(
                _post.title,
                style: Theme.of(context).textTheme.headlineSmall,
              ),
              const SizedBox(height: 16),
              Text(
                _post.body,
                style: Theme.of(context).textTheme.bodyLarge,
              ),
            ],
          ],
        ),
      ),
    );
  }
}
```

3.4.2.6 main.dart

```
import 'package:flutter/material.dart';
import 'pages/posts_page.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'HTTP Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
        useMaterial3: true,
      ),
      home: const PostsPage(),
    );
  }
```

```
}
```

3.4.2.7 Verständnisfragen - Antworten

1. Uri.parse() vs Uri.https()

- `Uri.parse()`: Parst einen kompletten URL-String
- `Uri.https()`: Konstruiert eine URL aus Teilen (Host, Path, Query-Parameter)

`Uri.https()` ist sicherer, da Query-Parameter automatisch URL-encoded werden.

2. Warum http.Client?

- Wiederverwendet TCP-Verbindungen (Connection Pooling)
- Effizienter bei mehreren Requests
- Ermöglicht sauberes Cleanup mit `close()`

3. Statuscode 201 vs 200

- **200 OK**: Allgemeiner Erfolg, Ressource zurückgegeben
- **201 Created**: Neue Ressource wurde erfolgreich erstellt

4. Warum `as http`?

Verhindert Namenskonflikte, z.B. wenn du auch ein `Response`-Objekt aus einem anderen Package hast. Mit `http.Response` ist klar, welches gemeint ist.

5. Was wenn close() fehlt?

- TCP-Verbindungen bleiben offen
- Ressourcen werden nicht freigegeben
- Bei vielen Requests: Memory Leaks und Connection-Limits

3.4.3 Ressourcen

3.4.3.1 Offizielle Dokumentation

- http Package auf pub.dev
- Fetch data from the internet (Flutter Cookbook)
- Send data to the internet
- Make authenticated requests

3.4.3.2 Test-APIs

- JSONPlaceholder - Fake REST API für Tests
- ReqRes - Fake API für Login/Register
- HTTPBin - HTTP Request & Response Service
- PokeAPI - Pokemon API (für Übungen)
- OpenWeatherMap - Wetter-API

3.4.3.3 Alternative HTTP Packages

- dio - Erweiterte Features (Interceptors, FormData, Cancel)
- chopper - HTTP Client Generator
- retrofit - Type-Safe HTTP Client

3.4.3.4 Tutorials

- Networking in Flutter
- REST API Integration
- HTTP Error Handling

3.4.3.5 Videos

- Flutter HTTP Requests - Flutter Official
- REST API in Flutter - Reso Coder

3.4.3.6 Zusätzliche Packages

- connectivity_plus - Netzwerk-Status prüfen
- http_interceptor - Request/Response Interceptors
- pretty_http_logger - HTTP Logging

3.4.3.7 Zum Vertiefen

- RESTful API Design
- HTTP Status Codes
- SSL Pinning in Flutter

3.5 Einheit 3.5: JSON & Models

3.5.0.1 Lernziele

Nach dieser Einheit kannst du: - JSON-Daten in Dart-Objekte konvertieren - `fromJson` und `toJson` Methoden implementieren - Verschachtelte JSON-Strukturen verarbeiten - `json_serializable` für Code-Generierung nutzen - Best Practices für Model-Klassen anwenden

3.5.0.2 1. JSON Grundlagen in Dart

Import

```
import 'dart:convert';
```

JSON parsen und erzeugen

```
// JSON String -> Dart Map
final jsonString = '{"name": "Max", "age": 25}';
final Map<String, dynamic> data = jsonDecode(jsonString);

print(data['name']);  // Max
print(data['age']);   // 25

// Dart Map -> JSON String
final map = {'name': 'Max', 'age': 25};
final json = jsonEncode(map);

print(json);  // {"name":"Max","age":25}
```

JSON Arrays

```
// JSON Array -> Dart List
final jsonArray = '[{"id": 1}, {"id": 2}]';
final List<dynamic> list = jsonDecode(jsonArray);

// Dart List -> JSON String
final items = [{'id': 1}, {'id': 2}];
final json = jsonEncode(items);
```

3.5.0.3 2. Model-Klassen erstellen

Einfaches Model

```
class User {
  final int id;
  final String name;
  final String email;

  User({
    required this.id,
    required this.name,
    required this.email,
  });

  // Factory Constructor für JSON -> Object
  factory User.fromJson(Map<String, dynamic> json) {
    return User(
      id: json['id'] as int,
      name: json['name'] as String,
      email: json['email'] as String,
    );
  }

  // Method für Object -> JSON
  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'name': name,
      'email': email,
    };
  }
}
```

Verwendung

```
// API Response parsen
final response = await http.get(Uri.parse('https://api.example.com/user/1'));
final json = jsonDecode(response.body);
final user = User.fromJson(json);

// Für POST Request
final newUser = User(id: 0, name: 'Max', email: 'max@example.com');
final body = jsonEncode(newUser.toJson());
```

3.5.0.4 3. Optionale Felder und Defaults

```
class Post {
  final int id;
  final String title;
  final String? subtitle;      // Nullable
  final int likes;
  final DateTime createdAt;

  Post({
    required this.id,
    required this.title,
    this.subtitle,              // Optional
    this.likes = 0,             // Default
    required this.createdAt,
  });

  factory Post.fromJson(Map<String, dynamic> json) {
    return Post(
      id: json['id'] as int,
      title: json['title'] as String,
      subtitle: json['subtitle'] as String?,  // Kann null sein
      likes: json['likes'] as int? ?? 0,      // Default wenn null
      createdAt: DateTime.parse(json['created_at'] as String),
    );
  }

  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'title': title,
      if (subtitle != null) 'subtitle': subtitle,  // Nur wenn nicht null
      'likes': likes,
      'created_at': createdAt.toIso8601String(),
    };
  }
}
```

3.5.0.5 4. Verschachtelte Objekte

Einfache Verschachtelung

```
class Address {
  final String street;
  final String city;
  final String zipCode;

  Address({
    required this.street,
```

```
    required this.city,
    required this.zipCode,
  });

  factory Address.fromJson(Map<String, dynamic> json) {
    return Address(
      street: json['street'] as String,
      city: json['city'] as String,
      zipCode: json['zip_code'] as String,
    );
  }

  Map<String, dynamic> toJson() => {
    'street': street,
    'city': city,
    'zip_code': zipCode,
  };
}

class User {
  final int id;
  final String name;
  final Address address;  // Verschachteltes Objekt

  User({
    required this.id,
    required this.name,
    required this.address,
  });

  factory User.fromJson(Map<String, dynamic> json) {
    return User(
      id: json['id'] as int,
      name: json['name'] as String,
      // Verschachteltes Objekt parsen
      address: Address.fromJson(json['address'] as Map<String, dynamic>),
    );
  }

  Map<String, dynamic> toJson() => {
    'id': id,
    'name': name,
    'address': address.toJson(),  // Rekursiv
  };
}
```

Listen von Objekten

```
class Author {
  final String name;
  final List<Book> books;
```

```
  Author({required this.name, required this.books});

  factory Author.fromJson(Map<String, dynamic> json) {
    return Author(
      name: json['name'] as String,
      books: (json['books'] as List<dynamic>)
          .map((bookJson) => Book.fromJson(bookJson as Map<String, dynamic>))
          .toList(),
    );
  }

  Map<String, dynamic> toJson() => {
    'name': name,
    'books': books.map((book) => book.toJson()).toList(),
  };
}

class Book {
  final String title;
  final int year;

  Book({required this.title, required this.year});

  factory Book.fromJson(Map<String, dynamic> json) {
    return Book(
      title: json['title'] as String,
      year: json['year'] as int,
    );
  }

  Map<String, dynamic> toJson() => {
    'title': title,
    'year': year,
  };
}
```

3.5.0.6 5. Enums in JSON

```
enum Status {
  pending,
  active,
  completed,
  cancelled;

  // String -> Enum
  static Status fromJson(String json) {
    return Status.values.firstWhere(
      (e) => e.name == json,
      orElse: () => Status.pending,
```

```
    );
  }

  // Enum -> String
  String toJson() => name;
}

class Task {
  final String title;
  final Status status;

  Task({required this.title, required this.status});

  factory Task.fromJson(Map<String, dynamic> json) {
    return Task(
      title: json['title'] as String,
      status: Status.fromJson(json['status'] as String),
    );
  }

  Map<String, dynamic> toJson() => {
    'title': title,
    'status': status.toJson(),
  };
}
```

3.5.0.7 6. copyWith Pattern

```
class User {
  final int id;
  final String name;
  final String email;

  User({
    required this.id,
    required this.name,
    required this.email,
  });

  // Kopie mit optionalen Änderungen
  User copyWith({
    int? id,
    String? name,
    String? email,
  }) {
    return User(
      id: id ?? this.id,
      name: name ?? this.name,
      email: email ?? this.email,
    );
```

```
  }

  factory User.fromJson(Map<String, dynamic> json) => User(
    id: json['id'] as int,
    name: json['name'] as String,
    email: json['email'] as String,
  );

  Map<String, dynamic> toJson() => {
    'id': id,
    'name': name,
    'email': email,
  };

  // Für Debugging
  @override
  String toString() => 'User(id: $id, name: $name, email: $email)';

  // Für Vergleiche
  @override
  bool operator ==(Object other) =>
      identical(this, other) ||
      other is User &&
          id == other.id &&
          name == other.name &&
          email == other.email;

  @override
  int get hashCode => Object.hash(id, name, email);
}
```

3.5.0.8 7. json_serializable (Code-Generierung)

Setup

```
# pubspec.yaml
dependencies:
  json_annotation: ^4.8.1

dev_dependencies:
  build_runner: ^2.4.8
  json_serializable: ^6.7.1
```

Model mit Annotations

```
import 'package:json_annotation/json_annotation.dart';

// Generierte Datei importieren (wird erstellt)
part 'user.g.dart';

@JsonSerializable()
class User {
```

```
  final int id;
  final String name;

  @JsonKey(name: 'email_address')  // Anderer JSON-Schlüssel
  final String email;

  @JsonKey(defaultValue: false)    // Default-Wert
  final bool isVerified;

  @JsonKey(includeIfNull: false)   // Nicht in JSON wenn null
  final String? bio;

  User({
    required this.id,
    required this.name,
    required this.email,
    required this.isVerified,
    this.bio,
  });

  // Generierte Methoden verwenden
  factory User.fromJson(Map<String, dynamic> json) => _$UserFromJson(json);
  Map<String, dynamic> toJson() => _$UserToJson(this);
}
```

Code generieren

```
# Einmalig generieren
flutter pub run build_runner build

# Kontinuierlich bei Änderungen (Watch-Mode)
flutter pub run build_runner watch

# Konflikte lösen
flutter pub run build_runner build --delete-conflicting-outputs
```

Generierte Datei (user.g.dart)

```
// GENERATED CODE - DO NOT MODIFY BY HAND

part of 'user.dart';

User _$UserFromJson(Map<String, dynamic> json) => User(
  id: json['id'] as int,
  name: json['name'] as String,
  email: json['email_address'] as String,
  isVerified: json['isVerified'] as bool? ?? false,
  bio: json['bio'] as String?,
);

Map<String, dynamic> _$UserToJson(User instance) {
  final val = <String, dynamic>{
```

```
    'id': instance.id,
    'name': instance.name,
    'email_address': instance.email,
    'isVerified': instance.isVerified,
  };

  if (instance.bio != null) {
    val['bio'] = instance.bio;
  }

  return val;
}
```

JsonKey Optionen

```
@JsonKey(
  name: 'user_name',           // JSON-Schlüssel
  defaultValue: 'Unknown',     // Default wenn null/fehlt
  includeIfNull: false,        // Nicht serialisieren wenn null
  fromJson: _dateFromJson,     // Custom Parser
  toJson: _dateToJson,         // Custom Serializer
  ignore: true,                // Komplett ignorieren
  required: true,              // Muss vorhanden sein
)
```

Verschachtelte Objekte

```
@JsonSerializable(explicitToJson: true)  // Wichtig für verschachtelte Objekte
class Order {
  final int id;
  final User customer;         // Verschachteltes Objekt
  final List<Product> items;   // Liste von Objekten

  Order({
    required this.id,
    required this.customer,
    required this.items,
  });

  factory Order.fromJson(Map<String, dynamic> json) => _$OrderFromJson(json);
  Map<String, dynamic> toJson() => _$OrderToJson(this);
}
```

3.5.0.9 8. Praktisches Beispiel: API Response

JSON vom Server

```
{
  "data": {
    "users": [
      {
        "id": 1,
```

```
        "name": "Max Mustermann",
        "email": "max@example.com",
        "profile": {
          "avatar_url": "https://...",
          "bio": "Flutter Developer"
        },
        "created_at": "2024-01-15T10:30:00Z"
      }
    ],
    "total": 42,
    "page": 1
  },
  "success": true
}
```

Model-Klassen

```
// api_response.dart
class ApiResponse<T> {
  final T data;
  final bool success;

  ApiResponse({required this.data, required this.success});

  factory ApiResponse.fromJson(
    Map<String, dynamic> json,
    T Function(Map<String, dynamic>) fromJsonT,
  ) {
    return ApiResponse(
      data: fromJsonT(json['data'] as Map<String, dynamic>),
      success: json['success'] as bool,
    );
  }
}

// users_data.dart
class UsersData {
  final List<User> users;
  final int total;
  final int page;

  UsersData({
    required this.users,
    required this.total,
    required this.page,
  });

  factory UsersData.fromJson(Map<String, dynamic> json) {
    return UsersData(
      users: (json['users'] as List<dynamic>)
          .map((u) => User.fromJson(u as Map<String, dynamic>))
```

```
          .toList(),
      total: json['total'] as int,
      page: json['page'] as int,
    );
  }
}

// user.dart
class User {
  final int id;
  final String name;
  final String email;
  final Profile profile;
  final DateTime createdAt;

  User({
    required this.id,
    required this.name,
    required this.email,
    required this.profile,
    required this.createdAt,
  });

  factory User.fromJson(Map<String, dynamic> json) {
    return User(
      id: json['id'] as int,
      name: json['name'] as String,
      email: json['email'] as String,
      profile: Profile.fromJson(json['profile'] as Map<String, dynamic>),
      createdAt: DateTime.parse(json['created_at'] as String),
    );
  }
}

// profile.dart
class Profile {
  final String avatarUrl;
  final String? bio;

  Profile({required this.avatarUrl, this.bio});

  factory Profile.fromJson(Map<String, dynamic> json) {
    return Profile(
      avatarUrl: json['avatar_url'] as String,
      bio: json['bio'] as String?,
    );
  }
}
```

Verwendung

```dart
final response = await http.get(Uri.parse('https://api.example.com/users'));
final json = jsonDecode(response.body) as Map<String, dynamic>;

final apiResponse = ApiResponse.fromJson(json, UsersData.fromJson);

if (apiResponse.success) {
  for (final user in apiResponse.data.users) {
    print('${user.name}: ${user.profile.bio}');
  }
}
```

3.5.0.10 Zusammenfassung

Ansatz	Vorteile	Nachteile
Manuell	Volle Kontrolle, kein Build-Step	Fehleranfällig, viel Code
json_serializable	Weniger Code, Type-Safe	Build-Step nötig

Best Practices: 1. Immutable Models (alle Felder `final`) 2. `copyWith` für Änderungen 3. `toString`, `==`, `hashCode` implementieren 4. Factory Constructor für `fromJson` 5. Nullable Felder explizit markieren 6. `explicitToJson: true` bei verschachtelten Objekten

3.5.1 Übung

3.5.1.1 Ziel

Model-Klassen für eine API erstellen und JSON-Daten verarbeiten.

3.5.1.2 Aufgabe 1: Einfaches Model (15 min)

Gegeben ist folgender JSON:

```json
{
  "id": 1,
  "title": "Flutter Basics",
  "author": "Max Mustermann",
  "pages": 350,
  "published": true,
  "rating": 4.5
}
```

Erstelle eine `Book`-Klasse mit: - `fromJson` Factory Constructor - `toJson` Methode - `copyWith` Methode - `toString` Override

3.5.1.3 Aufgabe 2: Verschachtelte Objekte (20 min)

JSON-Struktur:

```json
{
  "id": 1,
  "name": "Flutter Developers",
  "description": "Eine Gruppe für Flutter Enthusiasten",
```

```
  "admin": {
    "id": 42,
    "username": "flutter_pro",
    "email": "admin@example.com"
  },
  "members_count": 1250,
  "created_at": "2024-01-15T10:30:00Z",
  "tags": ["flutter", "dart", "mobile"]
}
```

Erstelle die Model-Klassen: - `Group` (Hauptklasse) - `GroupAdmin` (verschachteltes Objekt)

Achte auf: - DateTime Parsing für `created_at` - Liste für `tags`

3.5.1.4 Aufgabe 3: Liste von Objekten (20 min)

API Response:

```
{
  "success": true,
  "data": {
    "products": [
      {
        "id": 1,
        "name": "Laptop",
        "price": 999.99,
        "category": "electronics",
        "in_stock": true
      },
      {
        "id": 2,
        "name": "Headphones",
        "price": 149.99,
        "category": "electronics",
        "in_stock": false
      }
    ],
    "total": 2,
    "currency": "EUR"
  }
}
```

Erstelle: - `Product` Model - `ProductsResponse` Model (enthält Liste von Products) - `ApiResponse<T>` Generic Wrapper

3.5.1.5 Aufgabe 4: Enum Handling (15 min)

```
{
  "id": 1,
  "title": "Wichtige Aufgabe",
  "priority": "high",
```

```
  "status": "in_progress"
}
```

Erstelle: - `Priority` Enum (low, medium, high) - `TaskStatus` Enum (todo, in_progress, done, cancelled) - `Task` Model mit korrektem Enum-Parsing

Behandle unbekannte Enum-Werte mit einem Default.

3.5.1.6 Aufgabe 5: json_serializable (30 min)

Rüste die `Product`-Klasse aus Aufgabe 3 auf `json_serializable` um:

1. Füge Dependencies hinzu:

```
dependencies:
  json_annotation: ^4.8.1
dev_dependencies:
  build_runner: ^2.4.8
  json_serializable: ^6.7.1
```

2. Erstelle die annotierte Klasse:

```
@JsonSerializable()
class Product {
  final int id;

  @JsonKey(name: 'product_name')
  final String name;

  @JsonKey(defaultValue: 0.0)
  final double price;

  @JsonKey(name: 'in_stock', defaultValue: false)
  final bool inStock;

  // ...
}
```

3. Führe den Build-Runner aus
4. Teste mit verschiedenen JSON-Inputs

3.5.1.7 Aufgabe 6: Fehlerbehandlung (15 min)

Schreibe eine `safeParse`-Funktion, die JSON-Fehler abfängt:

```
T? safeParse<T>(
  String jsonString,
  T Function(Map<String, dynamic>) fromJson,
) {
  // Implementiere:
  // - FormatException abfangen (ungültiges JSON)
  // - TypeError abfangen (fehlende/falsche Felder)
```

```
  // - null zurückgeben bei Fehlern (oder Exception werfen)
}
```

Teste mit: - Gültigem JSON - Ungültigem JSON-String - JSON mit fehlenden Feldern - JSON mit falschen Typen

3.5.1.8 Aufgabe 7: Praxis-Integration (25 min)

Integriere die Models in eine echte API-Anbindung:

1. Nutze die API: `https://jsonplaceholder.typicode.com/users`
2. Erstelle das `User`-Model basierend auf der API-Response:

```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
}
```

3. Lade alle User und zeige sie in einer ListView an

3.5.1.9 Bonus: Freezed Package

Installiere `freezed` und erstelle ein immutables Model:

```
import 'package:freezed_annotation/freezed_annotation.dart';

part 'user.freezed.dart';
part 'user.g.dart';

@freezed
class User with _$User {
  const factory User({
    required int id,
```

```
    required String name,
    String? email,
    @Default(false) bool isAdmin,
  }) = _User;

  factory User.fromJson(Map<String, dynamic> json) => _$UserFromJson(json);
}
```

Vergleiche den generierten Code mit manuellem Code.

3.5.1.10 Abgabe-Checkliste

- ☐ Book Model mit allen Methoden
- ☐ Group/GroupAdmin verschachtelte Models
- ☐ Product Model mit Liste
- ☐ Enums korrekt geparst
- ☐ json_serializable funktioniert
- ☐ Fehlerbehandlung implementiert
- ☐ User-API angebunden
- ☐ (Bonus) Freezed ausprobiert

3.5.2 Lösung

3.5.2.1 Aufgabe 1: Book Model

```
class Book {
  final int id;
  final String title;
  final String author;
  final int pages;
  final bool published;
  final double rating;

  Book({
    required this.id,
    required this.title,
    required this.author,
    required this.pages,
    required this.published,
    required this.rating,
  });

  factory Book.fromJson(Map<String, dynamic> json) {
    return Book(
      id: json['id'] as int,
      title: json['title'] as String,
      author: json['author'] as String,
      pages: json['pages'] as int,
      published: json['published'] as bool,
      rating: (json['rating'] as num).toDouble(),
    );
```

```
  }

  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'title': title,
      'author': author,
      'pages': pages,
      'published': published,
      'rating': rating,
    };
  }

  Book copyWith({
    int? id,
    String? title,
    String? author,
    int? pages,
    bool? published,
    double? rating,
  }) {
    return Book(
      id: id ?? this.id,
      title: title ?? this.title,
      author: author ?? this.author,
      pages: pages ?? this.pages,
      published: published ?? this.published,
      rating: rating ?? this.rating,
    );
  }

  @override
  String toString() {
    return 'Book(id: $id, title: $title, author: $author, '
        'pages: $pages, published: $published, rating: $rating)';
  }

  @override
  bool operator ==(Object other) =>
      identical(this, other) ||
      other is Book &&
          id == other.id &&
          title == other.title &&
          author == other.author;

  @override
  int get hashCode => Object.hash(id, title, author);
}
```

3.5.2.2 Aufgabe 2: Verschachtelte Objekte

```
class GroupAdmin {
  final int id;
  final String username;
  final String email;

  GroupAdmin({
    required this.id,
    required this.username,
    required this.email,
  });

  factory GroupAdmin.fromJson(Map<String, dynamic> json) {
    return GroupAdmin(
      id: json['id'] as int,
      username: json['username'] as String,
      email: json['email'] as String,
    );
  }

  Map<String, dynamic> toJson() => {
        'id': id,
        'username': username,
        'email': email,
      };
}

class Group {
  final int id;
  final String name;
  final String description;
  final GroupAdmin admin;
  final int membersCount;
  final DateTime createdAt;
  final List<String> tags;

  Group({
    required this.id,
    required this.name,
    required this.description,
    required this.admin,
    required this.membersCount,
    required this.createdAt,
    required this.tags,
  });

  factory Group.fromJson(Map<String, dynamic> json) {
    return Group(
      id: json['id'] as int,
      name: json['name'] as String,
```

```
      description: json['description'] as String,
      admin: GroupAdmin.fromJson(json['admin'] as Map<String, dynamic>),
      membersCount: json['members_count'] as int,
      createdAt: DateTime.parse(json['created_at'] as String),
      tags: (json['tags'] as List<dynamic>).cast<String>(),
    );
  }

  Map<String, dynamic> toJson() => {
        'id': id,
        'name': name,
        'description': description,
        'admin': admin.toJson(),
        'members_count': membersCount,
        'created_at': createdAt.toIso8601String(),
        'tags': tags,
      };
}
```

3.5.2.3 Aufgabe 3: Liste von Objekten

```
class Product {
  final int id;
  final String name;
  final double price;
  final String category;
  final bool inStock;

  Product({
    required this.id,
    required this.name,
    required this.price,
    required this.category,
    required this.inStock,
  });

  factory Product.fromJson(Map<String, dynamic> json) {
    return Product(
      id: json['id'] as int,
      name: json['name'] as String,
      price: (json['price'] as num).toDouble(),
      category: json['category'] as String,
      inStock: json['in_stock'] as bool,
    );
  }

  Map<String, dynamic> toJson() => {
        'id': id,
        'name': name,
        'price': price,
```

```
        'category': category,
        'in_stock': inStock,
      };
}

class ProductsData {
  final List<Product> products;
  final int total;
  final String currency;

  ProductsData({
    required this.products,
    required this.total,
    required this.currency,
  });

  factory ProductsData.fromJson(Map<String, dynamic> json) {
    return ProductsData(
      products: (json['products'] as List<dynamic>)
          .map((p) => Product.fromJson(p as Map<String, dynamic>))
          .toList(),
      total: json['total'] as int,
      currency: json['currency'] as String,
    );
  }
}

class ApiResponse<T> {
  final bool success;
  final T data;

  ApiResponse({
    required this.success,
    required this.data,
  });

  factory ApiResponse.fromJson(
    Map<String, dynamic> json,
    T Function(Map<String, dynamic>) fromJsonT,
  ) {
    return ApiResponse(
      success: json['success'] as bool,
      data: fromJsonT(json['data'] as Map<String, dynamic>),
    );
  }
}

// Verwendung:
// final response = ApiResponse.fromJson(json, ProductsData.fromJson);
// print(response.data.products.length);
```

3.5.2.4 Aufgabe 4: Enum Handling

```
enum Priority {
  low,
  medium,
  high;

  static Priority fromJson(String json) {
    return Priority.values.firstWhere(
      (e) => e.name == json.toLowerCase(),
      orElse: () => Priority.medium,
    );
  }

  String toJson() => name;
}

enum TaskStatus {
  todo,
  inProgress,
  done,
  cancelled;

  static TaskStatus fromJson(String json) {
    // Handle snake_case from API
    switch (json.toLowerCase()) {
      case 'todo':
        return TaskStatus.todo;
      case 'in_progress':
        return TaskStatus.inProgress;
      case 'done':
        return TaskStatus.done;
      case 'cancelled':
        return TaskStatus.cancelled;
      default:
        return TaskStatus.todo;
    }
  }

  String toJson() {
    switch (this) {
      case TaskStatus.todo:
        return 'todo';
      case TaskStatus.inProgress:
        return 'in_progress';
      case TaskStatus.done:
        return 'done';
      case TaskStatus.cancelled:
        return 'cancelled';
    }
  }
```

```
}

class Task {
  final int id;
  final String title;
  final Priority priority;
  final TaskStatus status;

  Task({
    required this.id,
    required this.title,
    required this.priority,
    required this.status,
  });

  factory Task.fromJson(Map<String, dynamic> json) {
    return Task(
      id: json['id'] as int,
      title: json['title'] as String,
      priority: Priority.fromJson(json['priority'] as String),
      status: TaskStatus.fromJson(json['status'] as String),
    );
  }

  Map<String, dynamic> toJson() => {
        'id': id,
        'title': title,
        'priority': priority.toJson(),
        'status': status.toJson(),
      };
}
```

3.5.2.5 Aufgabe 5: json_serializable

```
// product.dart
import 'package:json_annotation/json_annotation.dart';

part 'product.g.dart';

@JsonSerializable()
class Product {
  final int id;

  @JsonKey(name: 'name')
  final String name;

  @JsonKey(defaultValue: 0.0)
  final double price;

  final String category;
```

```
  @JsonKey(name: 'in_stock', defaultValue: false)
  final bool inStock;

  Product({
    required this.id,
    required this.name,
    required this.price,
    required this.category,
    required this.inStock,
  });

  factory Product.fromJson(Map<String, dynamic> json) =>
      _$ProductFromJson(json);

  Map<String, dynamic> toJson() => _$ProductToJson(this);
}
```

Nach `flutter pub run build_runner build` wird `product.g.dart` generiert.

3.5.2.6 Aufgabe 6: Fehlerbehandlung

```
import 'dart:convert';

class JsonParseException implements Exception {
  final String message;
  final dynamic originalError;

  JsonParseException(this.message, [this.originalError]);

  @override
  String toString() => 'JsonParseException: $message';
}

T? safeParse<T>(
  String jsonString,
  T Function(Map<String, dynamic>) fromJson,
) {
  try {
    final decoded = jsonDecode(jsonString);

    if (decoded is! Map<String, dynamic>) {
      throw JsonParseException('Expected JSON object, got
↪  ${decoded.runtimeType}');
    }

    return fromJson(decoded);
  } on FormatException catch (e) {
    print('Invalid JSON format: $e');
    return null;
  } on TypeError catch (e) {
```

```
    print('Type error while parsing: $e');
    return null;
  } on JsonParseException catch (e) {
    print(e);
    return null;
  } catch (e) {
    print('Unexpected error: $e');
    return null;
  }
}

// Variante die Exception wirft
T parseOrThrow<T>(
  String jsonString,
  T Function(Map<String, dynamic>) fromJson,
) {
  try {
    final decoded = jsonDecode(jsonString);

    if (decoded is! Map<String, dynamic>) {
      throw JsonParseException('Expected JSON object');
    }

    return fromJson(decoded);
  } on FormatException catch (e) {
    throw JsonParseException('Invalid JSON format', e);
  } on TypeError catch (e) {
    throw JsonParseException('Missing or invalid field', e);
  }
}

// Tests
void main() {
  // Gültiges JSON
  final valid = '{"id": 1, "title": "Test", "author": "Max", "pages": 100,     ↪
 ↪  "published": true, "rating": 4.5}';
  final book = safeParse(valid, Book.fromJson);
  print(book);  // Book(...)

  // Ungültiges JSON
  final invalid = '{invalid json}';
  final result1 = safeParse(invalid, Book.fromJson);
  print(result1);  // null

  // Fehlende Felder
  final missing = '{"id": 1}';
  final result2 = safeParse(missing, Book.fromJson);
  print(result2);  // null

  // Falsche Typen
  final wrongType = '{"id": "not a number", "title": 123}';
```

```
  final result3 = safeParse(wrongType, Book.fromJson);
  print(result3);  // null
}
```

3.5.2.7 Aufgabe 7: User API Integration

```
// models/address.dart
class Geo {
  final String lat;
  final String lng;

  Geo({required this.lat, required this.lng});

  factory Geo.fromJson(Map<String, dynamic> json) {
    return Geo(
      lat: json['lat'] as String,
      lng: json['lng'] as String,
    );
  }
}

class Address {
  final String street;
  final String suite;
  final String city;
  final String zipcode;
  final Geo geo;

  Address({
    required this.street,
    required this.suite,
    required this.city,
    required this.zipcode,
    required this.geo,
  });

  factory Address.fromJson(Map<String, dynamic> json) {
    return Address(
      street: json['street'] as String,
      suite: json['suite'] as String,
      city: json['city'] as String,
      zipcode: json['zipcode'] as String,
      geo: Geo.fromJson(json['geo'] as Map<String, dynamic>),
    );
  }

  String get fullAddress => '$street, $suite, $city $zipcode';
}

// models/company.dart
```

```
class Company {
  final String name;
  final String catchPhrase;
  final String bs;

  Company({
    required this.name,
    required this.catchPhrase,
    required this.bs,
  });

  factory Company.fromJson(Map<String, dynamic> json) {
    return Company(
      name: json['name'] as String,
      catchPhrase: json['catchPhrase'] as String,
      bs: json['bs'] as String,
    );
  }
}

// models/user.dart
class User {
  final int id;
  final String name;
  final String username;
  final String email;
  final Address address;
  final String phone;
  final String website;
  final Company company;

  User({
    required this.id,
    required this.name,
    required this.username,
    required this.email,
    required this.address,
    required this.phone,
    required this.website,
    required this.company,
  });

  factory User.fromJson(Map<String, dynamic> json) {
    return User(
      id: json['id'] as int,
      name: json['name'] as String,
      username: json['username'] as String,
      email: json['email'] as String,
      address: Address.fromJson(json['address'] as Map<String, dynamic>),
      phone: json['phone'] as String,
      website: json['website'] as String,
```

```
      company: Company.fromJson(json['company'] as Map<String, dynamic>),
    );
  }
}

// services/user_service.dart
class UserService {
  static const baseUrl = 'https://jsonplaceholder.typicode.com';

  Future<List<User>> fetchUsers() async {
    final response = await http.get(Uri.parse('$baseUrl/users'));

    if (response.statusCode != 200) {
      throw Exception('Failed to load users');
    }

    final List<dynamic> jsonList = jsonDecode(response.body);
    return jsonList
        .map((json) => User.fromJson(json as Map<String, dynamic>))
        .toList();
  }
}

// Widget
class UsersListPage extends StatefulWidget {
  @override
  State<UsersListPage> createState() => _UsersListPageState();
}

class _UsersListPageState extends State<UsersListPage> {
  late Future<List<User>> _usersFuture;

  @override
  void initState() {
    super.initState();
    _usersFuture = UserService().fetchUsers();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Users')),
      body: FutureBuilder<List<User>>(
        future: _usersFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          }

          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
```

```
          }

          final users = snapshot.data!;

          return ListView.builder(
            itemCount: users.length,
            itemBuilder: (context, index) {
              final user = users[index];
              return ListTile(
                leading: CircleAvatar(child: Text(user.name[0])),
                title: Text(user.name),
                subtitle: Text('${user.email}\n${user.company.name}'),
                isThreeLine: true,
              );
            },
          );
        },
      ),
    );
  }
}
```

3.5.3 Ressourcen

3.5.3.1 Offizielle Dokumentation

- JSON and serialization (Flutter Docs)
- dart:convert library
- json_serializable Package
- json_annotation Package

3.5.3.2 Code-Generierung

- build_runner Package
- freezed Package - Immutable Classes + JSON
- built_value Package - Alternative zu json_serializable

3.5.3.3 Online Tools

- quicktype.io - JSON zu Dart Model Generator
- JSON to Dart - Online Converter
- JSON Formatter - JSON validieren und formatieren

3.5.3.4 Tutorials

- Flutter JSON Serialization
- JSON Serialization Deep Dive
- Freezed Tutorial

3.5.3.5 Videos

- JSON Parsing in Flutter
- json_serializable Tutorial

- Freezed Package Explained

3.5.3.6 Best Practices

- Effective Dart: Design
- Immutable Data Patterns
- Error Handling in JSON Parsing

3.5.3.7 Alternative Packages

- dart_mappable - Moderne Alternative
- equatable - Einfache == Implementierung
- copy_with_extension - copyWith Generator

3.6 Einheit 3.6: FutureBuilder & StreamBuilder

3.6.0.1 Lernziele

Nach dieser Einheit kannst du: - `FutureBuilder` für einmalige async Operationen verwenden - `StreamBuilder` für kontinuierliche Datenströme einsetzen - Loading und Error States korrekt anzeigen - Skeleton Screens für bessere UX implementieren - Häufige Fehler vermeiden

3.6.0.2 1. FutureBuilder Grundlagen

`FutureBuilder` baut Widgets basierend auf dem Zustand eines `Future`.

Einfaches Beispiel

```
class UserProfile extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return FutureBuilder<User>(
      future: fetchUser(),
      builder: (context, snapshot) {
        // Loading State
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const CircularProgressIndicator();
        }

        // Error State
        if (snapshot.hasError) {
          return Text('Fehler: ${snapshot.error}');
        }

        // Success State
        if (snapshot.hasData) {
          final user = snapshot.data!;
          return Text('Hallo, ${user.name}!');
        }

        // Kein Zustand (sollte nicht vorkommen)
        return const SizedBox.shrink();
      },
```

```
    );
  }
}
```

AsyncSnapshot verstehen

```
builder: (context, AsyncSnapshot<User> snapshot) {
  // ConnectionState
  snapshot.connectionState;  // none, waiting, active, done

  // Daten
  snapshot.hasData;          // true wenn data != null
  snapshot.data;             // Die Daten (nullable)

  // Fehler
  snapshot.hasError;         // true wenn error != null
  snapshot.error;            // Die Exception

  // Hilfreich für Debugging
  print('State: ${snapshot.connectionState}');
  print('Data: ${snapshot.data}');
  print('Error: ${snapshot.error}');
}
```

ConnectionState Werte

State	Bedeutung
`none`	Kein Future zugewiesen
`waiting`	Future läuft noch
`active`	Stream aktiv (nur bei StreamBuilder)
`done`	Future abgeschlossen

3.6.0.3 2. FutureBuilder Best Practices

Problem: Future wird bei jedem Build neu erstellt

```
// ❌ FALSCH: Future wird bei jedem setState() neu ausgeführt
class BadExample extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return FutureBuilder<User>(
      future: fetchUser(),  // Wird bei jedem Build aufgerufen!
      builder: (context, snapshot) => ...,
    );
  }
}
```

Lösung: Future im State speichern

```
// ✅ RICHTIG: Future wird einmal erstellt
class GoodExample extends StatefulWidget {
  @override
```

```
  State<GoodExample> createState() => _GoodExampleState();
}

class _GoodExampleState extends State<GoodExample> {
  late Future<User> _userFuture;

  @override
  void initState() {
    super.initState();
    _userFuture = fetchUser();  // Nur einmal!
  }

  @override
  Widget build(BuildContext context) {
    return FutureBuilder<User>(
      future: _userFuture,  // Gleiche Referenz
      builder: (context, snapshot) => ...,
    );
  }

  // Für Refresh
  void _refresh() {
    setState(() {
      _userFuture = fetchUser();
    });
  }
}
```

Vollständiges Pattern

```
class UserPage extends StatefulWidget {
  const UserPage({super.key});

  @override
  State<UserPage> createState() => _UserPageState();
}

class _UserPageState extends State<UserPage> {
  late Future<User> _userFuture;

  @override
  void initState() {
    super.initState();
    _loadUser();
  }

  void _loadUser() {
    _userFuture = UserService().fetchCurrentUser();
  }

  void _refresh() {
```

```
    setState(() {
      _loadUser();
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Profil'),
        actions: [
          IconButton(
            icon: const Icon(Icons.refresh),
            onPressed: _refresh,
          ),
        ],
      ),
      body: FutureBuilder<User>(
        future: _userFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          }

          if (snapshot.hasError) {
            return _ErrorWidget(
              error: snapshot.error!,
              onRetry: _refresh,
            );
          }

          final user = snapshot.data!;
          return _UserContent(user: user);
        },
      ),
    );
  }
}

class _ErrorWidget extends StatelessWidget {
  final Object error;
  final VoidCallback onRetry;

  const _ErrorWidget({required this.error, required this.onRetry});

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
```

```
          const Icon(Icons.error_outline, size: 48, color: Colors.red),
          const SizedBox(height: 16),
          Text('Fehler: $error'),
          const SizedBox(height: 16),
          ElevatedButton(
            onPressed: onRetry,
            child: const Text('Erneut versuchen'),
          ),
        ],
      ),
    );
  }
}
```

3.6.0.4 3. StreamBuilder Grundlagen

`StreamBuilder` reagiert auf jeden neuen Wert eines Streams.

Einfaches Beispiel

```
class MessageList extends StatelessWidget {
  final Stream<List<Message>> messagesStream;

  const MessageList({required this.messagesStream});

  @override
  Widget build(BuildContext context) {
    return StreamBuilder<List<Message>>(
      stream: messagesStream,
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const CircularProgressIndicator();
        }

        if (snapshot.hasError) {
          return Text('Fehler: ${snapshot.error}');
        }

        final messages = snapshot.data ?? [];

        return ListView.builder(
          itemCount: messages.length,
          itemBuilder: (context, index) {
            return MessageTile(message: messages[index]);
          },
        );
      },
    );
  }
}
```

Mit initialData

```
StreamBuilder<int>(
  stream: counterStream,
  initialData: 0,  // Startwert während wir auf ersten Stream-Wert warten
  builder: (context, snapshot) {
    final count = snapshot.data!;  // Sicher, da initialData gesetzt
    return Text('Count: $count');
  },
);
```

Timer-Beispiel

```
class Clock extends StatefulWidget {
  @override
  State<Clock> createState() => _ClockState();
}

class _ClockState extends State<Clock> {
  late Stream<DateTime> _timeStream;

  @override
  void initState() {
    super.initState();
    _timeStream = Stream.periodic(
      const Duration(seconds: 1),
      (_) => DateTime.now(),
    );
  }

  @override
  Widget build(BuildContext context) {
    return StreamBuilder<DateTime>(
      stream: _timeStream,
      builder: (context, snapshot) {
        if (!snapshot.hasData) {
          return const Text('--:--:--');
        }

        final time = snapshot.data!;
        final formatted =
            '${time.hour.toString().padLeft(2, '0')}:'
            '${time.minute.toString().padLeft(2, '0')}:'
            '${time.second.toString().padLeft(2, '0')}';

        return Text(
          formatted,
          style: const TextStyle(fontSize: 48),
        );
      },
    );
  }
}
```

3.6.0.5 4. Skeleton Screens

Skeleton Screens zeigen Platzhalter während des Ladens - bessere UX als Spinner.

Einfaches Skeleton

```
class SkeletonBox extends StatelessWidget {
  final double width;
  final double height;

  const SkeletonBox({
    this.width = double.infinity,
    this.height = 16,
  });

  @override
  Widget build(BuildContext context) {
    return Container(
      width: width,
      height: height,
      decoration: BoxDecoration(
        color: Colors.grey[300],
        borderRadius: BorderRadius.circular(4),
      ),
    );
  }
}

class SkeletonListTile extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return const Padding(
      padding: EdgeInsets.all(16),
      child: Row(
        children: [
          SkeletonBox(width: 48, height: 48),  // Avatar
          SizedBox(width: 16),
          Expanded(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                SkeletonBox(width: 150, height: 16),  // Title
                SizedBox(height: 8),
                SkeletonBox(width: 100, height: 12),  // Subtitle
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

Animiertes Skeleton (Shimmer)

```
class ShimmerBox extends StatefulWidget {
  final double width;
  final double height;

  const ShimmerBox({this.width = double.infinity, this.height = 16});

  @override
  State<ShimmerBox> createState() => _ShimmerBoxState();
}

class _ShimmerBoxState extends State<ShimmerBox>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      vsync: this,
      duration: const Duration(milliseconds: 1500),
    )..repeat();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return AnimatedBuilder(
      animation: _controller,
      builder: (context, child) {
        return Container(
          width: widget.width,
          height: widget.height,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(4),
            gradient: LinearGradient(
              colors: [
                Colors.grey[300]!,
                Colors.grey[100]!,
                Colors.grey[300]!,
              ],
              stops: [
                _controller.value - 0.3,
                _controller.value,
                _controller.value + 0.3,
              ].map((s) => s.clamp(0.0, 1.0)).toList(),
```

```
            ),
          ),
        );
      },
    );
  }
}
```

shimmer Package verwenden

```
dependencies:
  shimmer: ^3.0.0
```

```
import 'package:shimmer/shimmer.dart';

class SkeletonList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Shimmer.fromColors(
      baseColor: Colors.grey[300]!,
      highlightColor: Colors.grey[100]!,
      child: ListView.builder(
        itemCount: 5,
        itemBuilder: (_, __) => const ListTile(
          leading: CircleAvatar(backgroundColor: Colors.white),
          title: SkeletonBox(height: 16),
          subtitle: SkeletonBox(height: 12, width: 100),
        ),
      ),
    );
  }
}
```

In FutureBuilder integrieren

```
FutureBuilder<List<Post>>(
  future: _postsFuture,
  builder: (context, snapshot) {
    // Skeleton während Loading
    if (snapshot.connectionState == ConnectionState.waiting) {
      return ListView.builder(
        itemCount: 5,
        itemBuilder: (_, __) => const SkeletonPostCard(),
      );
    }

    if (snapshot.hasError) {
      return ErrorWidget(error: snapshot.error!);
    }

    final posts = snapshot.data!;
    return ListView.builder(
```

```
      itemCount: posts.length,
      itemBuilder: (_, index) => PostCard(post: posts[index]),
    );
  },
);
```

3.6.0.6 5. Fortgeschrittene Patterns

Kombinierte Futures

```
class DashboardPage extends StatefulWidget {
  @override
  State<DashboardPage> createState() => _DashboardPageState();
}

class _DashboardPageState extends State<DashboardPage> {
  late Future<(User, List<Post>, Stats)> _dataFuture;

  @override
  void initState() {
    super.initState();
    _loadData();
  }

  void _loadData() {
    _dataFuture = (
      fetchUser(),
      fetchPosts(),
      fetchStats(),
    ).wait;  // Dart 3 Records + Future.wait
  }

  @override
  Widget build(BuildContext context) {
    return FutureBuilder(
      future: _dataFuture,
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const DashboardSkeleton();
        }

        if (snapshot.hasError) {
          return ErrorWidget(error: snapshot.error!);
        }

        final (user, posts, stats) = snapshot.data!;

        return DashboardContent(
          user: user,
          posts: posts,
          stats: stats,
```

```
        );
      },
    );
  }
}
```

Stream mit Fehlerbehandlung

```
StreamBuilder<Result<List<Message>>>(
  stream: messageService.messagesStream,
  builder: (context, snapshot) {
    if (!snapshot.hasData) {
      return const LoadingIndicator();
    }

    return snapshot.data!.when(
      success: (messages) => MessageList(messages: messages),
      error: (error) => ErrorWidget(error: error),
    );
  },
);

// Result Sealed Class
sealed class Result<T> {
  const Result();

  R when<R>({
    required R Function(T data) success,
    required R Function(Object error) error,
  });
}

class Success<T> extends Result<T> {
  final T data;
  const Success(this.data);

  @override
  R when<R>({
    required R Function(T data) success,
    required R Function(Object error) error,
  }) => success(data);
}

class Error<T> extends Result<T> {
  final Object exception;
  const Error(this.exception);

  @override
  R when<R>({
    required R Function(T data) success,
    required R Function(Object error) error,
```

```
  }) => error(exception);
}
```

3.6.0.7 6. Häufige Fehler

Future in build() erstellen

```
// ❌ FALSCH
Widget build(BuildContext context) {
  return FutureBuilder(
    future: api.fetchData(),  // Jedes Mal neu!
    builder: ...,
  );
}

// ✅ RICHTIG
late Future<Data> _future;

@override
void initState() {
  super.initState();
  _future = api.fetchData();
}
```

Stream nicht disposen

```
// ❌ FALSCH: Stream läuft weiter
class BadWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final stream = StreamController<int>();
    // Stream wird nie geschlossen!
    return StreamBuilder(stream: stream.stream, ...);
  }
}

// ✅ RICHTIG: Cleanup in dispose
class GoodWidget extends StatefulWidget {
  @override
  State<GoodWidget> createState() => _GoodWidgetState();
}

class _GoodWidgetState extends State<GoodWidget> {
  final _controller = StreamController<int>();

  @override
  void dispose() {
    _controller.close();
    super.dispose();
  }
}
```

3.6.0.8 Zusammenfassung

Widget	Use Case
`FutureBuilder`	Einmalige async Operation (API Call, File Read)
`StreamBuilder`	Kontinuierlicher Datenstrom (Real-time, Timer)

Best Practices: 1. Future/Stream im `initState` erstellen, nicht in `build` 2. Alle drei States behandeln: Loading, Error, Data 3. Skeleton Screens für bessere UX 4. Streams in `dispose` schließen 5. `initialData` für sofortige Anzeige nutzen

3.6.1 Übung

3.6.1.1 Ziel

Loading States und asynchrone Daten elegant in der UI darstellen.

3.6.1.2 Aufgabe 1: FutureBuilder Basics (20 min)

Erstelle eine `QuotePage`, die ein Zitat von einer API lädt:

API: `https://api.quotable.io/random`

Anforderungen: 1. Loading State: Zeige `CircularProgressIndicator` 2. Error State: Zeige Fehlermeldung mit Retry-Button 3. Success State: Zeige das Zitat mit Autor 4. Refresh-Button in der AppBar

```
+---------------------------------+
| Quote of the Day       [□]   |
+---------------------------------+
|                                 |
|    "The only way to do great   |
|     work is to love what       |
|     you do."                   |
|                                 |
|           - Steve Jobs         |
|                                 |
+---------------------------------+
```

Wichtig: Achte darauf, das Future NICHT in `build()` zu erstellen!

3.6.1.3 Aufgabe 2: Skeleton Screen (25 min)

Erstelle Skeleton-Komponenten für eine User-Liste:

1. `SkeletonBox` - Einfacher grauer Platzhalter
2. `SkeletonCircle` - Runder Platzhalter (Avatar)
3. `SkeletonUserTile` - Komplettes Skeleton für einen User

```
+---------------------------------+
| □ ████████████            |
|    ████████                 |
+---------------------------------+
| □ ████████████            |
|    ████████                 |
+---------------------------------+
```

Integriere in `FutureBuilder`: - Zeige 5 Skeleton-Tiles während Loading - Zeige echte Daten nach dem Laden

3.6.1.4 Aufgabe 3: StreamBuilder - Live Clock (15 min)

Erstelle eine digitale Uhr, die jede Sekunde aktualisiert:

```
Stream<DateTime> _createTimeStream() {
  return Stream.periodic(
    const Duration(seconds: 1),
    (_) => DateTime.now(),
  );
}
```

Anforderungen: 1. Format: HH:MM:SS 2. Zeige auch das Datum 3. Responsive Schriftgröße

3.6.1.5 Aufgabe 4: StreamBuilder - Counter mit Buttons (20 min)

Erstelle einen Counter, der über einen `StreamController` gesteuert wird:

```
class CounterController {
  final _controller = StreamController<int>.broadcast();
  int _count = 0;

  Stream<int> get stream => _controller.stream;

  void increment() {
    _count++;
    _controller.add(_count);
  }

  void decrement() {
    _count--;
    _controller.add(_count);
  }

  void reset() {
    _count = 0;
    _controller.add(_count);
  }

  void dispose() {
    _controller.close();
  }
}
```

UI: - Zeige aktuellen Wert - Buttons: +, -, Reset - `initialData: 0` für sofortige Anzeige

3.6.1.6 Aufgabe 5: Kombinierte Futures (25 min)

Erstelle ein Dashboard, das mehrere API-Calls parallel ausführt:

```
// Simulierte API-Calls
Future<int> fetchUserCount() async {
  await Future.delayed(Duration(seconds: 1));
  return 1234;
}

Future<int> fetchOrderCount() async {
  await Future.delayed(Duration(seconds: 2));
  return 567;
}

Future<double> fetchRevenue() async {
  await Future.delayed(Duration(seconds: 1, milliseconds: 500));
  return 12345.67;
}
```

Anforderungen: 1. Alle drei Calls parallel starten 2. Dashboard erst anzeigen wenn ALLE fertig sind 3. Einzelne Skeleton-Karten während Loading 4. Refresh lädt alle neu

```
+---------------------------------+
| Dashboard               [□]   |
+---------------------------------+
| +---------+ +---------+        |
| |  Users  | | Orders  |        |
| |  1,234  | |   567   |        |
| +---------+ +---------+        |
| +-----------------------+      |
| |      Revenue          |      |
| |    €12,345.67         |      |
| +-----------------------+      |
+---------------------------------+
```

3.6.1.7 Aufgabe 6: Pull-to-Refresh mit FutureBuilder (15 min)

Erweitere die QuotePage aus Aufgabe 1:

1. Implementiere Pull-to-Refresh mit `RefreshIndicator`
2. Zeige eine Snackbar bei erfolgreichem Refresh
3. Behalte das alte Zitat während des Ladens sichtbar

3.6.1.8 Aufgabe 7: Verständnisfragen

1. Warum sollte man das Future nicht direkt in `build()` erstellen?
2. Was ist der Unterschied zwischen `snapshot.hasData` und `snapshot.data != null`?
3. Wann verwendet man `StreamBuilder.initialData`?
4. Wie verhindert man Memory Leaks bei StreamControllern?
5. Was passiert, wenn der Stream einen Fehler emittiert?

3.6.1.9 Bonus: Shimmer Animation

Installiere das `shimmer` Package und erstelle animierte Skeleton Screens:

```
Shimmer.fromColors(
  baseColor: Colors.grey[300]!,
  highlightColor: Colors.grey[100]!,
  child: YourSkeletonWidget(),
);
```

Erstelle ein komplettes Skeleton für eine Produktkarte mit: - Bild-Platzhalter - Titel - Preis - Rating-Sterne

3.6.1.10 Abgabe-Checkliste

- ☐ QuotePage mit allen 3 States
- ☐ Skeleton-Komponenten erstellt
- ☐ Skeleton in FutureBuilder integriert
- ☐ Live Clock funktioniert
- ☐ Counter mit StreamController
- ☐ Dashboard mit kombinierten Futures
- ☐ Pull-to-Refresh implementiert
- ☐ Verständnisfragen beantwortet
- ☐ (Bonus) Shimmer Animation

3.6.2 Lösung

3.6.2.1 Aufgabe 1: QuotePage

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class Quote {
  final String content;
  final String author;

  Quote({required this.content, required this.author});

  factory Quote.fromJson(Map<String, dynamic> json) {
    return Quote(
      content: json['content'] as String,
      author: json['author'] as String,
    );
  }
}

class QuotePage extends StatefulWidget {
  const QuotePage({super.key});

  @override
  State<QuotePage> createState() => _QuotePageState();
}
```

```
class _QuotePageState extends State<QuotePage> {
  late Future<Quote> _quoteFuture;

  @override
  void initState() {
    super.initState();
    _loadQuote();
  }

  void _loadQuote() {
    _quoteFuture = _fetchQuote();
  }

  Future<Quote> _fetchQuote() async {
    final response = await http.get(
      Uri.parse('https://api.quotable.io/random'),
    );

    if (response.statusCode != 200) {
      throw Exception('Failed to load quote');
    }

    return Quote.fromJson(jsonDecode(response.body));
  }

  void _refresh() {
    setState(() {
      _loadQuote();
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Quote of the Day'),
        actions: [
          IconButton(
            icon: const Icon(Icons.refresh),
            onPressed: _refresh,
          ),
        ],
      ),
      body: FutureBuilder<Quote>(
        future: _quoteFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          }
```

```
          if (snapshot.hasError) {
            return Center(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  const Icon(Icons.error_outline, size: 48, color: Colors.red),
                  const SizedBox(height: 16),
                  Text('Fehler: ${snapshot.error}'),
                  const SizedBox(height: 16),
                  ElevatedButton.icon(
                    onPressed: _refresh,
                    icon: const Icon(Icons.refresh),
                    label: const Text('Erneut versuchen'),
                  ),
                ],
              ),
            );
          }

          final quote = snapshot.data!;
          return Padding(
            padding: const EdgeInsets.all(24),
            child: Center(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  const Icon(Icons.format_quote, size: 48, color: Colors.grey),
                  const SizedBox(height: 24),
                  Text(
                    '"${quote.content}"',
                    style: Theme.of(context).textTheme.headlineSmall,
                    textAlign: TextAlign.center,
                  ),
                  const SizedBox(height: 24),
                  Text(
                    '- ${quote.author}',
                    style: Theme.of(context).textTheme.titleMedium?.copyWith(
                          fontStyle: FontStyle.italic,
                        ),
                  ),
                ],
              ),
            ),
          );
        },
      ),
    );
  }
}
```

3.6.2.2 Aufgabe 2: Skeleton Components

```
class SkeletonBox extends StatelessWidget {
  final double width;
  final double height;
  final double borderRadius;

  const SkeletonBox({
    super.key,
    this.width = double.infinity,
    this.height = 16,
    this.borderRadius = 4,
  });

  @override
  Widget build(BuildContext context) {
    return Container(
      width: width,
      height: height,
      decoration: BoxDecoration(
        color: Colors.grey[300],
        borderRadius: BorderRadius.circular(borderRadius),
      ),
    );
  }
}

class SkeletonCircle extends StatelessWidget {
  final double size;

  const SkeletonCircle({super.key, this.size = 48});

  @override
  Widget build(BuildContext context) {
    return Container(
      width: size,
      height: size,
      decoration: BoxDecoration(
        color: Colors.grey[300],
        shape: BoxShape.circle,
      ),
    );
  }
}

class SkeletonUserTile extends StatelessWidget {
  const SkeletonUserTile({super.key});

  @override
  Widget build(BuildContext context) {
    return const Padding(
```

```
      padding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),
      child: Row(
        children: [
          SkeletonCircle(size: 48),
          SizedBox(width: 16),
          Expanded(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                SkeletonBox(width: 150, height: 16),
                SizedBox(height: 8),
                SkeletonBox(width: 100, height: 12),
              ],
            ),
          ),
        ],
      ),
    );
  }
}

// Integration
class UserListPage extends StatefulWidget {
  const UserListPage({super.key});

  @override
  State<UserListPage> createState() => _UserListPageState();
}

class _UserListPageState extends State<UserListPage> {
  late Future<List<User>> _usersFuture;

  @override
  void initState() {
    super.initState();
    _usersFuture = _fetchUsers();
  }

  Future<List<User>> _fetchUsers() async {
    await Future.delayed(const Duration(seconds: 2)); // Simulate network
    // ... fetch from API
    return [];
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Users')),
      body: FutureBuilder<List<User>>(
        future: _usersFuture,
        builder: (context, snapshot) {
```

```
          if (snapshot.connectionState == ConnectionState.waiting) {
            return ListView.builder(
              itemCount: 5,
              itemBuilder: (_, __) => const SkeletonUserTile(),
            );
          }

          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }

          final users = snapshot.data!;
          return ListView.builder(
            itemCount: users.length,
            itemBuilder: (_, index) => UserTile(user: users[index]),
          );
        },
      ),
    );
  }
}
```

3.6.2.3 Aufgabe 3: Live Clock

```
class LiveClock extends StatefulWidget {
  const LiveClock({super.key});

  @override
  State<LiveClock> createState() => _LiveClockState();
}

class _LiveClockState extends State<LiveClock> {
  late Stream<DateTime> _timeStream;

  @override
  void initState() {
    super.initState();
    _timeStream = Stream.periodic(
      const Duration(seconds: 1),
      (_) => DateTime.now(),
    );
  }

  String _formatTime(DateTime time) {
    return '${time.hour.toString().padLeft(2, '0')}:'
        '${time.minute.toString().padLeft(2, '0')}:'
        '${time.second.toString().padLeft(2, '0')}';
  }

  String _formatDate(DateTime time) {
```

```
    const weekdays = ['Mo', 'Di', 'Mi', 'Do', 'Fr', 'Sa', 'So'];
    const months = [
      'Jan', 'Feb', 'Mär', 'Apr', 'Mai', 'Jun',
      'Jul', 'Aug', 'Sep', 'Okt', 'Nov', 'Dez'
    ];

    return '${weekdays[time.weekday - 1]}, '
        '${time.day}. ${months[time.month - 1]} ${time.year}';
  }

  @override
  Widget build(BuildContext context) {
    return StreamBuilder<DateTime>(
      stream: _timeStream,
      initialData: DateTime.now(),
      builder: (context, snapshot) {
        final time = snapshot.data!;

        return Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              _formatTime(time),
              style: Theme.of(context).textTheme.displayLarge?.copyWith(
                    fontFamily: 'monospace',
                    fontWeight: FontWeight.bold,
                  ),
            ),
            const SizedBox(height: 8),
            Text(
              _formatDate(time),
              style: Theme.of(context).textTheme.titleLarge,
            ),
          ],
        );
      },
    );
  }
}
```

3.6.2.4 Aufgabe 4: Counter mit StreamController

```
import 'dart:async';

class CounterController {
  final _controller = StreamController<int>.broadcast();
  int _count = 0;

  Stream<int> get stream => _controller.stream;
  int get currentValue => _count;
```

```
  void increment() {
    _count++;
    _controller.add(_count);
  }

  void decrement() {
    _count--;
    _controller.add(_count);
  }

  void reset() {
    _count = 0;
    _controller.add(_count);
  }

  void dispose() {
    _controller.close();
  }
}

class StreamCounterPage extends StatefulWidget {
  const StreamCounterPage({super.key});

  @override
  State<StreamCounterPage> createState() => _StreamCounterPageState();
}

class _StreamCounterPageState extends State<StreamCounterPage> {
  final _counterController = CounterController();

  @override
  void dispose() {
    _counterController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Stream Counter')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            StreamBuilder<int>(
              stream: _counterController.stream,
              initialData: 0,
              builder: (context, snapshot) {
                return Text(
                  '${snapshot.data}',
```

```
                  style: Theme.of(context).textTheme.displayLarge,
                );
              },
            ),
            const SizedBox(height: 32),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                FloatingActionButton(
                  onPressed: _counterController.decrement,
                  heroTag: 'decrement',
                  child: const Icon(Icons.remove),
                ),
                const SizedBox(width: 16),
                FloatingActionButton(
                  onPressed: _counterController.reset,
                  heroTag: 'reset',
                  child: const Icon(Icons.refresh),
                ),
                const SizedBox(width: 16),
                FloatingActionButton(
                  onPressed: _counterController.increment,
                  heroTag: 'increment',
                  child: const Icon(Icons.add),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

3.6.2.5 Aufgabe 5: Dashboard mit kombinierten Futures

```
class DashboardData {
  final int userCount;
  final int orderCount;
  final double revenue;

  DashboardData({
    required this.userCount,
    required this.orderCount,
    required this.revenue,
  });
}

class DashboardPage extends StatefulWidget {
  const DashboardPage({super.key});
```

```
  @override
  State<DashboardPage> createState() => _DashboardPageState();
}

class _DashboardPageState extends State<DashboardPage> {
  late Future<DashboardData> _dataFuture;

  @override
  void initState() {
    super.initState();
    _loadData();
  }

  void _loadData() {
    _dataFuture = _fetchAllData();
  }

  Future<DashboardData> _fetchAllData() async {
    // Parallel ausführen
    final results = await Future.wait([
      _fetchUserCount(),
      _fetchOrderCount(),
      _fetchRevenue(),
    ]);

    return DashboardData(
      userCount: results[0] as int,
      orderCount: results[1] as int,
      revenue: results[2] as double,
    );
  }

  Future<int> _fetchUserCount() async {
    await Future.delayed(const Duration(seconds: 1));
    return 1234;
  }

  Future<int> _fetchOrderCount() async {
    await Future.delayed(const Duration(seconds: 2));
    return 567;
  }

  Future<double> _fetchRevenue() async {
    await Future.delayed(const Duration(milliseconds: 1500));
    return 12345.67;
  }

  void _refresh() {
    setState(() {
      _loadData();
```

```
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Dashboard'),
        actions: [
          IconButton(
            icon: const Icon(Icons.refresh),
            onPressed: _refresh,
          ),
        ],
      ),
      body: FutureBuilder<DashboardData>(
        future: _dataFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const _DashboardSkeleton();
          }

          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }

          final data = snapshot.data!;
          return _DashboardContent(data: data);
        },
      ),
    );
  }
}

class _DashboardSkeleton extends StatelessWidget {
  const _DashboardSkeleton();

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(16),
      child: Column(
        children: [
          Row(
            children: [
              Expanded(child: _SkeletonCard()),
              const SizedBox(width: 16),
              Expanded(child: _SkeletonCard()),
            ],
          ),
          const SizedBox(height: 16),
```

```
          _SkeletonCard(height: 120),
        ],
      ),
    );
  }
}

class _SkeletonCard extends StatelessWidget {
  final double height;

  const _SkeletonCard({this.height = 100});

  @override
  Widget build(BuildContext context) {
    return Container(
      height: height,
      decoration: BoxDecoration(
        color: Colors.grey[300],
        borderRadius: BorderRadius.circular(12),
      ),
    );
  }
}

class _DashboardContent extends StatelessWidget {
  final DashboardData data;

  const _DashboardContent({required this.data});

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(16),
      child: Column(
        children: [
          Row(
            children: [
              Expanded(
                child: _StatCard(
                  title: 'Users',
                  value: data.userCount.toString(),
                  icon: Icons.people,
                  color: Colors.blue,
                ),
              ),
              const SizedBox(width: 16),
              Expanded(
                child: _StatCard(
                  title: 'Orders',
                  value: data.orderCount.toString(),
                  icon: Icons.shopping_cart,
```

```
                  color: Colors.green,
                ),
              ),
            ],
          ),
          const SizedBox(height: 16),
          _StatCard(
            title: 'Revenue',
            value: '€${data.revenue.toStringAsFixed(2)}',
            icon: Icons.euro,
            color: Colors.orange,
          ),
        ],
      ),
    );
  }
}

class _StatCard extends StatelessWidget {
  final String title;
  final String value;
  final IconData icon;
  final Color color;

  const _StatCard({
    required this.title,
    required this.value,
    required this.icon,
    required this.color,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          children: [
            Icon(icon, color: color, size: 32),
            const SizedBox(height: 8),
            Text(title, style: Theme.of(context).textTheme.titleMedium),
            const SizedBox(height: 4),
            Text(
              value,
              style: Theme.of(context).textTheme.headlineMedium?.copyWith(
                    fontWeight: FontWeight.bold,
                  ),
            ),
          ],
        ),
      ),
```

```
    );
  }
}
```

3.6.2.6 Verständnisfragen - Antworten

1. **Warum Future nicht in build()?**
   - `build()` wird bei jedem `setState()` aufgerufen
   - Ein neues Future würde jedes Mal den API-Call erneut starten
   - Das führt zu unnötigen Requests und Flackern
2. **hasData vs. data != null?**
   - `hasData` prüft `data != null` UND ob ein Wert angekommen ist
   - Bei `initialData` ist `hasData` sofort `true`
   - Für die meisten Fälle ist `hasData` die bessere Wahl
3. **Wann initialData?**
   - Wenn ein Default-Wert sofort angezeigt werden soll
   - Um den initialen `waiting`-State zu überspringen
   - Bei Streams mit bekanntem Startwert
4. **Memory Leaks verhindern?**
   - `StreamController.close()` in `dispose()` aufrufen
   - `StreamSubscription.cancel()` bei manuellen Subscriptions
   - Bei Provider/Riverpod: automatisches Lifecycle-Management
5. **Stream-Fehler?**
   - `snapshot.hasError` wird `true`
   - `snapshot.error` enthält die Exception
   - Stream kann danach keine weiteren Events senden (bei Single-Subscription)

3.6.3 Ressourcen

3.6.3.1 Offizielle Dokumentation

- FutureBuilder API
- StreamBuilder API
- AsyncSnapshot API
- Fetch data from the internet

3.6.3.2 Skeleton/Shimmer Packages

- shimmer
- skeletons
- flutter_skeleton_plus
- skeleton_loader

3.6.3.3 Tutorials

- FutureBuilder in Depth
- StreamBuilder Explained
- Skeleton Screens in Flutter
- Async UI Patterns

3.6.3.4 Videos

- FutureBuilder - Widget of the Week

- StreamBuilder - Widget of the Week
- Loading States in Flutter

3.6.3.5 Alternativen

- flutter_hooks: useFuture/useStream
- Riverpod AsyncValue
- BLoC Pattern

3.6.3.6 UX Best Practices

- Skeleton Screens UX Guide
- Loading State Best Practices

3.6.3.7 Zum Vertiefen

- Error Handling with FutureBuilder
- Combining Multiple Futures
- Streams in Dart

3.7 Einheit 3.7: SharedPreferences

3.7.0.1 Lernziele

Nach dieser Einheit kannst du: - Das `shared_preferences` Package verwenden - Einfache Daten persistent speichern - Settings und User-Präferenzen implementieren - Wissen, wann SharedPreferences vs. Datenbank sinnvoll ist

3.7.0.2 1. Setup

Installation

```
# pubspec.yaml
dependencies:
  shared_preferences: ^2.2.2
```

```
flutter pub get
```

Import

```
import 'package:shared_preferences/shared_preferences.dart';
```

3.7.0.3 2. Grundlegende Operationen

Instanz holen

```
// Async - muss awaited werden
final prefs = await SharedPreferences.getInstance();
```

Daten speichern

```
// String
await prefs.setString('username', 'max_mustermann');

// Int
```

```
await prefs.setInt('highscore', 1000);

// Double
await prefs.setDouble('volume', 0.8);

// Bool
await prefs.setBool('darkMode', true);

// String List
await prefs.setStringList('recentSearches', ['flutter', 'dart', 'widgets']);
```

Daten lesen

```
// Mit Default-Wert über ?? Operator
final username = prefs.getString('username') ?? 'Guest';
final highscore = prefs.getInt('highscore') ?? 0;
final volume = prefs.getDouble('volume') ?? 1.0;
final darkMode = prefs.getBool('darkMode') ?? false;
final searches = prefs.getStringList('recentSearches') ?? [];
```

Daten löschen

```
// Einzelnen Key löschen
await prefs.remove('username');

// Alle Daten löschen
await prefs.clear();
```

Prüfen ob Key existiert

```
final hasUsername = prefs.containsKey('username');
```

3.7.0.4 3. Settings Service Pattern

Strukturierte Implementierung

```
class SettingsService {
  static const _darkModeKey = 'darkMode';
  static const _languageKey = 'language';
  static const _notificationsKey = 'notifications';
  static const _fontSizeKey = 'fontSize';

  final SharedPreferences _prefs;

  SettingsService(this._prefs);

  // Factory für async Initialisierung
  static Future<SettingsService> create() async {
    final prefs = await SharedPreferences.getInstance();
    return SettingsService(prefs);
  }

  // Dark Mode
```

```
  bool get darkMode => _prefs.getBool(_darkModeKey) ?? false;
  Future<void> setDarkMode(bool value) =>
      _prefs.setBool(_darkModeKey, value);

  // Language
  String get language => _prefs.getString(_languageKey) ?? 'de';
  Future<void> setLanguage(String value) =>
      _prefs.setString(_languageKey, value);

  // Notifications
  bool get notificationsEnabled =>
      _prefs.getBool(_notificationsKey) ?? true;
  Future<void> setNotificationsEnabled(bool value) =>
      _prefs.setBool(_notificationsKey, value);

  // Font Size
  double get fontSize => _prefs.getDouble(_fontSizeKey) ?? 16.0;
  Future<void> setFontSize(double value) =>
      _prefs.setDouble(_fontSizeKey, value);

  // Reset all settings
  Future<void> resetToDefaults() async {
    await _prefs.remove(_darkModeKey);
    await _prefs.remove(_languageKey);
    await _prefs.remove(_notificationsKey);
    await _prefs.remove(_fontSizeKey);
  }
}
```

Integration mit Provider

```
// main.dart
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  final settingsService = await SettingsService.create();

  runApp(
    Provider.value(
      value: settingsService,
      child: const MyApp(),
    ),
  );
}

// Verwendung
class SettingsPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final settings = context.read<SettingsService>();
```

```
    return SwitchListTile(
      title: const Text('Dark Mode'),
      value: settings.darkMode,
      onChanged: (value) async {
        await settings.setDarkMode(value);
        // UI Update triggern...
      },
    );
  }
}
```

3.7.0.5 4. Mit ChangeNotifier kombinieren

```
class SettingsNotifier extends ChangeNotifier {
  final SharedPreferences _prefs;

  SettingsNotifier(this._prefs);

  static Future<SettingsNotifier> create() async {
    final prefs = await SharedPreferences.getInstance();
    return SettingsNotifier(prefs);
  }

  // Dark Mode
  bool get darkMode => _prefs.getBool('darkMode') ?? false;

  Future<void> setDarkMode(bool value) async {
    await _prefs.setBool('darkMode', value);
    notifyListeners();  // UI wird automatisch aktualisiert
  }

  // Notifications
  bool get notifications => _prefs.getBool('notifications') ?? true;

  Future<void> setNotifications(bool value) async {
    await _prefs.setBool('notifications', value);
    notifyListeners();
  }

  // Language
  String get language => _prefs.getString('language') ?? 'de';

  Future<void> setLanguage(String value) async {
    await _prefs.setString('language', value);
    notifyListeners();
  }
}

// main.dart
void main() async {
```

```
  WidgetsFlutterBinding.ensureInitialized();

  final settingsNotifier = await SettingsNotifier.create();

  runApp(
    ChangeNotifierProvider.value(
      value: settingsNotifier,
      child: const MyApp(),
    ),
  );
}

// App mit Theme-Reaktivität
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Consumer<SettingsNotifier>(
      builder: (context, settings, _) {
        return MaterialApp(
          themeMode: settings.darkMode ? ThemeMode.dark : ThemeMode.light,
          theme: ThemeData.light(),
          darkTheme: ThemeData.dark(),
          home: const HomePage(),
        );
      },
    );
  }
}
```

3.7.0.6 5. Komplexe Objekte speichern

SharedPreferences unterstützt nur primitive Typen. Für komplexe Objekte: JSON verwenden.

```
class UserPreferences {
  final String theme;
  final int fontSize;
  final List<String> favoriteCategories;

  UserPreferences({
    required this.theme,
    required this.fontSize,
    required this.favoriteCategories,
  });

  factory UserPreferences.fromJson(Map<String, dynamic> json) {
    return UserPreferences(
      theme: json['theme'] as String,
      fontSize: json['fontSize'] as int,
      favoriteCategories:
          (json['favoriteCategories'] as List<dynamic>).cast<String>(),
    );
```

```
  }

  Map<String, dynamic> toJson() => {
        'theme': theme,
        'fontSize': fontSize,
        'favoriteCategories': favoriteCategories,
      };

  static UserPreferences get defaults => UserPreferences(
        theme: 'light',
        fontSize: 16,
        favoriteCategories: [],
      );
}

// Speichern und Laden
class PreferencesService {
  static const _key = 'userPreferences';
  final SharedPreferences _prefs;

  PreferencesService(this._prefs);

  UserPreferences get userPreferences {
    final json = _prefs.getString(_key);
    if (json == null) return UserPreferences.defaults;

    try {
      return UserPreferences.fromJson(jsonDecode(json));
    } catch (e) {
      return UserPreferences.defaults;
    }
  }

  Future<void> saveUserPreferences(UserPreferences prefs) async {
    await _prefs.setString(_key, jsonEncode(prefs.toJson()));
  }
}
```

3.7.0.7 6. First Launch / Onboarding

```
class OnboardingService {
  static const _hasSeenOnboardingKey = 'hasSeenOnboarding';
  static const _firstLaunchDateKey = 'firstLaunchDate';

  final SharedPreferences _prefs;

  OnboardingService(this._prefs);

  bool get hasSeenOnboarding =>
      _prefs.getBool(_hasSeenOnboardingKey) ?? false;
```

```
  Future<void> completeOnboarding() async {
    await _prefs.setBool(_hasSeenOnboardingKey, true);
  }

  DateTime? get firstLaunchDate {
    final timestamp = _prefs.getString(_firstLaunchDateKey);
    if (timestamp == null) return null;
    return DateTime.parse(timestamp);
  }

  Future<void> recordFirstLaunch() async {
    if (firstLaunchDate == null) {
      await _prefs.setString(
        _firstLaunchDateKey,
        DateTime.now().toIso8601String(),
      );
    }
  }

  bool get isFirstLaunch => firstLaunchDate == null;
}

// Verwendung
class SplashScreen extends StatefulWidget {
  @override
  State<SplashScreen> createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    _checkOnboarding();
  }

  Future<void> _checkOnboarding() async {
    final prefs = await SharedPreferences.getInstance();
    final onboarding = OnboardingService(prefs);

    await onboarding.recordFirstLaunch();

    if (!mounted) return;

    if (onboarding.hasSeenOnboarding) {
      Navigator.pushReplacementNamed(context, '/home');
    } else {
      Navigator.pushReplacementNamed(context, '/onboarding');
    }
  }
```

```
  @override
  Widget build(BuildContext context) {
    return const Center(child: CircularProgressIndicator());
  }
}
```

3.7.0.8 7. Recent Items / History

```
class SearchHistoryService {
  static const _key = 'searchHistory';
  static const _maxItems = 10;

  final SharedPreferences _prefs;

  SearchHistoryService(this._prefs);

  List<String> get history => _prefs.getStringList(_key) ?? [];

  Future<void> addSearch(String query) async {
    final current = history;

    // Duplikat entfernen
    current.remove(query);

    // Am Anfang einfügen
    current.insert(0, query);

    // Auf max Größe begrenzen
    if (current.length > _maxItems) {
      current.removeLast();
    }

    await _prefs.setStringList(_key, current);
  }

  Future<void> removeSearch(String query) async {
    final current = history;
    current.remove(query);
    await _prefs.setStringList(_key, current);
  }

  Future<void> clearHistory() async {
    await _prefs.remove(_key);
  }
}
```

3.7.0.9 8. Wann SharedPreferences vs. Datenbank?

SharedPreferences verwenden für:

Use Case	Beispiel
App-Einstellungen	Dark Mode, Sprache, Notifications
Einfache Flags	hasSeenOnboarding, isLoggedIn
Kleine Listen	Recent Searches, Favorites (wenige Items)
Einzelne Werte	Highscore, Last Sync Date
Cache-Marker	lastFetchTime für API Cache

Datenbank verwenden für:

Use Case	Beispiel
Strukturierte Daten	Todos, Kontakte, Nachrichten
Große Datenmengen	100+ Items
Komplexe Abfragen	Filtern, Sortieren, Suchen
Relationen	User -> Posts -> Comments
Offline-First Apps	Sync mit Server

Faustregel

```
SharedPreferences: Key-Value, < 50 Items, keine Queries
Datenbank: Strukturiert, viele Items, Queries nötig
```

3.7.0.10 9. Secure Storage Alternative

Für **sensible Daten** (Tokens, Passwörter) besser `flutter_secure_storage`:

```yaml
dependencies:
  flutter_secure_storage: ^9.0.0
```

```dart
import 'package:flutter_secure_storage/flutter_secure_storage.dart';

class SecureStorageService {
  final _storage = const FlutterSecureStorage();

  Future<void> saveToken(String token) async {
    await _storage.write(key: 'auth_token', value: token);
  }

  Future<String?> getToken() async {
    return await _storage.read(key: 'auth_token');
  }

  Future<void> deleteToken() async {
    await _storage.delete(key: 'auth_token');
  }
}
```

3.7.0.11 Zusammenfassung

Methode	Verwendung
`setString/getString`	Text, JSON
`setInt/getInt`	Zahlen
`setBool/getBool`	Flags
`setDouble/getDouble`	Dezimalzahlen
`setStringList/getStringList`	Listen
`remove`	Einzelnen Key löschen
`clear`	Alles löschen

Best Practices: 1. Keys als Konstanten definieren 2. Default-Werte immer angeben 3. Service-Klasse für Kapselung 4. Mit ChangeNotifier für reaktive UI 5. Sensible Daten -> `flutter_secure_storage`

3.7.1 Übung

3.7.1.1 Ziel

Eine Settings-Seite mit persistenten Einstellungen implementieren.

3.7.1.2 Aufgabe 1: Setup & Basics (15 min)

1. Füge `shared_preferences` zu deinem Projekt hinzu
2. Erstelle eine einfache App, die einen Counter persistent speichert
3. Der Counter soll beim App-Neustart den letzten Wert anzeigen

```
// Speichern
await prefs.setInt('counter', value);

// Laden
final value = prefs.getInt('counter') ?? 0;
```

3.7.1.3 Aufgabe 2: SettingsService erstellen (25 min)

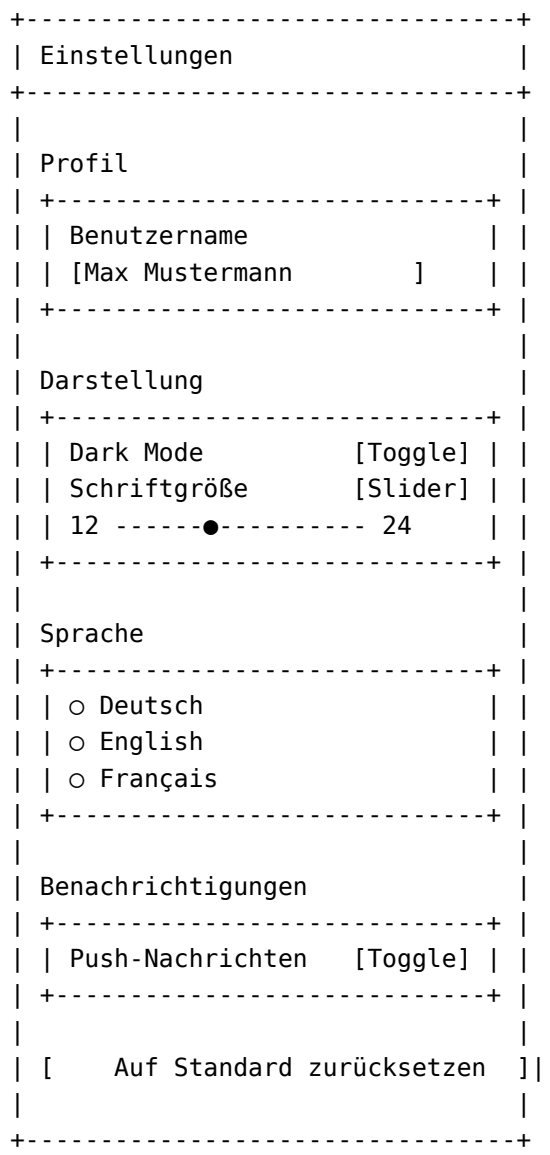
Erstelle einen `SettingsService` mit folgenden Einstellungen:

```
class SettingsService {
  // Keys als Konstanten
  static const _darkModeKey = 'darkMode';
  static const _languageKey = 'language';
  static const _notificationsKey = 'notifications';
  static const _fontSizeKey = 'fontSize';
  static const _usernameKey = 'username';

  // Implementiere:
  // - Getter für jeden Wert (mit sinnvollen Defaults)
  // - Setter für jeden Wert (async)
  // - resetToDefaults() Methode
}
```

Defaults: - darkMode: `false` - language: `'de'` - notifications: `true` - fontSize: `16.0` - username: `'Guest'`

3.7.1.4 Aufgabe 3: Settings UI bauen (30 min)

Erstelle eine Settings-Seite mit:



Anforderungen: - Einstellungen werden sofort gespeichert - Beim Öffnen werden gespeicherte Werte geladen - Dark Mode ändert sofort das App-Theme

3.7.1.5 Aufgabe 4: Mit ChangeNotifier (20 min)

Refaktoriere den `SettingsService` zu einem `SettingsNotifier`:

```
class SettingsNotifier extends ChangeNotifier {
  final SharedPreferences _prefs;

  SettingsNotifier(this._prefs);

  // Bei jedem Setter: notifyListeners() aufrufen
  Future<void> setDarkMode(bool value) async {
    await _prefs.setBool('darkMode', value);
    notifyListeners();  // UI aktualisiert sich automatisch
  }
```

```
}
```

Integriere mit Provider: - App-Theme reagiert auf Dark Mode Änderungen - Settings-Seite zeigt immer aktuelle Werte

3.7.1.6 Aufgabe 5: Search History (20 min)

Implementiere eine Suchhistorie:

```
class SearchHistoryService {
  static const _maxItems = 5;

  // Implementiere:
  List<String> get history;
  Future<void> addSearch(String query);
  Future<void> removeSearch(String query);
  Future<void> clearHistory();
}
```

Erstelle eine Such-UI: - TextField für Suche - Liste der letzten Suchen - Tippen auf Item -> In TextField übernehmen - Swipe zum Löschen - “Verlauf löschen” Button

3.7.1.7 Aufgabe 6: Onboarding Flow (20 min)

Implementiere einen “First Launch” Flow:

1. Beim ersten Start: Zeige Onboarding-Screens
2. Nach Onboarding: Speichere `hasSeenOnboarding = true`
3. Bei weiteren Starts: Direkt zur Home-Seite

```
class OnboardingService {
  bool get hasSeenOnboarding;
  Future<void> completeOnboarding();
  Future<void> resetOnboarding();  // Für Testing
}
```

Onboarding-Screens (3 Seiten mit PageView): 1. “Willkommen bei der App” 2. “Entdecke Features” 3. “Los geht’s!” mit Button zum Abschließen

3.7.1.8 Aufgabe 7: Verständnisfragen

1. Warum muss `SharedPreferences.getInstance()` awaited werden?
2. Was passiert, wenn man `getString('nonexistent')` aufruft?
3. Warum sollte man sensible Daten (Passwörter, Tokens) NICHT in SharedPreferences speichern?
4. Wann würdest du SharedPreferences verwenden, wann eine Datenbank?
5. Warum `WidgetsFlutterBinding.ensureInitialized()` vor `SharedPreferences.getInstance()`?

3.7.1.9 Bonus: Themes mit Persistenz

Erweitere die App um ein Theme-System:

```
enum AppTheme {
  light,
  dark,
  blue,
  green,
  purple;

  ThemeData get themeData {
    switch (this) {
      case AppTheme.light:
        return ThemeData.light();
      case AppTheme.dark:
        return ThemeData.dark();
      case AppTheme.blue:
        return ThemeData(colorScheme: ColorScheme.fromSeed(seedColor:
 ↪  Colors.blue));
      // ...
    }
  }
}
```

- Theme-Auswahl in Settings
- Theme wird persistent gespeichert
- App startet mit gespeichertem Theme

3.7.1.10 Abgabe-Checkliste

- ☐ SharedPreferences Package installiert
- ☐ Counter-App mit Persistenz
- ☐ SettingsService mit allen Methoden
- ☐ Settings-UI vollständig
- ☐ ChangeNotifier-Integration
- ☐ SearchHistory funktioniert
- ☐ Onboarding-Flow implementiert
- ☐ Verständnisfragen beantwortet

3.7.2 Lösung

3.7.2.1 Aufgabe 2: SettingsService

```
import 'package:shared_preferences/shared_preferences.dart';

class SettingsService {
  static const _darkModeKey = 'darkMode';
  static const _languageKey = 'language';
  static const _notificationsKey = 'notifications';
  static const _fontSizeKey = 'fontSize';
  static const _usernameKey = 'username';

  final SharedPreferences _prefs;
```

```
  SettingsService(this._prefs);

  static Future<SettingsService> create() async {
    final prefs = await SharedPreferences.getInstance();
    return SettingsService(prefs);
  }

  // Dark Mode
  bool get darkMode => _prefs.getBool(_darkModeKey) ?? false;
  Future<void> setDarkMode(bool value) => _prefs.setBool(_darkModeKey, value);

  // Language
  String get language => _prefs.getString(_languageKey) ?? 'de';
  Future<void> setLanguage(String value) =>
      _prefs.setString(_languageKey, value);

  // Notifications
  bool get notifications => _prefs.getBool(_notificationsKey) ?? true;
  Future<void> setNotifications(bool value) =>
      _prefs.setBool(_notificationsKey, value);

  // Font Size
  double get fontSize => _prefs.getDouble(_fontSizeKey) ?? 16.0;
  Future<void> setFontSize(double value) =>
      _prefs.setDouble(_fontSizeKey, value);

  // Username
  String get username => _prefs.getString(_usernameKey) ?? 'Guest';
  Future<void> setUsername(String value) =>
      _prefs.setString(_usernameKey, value);

  // Reset
  Future<void> resetToDefaults() async {
    await _prefs.remove(_darkModeKey);
    await _prefs.remove(_languageKey);
    await _prefs.remove(_notificationsKey);
    await _prefs.remove(_fontSizeKey);
    await _prefs.remove(_usernameKey);
  }
}
```

3.7.2.2 Aufgabe 4: SettingsNotifier

```
import 'package:flutter/foundation.dart';
import 'package:shared_preferences/shared_preferences.dart';

class SettingsNotifier extends ChangeNotifier {
  static const _darkModeKey = 'darkMode';
  static const _languageKey = 'language';
  static const _notificationsKey = 'notifications';
```

```
  static const _fontSizeKey = 'fontSize';
  static const _usernameKey = 'username';

  final SharedPreferences _prefs;

  SettingsNotifier(this._prefs);

  static Future<SettingsNotifier> create() async {
    final prefs = await SharedPreferences.getInstance();
    return SettingsNotifier(prefs);
  }

  // Dark Mode
  bool get darkMode => _prefs.getBool(_darkModeKey) ?? false;

  Future<void> setDarkMode(bool value) async {
    await _prefs.setBool(_darkModeKey, value);
    notifyListeners();
  }

  // Language
  String get language => _prefs.getString(_languageKey) ?? 'de';

  Future<void> setLanguage(String value) async {
    await _prefs.setString(_languageKey, value);
    notifyListeners();
  }

  // Notifications
  bool get notifications => _prefs.getBool(_notificationsKey) ?? true;

  Future<void> setNotifications(bool value) async {
    await _prefs.setBool(_notificationsKey, value);
    notifyListeners();
  }

  // Font Size
  double get fontSize => _prefs.getDouble(_fontSizeKey) ?? 16.0;

  Future<void> setFontSize(double value) async {
    await _prefs.setDouble(_fontSizeKey, value);
    notifyListeners();
  }

  // Username
  String get username => _prefs.getString(_usernameKey) ?? 'Guest';

  Future<void> setUsername(String value) async {
    await _prefs.setString(_usernameKey, value);
    notifyListeners();
  }
```

```
  // Reset
  Future<void> resetToDefaults() async {
    await _prefs.remove(_darkModeKey);
    await _prefs.remove(_languageKey);
    await _prefs.remove(_notificationsKey);
    await _prefs.remove(_fontSizeKey);
    await _prefs.remove(_usernameKey);
    notifyListeners();
  }
}
```

3.7.2.3 Settings Page UI

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

class SettingsPage extends StatelessWidget {
  const SettingsPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Einstellungen'),
      ),
      body: Consumer<SettingsNotifier>(
        builder: (context, settings, _) {
          return ListView(
            padding: const EdgeInsets.all(16),
            children: [
              // Profil
              _SectionHeader(title: 'Profil'),
              Card(
                child: Padding(
                  padding: const EdgeInsets.all(16),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      const Text('Benutzername'),
                      const SizedBox(height: 8),
                      TextFormField(
                        initialValue: settings.username,
                        decoration: const InputDecoration(
                          border: OutlineInputBorder(),
                        ),
                        onFieldSubmitted: (value) {
                          settings.setUsername(value);
                        },
                      ),
```

```
          ],
        ),
      ),
    ),
    const SizedBox(height: 16),

    // Darstellung
    _SectionHeader(title: 'Darstellung'),
    Card(
      child: Column(
        children: [
          SwitchListTile(
            title: const Text('Dark Mode'),
            value: settings.darkMode,
            onChanged: (value) => settings.setDarkMode(value),
          ),
          const Divider(height: 1),
          ListTile(
            title: const Text('Schriftgröße'),
            subtitle: Slider(
              value: settings.fontSize,
              min: 12,
              max: 24,
              divisions: 12,
              label: settings.fontSize.round().toString(),
              onChanged: (value) => settings.setFontSize(value),
            ),
          ),
        ],
      ),
    ),
    const SizedBox(height: 16),

    // Sprache
    _SectionHeader(title: 'Sprache'),
    Card(
      child: Column(
        children: [
          RadioListTile<String>(
            title: const Text('Deutsch'),
            value: 'de',
            groupValue: settings.language,
            onChanged: (value) => settings.setLanguage(value!),
          ),
          RadioListTile<String>(
            title: const Text('English'),
            value: 'en',
            groupValue: settings.language,
            onChanged: (value) => settings.setLanguage(value!),
          ),
          RadioListTile<String>(
```

```
                    title: const Text('Français'),
                    value: 'fr',
                    groupValue: settings.language,
                    onChanged: (value) => settings.setLanguage(value!),
                  ),
                ],
              ),
            ),
            const SizedBox(height: 16),

            // Benachrichtigungen
            _SectionHeader(title: 'Benachrichtigungen'),
            Card(
              child: SwitchListTile(
                title: const Text('Push-Nachrichten'),
                value: settings.notifications,
                onChanged: (value) => settings.setNotifications(value),
              ),
            ),
            const SizedBox(height: 24),

            // Reset Button
            OutlinedButton(
              onPressed: () => _showResetDialog(context, settings),
              child: const Text('Auf Standard zurücksetzen'),
            ),
          ],
        );
      },
    ),
  );
}

void _showResetDialog(BuildContext context, SettingsNotifier settings) {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text('Zurücksetzen?'),
      content: const Text(
        'Alle Einstellungen werden auf die Standardwerte zurückgesetzt.',
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('Abbrechen'),
        ),
        ElevatedButton(
          onPressed: () {
            settings.resetToDefaults();
            Navigator.pop(context);
            ScaffoldMessenger.of(context).showSnackBar(
```

```
                const SnackBar(content: Text('Einstellungen zurückgesetzt')),
              );
            },
            child: const Text('Zurücksetzen'),
          ),
        ],
      ),
    );
  }
}

class _SectionHeader extends StatelessWidget {
  final String title;

  const _SectionHeader({required this.title});

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(left: 4, bottom: 8),
      child: Text(
        title,
        style: Theme.of(context).textTheme.titleMedium?.copyWith(
              fontWeight: FontWeight.bold,
            ),
      ),
    );
  }
}
```

3.7.2.4 Aufgabe 5: SearchHistory

```
class SearchHistoryService {
  static const _key = 'searchHistory';
  static const _maxItems = 5;

  final SharedPreferences _prefs;

  SearchHistoryService(this._prefs);

  List<String> get history => _prefs.getStringList(_key) ?? [];

  Future<void> addSearch(String query) async {
    if (query.trim().isEmpty) return;

    final current = history.toList();
    current.remove(query);
    current.insert(0, query.trim());

    if (current.length > _maxItems) {
```

```
      current.removeLast();
    }

    await _prefs.setStringList(_key, current);
  }

  Future<void> removeSearch(String query) async {
    final current = history.toList();
    current.remove(query);
    await _prefs.setStringList(_key, current);
  }

  Future<void> clearHistory() async {
    await _prefs.remove(_key);
  }
}

// UI
class SearchPage extends StatefulWidget {
  const SearchPage({super.key});

  @override
  State<SearchPage> createState() => _SearchPageState();
}

class _SearchPageState extends State<SearchPage> {
  late SearchHistoryService _historyService;
  final _controller = TextEditingController();

  @override
  void initState() {
    super.initState();
    _initHistory();
  }

  Future<void> _initHistory() async {
    final prefs = await SharedPreferences.getInstance();
    setState(() {
      _historyService = SearchHistoryService(prefs);
    });
  }

  void _search() {
    final query = _controller.text;
    if (query.isNotEmpty) {
      _historyService.addSearch(query);
      setState(() {});
      // Perform actual search...
    }
  }
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Suche')),
    body: Column(
      children: [
        Padding(
          padding: const EdgeInsets.all(16),
          child: TextField(
            controller: _controller,
            decoration: InputDecoration(
              hintText: 'Suchen...',
              suffixIcon: IconButton(
                icon: const Icon(Icons.search),
                onPressed: _search,
              ),
            ),
            onSubmitted: (_) => _search(),
          ),
        ),
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 16),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              const Text('Letzte Suchen'),
              TextButton(
                onPressed: () {
                  _historyService.clearHistory();
                  setState(() {});
                },
                child: const Text('Löschen'),
              ),
            ],
          ),
        ),
        Expanded(
          child: ListView.builder(
            itemCount: _historyService.history.length,
            itemBuilder: (context, index) {
              final query = _historyService.history[index];
              return Dismissible(
                key: ValueKey(query),
                onDismissed: (_) {
                  _historyService.removeSearch(query);
                  setState(() {});
                },
                child: ListTile(
                  leading: const Icon(Icons.history),
                  title: Text(query),
                  onTap: () {
```

```
                    _controller.text = query;
                  },
                ),
              );
            },
          ),
        ),
      ],
    ),
  );
  }
}
```

3.7.2.5 Aufgabe 6: Onboarding

```
class OnboardingService {
  static const _key = 'hasSeenOnboarding';

  final SharedPreferences _prefs;

  OnboardingService(this._prefs);

  bool get hasSeenOnboarding => _prefs.getBool(_key) ?? false;

  Future<void> completeOnboarding() async {
    await _prefs.setBool(_key, true);
  }

  Future<void> resetOnboarding() async {
    await _prefs.remove(_key);
  }
}

class OnboardingPage extends StatefulWidget {
  const OnboardingPage({super.key});

  @override
  State<OnboardingPage> createState() => _OnboardingPageState();
}

class _OnboardingPageState extends State<OnboardingPage> {
  final _pageController = PageController();
  int _currentPage = 0;

  final _pages = [
    const _OnboardingContent(
      icon: Icons.flutter_dash,
      title: 'Willkommen!',
      description: 'Entdecke die Möglichkeiten unserer App.',
    ),
```

```
  const _OnboardingContent(
    icon: Icons.star,
    title: 'Tolle Features',
    description: 'Nutze alle Funktionen für deinen Erfolg.',
  ),
  const _OnboardingContent(
    icon: Icons.rocket_launch,
    title: 'Los geht\'s!',
    description: 'Starte jetzt durch.',
  ),
];

Future<void> _complete() async {
  final prefs = await SharedPreferences.getInstance();
  final onboarding = OnboardingService(prefs);
  await onboarding.completeOnboarding();

  if (mounted) {
    Navigator.pushReplacementNamed(context, '/home');
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SafeArea(
      child: Column(
        children: [
          Expanded(
            child: PageView.builder(
              controller: _pageController,
              itemCount: _pages.length,
              onPageChanged: (index) {
                setState(() => _currentPage = index);
              },
              itemBuilder: (_, index) => _pages[index],
            ),
          ),
          // Dots Indicator
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: List.generate(
              _pages.length,
              (index) => Container(
                margin: const EdgeInsets.all(4),
                width: _currentPage == index ? 24 : 8,
                height: 8,
                decoration: BoxDecoration(
                  color: _currentPage == index
                      ? Theme.of(context).primaryColor
                      : Colors.grey,
```

```
                    borderRadius: BorderRadius.circular(4),
                  ),
                ),
              ),
            ),
            const SizedBox(height: 24),
            // Button
            Padding(
              padding: const EdgeInsets.all(24),
              child: SizedBox(
                width: double.infinity,
                child: ElevatedButton(
                  onPressed: _currentPage == _pages.length - 1
                      ? _complete
                      : () => _pageController.nextPage(
                            duration: const Duration(milliseconds: 300),
                            curve: Curves.easeInOut,
                          ),
                  child: Text(
                    _currentPage == _pages.length - 1 ? 'Starten' : 'Weiter',
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

class _OnboardingContent extends StatelessWidget {
  final IconData icon;
  final String title;
  final String description;

  const _OnboardingContent({
    required this.icon,
    required this.title,
    required this.description,
  });

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(48),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(icon, size: 120, color: Theme.of(context).primaryColor),
          const SizedBox(height: 48),
```

```
          Text(title, style: Theme.of(context).textTheme.headlineMedium),
          const SizedBox(height: 16),
          Text(description, textAlign: TextAlign.center),
        ],
      ),
    );
  }
}
```

3.7.2.6 Verständnisfragen - Antworten

1. **Warum await bei getInstance()?**
   - SharedPreferences muss Platform-spezifischen Code aufrufen (Android/iOS)
   - Der Zugriff auf den Dateisystem ist asynchron
   - Die Instanz muss erst initialisiert werden
2. **getString('nonexistent')?**
   - Gibt `null` zurück
   - Deshalb immer Default-Wert mit `?? 'default'` angeben
3. **Warum keine sensiblen Daten?**
   - SharedPreferences speichert im Klartext
   - Auf gerooteten Geräten lesbar
   - Kein verschlüsseltes Storage
   - Besser: `flutter_secure_storage` für Tokens/Passwörter
4. **SharedPreferences vs. Datenbank?**
   - SharedPreferences: Einfache Key-Value Paare, wenige Einträge, keine Queries
   - Datenbank: Strukturierte Daten, viele Einträge, komplexe Abfragen
5. **Warum ensureInitialized()?**
   - `SharedPreferences.getInstance()` nutzt Platform Channels
   - Diese erfordern, dass Flutter Binding initialisiert ist
   - Ohne Binding: Crash beim Zugriff auf native Plattform

3.7.3 Ressourcen

3.7.3.1 Offizielle Dokumentation

- shared_preferences Package
- Store key-value data on disk (Flutter Cookbook)
- Data persistence overview

3.7.3.2 Sichere Speicherung

- flutter_secure_storage
- Security Best Practices

3.7.3.3 Alternative Packages

- hive - Schnelle NoSQL Datenbank
- get_storage - Alternative zu SharedPreferences
- isar - Moderne NoSQL Datenbank

3.7.3.4 Tutorials

- SharedPreferences Complete Guide

- User Settings with SharedPreferences
- Dark Mode with SharedPreferences

3.7.3.5 Videos

- SharedPreferences Tutorial
- Persist Data in Flutter
- Settings Screen Implementation

3.7.3.6 Best Practices

- State Management and Persistence
- When to use SharedPreferences vs SQLite

3.7.3.7 Zum Vertiefen

- Encrypted SharedPreferences
- SharedPreferences Mock for Testing
- Platform-specific storage

3.8 Einheit 3.8: Lokale Datenbanken

3.8.0.1 Lernziele

Nach dieser Einheit kannst du: - `sqflite` für SQLite-Datenbanken verwenden - `hive` als NoSQL-Alternative einsetzen - CRUD-Operationen implementieren - Die richtige Datenbank für deinen Use Case wählen

3.8.0.2 1. Übersicht: sqflite vs. Hive

Feature	sqflite (SQLite)	Hive (NoSQL)
Typ	Relational	Key-Value / Document
Query-Sprache	SQL	Dart API
Schema	Ja (Tabellen)	Schema-frei
Performance	Gut	Sehr schnell
Komplexe Queries	Ja (JOINs, etc.)	Begrenzt
Learning Curve	Mittel	Niedrig
Best für	Strukturierte, relationale Daten	Einfache Objekte, schneller Zugriff

3.8.0.3 2. sqflite - Setup

Installation

```
dependencies:
  sqflite: ^2.3.2
  path: ^1.8.3
```

Datenbank öffnen

```
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';
```

```
class DatabaseHelper {
  static Database? _database;

  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await _initDatabase();
    return _database!;
  }

  Future<Database> _initDatabase() async {
    final dbPath = await getDatabasesPath();
    final path = join(dbPath, 'app_database.db');

    return await openDatabase(
      path,
      version: 1,
      onCreate: _onCreate,
      onUpgrade: _onUpgrade,
    );
  }

  Future<void> _onCreate(Database db, int version) async {
    await db.execute('''
      CREATE TABLE todos (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        title TEXT NOT NULL,
        description TEXT,
        completed INTEGER DEFAULT 0,
        created_at TEXT DEFAULT CURRENT_TIMESTAMP
      )
    ''');
  }

  Future<void> _onUpgrade(Database db, int oldVersion, int newVersion) async {
    if (oldVersion < 2) {
      await db.execute('ALTER TABLE todos ADD COLUMN priority INTEGER DEFAULT
 ↪  0');
    }
  }
}
```

3.8.0.4 3. sqflite - CRUD Operationen

Model-Klasse

```
class Todo {
  final int? id;
  final String title;
  final String? description;
  final bool completed;
```

```
  final DateTime createdAt;

  Todo({
    this.id,
    required this.title,
    this.description,
    this.completed = false,
    DateTime? createdAt,
  }) : createdAt = createdAt ?? DateTime.now();

  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'title': title,
      'description': description,
      'completed': completed ? 1 : 0,
      'created_at': createdAt.toIso8601String(),
    };
  }

  factory Todo.fromMap(Map<String, dynamic> map) {
    return Todo(
      id: map['id'] as int,
      title: map['title'] as String,
      description: map['description'] as String?,
      completed: map['completed'] == 1,
      createdAt: DateTime.parse(map['created_at'] as String),
    );
  }

  Todo copyWith({
    int? id,
    String? title,
    String? description,
    bool? completed,
  }) {
    return Todo(
      id: id ?? this.id,
      title: title ?? this.title,
      description: description ?? this.description,
      completed: completed ?? this.completed,
      createdAt: createdAt,
    );
  }
}
```

Repository Pattern

```
class TodoRepository {
  final DatabaseHelper _dbHelper;
```

```
TodoRepository(this._dbHelper);

// CREATE
Future<int> insert(Todo todo) async {
  final db = await _dbHelper.database;
  return await db.insert('todos', todo.toMap());
}

// READ ALL
Future<List<Todo>> getAll() async {
  final db = await _dbHelper.database;
  final maps = await db.query('todos', orderBy: 'created_at DESC');
  return maps.map((map) => Todo.fromMap(map)).toList();
}

// READ ONE
Future<Todo?> getById(int id) async {
  final db = await _dbHelper.database;
  final maps = await db.query(
    'todos',
    where: 'id = ?',
    whereArgs: [id],
    limit: 1,
  );
  if (maps.isEmpty) return null;
  return Todo.fromMap(maps.first);
}

// UPDATE
Future<int> update(Todo todo) async {
  final db = await _dbHelper.database;
  return await db.update(
    'todos',
    todo.toMap(),
    where: 'id = ?',
    whereArgs: [todo.id],
  );
}

// DELETE
Future<int> delete(int id) async {
  final db = await _dbHelper.database;
  return await db.delete(
    'todos',
    where: 'id = ?',
    whereArgs: [id],
  );
}

// QUERY mit Filter
Future<List<Todo>> getByCompleted(bool completed) async {
```

```
    final db = await _dbHelper.database;
    final maps = await db.query(
      'todos',
      where: 'completed = ?',
      whereArgs: [completed ? 1 : 0],
    );
    return maps.map((map) => Todo.fromMap(map)).toList();
  }

  // COUNT
  Future<int> count() async {
    final db = await _dbHelper.database;
    final result = await db.rawQuery('SELECT COUNT(*) as count FROM todos');
    return result.first['count'] as int;
  }
}
```

3.8.0.5 4. sqflite - Komplexe Queries

```
// Raw Query
Future<List<Todo>> search(String query) async {
  final db = await _dbHelper.database;
  final maps = await db.rawQuery(
    'SELECT * FROM todos WHERE title LIKE ? OR description LIKE ?',
    ['%$query%', '%$query%'],
  );
  return maps.map((map) => Todo.fromMap(map)).toList();
}

// Aggregationen
Future<Map<String, int>> getStats() async {
  final db = await _dbHelper.database;
  final result = await db.rawQuery('''
    SELECT
      COUNT(*) as total,
      SUM(CASE WHEN completed = 1 THEN 1 ELSE 0 END) as completed
    FROM todos
  ''');

  return {
    'total': result.first['total'] as int,
    'completed': result.first['completed'] as int? ?? 0,
  };
}

// Transaction
Future<void> completeAll() async {
  final db = await _dbHelper.database;
  await db.transaction((txn) async {
    await txn.update('todos', {'completed': 1});
```

```
  });
}

// Batch Operations
Future<void> insertMany(List<Todo> todos) async {
  final db = await _dbHelper.database;
  final batch = db.batch();

  for (final todo in todos) {
    batch.insert('todos', todo.toMap());
  }

  await batch.commit(noResult: true);
}
```

3.8.0.6 5. Hive - Setup

Installation

```
dependencies:
  hive: ^2.2.3
  hive_flutter: ^1.1.0

dev_dependencies:
  hive_generator: ^2.0.1
  build_runner: ^2.4.8
```

Initialisierung

```
import 'package:hive_flutter/hive_flutter.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Hive.initFlutter();

  // Adapter registrieren (nach Code-Generierung)
  Hive.registerAdapter(TodoAdapter());

  // Box öffnen
  await Hive.openBox<Todo>('todos');

  runApp(const MyApp());
}
```

3.8.0.7 6. Hive - Model mit TypeAdapter

```
import 'package:hive/hive.dart';

part 'todo.g.dart';
```

```
@HiveType(typeId: 0)
class Todo extends HiveObject {
  @HiveField(0)
  final String id;

  @HiveField(1)
  String title;

  @HiveField(2)
  String? description;

  @HiveField(3)
  bool completed;

  @HiveField(4)
  DateTime createdAt;

  Todo({
    required this.id,
    required this.title,
    this.description,
    this.completed = false,
    DateTime? createdAt,
  }) : createdAt = createdAt ?? DateTime.now();
}
```

Nach `flutter pub run build_runner build` wird `todo.g.dart` generiert.

3.8.0.8 7. Hive - CRUD Operationen

```
class TodoRepository {
  static const _boxName = 'todos';

  Box<Todo> get _box => Hive.box<Todo>(_boxName);

  // CREATE
  Future<void> add(Todo todo) async {
    await _box.put(todo.id, todo);
  }

  // READ ALL
  List<Todo> getAll() {
    return _box.values.toList();
  }

  // READ ONE
  Todo? getById(String id) {
    return _box.get(id);
  }

  // UPDATE
```

```
  Future<void> update(Todo todo) async {
    await todo.save();  // HiveObject bietet save()
    // Oder: await _box.put(todo.id, todo);
  }

  // DELETE
  Future<void> delete(String id) async {
    await _box.delete(id);
  }

  // DELETE ALL
  Future<void> deleteAll() async {
    await _box.clear();
  }

  // Filter
  List<Todo> getCompleted() {
    return _box.values.where((todo) => todo.completed).toList();
  }

  List<Todo> getPending() {
    return _box.values.where((todo) => !todo.completed).toList();
  }

  // Count
  int get count => _box.length;

  // Listen to changes
  Stream<BoxEvent> watch() {
    return _box.watch();
  }
}
```

3.8.0.9 8. Hive - Reactive mit ValueListenableBuilder

```
class TodoListPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Todos')),
      body: ValueListenableBuilder<Box<Todo>>(
        valueListenable: Hive.box<Todo>('todos').listenable(),
        builder: (context, box, _) {
          final todos = box.values.toList();

          if (todos.isEmpty) {
            return const Center(child: Text('Keine Todos'));
          }

          return ListView.builder(
```

```
            itemCount: todos.length,
            itemBuilder: (context, index) {
              final todo = todos[index];
              return ListTile(
                title: Text(todo.title),
                leading: Checkbox(
                  value: todo.completed,
                  onChanged: (value) {
                    todo.completed = value!;
                    todo.save();
                  },
                ),
                trailing: IconButton(
                  icon: const Icon(Icons.delete),
                  onPressed: () => todo.delete(),
                ),
              );
            },
          );
        },
      ),
    );
  }
}
```

3.8.0.10 9. path_provider

Für Custom-Pfade:

```
dependencies:
  path_provider: ^2.1.2
```

```
import 'package:path_provider/path_provider.dart';

// App-Dokumente (persistent)
final docDir = await getApplicationDocumentsDirectory();
final dbPath = '${docDir.path}/my_database.db';

// Temporäre Dateien
final tempDir = await getTemporaryDirectory();

// App-Support (iOS: Nicht sichtbar für User)
final supportDir = await getApplicationSupportDirectory();
```

3.8.0.11 10. Vergleich der Ansätze

Wann sqflite?

- Komplexe Datenstrukturen mit Relationen
- SQL-Kenntnisse vorhanden
- Komplexe Queries (JOINs, Aggregationen)
- Migration von anderen SQL-Datenbanken

- Große Datenmengen mit Indizes

```
// Beispiel: Relationale Daten
// Users -> Posts -> Comments
await db.rawQuery('''
  SELECT p.*, u.name as author_name
  FROM posts p
  JOIN users u ON p.user_id = u.id
  WHERE p.id = ?
''', [postId]);
```

Wann Hive?

- Einfache Objekt-Speicherung
- Schneller Zugriff ohne Queries
- Kein SQL-Wissen nötig
- Reactive UI mit listenable()
- Offline-Cache

```
// Beispiel: Cache
final box = Hive.box<User>('userCache');
box.put('current_user', user);
final cachedUser = box.get('current_user');
```

3.8.0.12 11. Alternativen

Package	Beschreibung
`isar`	Schnelle NoSQL DB, ersetzt Hive
`drift`	Type-safe SQL mit Code-Gen
`objectbox`	NoSQL, sehr performant
`floor`	Room-ähnlich (Android), Type-safe
`sembast`	Simple NoSQL

3.8.0.13 Zusammenfassung

Operation	sqflite	Hive
Setup	`openDatabase()`	`Hive.initFlutter()`
Insert	`db.insert()`	`box.put()`
Read	`db.query()`	`box.get()` / `box.values`
Update	`db.update()`	`object.save()`
Delete	`db.delete()`	`box.delete()`
Watch	Nicht eingebaut	`box.listenable()`

Empfehlung: - Einfache Apps: **Hive** - Komplexe Daten/Queries: **sqflite** - Modern mit Type-Safety: **drift** oder **isar**

3.8.1 Übung

3.8.1.1 Ziel

Eine Notizen-App mit lokaler Datenbank implementieren.

3.8.1.2 Aufgabe 1: sqflite Setup (20 min)

1. Füge `sqflite` und `path` zu deinem Projekt hinzu
2. Erstelle einen `DatabaseHelper` mit:
   - Singleton-Pattern
   - Datenbank-Initialisierung
   - `onCreate` mit Tabellen-Erstellung

Tabellen-Schema für Notes:

```
CREATE TABLE notes (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  title TEXT NOT NULL,
  content TEXT,
  color INTEGER DEFAULT 0,
  is_pinned INTEGER DEFAULT 0,
  created_at TEXT,
  updated_at TEXT
);
```

3.8.1.3 Aufgabe 2: Note Model & Repository (25 min)

Erstelle das Note Model:

```
class Note {
  final int? id;
  final String title;
  final String content;
  final int color;  // Farb-Index (0-5)
  final bool isPinned;
  final DateTime createdAt;
  final DateTime updatedAt;

  // Implementiere:
  // - Konstruktor
  // - toMap()
  // - fromMap()
  // - copyWith()
}
```

Erstelle das `NoteRepository` mit: - `insert(Note note)` - `getAll()` - `getById(int id)` - `update(Note note)` - `delete(int id)` - `search(String query)` - Suche in title und content - `getPinned()` - Nur angepinnte Notizen

3.8.1.4 Aufgabe 3: Notes UI mit sqflite (30 min)

Erstelle eine Notizen-App:

```
+---------------------------------+
| Notizen                 [□]   |
+---------------------------------+
| Angepinnt                       |
| +-----------------------------+ |
| | □ Wichtige Notiz          | |
```

```
| | Text hier...                | |
| +---------------------------+ |
|                               |
| Alle Notizen                  |
| +---------------------------+ |
| | Einkaufsliste             | |
| | Milch, Brot, Eier...      | |
| +---------------------------+ |
| +---------------------------+ |
| | Meeting Notes             | |
| | Projekt besprechen...     | |
| +---------------------------+ |
|                          [+]  |
+-------------------------------+
```

Features: - Notizen erstellen/bearbeiten - Notizen löschen (Swipe oder Long Press) - Notizen anpinnen - Farbauswahl (6 Farben) - Suche

3.8.1.5 Aufgabe 4: Hive Alternative (25 min)

Implementiere dieselbe Notizen-App mit Hive:

1. Setup:

```yaml
dependencies:
  hive: ^2.2.3
  hive_flutter: ^1.1.0
dev_dependencies:
  hive_generator: ^2.0.1
  build_runner: ^2.4.8
```

2. Note Model mit Hive Annotations:

```
@HiveType(typeId: 0)
class Note extends HiveObject {
  @HiveField(0)
  late String id;

  @HiveField(1)
  late String title;

  // ...
}
```

3. Code generieren: `flutter pub run build_runner build`
4. Repository mit Hive implementieren
5. UI mit `ValueListenableBuilder` für reactive Updates

3.8.1.6 Aufgabe 5: Vergleich (15 min)

Dokumentiere deine Erfahrungen:

Kriterium	sqflite	Hive
Setup-Aufwand		
Code-Menge		
Performance (gefühlt)		
Flexibilität		
Reaktive UI		
Lernkurve		

Beantworte: 1. Welche Lösung würdest du für diese App bevorzugen? Warum? 2. Wann würdest du sqflite wählen? 3. Wann würdest du Hive wählen?

3.8.1.7 Aufgabe 6: Bonus - Kategorien (Optional)

Erweitere die App um Kategorien:

sqflite:

```
CREATE TABLE categories (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  color INTEGER
);

-- Notes hat jetzt category_id
ALTER TABLE notes ADD COLUMN category_id INTEGER REFERENCES categories(id);
```

Features: - Kategorien erstellen/bearbeiten/löschen - Notizen einer Kategorie zuweisen - Filter nach Kategorie - JOIN-Query für Notizen mit Kategorie-Namen

3.8.1.8 Aufgabe 7: Migration (Optional)

Implementiere eine Datenbank-Migration:

Version 1 -> Version 2: - Neues Feld `is_archived` hinzufügen

```
Future<void> _onUpgrade(Database db, int oldVersion, int newVersion) async {
  if (oldVersion < 2) {
    await db.execute(
      'ALTER TABLE notes ADD COLUMN is_archived INTEGER DEFAULT 0',
    );
  }
}
```

Teste: 1. App mit Version 1 installieren 2. Einige Notizen erstellen 3. Version auf 2 erhöhen 4. App neu starten -> Migration sollte laufen 5. Alte Daten müssen erhalten bleiben

3.8.1.9 Abgabe-Checkliste

- ☐ sqflite Setup funktioniert
- ☐ Note Model mit toMap/fromMap
- ☐ NoteRepository mit allen CRUD-Methoden
- ☐ Notes UI vollständig
- ☐ Suche funktioniert

- ☐ Pinnen funktioniert
- ☐ Hive-Alternative implementiert
- ☐ Vergleich dokumentiert
- ☐ (Bonus) Kategorien
- ☐ (Bonus) Migration

3.8.2 Lösung

3.8.2.1 sqflite Lösung

database_helper.dart

```
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class DatabaseHelper {
  static final DatabaseHelper _instance = DatabaseHelper._internal();
  static Database? _database;

  factory DatabaseHelper() => _instance;

  DatabaseHelper._internal();

  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await _initDatabase();
    return _database!;
  }

  Future<Database> _initDatabase() async {
    final dbPath = await getDatabasesPath();
    final path = join(dbPath, 'notes.db');

    return await openDatabase(
      path,
      version: 1,
      onCreate: _onCreate,
    );
  }

  Future<void> _onCreate(Database db, int version) async {
    await db.execute('''
      CREATE TABLE notes (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        title TEXT NOT NULL,
        content TEXT,
        color INTEGER DEFAULT 0,
        is_pinned INTEGER DEFAULT 0,
        created_at TEXT,
        updated_at TEXT
      )
    ''');
```

```
  }
}
```

note.dart

```
class Note {
  final int? id;
  final String title;
  final String content;
  final int color;
  final bool isPinned;
  final DateTime createdAt;
  final DateTime updatedAt;

  Note({
    this.id,
    required this.title,
    this.content = '',
    this.color = 0,
    this.isPinned = false,
    DateTime? createdAt,
    DateTime? updatedAt,
  })  : createdAt = createdAt ?? DateTime.now(),
        updatedAt = updatedAt ?? DateTime.now();

  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'title': title,
      'content': content,
      'color': color,
      'is_pinned': isPinned ? 1 : 0,
      'created_at': createdAt.toIso8601String(),
      'updated_at': updatedAt.toIso8601String(),
    };
  }

  factory Note.fromMap(Map<String, dynamic> map) {
    return Note(
      id: map['id'] as int?,
      title: map['title'] as String,
      content: map['content'] as String? ?? '',
      color: map['color'] as int? ?? 0,
      isPinned: map['is_pinned'] == 1,
      createdAt: DateTime.parse(map['created_at'] as String),
      updatedAt: DateTime.parse(map['updated_at'] as String),
    );
  }

  Note copyWith({
    int? id,
```

```
    String? title,
    String? content,
    int? color,
    bool? isPinned,
    DateTime? updatedAt,
  }) {
    return Note(
      id: id ?? this.id,
      title: title ?? this.title,
      content: content ?? this.content,
      color: color ?? this.color,
      isPinned: isPinned ?? this.isPinned,
      createdAt: createdAt,
      updatedAt: updatedAt ?? DateTime.now(),
    );
  }
}
```

note_repository.dart

```
import 'package:sqflite/sqflite.dart';
import 'database_helper.dart';
import 'note.dart';

class NoteRepository {
  final DatabaseHelper _dbHelper = DatabaseHelper();

  Future<int> insert(Note note) async {
    final db = await _dbHelper.database;
    return await db.insert('notes', note.toMap());
  }

  Future<List<Note>> getAll() async {
    final db = await _dbHelper.database;
    final maps = await db.query(
      'notes',
      orderBy: 'is_pinned DESC, updated_at DESC',
    );
    return maps.map((map) => Note.fromMap(map)).toList();
  }

  Future<Note?> getById(int id) async {
    final db = await _dbHelper.database;
    final maps = await db.query(
      'notes',
      where: 'id = ?',
      whereArgs: [id],
      limit: 1,
    );
    if (maps.isEmpty) return null;
    return Note.fromMap(maps.first);
```

```
  }

  Future<int> update(Note note) async {
    final db = await _dbHelper.database;
    final updatedNote = note.copyWith(updatedAt: DateTime.now());
    return await db.update(
      'notes',
      updatedNote.toMap(),
      where: 'id = ?',
      whereArgs: [note.id],
    );
  }

  Future<int> delete(int id) async {
    final db = await _dbHelper.database;
    return await db.delete(
      'notes',
      where: 'id = ?',
      whereArgs: [id],
    );
  }

  Future<List<Note>> search(String query) async {
    final db = await _dbHelper.database;
    final maps = await db.query(
      'notes',
      where: 'title LIKE ? OR content LIKE ?',
      whereArgs: ['%$query%', '%$query%'],
      orderBy: 'updated_at DESC',
    );
    return maps.map((map) => Note.fromMap(map)).toList();
  }

  Future<List<Note>> getPinned() async {
    final db = await _dbHelper.database;
    final maps = await db.query(
      'notes',
      where: 'is_pinned = 1',
      orderBy: 'updated_at DESC',
    );
    return maps.map((map) => Note.fromMap(map)).toList();
  }

  Future<void> togglePinned(int id) async {
    final db = await _dbHelper.database;
    await db.rawUpdate(
      'UPDATE notes SET is_pinned = 1 - is_pinned WHERE id = ?',
      [id],
    );
  }
}
```

notes_page.dart

```
import 'package:flutter/material.dart';
import 'note.dart';
import 'note_repository.dart';

class NotesPage extends StatefulWidget {
  const NotesPage({super.key});

  @override
  State<NotesPage> createState() => _NotesPageState();
}

class _NotesPageState extends State<NotesPage> {
  final _repository = NoteRepository();
  List<Note> _notes = [];
  String _searchQuery = '';
  bool _isLoading = true;

  static const _colors = [
    Colors.white,
    Colors.red,
    Colors.orange,
    Colors.yellow,
    Colors.green,
    Colors.blue,
  ];

  @override
  void initState() {
    super.initState();
    _loadNotes();
  }

  Future<void> _loadNotes() async {
    setState(() => _isLoading = true);

    final notes = _searchQuery.isEmpty
        ? await _repository.getAll()
        : await _repository.search(_searchQuery);

    setState(() {
      _notes = notes;
      _isLoading = false;
    });
  }

  @override
  Widget build(BuildContext context) {
    final pinnedNotes = _notes.where((n) => n.isPinned).toList();
```

```
    final otherNotes = _notes.where((n) => !n.isPinned).toList();

    return Scaffold(
      appBar: AppBar(
        title: const Text('Notizen'),
        actions: [
          IconButton(
            icon: const Icon(Icons.search),
            onPressed: _showSearchDialog,
          ),
        ],
      ),
      body: _isLoading
          ? const Center(child: CircularProgressIndicator())
          : _notes.isEmpty
              ? const Center(child: Text('Keine Notizen'))
              : ListView(
                  padding: const EdgeInsets.all(8),
                  children: [
                    if (pinnedNotes.isNotEmpty) ...[
                      const Padding(
                        padding: EdgeInsets.all(8),
                        child: Text('Angepinnt',
                            style: TextStyle(fontWeight: FontWeight.bold)),
                      ),
                      ...pinnedNotes.map((note) => _NoteCard(
                            note: note,
                            color: _colors[note.color],
                            onTap: () => _editNote(note),
                            onDelete: () => _deleteNote(note),
                            onTogglePin: () => _togglePin(note),
                          )),
                      const Divider(),
                    ],
                    if (otherNotes.isNotEmpty) ...[
                      const Padding(
                        padding: EdgeInsets.all(8),
                        child: Text('Alle Notizen',
                            style: TextStyle(fontWeight: FontWeight.bold)),
                      ),
                      ...otherNotes.map((note) => _NoteCard(
                            note: note,
                            color: _colors[note.color],
                            onTap: () => _editNote(note),
                            onDelete: () => _deleteNote(note),
                            onTogglePin: () => _togglePin(note),
                          )),
                    ],
                  ],
                ),
      floatingActionButton: FloatingActionButton(
```

```
      onPressed: _createNote,
      child: const Icon(Icons.add),
    ),
  );
}

Future<void> _createNote() async {
  final result = await Navigator.push<Note>(
    context,
    MaterialPageRoute(builder: (_) => const NoteEditPage()),
  );

  if (result != null) {
    await _repository.insert(result);
    _loadNotes();
  }
}

Future<void> _editNote(Note note) async {
  final result = await Navigator.push<Note>(
    context,
    MaterialPageRoute(builder: (_) => NoteEditPage(note: note)),
  );

  if (result != null) {
    await _repository.update(result);
    _loadNotes();
  }
}

Future<void> _deleteNote(Note note) async {
  await _repository.delete(note.id!);
  _loadNotes();
  if (mounted) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('"${note.title}" gelöscht')),
    );
  }
}

Future<void> _togglePin(Note note) async {
  await _repository.togglePinned(note.id!);
  _loadNotes();
}

void _showSearchDialog() {
  showDialog(
    context: context,
    builder: (context) {
      return AlertDialog(
        title: const Text('Suchen'),
```

```
          content: TextField(
            onChanged: (value) {
              _searchQuery = value;
              _loadNotes();
            },
            decoration: const InputDecoration(hintText: 'Suchbegriff...'),
          ),
          actions: [
            TextButton(
              onPressed: () {
                _searchQuery = '';
                _loadNotes();
                Navigator.pop(context);
              },
              child: const Text('Zurücksetzen'),
            ),
            TextButton(
              onPressed: () => Navigator.pop(context),
              child: const Text('Schließen'),
            ),
          ],
        );
      },
    );
  }
}

class _NoteCard extends StatelessWidget {
  final Note note;
  final Color color;
  final VoidCallback onTap;
  final VoidCallback onDelete;
  final VoidCallback onTogglePin;

  const _NoteCard({
    required this.note,
    required this.color,
    required this.onTap,
    required this.onDelete,
    required this.onTogglePin,
  });

  @override
  Widget build(BuildContext context) {
    return Dismissible(
      key: ValueKey(note.id),
      direction: DismissDirection.endToStart,
      background: Container(
        color: Colors.red,
        alignment: Alignment.centerRight,
        padding: const EdgeInsets.only(right: 16),
```

```
        child: const Icon(Icons.delete, color: Colors.white),
      ),
      onDismissed: (_) => onDelete(),
      child: Card(
        color: color,
        child: ListTile(
          leading: note.isPinned
              ? const Icon(Icons.push_pin, size: 20)
              : null,
          title: Text(note.title),
          subtitle: note.content.isNotEmpty
              ? Text(
                  note.content,
                  maxLines: 2,
                  overflow: TextOverflow.ellipsis,
                )
              : null,
          onTap: onTap,
          onLongPress: onTogglePin,
        ),
      ),
    );
  }
}
```

3.8.2.2 Hive Lösung

note.dart (Hive)

```
import 'package:hive/hive.dart';

part 'note.g.dart';

@HiveType(typeId: 0)
class Note extends HiveObject {
  @HiveField(0)
  late String id;

  @HiveField(1)
  late String title;

  @HiveField(2)
  late String content;

  @HiveField(3)
  late int color;

  @HiveField(4)
  late bool isPinned;

  @HiveField(5)
```

```
  late DateTime createdAt;

  @HiveField(6)
  late DateTime updatedAt;

  Note({
    String? id,
    required this.title,
    this.content = '',
    this.color = 0,
    this.isPinned = false,
    DateTime? createdAt,
    DateTime? updatedAt,
  }) {
    this.id = id ?? DateTime.now().millisecondsSinceEpoch.toString();
    this.createdAt = createdAt ?? DateTime.now();
    this.updatedAt = updatedAt ?? DateTime.now();
  }
}
```

note_repository.dart (Hive)

```
import 'package:hive/hive.dart';
import 'note.dart';

class NoteRepository {
  static const _boxName = 'notes';

  Box<Note> get _box => Hive.box<Note>(_boxName);

  Future<void> add(Note note) async {
    await _box.put(note.id, note);
  }

  List<Note> getAll() {
    final notes = _box.values.toList();
    notes.sort((a, b) {
      if (a.isPinned != b.isPinned) {
        return a.isPinned ? -1 : 1;
      }
      return b.updatedAt.compareTo(a.updatedAt);
    });
    return notes;
  }

  Note? getById(String id) => _box.get(id);

  Future<void> update(Note note) async {
    note.updatedAt = DateTime.now();
    await note.save();
  }
```

```
  Future<void> delete(String id) async {
    await _box.delete(id);
  }

  List<Note> search(String query) {
    final lowerQuery = query.toLowerCase();
    return _box.values.where((note) {
      return note.title.toLowerCase().contains(lowerQuery) ||
          note.content.toLowerCase().contains(lowerQuery);
    }).toList();
  }

  List<Note> getPinned() {
    return _box.values.where((note) => note.isPinned).toList();
  }
}
```

Reactive UI mit Hive

```
class NotesPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Notizen')),
      body: ValueListenableBuilder<Box<Note>>(
        valueListenable: Hive.box<Note>('notes').listenable(),
        builder: (context, box, _) {
          final notes = box.values.toList()
            ..sort((a, b) => b.updatedAt.compareTo(a.updatedAt));

          return ListView.builder(
            itemCount: notes.length,
            itemBuilder: (context, index) {
              final note = notes[index];
              return NoteCard(note: note);
            },
          );
        },
      ),
    );
  }
}
```

3.8.2.3 Aufgabe 5: Vergleich

Kriterium	sqflite	Hive
Setup-Aufwand	Mittel (SQL Schema)	Niedrig (Annotations)
Code-Menge	Mehr (SQL Queries)	Weniger (Dart API)
Performance	Gut	Sehr gut

Kriterium	sqflite	Hive
Flexibilität	Hoch (SQL)	Mittel
Reaktive UI	Manuell	`listenable()`
Lernkurve	Mittel (SQL)	Niedrig

Empfehlung für diese App: Hive, wegen einfacherem Setup und reaktiver UI.

3.8.3 Ressourcen

3.8.3.1 sqflite

- sqflite Package
- SQLite Tutorial
- Persist data with SQLite (Flutter Cookbook)

3.8.3.2 Hive

- Hive Package
- hive_flutter Package
- Hive Documentation
- Hive Tutorial

3.8.3.3 Alternative Datenbanken

- isar - Nachfolger von Hive
- drift - Type-safe SQL (ehem. moor)
- floor - SQLite mit Annotations
- objectbox - NoSQL Database
- sembast - Simple Embedded DB

3.8.3.4 path_provider

- path_provider Package
- Working with paths

3.8.3.5 Tutorials

- Complete SQLite Tutorial
- Hive Database Tutorial
- Local Database Comparison

3.8.3.6 Best Practices

- Repository Pattern
- Database Migrations
- Offline-First Architecture

3.8.3.7 Zum Vertiefen

- SQL Basics
- Database Design
- Data Persistence Overview

3.9 Einheit 3.9: Formulare Basics

3.9.0.1 Lernziele

Nach dieser Einheit kannst du: - `Form` und `GlobalKey<FormState>` verwenden - `TextFormField` mit `InputDecoration` gestalten - `TextEditingController` für Eingabekontrolle nutzen - Formulardaten auslesen und verarbeiten

3.9.0.2 1. Form Widget Grundlagen

Einfaches Formular

```
class LoginForm extends StatefulWidget {
  @override
  State<LoginForm> createState() => _LoginFormState();
}

class _LoginFormState extends State<LoginForm> {
  // GlobalKey für Formular-Zugriff
  final _formKey = GlobalKey<FormState>();

  // Controller für Eingabefelder
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();

  @override
  void dispose() {
    // Wichtig: Controller freigeben
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  void _submit() {
    // Validierung auslösen
    if (_formKey.currentState!.validate()) {
      // Formular ist gültig
      final email = _emailController.text;
      final password = _passwordController.text;

      print('Login: $email / $password');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        children: [
          TextFormField(
            controller: _emailController,
```

```
            decoration: const InputDecoration(
              labelText: 'E-Mail',
            ),
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Bitte E-Mail eingeben';
              }
              return null;
            },
          ),
          TextFormField(
            controller: _passwordController,
            decoration: const InputDecoration(
              labelText: 'Passwort',
            ),
            obscureText: true,
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Bitte Passwort eingeben';
              }
              return null;
            },
          ),
          ElevatedButton(
            onPressed: _submit,
            child: const Text('Anmelden'),
          ),
        ],
      ),
    );
  }
}
```

3.9.0.3 2. TextEditingController

Grundlegende Verwendung

```
class MyForm extends StatefulWidget {
  @override
  State<MyForm> createState() => _MyFormState();
}

class _MyFormState extends State<MyForm> {
  final _controller = TextEditingController();

  @override
  void initState() {
    super.initState();

    // Initialer Wert
    _controller.text = 'Startwert';
```

```
    // Auf Änderungen reagieren
    _controller.addListener(_onTextChanged);
  }

  void _onTextChanged() {
    print('Text: ${_controller.text}');
    print('Cursor: ${_controller.selection}');
  }

  @override
  void dispose() {
    _controller.removeListener(_onTextChanged);
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return TextField(
      controller: _controller,
      onChanged: (value) {
        // Alternative zu addListener
        print('Geändert: $value');
      },
    );
  }
}
```

Controller Methoden

```
// Text setzen
_controller.text = 'Neuer Text';

// Text lesen
final text = _controller.text;

// Text leeren
_controller.clear();

// Cursor-Position setzen
_controller.selection = TextSelection.fromPosition(
  TextPosition(offset: _controller.text.length),
);

// Bereich selektieren
_controller.selection = TextSelection(
  baseOffset: 0,
  extentOffset: 5,
);
```

3.9.0.4 3. InputDecoration im Detail

```
TextFormField(
  decoration: InputDecoration(
    // Labels
    labelText: 'E-Mail',
    hintText: 'name@example.com',
    helperText: 'Deine geschäftliche E-Mail',

    // Icons
    prefixIcon: const Icon(Icons.email),
    suffixIcon: IconButton(
      icon: const Icon(Icons.clear),
      onPressed: () => _controller.clear(),
    ),

    // Prefix/Suffix Text
    prefixText: 'https://',
    suffixText: '.com',

    // Rahmen
    border: const OutlineInputBorder(),
    enabledBorder: OutlineInputBorder(
      borderSide: BorderSide(color: Colors.grey),
    ),
    focusedBorder: OutlineInputBorder(
      borderSide: BorderSide(color: Colors.blue, width: 2),
    ),
    errorBorder: OutlineInputBorder(
      borderSide: BorderSide(color: Colors.red),
    ),

    // Styling
    filled: true,
    fillColor: Colors.grey[100],

    // Counter
    counterText: '0/100',

    // Constraints
    contentPadding: const EdgeInsets.symmetric(
      horizontal: 16,
      vertical: 12,
    ),
  ),
);
```

Underline vs. Outline

```
// Underline (Standard)
TextFormField(
  decoration: const InputDecoration(
```

```
    labelText: 'Standard',
  ),
);

// Outline
TextFormField(
  decoration: const InputDecoration(
    labelText: 'Outlined',
    border: OutlineInputBorder(),
  ),
);

// Ohne Rahmen
TextFormField(
  decoration: const InputDecoration(
    labelText: 'Borderless',
    border: InputBorder.none,
    filled: true,
  ),
);
```

3.9.0.5 4. Keyboard Types

```
// E-Mail
TextFormField(
  keyboardType: TextInputType.emailAddress,
);

// Zahlen
TextFormField(
  keyboardType: TextInputType.number,
);

// Telefon
TextFormField(
  keyboardType: TextInputType.phone,
);

// URL
TextFormField(
  keyboardType: TextInputType.url,
);

// Multiline
TextFormField(
  keyboardType: TextInputType.multiline,
  maxLines: 5,
);

// Dezimalzahlen
```

```
TextFormField(
  keyboardType: const TextInputType.numberWithOptions(decimal: true),
);
```

3.9.0.6 5. TextInputAction

```
// Standard Enter-Taste Verhalten ändern
TextFormField(
  textInputAction: TextInputAction.next,  // Zum nächsten Feld
  onFieldSubmitted: (_) {
    FocusScope.of(context).nextFocus();
  },
);

TextFormField(
  textInputAction: TextInputAction.done,  // Tastatur schließen
  onFieldSubmitted: (_) {
    FocusScope.of(context).unfocus();
  },
);

TextFormField(
  textInputAction: TextInputAction.search,  // Suche starten
  onFieldSubmitted: (query) {
    _performSearch(query);
  },
);

TextFormField(
  textInputAction: TextInputAction.send,  // Nachricht senden
  onFieldSubmitted: (_) {
    _sendMessage();
  },
);
```

3.9.0.7 6. FocusNode

```
class FocusDemo extends StatefulWidget {
  @override
  State<FocusDemo> createState() => _FocusDemoState();
}

class _FocusDemoState extends State<FocusDemo> {
  final _emailFocus = FocusNode();
  final _passwordFocus = FocusNode();

  @override
  void dispose() {
    _emailFocus.dispose();
```

```
    _passwordFocus.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        TextFormField(
          focusNode: _emailFocus,
          textInputAction: TextInputAction.next,
          onFieldSubmitted: (_) {
            // Fokus zum nächsten Feld
            FocusScope.of(context).requestFocus(_passwordFocus);
          },
          decoration: const InputDecoration(labelText: 'E-Mail'),
        ),
        TextFormField(
          focusNode: _passwordFocus,
          textInputAction: TextInputAction.done,
          obscureText: true,
          decoration: const InputDecoration(labelText: 'Passwort'),
        ),
        ElevatedButton(
          onPressed: () {
            // Fokus auf E-Mail Feld setzen
            _emailFocus.requestFocus();
          },
          child: const Text('E-Mail fokussieren'),
        ),
      ],
    );
  }
}
```

3.9.0.8 7. Formular speichern und zurücksetzen

```
class ContactForm extends StatefulWidget {
  @override
  State<ContactForm> createState() => _ContactFormState();
}

class _ContactFormState extends State<ContactForm> {
  final _formKey = GlobalKey<FormState>();
  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
  final _messageController = TextEditingController();

  void _submit() {
    if (_formKey.currentState!.validate()) {
```

```
      // Formular speichern (für onSaved Callbacks)
      _formKey.currentState!.save();

      // Daten verarbeiten
      _sendMessage();
    }
  }

  void _reset() {
    // Formular zurücksetzen
    _formKey.currentState!.reset();

    // Controller leeren
    _nameController.clear();
    _emailController.clear();
    _messageController.clear();
  }

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        children: [
          TextFormField(
            controller: _nameController,
            decoration: const InputDecoration(labelText: 'Name'),
            validator: (value) => value!.isEmpty ? 'Pflichtfeld' : null,
            onSaved: (value) {
              // Wird bei save() aufgerufen
              print('Name gespeichert: $value');
            },
          ),
          // Weitere Felder...
          Row(
            children: [
              ElevatedButton(
                onPressed: _submit,
                child: const Text('Senden'),
              ),
              TextButton(
                onPressed: _reset,
                child: const Text('Zurücksetzen'),
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

3.9.0.9 8. AutovalidateMode

```
Form(
  key: _formKey,
  // Validierung nur bei Submit
  autovalidateMode: AutovalidateMode.disabled,

  // Validierung sofort bei Eingabe
  // autovalidateMode: AutovalidateMode.always,

  // Validierung nach erstem Submit
  // autovalidateMode: AutovalidateMode.onUserInteraction,

  child: // ...
);
```

3.9.0.10 9. Vollständiges Beispiel

```
class RegistrationForm extends StatefulWidget {
  const RegistrationForm({super.key});

  @override
  State<RegistrationForm> createState() => _RegistrationFormState();
}

class _RegistrationFormState extends State<RegistrationForm> {
  final _formKey = GlobalKey<FormState>();

  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();

  final _nameFocus = FocusNode();
  final _emailFocus = FocusNode();
  final _passwordFocus = FocusNode();

  bool _obscurePassword = true;
  bool _isLoading = false;

  @override
  void dispose() {
    _nameController.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    _nameFocus.dispose();
    _emailFocus.dispose();
    _passwordFocus.dispose();
    super.dispose();
```

```
}

Future<void> _submit() async {
  if (!_formKey.currentState!.validate()) return;

  setState(() => _isLoading = true);

  // Simuliere API-Call
  await Future.delayed(const Duration(seconds: 2));

  setState(() => _isLoading = false);

  if (mounted) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Registrierung erfolgreich!')),
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Registrierung')),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(16),
      child: Form(
        key: _formKey,
        child: Column(
          children: [
            // Name
            TextFormField(
              controller: _nameController,
              focusNode: _nameFocus,
              decoration: const InputDecoration(
                labelText: 'Name',
                prefixIcon: Icon(Icons.person),
                border: OutlineInputBorder(),
              ),
              textInputAction: TextInputAction.next,
              textCapitalization: TextCapitalization.words,
              onFieldSubmitted: (_) {
                FocusScope.of(context).requestFocus(_emailFocus);
              },
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Name ist erforderlich';
                }
                if (value.length < 2) {
                  return 'Name muss mindestens 2 Zeichen haben';
                }
                return null;
```

```
    },
  ),
  const SizedBox(height: 16),

  // E-Mail
  TextFormField(
    controller: _emailController,
    focusNode: _emailFocus,
    decoration: const InputDecoration(
      labelText: 'E-Mail',
      prefixIcon: Icon(Icons.email),
      border: OutlineInputBorder(),
    ),
    keyboardType: TextInputType.emailAddress,
    textInputAction: TextInputAction.next,
    onFieldSubmitted: (_) {
      FocusScope.of(context).requestFocus(_passwordFocus);
    },
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'E-Mail ist erforderlich';
      }
      if (!value.contains('@')) {
        return 'Ungültige E-Mail';
      }
      return null;
    },
  ),
  const SizedBox(height: 16),

  // Passwort
  TextFormField(
    controller: _passwordController,
    focusNode: _passwordFocus,
    decoration: InputDecoration(
      labelText: 'Passwort',
      prefixIcon: const Icon(Icons.lock),
      border: const OutlineInputBorder(),
      suffixIcon: IconButton(
        icon: Icon(
          _obscurePassword
              ? Icons.visibility
              : Icons.visibility_off,
        ),
        onPressed: () {
          setState(() {
            _obscurePassword = !_obscurePassword;
          });
        },
      ),
    ),
```

```
                obscureText: _obscurePassword,
                textInputAction: TextInputAction.done,
                onFieldSubmitted: (_) => _submit(),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Passwort ist erforderlich';
                  }
                  if (value.length < 8) {
                    return 'Mindestens 8 Zeichen';
                  }
                  return null;
                },
              ),
              const SizedBox(height: 24),

              // Submit Button
              SizedBox(
                width: double.infinity,
                height: 48,
                child: ElevatedButton(
                  onPressed: _isLoading ? null : _submit,
                  child: _isLoading
                      ? const SizedBox(
                          height: 24,
                          width: 24,
                          child: CircularProgressIndicator(strokeWidth: 2),
                        )
                      : const Text('Registrieren'),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

3.9.0.11 Zusammenfassung

Element	Verwendung
`Form`	Container für Formularfelder
`GlobalKey<FormState>`	Zugriff auf `validate()`, `save()`, `reset()`
`TextFormField`	Eingabefeld mit Validierung
`TextEditingController`	Text lesen/schreiben, Cursor
`FocusNode`	Fokus-Steuerung
`InputDecoration`	Styling des Eingabefelds

Merke: - Controller und FocusNodes immer in `dispose()` freigeben - `validate()` vor Verar-

beitung aufrufen - `textInputAction` für bessere UX

3.9.1 Übung

3.9.1.1 Ziel

Ein Kontaktformular mit verschiedenen Eingabefeldern erstellen.

3.9.1.2 Aufgabe 1: Einfaches Formular (20 min)

Erstelle ein Login-Formular mit: - E-Mail Feld - Passwort Feld (verborgen) - Login Button - “Passwort vergessen?” Link

Anforderungen: - Beide Felder sind Pflichtfelder - E-Mail muss “@” enthalten - Button ist disabled während Validierung fehlschlägt

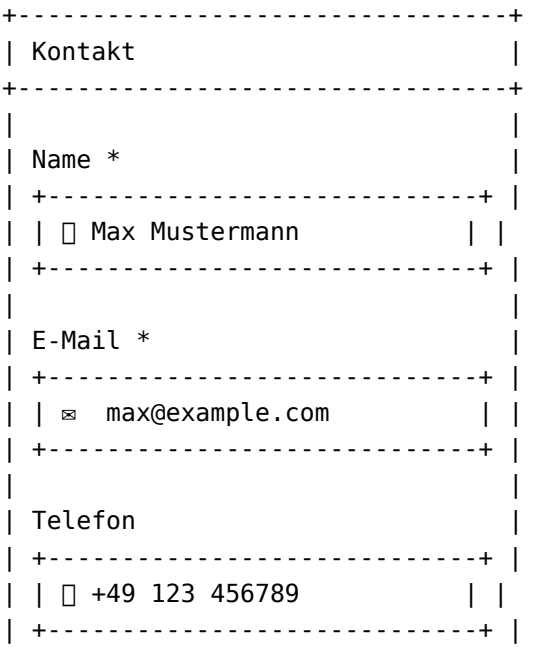
3.9.1.3 Aufgabe 2: InputDecoration Styling (20 min)

Erstelle drei unterschiedlich gestylte TextFormFields:

1. **Outline Style:**
   - Umrandetes Feld
   - Icon links
   - Clear-Button rechts
2. **Filled Style:**
   - Gefüllter Hintergrund
   - Keine Umrandung
   - Abgerundete Ecken
3. **Custom Style:**
   - Unterstrichen
   - Prefix-Text
   - Counter anzeigen

3.9.1.4 Aufgabe 3: Kontaktformular (30 min)

Erstelle ein vollständiges Kontaktformular:



```
|                                 |
| Betreff *                       |
| +-----------------------------+ |
| | Anfrage zu Produkt X        | |
| +-----------------------------+ |
|                                 |
| Nachricht *                     |
| +-----------------------------+ |
| | Ihre Nachricht hier...      | |
| |                             | |
| |                             | |
| |                             | |
| |                      42/500 | |
| +-----------------------------+ |
|                                 |
| [Zurücksetzen]  [Absenden]      |
|                                 |
+---------------------------------+
```

Features: - Pflichtfelder mit * - Passende Keyboard-Typen - Character Counter bei Nachricht - Tab-Navigation zwischen Feldern - Validierung bei Submit

3.9.1.5 Aufgabe 4: FocusNode Navigation (15 min)

Erweitere das Kontaktformular: 1. Enter auf einem Feld -> Fokus zum nächsten 2. Letzes Feld -> Submit 3. Button um Fokus auf erstes Feld zu setzen

```
TextFormField(
  focusNode: _nameFocus,
  textInputAction: TextInputAction.next,
  onFieldSubmitted: (_) {
    FocusScope.of(context).requestFocus(_emailFocus);
  },
)
```

3.9.1.6 Aufgabe 5: Passwort mit Toggle (15 min)

Erstelle ein Passwort-Feld mit: - Auge-Icon zum Ein-/Ausblenden - Passwort-Stärke Anzeige - Mindestens 8 Zeichen - Mindestens eine Zahl

```
+---------------------------------+
| Passwort *                      |
| +-----------------------------+ |
| |  ●●●●●●●●          |    | |
| +-----------------------------+ |
| Stärke: ████████░░ Mittel       |
+---------------------------------+
```

3.9.1.7 Aufgabe 6: Formular-Daten Model (20 min)

Erstelle ein Model für die Formulardaten:

```
class ContactFormData {
  final String name;
```

```
  final String email;
  final String? phone;
  final String subject;
  final String message;

  ContactFormData({
    required this.name,
    required this.email,
    this.phone,
    required this.subject,
    required this.message,
  });
}
```

1. Bei Submit: Erstelle `ContactFormData` Objekt
2. Zeige die Daten in einem Dialog
3. Nach Bestätigung: Formular zurücksetzen

3.9.1.8 Aufgabe 7: Verständnisfragen

1. Wann braucht man einen `TextEditingController`?
2. Was ist der Unterschied zwischen `TextField` und `TextFormField`?
3. Warum muss man Controller und FocusNodes in `dispose()` freigeben?
4. Was macht `AutovalidateMode.onUserInteraction`?
5. Wie kann man den Fokus programmatisch setzen?

3.9.1.9 Abgabe-Checkliste

- ☐ Login-Formular funktioniert
- ☐ Drei verschiedene InputDecoration Styles
- ☐ Kontaktformular mit allen Feldern
- ☐ Validierung funktioniert
- ☐ FocusNode Navigation implementiert
- ☐ Passwort Toggle funktioniert
- ☐ Formular-Daten in Model
- ☐ Verständnisfragen beantwortet

3.9.2 Lösung

3.9.2.1 Aufgabe 3 & 4: Kontaktformular

```
import 'package:flutter/material.dart';

class ContactFormData {
  final String name;
  final String email;
  final String? phone;
  final String subject;
  final String message;
```

```
  ContactFormData({
    required this.name,
    required this.email,
    this.phone,
    required this.subject,
    required this.message,
  });
}

class ContactFormPage extends StatefulWidget {
  const ContactFormPage({super.key});

  @override
  State<ContactFormPage> createState() => _ContactFormPageState();
}

class _ContactFormPageState extends State<ContactFormPage> {
  final _formKey = GlobalKey<FormState>();

  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
  final _phoneController = TextEditingController();
  final _subjectController = TextEditingController();
  final _messageController = TextEditingController();

  final _nameFocus = FocusNode();
  final _emailFocus = FocusNode();
  final _phoneFocus = FocusNode();
  final _subjectFocus = FocusNode();
  final _messageFocus = FocusNode();

  int _messageLength = 0;
  static const _maxMessageLength = 500;

  @override
  void initState() {
    super.initState();
    _messageController.addListener(() {
      setState(() {
        _messageLength = _messageController.text.length;
      });
    });
  }

  @override
  void dispose() {
    _nameController.dispose();
    _emailController.dispose();
    _phoneController.dispose();
    _subjectController.dispose();
    _messageController.dispose();
```

```
    _nameFocus.dispose();
    _emailFocus.dispose();
    _phoneFocus.dispose();
    _subjectFocus.dispose();
    _messageFocus.dispose();
    super.dispose();
  }

  void _submit() {
    if (_formKey.currentState!.validate()) {
      final data = ContactFormData(
        name: _nameController.text,
        email: _emailController.text,
        phone: _phoneController.text.isEmpty ? null : _phoneController.text,
        subject: _subjectController.text,
        message: _messageController.text,
      );

      _showConfirmDialog(data);
    }
  }

  void _showConfirmDialog(ContactFormData data) {
    showDialog(
      context: context,
      builder: (context) => AlertDialog(
        title: const Text('Nachricht senden?'),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text('Name: ${data.name}'),
            Text('E-Mail: ${data.email}'),
            if (data.phone != null) Text('Telefon: ${data.phone}'),
            Text('Betreff: ${data.subject}'),
            const Divider(),
            Text(data.message),
          ],
        ),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context),
            child: const Text('Bearbeiten'),
          ),
          ElevatedButton(
            onPressed: () {
              Navigator.pop(context);
              _reset();
              ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text('Nachricht gesendet!')),
              );
```

```
          },
          child: const Text('Senden'),
        ),
      ],
    ),
  );
}

void _reset() {
  _formKey.currentState!.reset();
  _nameController.clear();
  _emailController.clear();
  _phoneController.clear();
  _subjectController.clear();
  _messageController.clear();
  _nameFocus.requestFocus();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Kontakt')),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(16),
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            // Name
            TextFormField(
              controller: _nameController,
              focusNode: _nameFocus,
              decoration: const InputDecoration(
                labelText: 'Name *',
                prefixIcon: Icon(Icons.person),
                border: OutlineInputBorder(),
              ),
              textCapitalization: TextCapitalization.words,
              textInputAction: TextInputAction.next,
              onFieldSubmitted: (_) =>
                  FocusScope.of(context).requestFocus(_emailFocus),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Name ist erforderlich';
                }
                return null;
              },
            ),
            const SizedBox(height: 16),
```

```
              // E-Mail
              TextFormField(
                controller: _emailController,
                focusNode: _emailFocus,
                decoration: const InputDecoration(
                  labelText: 'E-Mail *',
                  prefixIcon: Icon(Icons.email),
                  border: OutlineInputBorder(),
                ),
                keyboardType: TextInputType.emailAddress,
                textInputAction: TextInputAction.next,
                onFieldSubmitted: (_) =>
                    FocusScope.of(context).requestFocus(_phoneFocus),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'E-Mail ist erforderlich';
                  }
                  if (!value.contains('@') || !value.contains('.')) {
                    return 'Ungültige E-Mail Adresse';
                  }
                  return null;
                },
              ),
              const SizedBox(height: 16),

              // Telefon
              TextFormField(
                controller: _phoneController,
                focusNode: _phoneFocus,
                decoration: const InputDecoration(
                  labelText: 'Telefon',
                  prefixIcon: Icon(Icons.phone),
                  border: OutlineInputBorder(),
                  hintText: '+49 123 456789',
                ),
                keyboardType: TextInputType.phone,
                textInputAction: TextInputAction.next,
                onFieldSubmitted: (_) =>
                    FocusScope.of(context).requestFocus(_subjectFocus),
              ),
              const SizedBox(height: 16),

              // Betreff
              TextFormField(
                controller: _subjectController,
                focusNode: _subjectFocus,
                decoration: const InputDecoration(
                  labelText: 'Betreff *',
                  border: OutlineInputBorder(),
                ),
                textInputAction: TextInputAction.next,
```

```
                onFieldSubmitted: (_) =>
                    FocusScope.of(context).requestFocus(_messageFocus),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Betreff ist erforderlich';
                  }
                  return null;
                },
              ),
              const SizedBox(height: 16),

              // Nachricht
              TextFormField(
                controller: _messageController,
                focusNode: _messageFocus,
                decoration: InputDecoration(
                  labelText: 'Nachricht *',
                  border: const OutlineInputBorder(),
                  alignLabelWithHint: true,
                  counterText: '$_messageLength/$_maxMessageLength',
                ),
                maxLines: 5,
                maxLength: _maxMessageLength,
                textInputAction: TextInputAction.newline,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Nachricht ist erforderlich';
                  }
                  if (value.length < 10) {
                    return 'Mindestens 10 Zeichen';
                  }
                  return null;
                },
              ),
              const SizedBox(height: 24),

              // Buttons
              Row(
                children: [
                  Expanded(
                    child: OutlinedButton(
                      onPressed: _reset,
                      child: const Text('Zurücksetzen'),
                    ),
                  ),
                  const SizedBox(width: 16),
                  Expanded(
                    child: ElevatedButton(
                      onPressed: _submit,
                      child: const Text('Absenden'),
                    ),
```

```
                ),
              ],
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

3.9.2.2 Aufgabe 5: Passwort mit Toggle und Stärke

```
class PasswordField extends StatefulWidget {
  final TextEditingController controller;

  const PasswordField({super.key, required this.controller});

  @override
  State<PasswordField> createState() => _PasswordFieldState();
}

class _PasswordFieldState extends State<PasswordField> {
  bool _obscure = true;

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        TextFormField(
          controller: widget.controller,
          decoration: InputDecoration(
            labelText: 'Passwort *',
            prefixIcon: const Icon(Icons.lock),
            border: const OutlineInputBorder(),
            suffixIcon: IconButton(
              icon: Icon(_obscure ? Icons.visibility : Icons.visibility_off),
              onPressed: () => setState(() => _obscure = !_obscure),
            ),
          ),
          obscureText: _obscure,
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Passwort ist erforderlich';
            }
            if (value.length < 8) {
              return 'Mindestens 8 Zeichen';
            }
            if (!value.contains(RegExp(r'[0-9]'))) {
```

```
              return 'Mindestens eine Zahl';
            }
            return null;
          },
        ),
        const SizedBox(height: 8),
        _PasswordStrengthIndicator(password: widget.controller.text),
      ],
    );
  }
}

class _PasswordStrengthIndicator extends StatelessWidget {
  final String password;

  const _PasswordStrengthIndicator({required this.password});

  int get _strength {
    int score = 0;
    if (password.length >= 8) score++;
    if (password.length >= 12) score++;
    if (password.contains(RegExp(r'[0-9]'))) score++;
    if (password.contains(RegExp(r'[A-Z]'))) score++;
    if (password.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]'))) score++;
    return score;
  }

  String get _label {
    switch (_strength) {
      case 0:
      case 1:
        return 'Schwach';
      case 2:
      case 3:
        return 'Mittel';
      default:
        return 'Stark';
    }
  }

  Color get _color {
    switch (_strength) {
      case 0:
      case 1:
        return Colors.red;
      case 2:
      case 3:
        return Colors.orange;
      default:
        return Colors.green;
    }
```

```
  }

  @override
  Widget build(BuildContext context) {
    if (password.isEmpty) return const SizedBox.shrink();

    return Row(
      children: [
        Expanded(
          child: LinearProgressIndicator(
            value: _strength / 5,
            backgroundColor: Colors.grey[300],
            valueColor: AlwaysStoppedAnimation(_color),
          ),
        ),
        const SizedBox(width: 8),
        Text(
          'Stärke: $_label',
          style: TextStyle(color: _color, fontWeight: FontWeight.bold),
        ),
      ],
    );
  }
}
```

3.9.2.3 Verständnisfragen - Antworten

1. **Wann TextEditingController?**
   - Wenn man den Text programmatisch setzen will
   - Wenn man auf Änderungen reagieren will (addListener)
   - Wenn man die Cursor-Position steuern will
   - Wenn man den Text außerhalb von onChanged lesen will
2. **TextField vs. TextFormField?**
   - `TextField`: Basis-Widget ohne Validierung
   - `TextFormField`: Kann in `Form` verwendet werden, hat `validator`, `onSaved`
   - `TextFormField` ist ein `TextField` mit `FormField` gewrappt
3. **Warum dispose()?**
   - Controller/FocusNodes registrieren Listener
   - Ohne dispose: Memory Leaks
   - Flutter Framework verlangt Cleanup
4. **AutovalidateMode.onUserInteraction?**
   - Validiert erst nachdem User das Feld berührt hat
   - Keine Fehler beim ersten Render
   - Bessere UX als `always`
5. **Fokus programmatisch setzen?**

```
// Mit FocusNode
myFocusNode.requestFocus();

// Mit FocusScope
FocusScope.of(context).requestFocus(otherNode);

// Zum nächsten Feld
FocusScope.of(context).nextFocus();

// Fokus entfernen
FocusScope.of(context).unfocus();
```

3.9.3 Ressourcen

3.9.3.1 Offizielle Dokumentation

- Build a form with validation
- Create and style a text field
- Handle changes to a text field
- Retrieve the value of a text field
- Focus and text fields

3.9.3.2 API Referenz

- Form Class
- TextFormField Class
- InputDecoration Class
- TextEditingController
- FocusNode

3.9.3.3 Tutorials

- Flutter Forms Tutorial
- Complete Guide to Forms
- TextFormField Deep Dive

3.9.3.4 Videos

- Forms - Flutter in Focus
- TextField & TextFormField
- Form Validation

3.9.3.5 Best Practices

- Form Design UX
- Input Validation Patterns
- Keyboard Handling

3.9.3.6 Packages

- flutter_form_builder - Formular-Generator
- reactive_forms - Reaktive Formulare
- mask_text_input_formatter - Input Masken

3.10 Einheit 3.10: Formular Validierung

3.10.0.1 Lernziele

Nach dieser Einheit kannst du: - Validatoren für verschiedene Eingabetypen schreiben - Regex-Validierung anwenden - Cross-Field-Validierung implementieren - AutovalidateMode sinnvoll einsetzen - Async-Validierung durchführen

3.10.0.2 1. Validator Grundlagen

Validator Signatur

```
String? validator(String? value) {
  // Gibt null zurück wenn gültig
  // Gibt Fehlermeldung zurück wenn ungültig
}
```

Einfache Validatoren

```
// Pflichtfeld
String? requiredValidator(String? value) {
  if (value == null || value.isEmpty) {
    return 'Dieses Feld ist erforderlich';
  }
  return null;
}

// Mindestlänge
String? minLengthValidator(String? value, int minLength) {
  if (value == null || value.length < minLength) {
    return 'Mindestens $minLength Zeichen';
  }
  return null;
}

// Maximallänge
String? maxLengthValidator(String? value, int maxLength) {
  if (value != null && value.length > maxLength) {
    return 'Maximal $maxLength Zeichen';
  }
  return null;
}
```

3.10.0.3 2. Regex-Validierung

E-Mail Validierung

```
String? emailValidator(String? value) {
  if (value == null || value.isEmpty) {
    return 'E-Mail ist erforderlich';
  }

  // Einfache Regex
```

```
  final emailRegex = RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$');

  if (!emailRegex.hasMatch(value)) {
    return 'Ungültige E-Mail Adresse';
  }

  return null;
}
```

Passwort Validierung

```
String? passwordValidator(String? value) {
  if (value == null || value.isEmpty) {
    return 'Passwort ist erforderlich';
  }

  final errors = <String>[];

  if (value.length < 8) {
    errors.add('mindestens 8 Zeichen');
  }
  if (!value.contains(RegExp(r'[A-Z]'))) {
    errors.add('einen Großbuchstaben');
  }
  if (!value.contains(RegExp(r'[a-z]'))) {
    errors.add('einen Kleinbuchstaben');
  }
  if (!value.contains(RegExp(r'[0-9]'))) {
    errors.add('eine Zahl');
  }
  if (!value.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]'))) {
    errors.add('ein Sonderzeichen');
  }

  if (errors.isNotEmpty) {
    return 'Passwort benötigt: ${errors.join(', ')}';
  }

  return null;
}
```

Telefonnummer

```
String? phoneValidator(String? value) {
  if (value == null || value.isEmpty) return null;  // Optional

  // Nur Ziffern, Leerzeichen, +, -, ()
  final cleaned = value.replaceAll(RegExp(r'[\s\-\(\)]'), '');

  if (!RegExp(r'^\+?[0-9]{6,15}$').hasMatch(cleaned)) {
    return 'Ungültige Telefonnummer';
  }
```

```
  return null;
}
```

URL Validierung

```
String? urlValidator(String? value) {
  if (value == null || value.isEmpty) return null;

  final urlRegex = RegExp(
    r'^https?://'
    r'([\w-]+\.)+[\w-]+'
    r'(/[\w-./?%&=]*)?$',
    caseSensitive: false,
  );

  if (!urlRegex.hasMatch(value)) {
    return 'Ungültige URL';
  }

  return null;
}
```

3.10.0.4 3. Validator Kombinieren

Validator-Chain

```
class Validators {
  static String? Function(String?) combine(
    List<String? Function(String?)> validators,
  ) {
    return (value) {
      for (final validator in validators) {
        final error = validator(value);
        if (error != null) return error;
      }
      return null;
    };
  }

  static String? required(String? value) {
    if (value == null || value.isEmpty) {
      return 'Pflichtfeld';
    }
    return null;
  }

  static String? Function(String?) minLength(int length) {
    return (value) {
      if (value != null && value.length < length) {
        return 'Mindestens $length Zeichen';
      }
```

```
      return null;
    };
  }

  static String? Function(String?) maxLength(int length) {
    return (value) {
      if (value != null && value.length > length) {
        return 'Maximal $length Zeichen';
      }
      return null;
    };
  }

  static String? email(String? value) {
    if (value == null || value.isEmpty) return null;
    if (!RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$').hasMatch(value)) {
      return 'Ungültige E-Mail';
    }
    return null;
  }

  static String? Function(String?) pattern(RegExp regex, String message) {
    return (value) {
      if (value != null && !regex.hasMatch(value)) {
        return message;
      }
      return null;
    };
  }
}

// Verwendung
TextFormField(
  validator: Validators.combine([
    Validators.required,
    Validators.minLength(3),
    Validators.maxLength(20),
    Validators.pattern(
      RegExp(r'^[a-zA-Z]+$'),
      'Nur Buchstaben erlaubt',
    ),
  ]),
);
```

3.10.0.5 4. Cross-Field Validierung

Passwort Bestätigung

```
class PasswordForm extends StatefulWidget {
  @override
  State<PasswordForm> createState() => _PasswordFormState();
```

```
}

class _PasswordFormState extends State<PasswordForm> {
  final _formKey = GlobalKey<FormState>();
  final _passwordController = TextEditingController();
  final _confirmController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        children: [
          TextFormField(
            controller: _passwordController,
            decoration: const InputDecoration(labelText: 'Passwort'),
            obscureText: true,
            validator: (value) {
              if (value == null || value.length < 8) {
                return 'Mindestens 8 Zeichen';
              }
              return null;
            },
          ),
          TextFormField(
            controller: _confirmController,
            decoration: const InputDecoration(labelText: 'Passwort bestätigen'),
            obscureText: true,
            validator: (value) {
              if (value != _passwordController.text) {
                return 'Passwörter stimmen nicht überein';
              }
              return null;
            },
          ),
        ],
      ),
    );
  }
}
```

Datum-Validierung (Start < Ende)

```
class DateRangeForm extends StatefulWidget {
  @override
  State<DateRangeForm> createState() => _DateRangeFormState();
}

class _DateRangeFormState extends State<DateRangeForm> {
  DateTime? _startDate;
  DateTime? _endDate;
```

```
  String? _validateStartDate(DateTime? value) {
    if (value == null) return 'Startdatum erforderlich';
    if (_endDate != null && value.isAfter(_endDate!)) {
      return 'Startdatum muss vor Enddatum liegen';
    }
    return null;
  }

  String? _validateEndDate(DateTime? value) {
    if (value == null) return 'Enddatum erforderlich';
    if (_startDate != null && value.isBefore(_startDate!)) {
      return 'Enddatum muss nach Startdatum liegen';
    }
    return null;
  }
}
```

3.10.0.6 5. AutovalidateMode

```
Form(
  // Nie automatisch validieren
  autovalidateMode: AutovalidateMode.disabled,

  // Immer validieren (bei jeder Eingabe)
  autovalidateMode: AutovalidateMode.always,

  // Nach erster User-Interaktion
  autovalidateMode: AutovalidateMode.onUserInteraction,
)
```

Dynamisch umschalten

```
class SmartForm extends StatefulWidget {
  @override
  State<SmartForm> createState() => _SmartFormState();
}

class _SmartFormState extends State<SmartForm> {
  final _formKey = GlobalKey<FormState>();
  var _autovalidate = AutovalidateMode.disabled;

  void _submit() {
    if (_formKey.currentState!.validate()) {
      // Success
    } else {
      // Nach erstem Submit: Autovalidierung aktivieren
      setState(() {
        _autovalidate = AutovalidateMode.onUserInteraction;
      });
    }
```

```
  }

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      autovalidateMode: _autovalidate,
      child: Column(
        children: [
          TextFormField(
            validator: (v) => v!.isEmpty ? 'Erforderlich' : null,
          ),
          ElevatedButton(
            onPressed: _submit,
            child: const Text('Submit'),
          ),
        ],
      ),
    );
  }
}
```

3.10.0.7 6. Async Validierung

Benutzername Verfügbarkeit

```
class UsernameField extends StatefulWidget {
  @override
  State<UsernameField> createState() => _UsernameFieldState();
}

class _UsernameFieldState extends State<UsernameField> {
  final _controller = TextEditingController();
  String? _asyncError;
  bool _isChecking = false;
  Timer? _debounce;

  @override
  void dispose() {
    _debounce?.cancel();
    _controller.dispose();
    super.dispose();
  }

  Future<void> _checkUsername(String username) async {
    setState(() {
      _isChecking = true;
      _asyncError = null;
    });

    // Simuliere API-Call
```

```
    await Future.delayed(const Duration(milliseconds: 500));

    // Beispiel: "admin" ist vergeben
    final isAvailable = username.toLowerCase() != 'admin';

    setState(() {
      _isChecking = false;
      _asyncError = isAvailable ? null : 'Benutzername bereits vergeben';
    });
  }

  @override
  Widget build(BuildContext context) {
    return TextFormField(
      controller: _controller,
      decoration: InputDecoration(
        labelText: 'Benutzername',
        errorText: _asyncError,
        suffixIcon: _isChecking
            ? const SizedBox(
                width: 20,
                height: 20,
                child: CircularProgressIndicator(strokeWidth: 2),
              )
            : _asyncError == null && _controller.text.isNotEmpty
                ? const Icon(Icons.check, color: Colors.green)
                : null,
      ),
      onChanged: (value) {
        _debounce?.cancel();
        if (value.length >= 3) {
          _debounce = Timer(
            const Duration(milliseconds: 500),
            () => _checkUsername(value),
          );
        }
      },
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Benutzername erforderlich';
        }
        if (value.length < 3) {
          return 'Mindestens 3 Zeichen';
        }
        // Async-Fehler in Standard-Validator einbeziehen
        if (_asyncError != null) {
          return _asyncError;
        }
        return null;
      },
    );
```

```
  }
}
```

3.10.0.8 7. Form-Level Validierung

```
class PaymentForm extends StatefulWidget {
  @override
  State<PaymentForm> createState() => _PaymentFormState();
}

class _PaymentFormState extends State<PaymentForm> {
  final _formKey = GlobalKey<FormState>();
  String? _formError;

  String _cardNumber = '';
  String _expiry = '';
  String _cvv = '';

  void _submit() {
    setState(() => _formError = null);

    if (!_formKey.currentState!.validate()) return;

    // Form-Level Validierung
    if (_cardNumber.startsWith('4') && _cvv.length != 3) {
      setState(() => _formError = 'Visa-Karten benötigen 3-stelligen CVV');
      return;
    }

    // Expiry-Datum prüfen
    final parts = _expiry.split('/');
    final month = int.tryParse(parts[0]);
    final year = int.tryParse('20${parts[1]}');

    if (month != null && year != null) {
      final expiry = DateTime(year, month + 1, 0);
      if (expiry.isBefore(DateTime.now())) {
        setState(() => _formError = 'Karte ist abgelaufen');
        return;
      }
    }

    // Alles OK
    print('Payment valid!');
  }

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
```

```
      child: Column(
        children: [
          if (_formError != null)
            Container(
              padding: const EdgeInsets.all(8),
              color: Colors.red[100],
              child: Text(
                _formError!,
                style: const TextStyle(color: Colors.red),
              ),
            ),
          TextFormField(
            decoration: const InputDecoration(labelText: 'Kartennummer'),
            onChanged: (v) => _cardNumber = v,
            validator: (v) =>
                v!.length < 16 ? 'Ungültige Kartennummer' : null,
          ),
          // Weitere Felder...
          ElevatedButton(onPressed: _submit, child: const Text('Zahlen')),
        ],
      ),
    );
  }
}
```

3.10.0.9 8. Validator-Klasse Pattern

```
abstract class FormValidator<T> {
  String? validate(T? value);
}

class RequiredValidator implements FormValidator<String> {
  final String message;
  RequiredValidator([this.message = 'Pflichtfeld']);

  @override
  String? validate(String? value) {
    if (value == null || value.isEmpty) return message;
    return null;
  }
}

class EmailValidator implements FormValidator<String> {
  @override
  String? validate(String? value) {
    if (value == null || value.isEmpty) return null;
    if (!value.contains('@')) return 'Ungültige E-Mail';
    return null;
  }
}
```

```
class CompositeValidator implements FormValidator<String> {
  final List<FormValidator<String>> validators;

  CompositeValidator(this.validators);

  @override
  String? validate(String? value) {
    for (final v in validators) {
      final error = v.validate(value);
      if (error != null) return error;
    }
    return null;
  }
}

// Verwendung
final emailValidator = CompositeValidator([
  RequiredValidator('E-Mail erforderlich'),
  EmailValidator(),
]);

TextFormField(
  validator: emailValidator.validate,
);
```

3.10.0.10 Zusammenfassung

Konzept	Verwendung
Einfache Validatoren	Pflichtfeld, Länge, Format
Regex	E-Mail, Telefon, Passwort-Regeln
Validator.combine	Mehrere Regeln kombinieren
Cross-Field	Passwort-Bestätigung, Datum-Bereiche
AutovalidateMode	Wann validiert wird
Async-Validierung	Server-Checks (Username, etc.)
Form-Level	Komplexe Abhängigkeiten

Best Practices: 1. Wiederverwendbare Validator-Funktionen 2. User-freundliche Fehlermeldungen 3. Autovalidate erst nach erstem Submit 4. Async-Validierung mit Debounce 5. Form-Level für komplexe Logik

3.10.1 Übung

3.10.1.1 Ziel

Verschiedene Validierungsszenarien implementieren.

3.10.1.2 Aufgabe 1: Validator-Bibliothek (25 min)

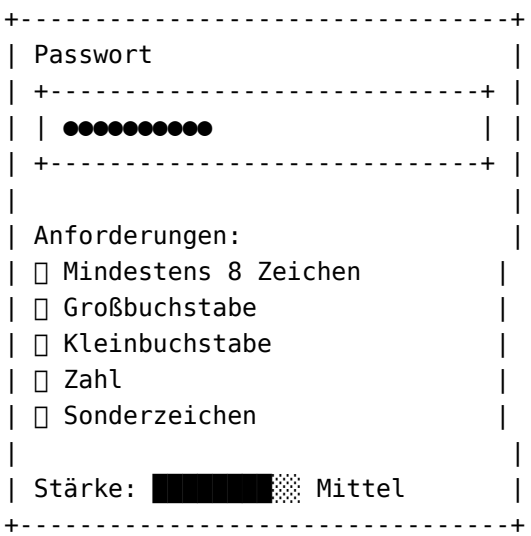
Erstelle eine `Validators` Klasse mit wiederverwendbaren Validatoren:

```
class Validators {
  // Pflichtfeld
  static String? required(String? value);

  // Mindestlänge
  static String? Function(String?) minLength(int length);

  // Maximallänge
  static String? Function(String?) maxLength(int length);

  // E-Mail Format
  static String? email(String? value);

  // Nur Zahlen
  static String? numeric(String? value);

  // Nur Buchstaben
  static String? alpha(String? value);

  // Alphanumerisch
  static String? alphanumeric(String? value);

  // Benutzerdefiniertes Pattern
  static String? Function(String?) pattern(RegExp regex, String message);

  // Mehrere Validatoren kombinieren
  static String? Function(String?) combine(List<String? Function(String?)>     ↪
 ↪  validators);
}
```

Teste jeden Validator mit verschiedenen Eingaben.

3.10.1.3 Aufgabe 2: Passwort-Validierung (20 min)

Erstelle einen Passwort-Validator mit Echtzeit-Feedback:



- Zeige jede Anforderung einzeln
- Grünes Häkchen wenn erfüllt
- Stärke-Anzeige aktualisiert sich live

3.10.1.4 Aufgabe 3: Registrierungsformular (30 min)

Erstelle ein Registrierungsformular mit:

1. **Benutzername**
   - Pflichtfeld, 3-20 Zeichen
   - Nur Buchstaben, Zahlen, Unterstriche
   - Async-Check: Simuliere "admin" und "test" als vergeben
2. **E-Mail**
   - Pflichtfeld
   - Gültiges E-Mail Format
3. **Passwort**
   - Alle Anforderungen aus Aufgabe 2
4. **Passwort bestätigen**
   - Muss mit Passwort übereinstimmen
5. **Geburtsdatum**
   - Muss in der Vergangenheit liegen
   - Mindestens 13 Jahre alt
6. **AGBs akzeptieren**
   - Checkbox muss angehakt sein

3.10.1.5 Aufgabe 4: Cross-Field Validierung (20 min)

Erstelle ein Datum-Range Formular:

```
class DateRangeForm {
  DateTime? startDate;
  DateTime? endDate;
  int? maxDays;  // Optional: Maximale Dauer
}
```

Validierungen: - Startdatum erforderlich - Enddatum erforderlich - Enddatum muss nach Startdatum liegen - Wenn maxDays gesetzt: Differenz darf maxDays nicht überschreiten

`Fehler: Enddatum (15.03.2024) liegt vor Startdatum (20.03.2024)`

3.10.1.6 Aufgabe 5: AutovalidateMode verstehen (15 min)

Erstelle drei identische Formulare mit verschiedenen AutovalidateModes:

1. `AutovalidateMode.disabled`
2. `AutovalidateMode.always`
3. `AutovalidateMode.onUserInteraction`

Beobachte und dokumentiere: - Wann werden Fehler angezeigt? - Wie fühlt sich die UX an? - Welcher Modus ist wann am besten?

3.10.1.7 Aufgabe 6: Async-Validierung mit Debounce (20 min)

Implementiere ein E-Mail Feld mit Server-Validierung:

```
Future<bool> checkEmailExists(String email) async {
  await Future.delayed(Duration(milliseconds: 500));
  // Simuliere: Diese E-Mails sind "registriert"
  return ['test@example.com', 'admin@example.com'].contains(email);
}
```

Anforderungen: - Debounce: Erst nach 500ms Pause prüfen - Loading-Indikator während der Prüfung - Grünes Häkchen wenn E-Mail verfügbar - Rotes X wenn bereits registriert - Fehler im Validator berücksichtigen

3.10.1.8 Aufgabe 7: Formular mit Conditional Fields (25 min)

Erstelle ein Versandformular:

```
Lieferart: ○ Abholung  ○ Versand

[Wenn Versand ausgewählt:]
Straße: _____________
PLZ: ____
Stadt: _____________
Land: [Dropdown: DE, AT, CH]

[Wenn DE ausgewählt:]
Packstation: ○ Ja  ○ Nein

[Wenn Packstation Ja:]
Packstation Nr: ________
Postnummer: __________
```

- Felder werden nur angezeigt wenn relevant
- Validierung nur für sichtbare Felder
- PLZ-Format abhängig vom Land

3.10.1.9 Abgabe-Checkliste

- ☐ Validators Klasse mit allen Methoden
- ☐ Passwort-Validierung mit Live-Feedback
- ☐ Registrierungsformular vollständig
- ☐ Cross-Field Validierung für Datum
- ☐ AutovalidateMode Dokumentation
- ☐ Async-Validierung mit Debounce
- ☐ Conditional Fields funktionieren

3.10.2 Lösung

3.10.2.1 Aufgabe 1: Validators Klasse

```
class Validators {
  static String? required(String? value) {
    if (value == null || value.trim().isEmpty) {
      return 'Dieses Feld ist erforderlich';
    }
    return null;
```

```
}

static String? Function(String?) minLength(int length) {
  return (value) {
    if (value != null && value.length < length) {
      return 'Mindestens $length Zeichen erforderlich';
    }
    return null;
  };
}

static String? Function(String?) maxLength(int length) {
  return (value) {
    if (value != null && value.length > length) {
      return 'Maximal $length Zeichen erlaubt';
    }
    return null;
  };
}

static String? email(String? value) {
  if (value == null || value.isEmpty) return null;

  final regex = RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$');
  if (!regex.hasMatch(value)) {
    return 'Ungültige E-Mail Adresse';
  }
  return null;
}

static String? numeric(String? value) {
  if (value == null || value.isEmpty) return null;

  if (!RegExp(r'^[0-9]+$').hasMatch(value)) {
    return 'Nur Zahlen erlaubt';
  }
  return null;
}

static String? alpha(String? value) {
  if (value == null || value.isEmpty) return null;

  if (!RegExp(r'^[a-zA-Z]+$').hasMatch(value)) {
    return 'Nur Buchstaben erlaubt';
  }
  return null;
}

static String? alphanumeric(String? value) {
  if (value == null || value.isEmpty) return null;
```

```
    if (!RegExp(r'^[a-zA-Z0-9]+$').hasMatch(value)) {
      return 'Nur Buchstaben und Zahlen erlaubt';
    }
    return null;
  }

  static String? Function(String?) pattern(RegExp regex, String message) {
    return (value) {
      if (value != null && value.isNotEmpty && !regex.hasMatch(value)) {
        return message;
      }
      return null;
    };
  }

  static String? Function(String?) combine(
      List<String? Function(String?)> validators) {
    return (value) {
      for (final validator in validators) {
        final error = validator(value);
        if (error != null) return error;
      }
      return null;
    };
  }
}
```

3.10.2.2 Aufgabe 2: Passwort-Validierung mit Live-Feedback

```
class PasswordRequirement {
  final String label;
  final bool Function(String) check;

  PasswordRequirement(this.label, this.check);
}

class PasswordFieldWithRequirements extends StatefulWidget {
  final TextEditingController controller;

  const PasswordFieldWithRequirements({
    super.key,
    required this.controller,
  });

  @override
  State<PasswordFieldWithRequirements> createState() =>
      _PasswordFieldWithRequirementsState();
}

class _PasswordFieldWithRequirementsState
```

```
  extends State<PasswordFieldWithRequirements> {
bool _obscure = true;
String _password = '';

final _requirements = [
  PasswordRequirement('Mindestens 8 Zeichen', (p) => p.length >= 8),
  PasswordRequirement('Großbuchstabe', (p) => p.contains(RegExp(r'[A-Z]'))),
  PasswordRequirement('Kleinbuchstabe', (p) => p.contains(RegExp(r'[a-z]'))),
  PasswordRequirement('Zahl', (p) => p.contains(RegExp(r'[0-9]'))),
  PasswordRequirement(
      'Sonderzeichen', (p) => p.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]'))),
];

int get _fulfilledCount =>
    _requirements.where((r) => r.check(_password)).length;

double get _strength => _requirements.isEmpty
    ? 0
    : _fulfilledCount / _requirements.length;

String get _strengthLabel {
  if (_strength < 0.4) return 'Schwach';
  if (_strength < 0.8) return 'Mittel';
  return 'Stark';
}

Color get _strengthColor {
  if (_strength < 0.4) return Colors.red;
  if (_strength < 0.8) return Colors.orange;
  return Colors.green;
}

@override
void initState() {
  super.initState();
  widget.controller.addListener(() {
    setState(() => _password = widget.controller.text);
  });
}

@override
Widget build(BuildContext context) {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      TextFormField(
        controller: widget.controller,
        obscureText: _obscure,
        decoration: InputDecoration(
          labelText: 'Passwort',
          border: const OutlineInputBorder(),
```

```
            suffixIcon: IconButton(
              icon: Icon(_obscure ? Icons.visibility : Icons.visibility_off),
              onPressed: () => setState(() => _obscure = !_obscure),
            ),
          ),
          validator: (value) {
            if (_fulfilledCount < _requirements.length) {
              return 'Nicht alle Anforderungen erfüllt';
            }
            return null;
          },
        ),
        if (_password.isNotEmpty) ...[
          const SizedBox(height: 16),
          const Text('Anforderungen:',
              style: TextStyle(fontWeight: FontWeight.bold)),
          const SizedBox(height: 8),
          ..._requirements.map((req) {
            final fulfilled = req.check(_password);
            return Row(
              children: [
                Icon(
                  fulfilled ? Icons.check_circle : Icons.cancel,
                  color: fulfilled ? Colors.green : Colors.red,
                  size: 20,
                ),
                const SizedBox(width: 8),
                Text(req.label),
              ],
            );
          }),
          const SizedBox(height: 16),
          Row(
            children: [
              Expanded(
                child: LinearProgressIndicator(
                  value: _strength,
                  backgroundColor: Colors.grey[300],
                  valueColor: AlwaysStoppedAnimation(_strengthColor),
                ),
              ),
              const SizedBox(width: 8),
              Text(
                'Stärke: $_strengthLabel',
                style: TextStyle(
                  color: _strengthColor,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ],
          ),
```

```
        ],
      ],
    );
  }
}
```

3.10.2.3 Aufgabe 3: Registrierungsformular

```
class RegistrationForm extends StatefulWidget {
  const RegistrationForm({super.key});

  @override
  State<RegistrationForm> createState() => _RegistrationFormState();
}

class _RegistrationFormState extends State<RegistrationForm> {
  final _formKey = GlobalKey<FormState>();

  final _usernameController = TextEditingController();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  final _confirmController = TextEditingController();

  DateTime? _birthDate;
  bool _acceptTerms = false;

  String? _usernameAsyncError;
  bool _checkingUsername = false;
  Timer? _debounce;

  Future<void> _checkUsername(String username) async {
    setState(() {
      _checkingUsername = true;
      _usernameAsyncError = null;
    });

    await Future.delayed(const Duration(milliseconds: 500));

    final taken = ['admin', 'test', 'user'].contains(username.toLowerCase());

    setState(() {
      _checkingUsername = false;
      _usernameAsyncError = taken ? 'Benutzername bereits vergeben' : null;
    });
  }

  void _submit() {
    if (!_acceptTerms) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Bitte AGBs akzeptieren')),
```

```
    );
    return;
  }

  if (_formKey.currentState!.validate() && _usernameAsyncError == null) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Registrierung erfolgreich!')),
    );
  }
}

@override
void dispose() {
  _debounce?.cancel();
  _usernameController.dispose();
  _emailController.dispose();
  _passwordController.dispose();
  _confirmController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Registrierung')),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(16),
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            // Benutzername
            TextFormField(
              controller: _usernameController,
              decoration: InputDecoration(
                labelText: 'Benutzername',
                border: const OutlineInputBorder(),
                errorText: _usernameAsyncError,
                suffixIcon: _checkingUsername
                    ? const SizedBox(
                        width: 20,
                        height: 20,
                        child: CircularProgressIndicator(strokeWidth: 2),
                      )
                    : _usernameAsyncError == null &&
                            _usernameController.text.length >= 3
                        ? const Icon(Icons.check, color: Colors.green)
                        : null,
              ),
              onChanged: (value) {
```

```
                _debounce?.cancel();
                if (value.length >= 3) {
                  _debounce = Timer(
                    const Duration(milliseconds: 500),
                    () => _checkUsername(value),
                  );
                }
              },
              validator: Validators.combine([
                Validators.required,
                Validators.minLength(3),
                Validators.maxLength(20),
                Validators.pattern(
                  RegExp(r'^[a-zA-Z0-9_]+$'),
                  'Nur Buchstaben, Zahlen und Unterstriche',
                ),
              ]),
            ),
            const SizedBox(height: 16),

            // E-Mail
            TextFormField(
              controller: _emailController,
              decoration: const InputDecoration(
                labelText: 'E-Mail',
                border: OutlineInputBorder(),
              ),
              keyboardType: TextInputType.emailAddress,
              validator: Validators.combine([
                Validators.required,
                Validators.email,
              ]),
            ),
            const SizedBox(height: 16),

            // Passwort
            PasswordFieldWithRequirements(controller: _passwordController),
            const SizedBox(height: 16),

            // Passwort bestätigen
            TextFormField(
              controller: _confirmController,
              decoration: const InputDecoration(
                labelText: 'Passwort bestätigen',
                border: OutlineInputBorder(),
              ),
              obscureText: true,
              validator: (value) {
                if (value != _passwordController.text) {
                  return 'Passwörter stimmen nicht überein';
                }
```

```
                return null;
              },
            ),
            const SizedBox(height: 16),

            // Geburtsdatum
            ListTile(
              title: Text(_birthDate == null
                  ? 'Geburtsdatum wählen'
                  : 'Geburtsdatum:
↪ ${_birthDate!.day}.${_birthDate!.month}.${_birthDate!.year}'),
              trailing: const Icon(Icons.calendar_today),
              onTap: () async {
                final date = await showDatePicker(
                  context: context,
                  initialDate: DateTime.now().subtract(
                    const Duration(days: 365 * 18),
                  ),
                  firstDate: DateTime(1900),
                  lastDate: DateTime.now(),
                );
                if (date != null) {
                  setState(() => _birthDate = date);
                }
              },
            ),
            if (_birthDate != null)
              Builder(
                builder: (context) {
                  final age = DateTime.now().year - _birthDate!.year;
                  if (age < 13) {
                    return const Text(
                      'Du musst mindestens 13 Jahre alt sein',
                      style: TextStyle(color: Colors.red),
                    );
                  }
                  return const SizedBox.shrink();
                },
              ),
            const SizedBox(height: 16),

            // AGBs
            CheckboxListTile(
              title: const Text('Ich akzeptiere die AGBs'),
              value: _acceptTerms,
              onChanged: (value) {
                setState(() => _acceptTerms = value ?? false);
              },
              controlAffinity: ListTileControlAffinity.leading,
            ),
            const SizedBox(height: 24),
```

```
              ElevatedButton(
                onPressed: _submit,
                child: const Text('Registrieren'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

3.10.2.4 Aufgabe 5: AutovalidateMode Vergleich

Modus	Verhalten	Beste Verwendung
`disabled`	Validiert nur bei `validate()` Aufruf	Standard-Formulare, Fehler erst bei Submit zeigen
`always`	Validiert bei jeder Eingabe sofort	Echtzeit-Feedback wichtig, aber kann störend sein
`onUserInteraction`	Validiert nach erster Interaktion	Bester Kompromiss - keine Fehler bei leerem Formular, aber Feedback nach Eingabe

Empfehlung: Start mit `disabled`, nach erstem Submit auf `onUserInteraction` wechseln:

```
var _autovalidate = AutovalidateMode.disabled;

void _submit() {
  if (_formKey.currentState!.validate()) {
    // Success
  } else {
    setState(() {
      _autovalidate = AutovalidateMode.onUserInteraction;
    });
  }
}
```

3.10.3 Ressourcen

3.10.3.1 Offizielle Dokumentation

- Build a form with validation
- FormState class
- FormFieldValidator

3.10.3.2 Regex Referenzen

- Regex101 - Online Regex Tester
- RegExr - Regex Lernen und Testen
- Dart RegExp

- Common Regex Patterns

3.10.3.3 Validierung Packages

- form_validator - Fertige Validatoren
- validators - String Validation
- email_validator - E-Mail Validierung
- phone_number - Telefonnummer Validierung

3.10.3.4 Form Builder

- flutter_form_builder
- reactive_forms
- formz - Form State Management

3.10.3.5 Tutorials

- Form Validation Best Practices
- Advanced Form Validation
- Async Validation

3.10.3.6 Videos

- Form Validation Deep Dive
- Custom Validators

3.10.3.7 UX Best Practices

- Form Design Guidelines
- Inline Validation
- Error Message Guidelines

3.11 Einheit 3.11: Dropdowns & Checkboxen

3.11.0.1 Lernziele

Nach dieser Einheit kannst du: - `DropdownButtonFormField` in Formularen verwenden - `Checkbox`, `Switch` und `Radio` einsetzen - `ChoiceChip` und `FilterChip` verwenden - Custom FormFields erstellen

3.11.0.2 1. DropdownButtonFormField

Basis-Verwendung

```
class MyForm extends StatefulWidget {
  @override
  State<MyForm> createState() => _MyFormState();
}

class _MyFormState extends State<MyForm> {
  String? _selectedCountry;

  final _countries = ['Deutschland', 'Österreich', 'Schweiz'];
```

```
  @override
  Widget build(BuildContext context) {
    return DropdownButtonFormField<String>(
      value: _selectedCountry,
      decoration: const InputDecoration(
        labelText: 'Land',
        border: OutlineInputBorder(),
      ),
      items: _countries.map((country) {
        return DropdownMenuItem(
          value: country,
          child: Text(country),
        );
      }).toList(),
      onChanged: (value) {
        setState(() => _selectedCountry = value);
      },
      validator: (value) {
        if (value == null) return 'Bitte Land auswählen';
        return null;
      },
    );
  }
}
```

Mit Enum

```
enum Priority { low, medium, high }

extension PriorityLabel on Priority {
  String get label {
    switch (this) {
      case Priority.low:
        return 'Niedrig';
      case Priority.medium:
        return 'Mittel';
      case Priority.high:
        return 'Hoch';
    }
  }

  Color get color {
    switch (this) {
      case Priority.low:
        return Colors.green;
      case Priority.medium:
        return Colors.orange;
      case Priority.high:
        return Colors.red;
    }
  }
```

```
}

// Verwendung
DropdownButtonFormField<Priority>(
  value: _priority,
  decoration: const InputDecoration(labelText: 'Priorität'),
  items: Priority.values.map((p) {
    return DropdownMenuItem(
      value: p,
      child: Row(
        children: [
          Icon(Icons.circle, color: p.color, size: 12),
          const SizedBox(width: 8),
          Text(p.label),
        ],
      ),
    );
  }).toList(),
  onChanged: (value) => setState(() => _priority = value),
);
```

Hint und DisabledHint

```
DropdownButtonFormField<String>(
  value: _value,
  hint: const Text('Bitte auswählen'),       // Wenn value == null
  disabledHint: const Text('Nicht verfügbar'), // Wenn onChanged == null
  items: _items,
  onChanged: _isEnabled ? (v) => setState(() => _value = v) : null,
);
```

3.11.0.3 2. Checkbox

Einfache Checkbox

```
class CheckboxDemo extends StatefulWidget {
  @override
  State<CheckboxDemo> createState() => _CheckboxDemoState();
}

class _CheckboxDemoState extends State<CheckboxDemo> {
  bool _isChecked = false;

  @override
  Widget build(BuildContext context) {
    return Checkbox(
      value: _isChecked,
      onChanged: (value) {
        setState(() => _isChecked = value ?? false);
      },
    );
  }
```

```
}
```

CheckboxListTile

```
CheckboxListTile(
  title: const Text('Newsletter abonnieren'),
  subtitle: const Text('Wöchentliche Updates erhalten'),
  secondary: const Icon(Icons.mail),
  value: _subscribeNewsletter,
  onChanged: (value) {
    setState(() => _subscribeNewsletter = value ?? false);
  },
  controlAffinity: ListTileControlAffinity.leading,  // Checkbox links
);
```

Tristate Checkbox

```
bool? _value;  // null, true, oder false

Checkbox(
  tristate: true,
  value: _value,
  onChanged: (value) {
    setState(() => _value = value);
  },
);

// Visuell:
// null -> Strich (-)
// true -> Häkchen ([x])
// false -> Leer
```

AGBs Checkbox mit Validierung

```
class TermsCheckbox extends FormField<bool> {
  TermsCheckbox({
    super.key,
    required bool value,
    required ValueChanged<bool> onChanged,
    String? Function(bool?)? validator,
  }) : super(
          initialValue: value,
          validator: validator,
          builder: (state) {
            return Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                CheckboxListTile(
                  title: const Text('Ich akzeptiere die AGBs'),
                  value: state.value,
                  onChanged: (v) {
                    state.didChange(v);
```

```
                  onChanged(v ?? false);
                },
                controlAffinity: ListTileControlAffinity.leading,
              ),
              if (state.hasError)
                Padding(
                  padding: const EdgeInsets.only(left: 16),
                  child: Text(
                    state.errorText!,
                    style: TextStyle(
                      color: Theme.of(state.context).colorScheme.error,
                      fontSize: 12,
                    ),
                  ),
                ),
            ],
          );
        },
      );
}

// Verwendung
TermsCheckbox(
  value: _acceptTerms,
  onChanged: (value) => setState(() => _acceptTerms = value),
  validator: (value) {
    if (value != true) return 'Bitte AGBs akzeptieren';
    return null;
  },
);
```

3.11.0.4 3. Switch

Einfacher Switch

```
Switch(
  value: _darkMode,
  onChanged: (value) {
    setState(() => _darkMode = value);
  },
);
```

SwitchListTile

```
SwitchListTile(
  title: const Text('Dark Mode'),
  subtitle: const Text('Dunkles Farbschema verwenden'),
  secondary: const Icon(Icons.dark_mode),
  value: _darkMode,
  onChanged: (value) {
    setState(() => _darkMode = value);
  },
```

```
);
```

Adaptive Switch (Platform-spezifisch)

```
Switch.adaptive(
  value: _value,
  onChanged: (value) => setState(() => _value = value),
);
// iOS: CupertinoSwitch
// Android: Material Switch
```

3.11.0.5 4. Radio

Radio Buttons

```
enum Gender { male, female, other }

class GenderSelector extends StatefulWidget {
  @override
  State<GenderSelector> createState() => _GenderSelectorState();
}

class _GenderSelectorState extends State<GenderSelector> {
  Gender? _gender;

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        RadioListTile<Gender>(
          title: const Text('Männlich'),
          value: Gender.male,
          groupValue: _gender,
          onChanged: (value) => setState(() => _gender = value),
        ),
        RadioListTile<Gender>(
          title: const Text('Weiblich'),
          value: Gender.female,
          groupValue: _gender,
          onChanged: (value) => setState(() => _gender = value),
        ),
        RadioListTile<Gender>(
          title: const Text('Divers'),
          value: Gender.other,
          groupValue: _gender,
          onChanged: (value) => setState(() => _gender = value),
        ),
      ],
    );
  }
}
```

Horizontale Radio Buttons

```
Row(
  children: Gender.values.map((g) {
    return Expanded(
      child: RadioListTile<Gender>(
        title: Text(g.name),
        value: g,
        groupValue: _gender,
        onChanged: (v) => setState(() => _gender = v),
        contentPadding: EdgeInsets.zero,
      ),
    );
  }).toList(),
);
```

3.11.0.6 5. ChoiceChip & FilterChip

ChoiceChip (Single Selection)

```
class SizeSelector extends StatefulWidget {
  @override
  State<SizeSelector> createState() => _SizeSelectorState();
}

class _SizeSelectorState extends State<SizeSelector> {
  String? _selectedSize;
  final _sizes = ['XS', 'S', 'M', 'L', 'XL'];

  @override
  Widget build(BuildContext context) {
    return Wrap(
      spacing: 8,
      children: _sizes.map((size) {
        return ChoiceChip(
          label: Text(size),
          selected: _selectedSize == size,
          onSelected: (selected) {
            setState(() {
              _selectedSize = selected ? size : null;
            });
          },
        );
      }).toList(),
    );
  }
}
```

FilterChip (Multi Selection)

```
class TagSelector extends StatefulWidget {
  @override
  State<TagSelector> createState() => _TagSelectorState();
```

```
}

class _TagSelectorState extends State<TagSelector> {
  final _allTags = ['Flutter', 'Dart', 'Firebase', 'API', 'UI/UX'];
  final Set<String> _selectedTags = {};

  @override
  Widget build(BuildContext context) {
    return Wrap(
      spacing: 8,
      runSpacing: 8,
      children: _allTags.map((tag) {
        return FilterChip(
          label: Text(tag),
          selected: _selectedTags.contains(tag),
          onSelected: (selected) {
            setState(() {
              if (selected) {
                _selectedTags.add(tag);
              } else {
                _selectedTags.remove(tag);
              }
            });
          },
        );
      }).toList(),
    );
  }
}
```

InputChip (mit Löschfunktion)

```
Wrap(
  spacing: 8,
  children: _selectedItems.map((item) {
    return InputChip(
      label: Text(item),
      onDeleted: () {
        setState(() => _selectedItems.remove(item));
      },
      deleteIcon: const Icon(Icons.close, size: 18),
    );
  }).toList(),
);
```

3.11.0.7 6. Custom FormField

Multi-Select als FormField

```
class MultiSelectFormField<T> extends FormField<Set<T>> {
  MultiSelectFormField({
    super.key,
```

```
    required List<T> options,
    required String Function(T) labelBuilder,
    Set<T>? initialValue,
    String? Function(Set<T>?)? validator,
    void Function(Set<T>?)? onSaved,
  }) : super(
          initialValue: initialValue ?? {},
          validator: validator,
          onSaved: onSaved,
          builder: (state) {
            return Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Wrap(
                  spacing: 8,
                  runSpacing: 8,
                  children: options.map((option) {
                    final selected = state.value?.contains(option) ?? false;
                    return FilterChip(
                      label: Text(labelBuilder(option)),
                      selected: selected,
                      onSelected: (isSelected) {
                        final newValue = Set<T>.from(state.value ?? {});
                        if (isSelected) {
                          newValue.add(option);
                        } else {
                          newValue.remove(option);
                        }
                        state.didChange(newValue);
                      },
                    );
                  }).toList(),
                ),
                if (state.hasError)
                  Padding(
                    padding: const EdgeInsets.only(top: 8),
                    child: Text(
                      state.errorText!,
                      style: TextStyle(
                        color: Theme.of(state.context).colorScheme.error,
                        fontSize: 12,
                      ),
                    ),
                  ),
              ],
            );
          },
        );
}

// Verwendung
```

```
MultiSelectFormField<String>(
  options: ['Option A', 'Option B', 'Option C'],
  labelBuilder: (option) => option,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Bitte mindestens eine Option wählen';
    }
    return null;
  },
  onSaved: (value) {
    print('Selected: $value');
  },
);
```

3.11.0.8 7. Segmented Button (Material 3)

```
enum View { list, grid, table }

class ViewSelector extends StatefulWidget {
  @override
  State<ViewSelector> createState() => _ViewSelectorState();
}

class _ViewSelectorState extends State<ViewSelector> {
  View _selectedView = View.list;

  @override
  Widget build(BuildContext context) {
    return SegmentedButton<View>(
      segments: const [
        ButtonSegment(
          value: View.list,
          label: Text('Liste'),
          icon: Icon(Icons.list),
        ),
        ButtonSegment(
          value: View.grid,
          label: Text('Grid'),
          icon: Icon(Icons.grid_view),
        ),
        ButtonSegment(
          value: View.table,
          label: Text('Tabelle'),
          icon: Icon(Icons.table_chart),
        ),
      ],
      selected: {_selectedView},
      onSelectionChanged: (selected) {
        setState(() => _selectedView = selected.first);
      },
```

```
    );
  }
}
```

Multi-Select Segmented Button

```
Set<String> _selectedDays = {'Mo', 'Mi', 'Fr'};

SegmentedButton<String>(
  segments: ['Mo', 'Di', 'Mi', 'Do', 'Fr', 'Sa', 'So']
      .map((d) => ButtonSegment(value: d, label: Text(d)))
      .toList(),
  selected: _selectedDays,
  onSelectionChanged: (selected) {
    setState(() => _selectedDays = selected);
  },
  multiSelectionEnabled: true,
);
```

3.11.0.9 Zusammenfassung

Widget	Use Case
`DropdownButtonFormField`	Einzelauswahl aus Liste
`Checkbox` / `CheckboxListTile`	Boolean Werte
`Switch` / `SwitchListTile`	Ein/Aus Toggle
`Radio` / `RadioListTile`	Exklusive Auswahl
`ChoiceChip`	Einzelauswahl (kompakt)
`FilterChip`	Mehrfachauswahl
`InputChip`	Auswahl mit Löschfunktion
`SegmentedButton`	Exklusive/Multi Auswahl (Material 3)

Tipps: - `ListTile` Varianten für bessere Beschriftung - `Wrap` für flexible Chip-Layouts - Custom `FormField` für komplexe Validierung - Enum + Extension für typsichere Auswahlen

3.11.1 Übung

3.11.1.1 Ziel

Ein Einstellungsformular mit verschiedenen Auswahloptionen erstellen.

3.11.1.2 Aufgabe 1: Dropdown Basics (20 min)

Erstelle ein Bestellformular mit Dropdowns:

1. **Produktkategorie** (Pflichtfeld)
   - Elektronik, Kleidung, Bücher, Sport
2. **Zahlungsart**
   - Kreditkarte, PayPal, Rechnung, Vorkasse
3. **Lieferzeit**
   - Standard (3-5 Tage), Express (1-2 Tage), Overnight

Anforderungen: - Hint-Text wenn nichts ausgewählt - Validierung: Kategorie ist Pflichtfeld - Icons vor den Optionen

3.11.1.3 Aufgabe 2: Checkboxen (20 min)

Erstelle einen Newsletter-Bereich:

```
+---------------------------------+
| Newsletter Einstellungen        |
+---------------------------------+
| ☑ Newsletter abonnieren        |
|                                 |
| Wenn abonniert:                 |
| ☑ Wöchentliche Updates         |
| ☐ Monatliche Zusammenfassung   |
| ☑ Produktneuheiten             |
| ☐ Sonderangebote               |
|                                 |
| [Alle auswählen] [Keine]        |
+---------------------------------+
```

- Haupt-Checkbox aktiviert/deaktiviert Unteroptionen
- “Alle auswählen” und “Keine” Buttons
- Unteroptionen nur sichtbar wenn Newsletter aktiv

3.11.1.4 Aufgabe 3: Radio Buttons (15 min)

Erstelle einen Versandoptionen-Wähler:

```
enum ShippingOption {
  standard,   // Kostenlos, 5-7 Tage
  express,    // 4,99€, 2-3 Tage
  overnight,  // 9,99€, Nächster Tag
}
```

Zeige für jede Option: - Titel - Preis - Lieferzeit - Icon

Berechne den Gesamtpreis basierend auf der Auswahl.

3.11.1.5 Aufgabe 4: Switches (15 min)

Erstelle einen Settings-Bereich:

```
+---------------------------------+
| App Einstellungen               |
+---------------------------------+
| Dark Mode                [===] |
| Benachrichtigungen       [===] |
| Standortdienste          [   ] |
| Automatische Updates     [===] |
| Datensammlung            [   ] |
+---------------------------------+
```

- Jeder Switch hat Titel und Icon
- Änderungen werden sofort “gespeichert” (State)
- Zeige Snackbar bei Änderung

3.11.1.6 Aufgabe 5: Chips (25 min)

Erstelle einen Tag-Filter:

```
+---------------------------------+
| Kategorien filtern              |
+---------------------------------+
| +----+ +----+ +----+ +----+    |
| |Web | |App | |API | | DB |    |
| +----+ +----+ +----+ +----+    |
| +-----+ +------+ +-------+     |
| |UI/UX| |DevOps| |Testing|     |
| +-----+ +------+ +-------+     |
|                                 |
| Ausgewählt: Web, API            |
| [Filter anwenden]               |
+---------------------------------+
```

Features: - FilterChip für Multi-Select - Zeige ausgewählte Tags als Text - “Filter anwenden” Button - Mindestens eine Auswahl erforderlich

3.11.1.7 Aufgabe 6: Größenauswahl mit ChoiceChip (15 min)

Erstelle eine Größenauswahl für einen Shop:

```
Größe auswählen:
[XS] [S] [M*] [L] [XL] [XXL]

* = nicht verfügbar (disabled)
```

- Nur eine Größe auswählbar
- Einige Größen als “nicht verfügbar” markieren
- Disabled Chips visuell unterscheiden
- Validierung: Größe muss ausgewählt sein

3.11.1.8 Aufgabe 7: Komplett-Formular (30 min)

Erstelle ein vollständiges Produktkonfigurations-Formular:

```
+---------------------------------+
| Produkt konfigurieren           |
+---------------------------------+
| Modell: [MacBook Pro    ▼]      |
|                                 |
| Farbe:                          |
| ○ Space Grau  ○ Silber          |
|                                 |
| Speicher:                       |
| [256GB] [512GB] [1TB] [2TB]     |
|                                 |
| Extras:                         |
| ☑ AppleCare+ (+149€)            |
| □ Magic Mouse (+99€)            |
| □ Magic Keyboard (+129€)        |
|                                 |
```

```
| Gravur: [Toggle]                |
| [Wenn aktiv: TextField]         |
|                                 |
| Geschenkverpackung: [Toggle]    |
|                                 |
| ------------------------------- |
| Gesamtpreis: 2.497€             |
|                                 |
| [In den Warenkorb]              |
+---------------------------------+
```

Anforderungen: - Dropdown für Modell (verschiedene Basispreise) - Radio für Farbe - ChoiceChip für Speicher (Aufpreis) - Checkboxen für Extras (mit Preisen) - Switch für Gravur (zeigt TextField wenn aktiv) - Switch für Geschenkverpackung - Echtzeit-Preisberechnung - Validierung vor "In den Warenkorb"

3.11.1.9 Abgabe-Checkliste

- ☐ Dropdowns mit Validierung
- ☐ Checkbox-Gruppe mit Master-Checkbox
- ☐ Radio Buttons mit Beschreibung
- ☐ Switches in Settings-Stil
- ☐ FilterChip Multi-Select
- ☐ ChoiceChip Single-Select
- ☐ Komplett-Formular mit Preisberechnung

3.11.2 Lösung

3.11.2.1 Aufgabe 7: Komplett-Formular

```
import 'package:flutter/material.dart';

enum MacModel {
  air('MacBook Air', 1199),
  pro14('MacBook Pro 14"', 1999),
  pro16('MacBook Pro 16"', 2499);

  final String name;
  final int basePrice;

  const MacModel(this.name, this.basePrice);
}

enum MacColor { spaceGray, silver }

class ProductConfigForm extends StatefulWidget {
  const ProductConfigForm({super.key});

  @override
  State<ProductConfigForm> createState() => _ProductConfigFormState();
}
```

```
class _ProductConfigFormState extends State<ProductConfigForm> {
  final _formKey = GlobalKey<FormState>();

  // Selections
  MacModel? _selectedModel;
  MacColor _selectedColor = MacColor.spaceGray;
  String _selectedStorage = '256GB';
  bool _appleCare = false;
  bool _magicMouse = false;
  bool _magicKeyboard = false;
  bool _engraving = false;
  bool _giftWrap = false;

  final _engravingController = TextEditingController();

  // Preise
  final _storagePrices = {
    '256GB': 0,
    '512GB': 200,
    '1TB': 400,
    '2TB': 600,
  };

  int get _totalPrice {
    if (_selectedModel == null) return 0;

    int price = _selectedModel!.basePrice;
    price += _storagePrices[_selectedStorage] ?? 0;
    if (_appleCare) price += 149;
    if (_magicMouse) price += 99;
    if (_magicKeyboard) price += 129;
    if (_giftWrap) price += 10;

    return price;
  }

  void _addToCart() {
    if (_formKey.currentState!.validate()) {
      showDialog(
        context: context,
        builder: (context) => AlertDialog(
          title: const Text('Zum Warenkorb hinzugefügt'),
          content: Column(
            mainAxisSize: MainAxisSize.min,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text('Modell: ${_selectedModel!.name}'),
              Text('Farbe: ${_selectedColor.name}'),
              Text('Speicher: $_selectedStorage'),
              if (_appleCare) const Text('+ AppleCare+'),
              if (_magicMouse) const Text('+ Magic Mouse'),
```

```
            if (_magicKeyboard) const Text('+ Magic Keyboard'),
            if (_engraving) Text('Gravur: ${_engravingController.text}'),
            if (_giftWrap) const Text('+ Geschenkverpackung'),
            const Divider(),
            Text(
              'Gesamt: ${_totalPrice}€',
              style: const TextStyle(fontWeight: FontWeight.bold),
            ),
          ],
        ),
        actions: [
          ElevatedButton(
            onPressed: () => Navigator.pop(context),
            child: const Text('OK'),
          ),
        ],
      ),
    );
  }
}

@override
void dispose() {
  _engravingController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Produkt konfigurieren')),
    body: Form(
      key: _formKey,
      child: ListView(
        padding: const EdgeInsets.all(16),
        children: [
          // Modell Dropdown
          DropdownButtonFormField<MacModel>(
            value: _selectedModel,
            decoration: const InputDecoration(
              labelText: 'Modell',
              border: OutlineInputBorder(),
            ),
            hint: const Text('Modell auswählen'),
            items: MacModel.values.map((model) {
              return DropdownMenuItem(
                value: model,
                child: Text('${model.name} (ab ${model.basePrice}€)'),
              );
            }).toList(),
            onChanged: (value) {
```

```
            setState(() => _selectedModel = value);
          },
          validator: (value) {
            if (value == null) return 'Bitte Modell auswählen';
            return null;
          },
        ),
        const SizedBox(height: 24),

        // Farbe Radio
        const Text('Farbe',
            style: TextStyle(fontWeight: FontWeight.bold)),
        Row(
          children: [
            Expanded(
              child: RadioListTile<MacColor>(
                title: const Text('Space Grau'),
                value: MacColor.spaceGray,
                groupValue: _selectedColor,
                onChanged: (v) => setState(() => _selectedColor = v!),
              ),
            ),
            Expanded(
              child: RadioListTile<MacColor>(
                title: const Text('Silber'),
                value: MacColor.silver,
                groupValue: _selectedColor,
                onChanged: (v) => setState(() => _selectedColor = v!),
              ),
            ),
          ],
        ),
        const SizedBox(height: 16),

        // Speicher ChoiceChips
        const Text('Speicher',
            style: TextStyle(fontWeight: FontWeight.bold)),
        const SizedBox(height: 8),
        Wrap(
          spacing: 8,
          children: _storagePrices.entries.map((entry) {
            final label = entry.value > 0
                ? '${entry.key} (+${entry.value}€)'
                : entry.key;
            return ChoiceChip(
              label: Text(label),
              selected: _selectedStorage == entry.key,
              onSelected: (selected) {
                if (selected) {
                  setState(() => _selectedStorage = entry.key);
                }
```

```
              },
            );
          }).toList(),
        ),
        const SizedBox(height: 24),

        // Extras Checkboxes
        const Text('Extras',
            style: TextStyle(fontWeight: FontWeight.bold)),
        CheckboxListTile(
          title: const Text('AppleCare+'),
          subtitle: const Text('+149€'),
          value: _appleCare,
          onChanged: (v) => setState(() => _appleCare = v!),
          controlAffinity: ListTileControlAffinity.leading,
        ),
        CheckboxListTile(
          title: const Text('Magic Mouse'),
          subtitle: const Text('+99€'),
          value: _magicMouse,
          onChanged: (v) => setState(() => _magicMouse = v!),
          controlAffinity: ListTileControlAffinity.leading,
        ),
        CheckboxListTile(
          title: const Text('Magic Keyboard'),
          subtitle: const Text('+129€'),
          value: _magicKeyboard,
          onChanged: (v) => setState(() => _magicKeyboard = v!),
          controlAffinity: ListTileControlAffinity.leading,
        ),
        const SizedBox(height: 16),

        // Gravur Switch
        SwitchListTile(
          title: const Text('Gravur'),
          subtitle: const Text('Personalisierte Gravur hinzufügen'),
          value: _engraving,
          onChanged: (v) => setState(() => _engraving = v),
        ),
        if (_engraving)
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 16),
            child: TextFormField(
              controller: _engravingController,
              decoration: const InputDecoration(
                labelText: 'Gravurtext',
                hintText: 'Max. 20 Zeichen',
                border: OutlineInputBorder(),
              ),
              maxLength: 20,
              validator: (value) {
```

```
            if (_engraving && (value == null || value.isEmpty)) {
              return 'Bitte Gravurtext eingeben';
            }
            return null;
          },
        ),
      ),
    const SizedBox(height: 8),

    // Geschenkverpackung Switch
    SwitchListTile(
      title: const Text('Geschenkverpackung'),
      subtitle: const Text('+10€'),
      value: _giftWrap,
      onChanged: (v) => setState(() => _giftWrap = v),
    ),

    const Divider(height: 32),

    // Preis
    Container(
      padding: const EdgeInsets.all(16),
      decoration: BoxDecoration(
        color: Colors.grey[100],
        borderRadius: BorderRadius.circular(8),
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          const Text(
            'Gesamtpreis:',
            style: TextStyle(fontSize: 18),
          ),
          Text(
            '${_totalPrice}€',
            style: const TextStyle(
              fontSize: 24,
              fontWeight: FontWeight.bold,
            ),
          ),
        ],
      ),
    ),
    const SizedBox(height: 24),

    // Submit Button
    SizedBox(
      height: 48,
      child: ElevatedButton.icon(
        onPressed: _selectedModel == null ? null : _addToCart,
        icon: const Icon(Icons.shopping_cart),
```

```
                label: const Text('In den Warenkorb'),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

3.11.2.2 Aufgabe 2: Newsletter Checkboxen

```
class NewsletterSettings extends StatefulWidget {
  const NewsletterSettings({super.key});

  @override
  State<NewsletterSettings> createState() => _NewsletterSettingsState();
}

class _NewsletterSettingsState extends State<NewsletterSettings> {
  bool _subscribed = false;
  bool _weekly = true;
  bool _monthly = false;
  bool _products = true;
  bool _offers = false;

  List<bool> get _options => [_weekly, _monthly, _products, _offers];

  bool get _allSelected => _options.every((o) => o);
  bool get _noneSelected => _options.every((o) => !o);

  void _selectAll() {
    setState(() {
      _weekly = true;
      _monthly = true;
      _products = true;
      _offers = true;
    });
  }

  void _selectNone() {
    setState(() {
      _weekly = false;
      _monthly = false;
      _products = false;
      _offers = false;
    });
  }

  @override
```

```
Widget build(BuildContext context) {
  return Card(
    child: Padding(
      padding: const EdgeInsets.all(16),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          const Text(
            'Newsletter Einstellungen',
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
          ),
          const SizedBox(height: 8),

          // Haupt-Checkbox
          CheckboxListTile(
            title: const Text('Newsletter abonnieren'),
            value: _subscribed,
            onChanged: (v) => setState(() => _subscribed = v!),
            controlAffinity: ListTileControlAffinity.leading,
          ),

          // Unteroptionen
          if (_subscribed) ...[
            const Divider(),
            CheckboxListTile(
              title: const Text('Wöchentliche Updates'),
              value: _weekly,
              onChanged: (v) => setState(() => _weekly = v!),
              controlAffinity: ListTileControlAffinity.leading,
            ),
            CheckboxListTile(
              title: const Text('Monatliche Zusammenfassung'),
              value: _monthly,
              onChanged: (v) => setState(() => _monthly = v!),
              controlAffinity: ListTileControlAffinity.leading,
            ),
            CheckboxListTile(
              title: const Text('Produktneuheiten'),
              value: _products,
              onChanged: (v) => setState(() => _products = v!),
              controlAffinity: ListTileControlAffinity.leading,
            ),
            CheckboxListTile(
              title: const Text('Sonderangebote'),
              value: _offers,
              onChanged: (v) => setState(() => _offers = v!),
              controlAffinity: ListTileControlAffinity.leading,
            ),
            const SizedBox(height: 8),
            Row(
              children: [
```

```
                  TextButton(
                    onPressed: _allSelected ? null : _selectAll,
                    child: const Text('Alle auswählen'),
                  ),
                  TextButton(
                    onPressed: _noneSelected ? null : _selectNone,
                    child: const Text('Keine'),
                  ),
                ],
              ),
            ],
          ],
        ),
      ),
    );
  }
}
```

3.11.2.3 Aufgabe 5: Filter Chips

```
class TagFilter extends StatefulWidget {
  const TagFilter({super.key});

  @override
  State<TagFilter> createState() => _TagFilterState();
}

class _TagFilterState extends State<TagFilter> {
  final _allTags = ['Web', 'App', 'API', 'DB', 'UI/UX', 'DevOps', 'Testing'];
  final Set<String> _selectedTags = {};

  void _applyFilter() {
    if (_selectedTags.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Bitte mindestens einen Tag auswählen')),
      );
      return;
    }

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Filter angewendet: ${_selectedTags.join(', ')}')),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
```

```
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const Text(
              'Kategorien filtern',
              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ),
            const SizedBox(height: 16),
            Wrap(
              spacing: 8,
              runSpacing: 8,
              children: _allTags.map((tag) {
                return FilterChip(
                  label: Text(tag),
                  selected: _selectedTags.contains(tag),
                  onSelected: (selected) {
                    setState(() {
                      if (selected) {
                        _selectedTags.add(tag);
                      } else {
                        _selectedTags.remove(tag);
                      }
                    });
                  },
                );
              }).toList(),
            ),
            const SizedBox(height: 16),
            Text('Ausgewählt: ${_selectedTags.isEmpty ? "Keine" :
↪  _selectedTags.join(", ")}'),
            const SizedBox(height: 16),
            ElevatedButton(
              onPressed: _applyFilter,
              child: const Text('Filter anwenden'),
            ),
          ],
        ),
      ),
    );
  }
}
```

3.11.3 Ressourcen

3.11.3.1 Offizielle Dokumentation

- DropdownButton
- DropdownButtonFormField
- Checkbox
- Switch
- Radio
- Chips

- SegmentedButton

3.11.3.2 Material Design Guidelines

- Selection Controls
- Switches
- Radio Buttons
- Chips
- Menus (Dropdowns)

3.11.3.3 Tutorials

- Dropdown Tutorial
- Custom Form Fields
- Selection Widgets Guide

3.11.3.4 Videos

- DropdownButton Widget
- Checkbox & Switch
- Chips in Flutter

3.11.3.5 Packages

- dropdown_search - Erweiterte Dropdown Funktionen
- multi_select_flutter - Multi-Select Dialoge
- flutter_chip_tags - Tag-Input Feld
- group_button - Gruppierte Buttons

3.11.3.6 Zum Vertiefen

- FormField Class
- Creating Custom Input Controls
- Accessibility for Selection Controls

3.12 Einheit 3.12: DatePicker & Dialoge

3.12.0.1 Lernziele

Nach dieser Einheit kannst du: - `showDatePicker` und `showTimePicker` verwenden - Verschiedene Dialog-Typen einsetzen - `BottomSheet` für mobile UX nutzen - Debouncing für Input-Felder implementieren

3.12.0.2 1. DatePicker

Einfacher DatePicker

```
class DatePickerDemo extends StatefulWidget {
  @override
  State<DatePickerDemo> createState() => _DatePickerDemoState();
}

class _DatePickerDemoState extends State<DatePickerDemo> {
  DateTime? _selectedDate;
```

```
  Future<void> _pickDate() async {
    final date = await showDatePicker(
      context: context,
      initialDate: _selectedDate ?? DateTime.now(),
      firstDate: DateTime(2000),
      lastDate: DateTime(2100),
    );

    if (date != null) {
      setState(() => _selectedDate = date);
    }
  }

  @override
  Widget build(BuildContext context) {
    return ListTile(
      title: Text(_selectedDate == null
          ? 'Datum auswählen'
          : '${_selectedDate!.day}.${_selectedDate!.month}.${_selectedDate!.year}'),
      trailing: const Icon(Icons.calendar_today),
      onTap: _pickDate,
    );
  }
}
```

DatePicker Optionen

```
final date = await showDatePicker(
  context: context,
  initialDate: DateTime.now(),
  firstDate: DateTime(2000),
  lastDate: DateTime(2100),

  // Lokalisierung
  locale: const Locale('de', 'DE'),

  // Initiale Ansicht
  initialDatePickerMode: DatePickerMode.year,  // oder .day

  // Hilfetext
  helpText: 'Geburtsdatum auswählen',
  cancelText: 'Abbrechen',
  confirmText: 'OK',

  // Bestimmte Tage deaktivieren
  selectableDayPredicate: (date) {
    // Wochenenden deaktivieren
    return date.weekday != DateTime.saturday &&
           date.weekday != DateTime.sunday;
```

```
  },

  // Stil anpassen
  builder: (context, child) {
    return Theme(
      data: Theme.of(context).copyWith(
        colorScheme: const ColorScheme.light(
          primary: Colors.blue,
          onPrimary: Colors.white,
          surface: Colors.white,
          onSurface: Colors.black,
        ),
      ),
      child: child!,
    );
  },
);
```

Date Range Picker

```
Future<void> _pickDateRange() async {
  final range = await showDateRangePicker(
    context: context,
    firstDate: DateTime(2000),
    lastDate: DateTime(2100),
    initialDateRange: DateTimeRange(
      start: DateTime.now(),
      end: DateTime.now().add(const Duration(days: 7)),
    ),
    helpText: 'Zeitraum auswählen',
    saveText: 'Speichern',
  );

  if (range != null) {
    print('Von: ${range.start} bis ${range.end}');
  }
}
```

3.12.0.3 2. TimePicker

Einfacher TimePicker

```
class TimePickerDemo extends StatefulWidget {
  @override
  State<TimePickerDemo> createState() => _TimePickerDemoState();
}

class _TimePickerDemoState extends State<TimePickerDemo> {
  TimeOfDay? _selectedTime;

  Future<void> _pickTime() async {
    final time = await showTimePicker(
```

```
      context: context,
      initialTime: _selectedTime ?? TimeOfDay.now(),
    );

    if (time != null) {
      setState(() => _selectedTime = time);
    }
  }

  String get _formattedTime {
    if (_selectedTime == null) return 'Zeit auswählen';
    final hour = _selectedTime!.hour.toString().padLeft(2, '0');
    final minute = _selectedTime!.minute.toString().padLeft(2, '0');
    return '$hour:$minute';
  }

  @override
  Widget build(BuildContext context) {
    return ListTile(
      title: Text(_formattedTime),
      trailing: const Icon(Icons.access_time),
      onTap: _pickTime,
    );
  }
}
```

TimePicker Optionen

```
final time = await showTimePicker(
  context: context,
  initialTime: TimeOfDay.now(),

  // 24h Format erzwingen
  builder: (context, child) {
    return MediaQuery(
      data: MediaQuery.of(context).copyWith(
        alwaysUse24HourFormat: true,
      ),
      child: child!,
    );
  },

  // Hilfetext
  helpText: 'Uhrzeit wählen',
  cancelText: 'Abbrechen',
  confirmText: 'OK',

  // Einstiegsmodus
  initialEntryMode: TimePickerEntryMode.input,  // oder .dial
);
```

3.12.0.4 3. AlertDialog

Einfacher Alert

```
void _showAlert() {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text('Hinweis'),
      content: const Text('Das ist eine wichtige Nachricht.'),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('OK'),
        ),
      ],
    ),
  );
}
```

Bestätigungsdialog

```
Future<bool> _showConfirmDialog() async {
  final result = await showDialog<bool>(
    context: context,
    barrierDismissible: false,  // Muss mit Button geschlossen werden
    builder: (context) => AlertDialog(
      title: const Text('Löschen bestätigen'),
      content: const Text('Möchtest du dieses Element wirklich löschen?'),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context, false),
          child: const Text('Abbrechen'),
        ),
        TextButton(
          onPressed: () => Navigator.pop(context, true),
          style: TextButton.styleFrom(foregroundColor: Colors.red),
          child: const Text('Löschen'),
        ),
      ],
    ),
  );

  return result ?? false;
}

// Verwendung
void _deleteItem() async {
  final confirmed = await _showConfirmDialog();
  if (confirmed) {
    // Löschen durchführen
  }
}
```

Dialog mit Eingabefeld

```
Future<String?> _showInputDialog() async {
  final controller = TextEditingController();

  final result = await showDialog<String>(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text('Name eingeben'),
      content: TextField(
        controller: controller,
        autofocus: true,
        decoration: const InputDecoration(
          hintText: 'Dein Name',
        ),
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('Abbrechen'),
        ),
        ElevatedButton(
          onPressed: () => Navigator.pop(context, controller.text),
          child: const Text('OK'),
        ),
      ],
    ),
  );

  return result;
}
```

3.12.0.5 4. SimpleDialog

Auswahlliste

```
Future<String?> _showSelectionDialog() async {
  return await showDialog<String>(
    context: context,
    builder: (context) => SimpleDialog(
      title: const Text('Kategorie wählen'),
      children: [
        SimpleDialogOption(
          onPressed: () => Navigator.pop(context, 'arbeit'),
          child: const ListTile(
            leading: Icon(Icons.work),
            title: Text('Arbeit'),
          ),
        ),
        SimpleDialogOption(
          onPressed: () => Navigator.pop(context, 'privat'),
```

```
          child: const ListTile(
            leading: Icon(Icons.home),
            title: Text('Privat'),
          ),
        ),
        SimpleDialogOption(
          onPressed: () => Navigator.pop(context, 'einkauf'),
          child: const ListTile(
            leading: Icon(Icons.shopping_cart),
            title: Text('Einkauf'),
          ),
        ),
      ],
    ),
  );
}
```

3.12.0.6 5. BottomSheet

Modal Bottom Sheet

```
void _showBottomSheet() {
  showModalBottomSheet(
    context: context,
    builder: (context) => Container(
      padding: const EdgeInsets.all(16),
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          ListTile(
            leading: const Icon(Icons.share),
            title: const Text('Teilen'),
            onTap: () {
              Navigator.pop(context);
              // Teilen-Logik
            },
          ),
          ListTile(
            leading: const Icon(Icons.edit),
            title: const Text('Bearbeiten'),
            onTap: () {
              Navigator.pop(context);
              // Bearbeiten-Logik
            },
          ),
          ListTile(
            leading: const Icon(Icons.delete),
            title: const Text('Löschen'),
            onTap: () {
              Navigator.pop(context);
              // Löschen-Logik
```

```
            },
          ),
        ],
      ),
    ),
  );
}
```

Scrollbares Bottom Sheet

```
void _showScrollableBottomSheet() {
  showModalBottomSheet(
    context: context,
    isScrollControlled: true,  // Wichtig für DraggableScrollableSheet
    builder: (context) => DraggableScrollableSheet(
      initialChildSize: 0.5,
      minChildSize: 0.25,
      maxChildSize: 0.9,
      expand: false,
      builder: (context, scrollController) {
        return Container(
          decoration: const BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.vertical(top: Radius.circular(16)),
          ),
          child: ListView.builder(
            controller: scrollController,
            itemCount: 50,
            itemBuilder: (context, index) => ListTile(
              title: Text('Item $index'),
            ),
          ),
        );
      },
    ),
  );
}
```

Bottom Sheet mit Formular

```
void _showFormBottomSheet() {
  showModalBottomSheet(
    context: context,
    isScrollControlled: true,  // Keyboard-aware
    builder: (context) => Padding(
      // Keyboard-Abstand
      padding: EdgeInsets.only(
        bottom: MediaQuery.of(context).viewInsets.bottom,
      ),
      child: Container(
        padding: const EdgeInsets.all(16),
        child: Column(
```

```
          mainAxisSize: MainAxisSize.min,
          children: [
            const Text('Neue Notiz',
                style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),
            const SizedBox(height: 16),
            const TextField(
              decoration: InputDecoration(
                labelText: 'Titel',
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 16),
            const TextField(
              decoration: InputDecoration(
                labelText: 'Inhalt',
                border: OutlineInputBorder(),
              ),
              maxLines: 3,
            ),
            const SizedBox(height: 16),
            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: () => Navigator.pop(context),
                child: const Text('Speichern'),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
```

3.12.0.7 6. Snackbar

```
// Einfache Snackbar
ScaffoldMessenger.of(context).showSnackBar(
  const SnackBar(content: Text('Gespeichert!')),
);

// Mit Action
ScaffoldMessenger.of(context).showSnackBar(
  SnackBar(
    content: const Text('Element gelöscht'),
    action: SnackBarAction(
      label: 'Rückgängig',
      onPressed: () {
        // Rückgängig-Logik
      },
```

```
    ),
    duration: const Duration(seconds: 5),
  ),
);

// Mit Styling
ScaffoldMessenger.of(context).showSnackBar(
  SnackBar(
    content: const Text('Fehler aufgetreten'),
    backgroundColor: Colors.red,
    behavior: SnackBarBehavior.floating,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(8),
    ),
    margin: const EdgeInsets.all(16),
  ),
);
```

3.12.0.8 7. Debouncing

Für Suchfelder

```
class SearchField extends StatefulWidget {
  final void Function(String) onSearch;

  const SearchField({super.key, required this.onSearch});

  @override
  State<SearchField> createState() => _SearchFieldState();
}

class _SearchFieldState extends State<SearchField> {
  final _controller = TextEditingController();
  Timer? _debounce;

  @override
  void dispose() {
    _debounce?.cancel();
    _controller.dispose();
    super.dispose();
  }

  void _onChanged(String value) {
    _debounce?.cancel();
    _debounce = Timer(const Duration(milliseconds: 500), () {
      widget.onSearch(value);
    });
  }

  @override
  Widget build(BuildContext context) {
```

```
    return TextField(
      controller: _controller,
      decoration: InputDecoration(
        hintText: 'Suchen...',
        prefixIcon: const Icon(Icons.search),
        suffixIcon: _controller.text.isNotEmpty
            ? IconButton(
                icon: const Icon(Icons.clear),
                onPressed: () {
                  _controller.clear();
                  widget.onSearch('');
                },
              )
            : null,
      ),
      onChanged: _onChanged,
    );
  }
}
```

Debounce Utility

```
class Debouncer {
  final Duration delay;
  Timer? _timer;

  Debouncer({this.delay = const Duration(milliseconds: 500)});

  void run(VoidCallback action) {
    _timer?.cancel();
    _timer = Timer(delay, action);
  }

  void cancel() {
    _timer?.cancel();
  }
}

// Verwendung
final _debouncer = Debouncer(delay: const Duration(milliseconds: 300));

void _onTextChanged(String value) {
  _debouncer.run(() {
    // API-Call oder Filterung
    _performSearch(value);
  });
}

@override
void dispose() {
  _debouncer.cancel();
```

```
  super.dispose();
}
```

3.12.0.9 8. DatePicker als FormField

```
class DateFormField extends FormField<DateTime> {
  DateFormField({
    super.key,
    DateTime? initialValue,
    required BuildContext context,
    InputDecoration? decoration,
    String? Function(DateTime?)? validator,
    void Function(DateTime?)? onSaved,
  }) : super(
          initialValue: initialValue,
          validator: validator,
          onSaved: onSaved,
          builder: (state) {
            return InkWell(
              onTap: () async {
                final date = await showDatePicker(
                  context: context,
                  initialDate: state.value ?? DateTime.now(),
                  firstDate: DateTime(1900),
                  lastDate: DateTime(2100),
                );
                if (date != null) {
                  state.didChange(date);
                }
              },
              child: InputDecorator(
                decoration: (decoration ?? const InputDecoration()).copyWith(
                  errorText: state.errorText,
                ),
                child: Text(
                  state.value == null
                      ? 'Datum auswählen'
                      :
 '${state.value!.day}.${state.value!.month}.${state.value!.year}',
                ),
              ),
            );
          },
        );
}
```

3.12.0.10 Zusammenfassung

Widget/Funktion	Verwendung
`showDatePicker`	Datum auswählen
`showDateRangePicker`	Zeitraum auswählen
`showTimePicker`	Uhrzeit auswählen
`AlertDialog`	Hinweise, Bestätigungen
`SimpleDialog`	Einfache Auswahllisten
`showModalBottomSheet`	Mobile-freundliche Optionen
`SnackBar`	Kurzfristige Benachrichtigungen
`Debouncer`	Verzögerte Aktionen (Suche)

Best Practices: - Bottom Sheet für mobile UX bevorzugen - Debounce für Suchfelder (300-500ms) - Dialoge kurz halten - `barrierDismissible: false` für wichtige Bestätigungen

3.12.1 Übung

3.12.1.1 Ziel

Eine Termin-Planungs-App mit verschiedenen Dialogen erstellen.

3.12.1.2 Aufgabe 1: DatePicker Integration (20 min)

Erstelle ein Formular für einen Termin:

```
+---------------------------------+
| Neuer Termin                    |
+---------------------------------+
| Datum: [15.03.2024        ]  |
| Uhrzeit: [14:30          ]   |
| Dauer: [1 Stunde         ▼]    |
+---------------------------------+
```

Anforderungen: - Datum darf nicht in der Vergangenheit liegen - Wochenenden deaktivieren - Uhrzeit in 15-Minuten-Schritten (z.B. 14:00, 14:15, 14:30) - Dauer als Dropdown (30 Min, 1h, 1.5h, 2h)

3.12.1.3 Aufgabe 2: Termin-Übersicht mit Dialog (25 min)

Erstelle eine Liste von Terminen mit Aktions-Dialog:

```
+---------------------------------+
| Termine                         |
+---------------------------------+
|  Meeting mit Team             |
|    15.03.2024, 14:00            |
| ------------------------------- |
|  Arzttermin                   |
|    16.03.2024, 10:30            |
+---------------------------------+
```

Bei Tap auf einen Termin: - Bottom Sheet mit Optionen: - Details anzeigen - Bearbeiten - Verschieben - Löschen

Bei “Löschen”: - Bestätigungsdialog anzeigen

3.12.1.4 Aufgabe 3: Date Range Picker (15 min)

Erstelle einen Urlaubsplaner:

```
+---------------------------------+
| Urlaub planen                   |
+---------------------------------+
| Zeitraum auswählen:             |
| [01.04.2024 - 14.04.2024]      |
|                                 |
| Dauer: 14 Tage (10 Werktage)   |
+---------------------------------+
```

- Zeige ausgewählten Zeitraum
- Berechne Anzahl der Tage
- Berechne Werktage (ohne Wochenenden)

3.12.1.5 Aufgabe 4: Debounced Search (20 min)

Erstelle eine Suchfunktion mit Debouncing:

```
// Simulierte API
Future<List<String>> searchContacts(String query) async {
  await Future.delayed(const Duration(milliseconds: 300));

  final contacts = [
    'Max Mustermann', 'Maria Müller', 'Michael Meyer',
    'Anna Schmidt', 'Andreas Fischer', 'Petra Wagner',
  ];

  return contacts
      .where((c) => c.toLowerCase().contains(query.toLowerCase()))
      .toList();
}
```

Anforderungen: - Debounce von 500ms - Loading-Indikator während Suche - Ergebnisse live anzeigen - “Keine Ergebnisse” wenn leer

3.12.1.6 Aufgabe 5: Custom Dialog (20 min)

Erstelle einen wiederverwendbaren Bewertungs-Dialog:

```
+---------------------------------+
|         Bewertung               |
|                                 |
|     ☆ ☆ ☆ ★ ★                  |
|                                 |
| +-----------------------------+ |
| | Dein Kommentar...           | |
| |                             | |
| +-----------------------------+ |
|                                 |
|   [Abbrechen]    [Bewerten]    |
+---------------------------------+
```

- Sterne-Bewertung (1-5)
- Optionaler Kommentar
- Rückgabe: `(int rating, String? comment)`

```
final result = await showRatingDialog(context);
if (result != null) {
  print('Rating: ${result.$1}, Comment: ${result.$2}');
}
```

3.12.1.7 Aufgabe 6: Snackbar Actions (15 min)

Implementiere verschiedene Snackbar-Szenarien:

1. **Erfolg:** "Termin gespeichert" (grün, 2 Sekunden)
2. **Warnung:** "Offline-Modus aktiv" (orange, persistent, mit "Einstellungen" Action)
3. **Fehler:** "Speichern fehlgeschlagen" (rot, mit "Erneut versuchen" Action)
4. **Undo:** "Termin gelöscht" (mit "Rückgängig" Action, 5 Sekunden)

3.12.1.8 Aufgabe 7: Komplette Termin-App (30 min)

Kombiniere alles zu einer Termin-App:

```
+---------------------------------+
| Meine Termine       [□] [+]   |
+---------------------------------+
| Heute                           |
| +-- 09:00 Standup Meeting       |
| +-- 14:00 Code Review           |
|                                 |
| Morgen                          |
| +-- 10:30 Kundengespräch        |
|                                 |
| Diese Woche                     |
| +-- Fr 15:00 Team-Event        |
+---------------------------------+
```

Features: - Suche mit Debouncing - FAB zum Erstellen (öffnet Bottom Sheet) - Tap auf Termin -> Details Dialog - Long Press -> Aktions-Bottom Sheet - Löschen mit Undo-Snackbar - Datum/Zeit Picker beim Erstellen - Bewertung nach abgeschlossenem Termin

3.12.1.9 Abgabe-Checkliste

- ☐ DatePicker mit Einschränkungen
- ☐ TimePicker integriert
- ☐ Bottom Sheet für Aktionen
- ☐ Bestätigungsdialog für Löschen
- ☐ Date Range Picker funktioniert
- ☐ Debounced Search implementiert
- ☐ Custom Rating Dialog
- ☐ Verschiedene Snackbar-Typen
- ☐ Komplette Termin-App

3.12.2 Lösung

3.12.2.1 Aufgabe 1: DatePicker Integration

```
class AppointmentForm extends StatefulWidget {
  const AppointmentForm({super.key});

  @override
  State<AppointmentForm> createState() => _AppointmentFormState();
}

class _AppointmentFormState extends State<AppointmentForm> {
  DateTime? _selectedDate;
  TimeOfDay? _selectedTime;
  String _duration = '1 Stunde';

  final _durations = ['30 Minuten', '1 Stunde', '1.5 Stunden', '2 Stunden'];

  Future<void> _pickDate() async {
    final date = await showDatePicker(
      context: context,
      initialDate: _selectedDate ?? DateTime.now().add(const Duration(days: 1)),
      firstDate: DateTime.now(),
      lastDate: DateTime.now().add(const Duration(days: 365)),
      selectableDayPredicate: (date) {
        // Wochenenden deaktivieren
        return date.weekday != DateTime.saturday &&
               date.weekday != DateTime.sunday;
      },
    );

    if (date != null) {
      setState(() => _selectedDate = date);
    }
  }

  Future<void> _pickTime() async {
    final time = await showTimePicker(
      context: context,
      initialTime: _selectedTime ?? const TimeOfDay(hour: 9, minute: 0),
      builder: (context, child) {
        return MediaQuery(
          data: MediaQuery.of(context).copyWith(alwaysUse24HourFormat: true),
          child: child!,
        );
      },
    );

    if (time != null) {
      // Auf 15-Minuten-Schritte runden
      final roundedMinute = (time.minute / 15).round() * 15;
      setState(() {
```

```
        _selectedTime = TimeOfDay(
          hour: time.hour,
          minute: roundedMinute % 60,
        );
      });
    }
  }

  String get _formattedDate {
    if (_selectedDate == null) return 'Datum auswählen';
    return
  '${_selectedDate!.day}.${_selectedDate!.month}.${_selectedDate!.year}';
  }

  String get _formattedTime {
    if (_selectedTime == null) return 'Zeit auswählen';
    return '${_selectedTime!.hour.toString().padLeft(2, '0')}:'
           '${_selectedTime!.minute.toString().padLeft(2, '0')}';
  }

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const Text('Neuer Termin',
                style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),
            const SizedBox(height: 16),

            // Datum
            ListTile(
              title: const Text('Datum'),
              trailing: Text(_formattedDate),
              leading: const Icon(Icons.calendar_today),
              onTap: _pickDate,
            ),

            // Uhrzeit
            ListTile(
              title: const Text('Uhrzeit'),
              trailing: Text(_formattedTime),
              leading: const Icon(Icons.access_time),
              onTap: _pickTime,
            ),

            // Dauer
            ListTile(
              title: const Text('Dauer'),
```

```
              leading: const Icon(Icons.timelapse),
              trailing: DropdownButton<String>(
                value: _duration,
                underline: const SizedBox(),
                items: _durations.map((d) {
                  return DropdownMenuItem(value: d, child: Text(d));
                }).toList(),
                onChanged: (v) => setState(() => _duration = v!),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

3.12.2.2 Aufgabe 4: Debounced Search

```
class Debouncer {
  final Duration delay;
  Timer? _timer;

  Debouncer({this.delay = const Duration(milliseconds: 500)});

  void run(VoidCallback action) {
    _timer?.cancel();
    _timer = Timer(delay, action);
  }

  void cancel() {
    _timer?.cancel();
  }
}

class ContactSearch extends StatefulWidget {
  const ContactSearch({super.key});

  @override
  State<ContactSearch> createState() => _ContactSearchState();
}

class _ContactSearchState extends State<ContactSearch> {
  final _controller = TextEditingController();
  final _debouncer = Debouncer();

  List<String> _results = [];
  bool _isLoading = false;
  String _query = '';
```

```
Future<List<String>> _searchContacts(String query) async {
  await Future.delayed(const Duration(milliseconds: 300));

  final contacts = [
    'Max Mustermann', 'Maria Müller', 'Michael Meyer',
    'Anna Schmidt', 'Andreas Fischer', 'Petra Wagner',
  ];

  if (query.isEmpty) return [];

  return contacts
      .where((c) => c.toLowerCase().contains(query.toLowerCase()))
      .toList();
}

void _onSearchChanged(String value) {
  _query = value;

  if (value.isEmpty) {
    setState(() {
      _results = [];
      _isLoading = false;
    });
    return;
  }

  setState(() => _isLoading = true);

  _debouncer.run(() async {
    final results = await _searchContacts(value);
    if (_query == value) {  // Nur wenn noch aktuell
      setState(() {
        _results = results;
        _isLoading = false;
      });
    }
  });
}

@override
void dispose() {
  _debouncer.cancel();
  _controller.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Column(
    children: [
      Padding(
```

```
          padding: const EdgeInsets.all(16),
          child: TextField(
            controller: _controller,
            decoration: InputDecoration(
              hintText: 'Kontakt suchen...',
              prefixIcon: const Icon(Icons.search),
              suffixIcon: _isLoading
                  ? const Padding(
                      padding: EdgeInsets.all(12),
                      child: SizedBox(
                        width: 20,
                        height: 20,
                        child: CircularProgressIndicator(strokeWidth: 2),
                      ),
                    )
                  : _controller.text.isNotEmpty
                      ? IconButton(
                          icon: const Icon(Icons.clear),
                          onPressed: () {
                            _controller.clear();
                            _onSearchChanged('');
                          },
                        )
                      : null,
              border: const OutlineInputBorder(),
            ),
            onChanged: _onSearchChanged,
          ),
        ),
        Expanded(
          child: _query.isEmpty
              ? const Center(child: Text('Suchbegriff eingeben'))
              : _results.isEmpty && !_isLoading
                  ? const Center(child: Text('Keine Ergebnisse'))
                  : ListView.builder(
                      itemCount: _results.length,
                      itemBuilder: (context, index) {
                        return ListTile(
                          leading: const CircleAvatar(
                            child: Icon(Icons.person),
                          ),
                          title: Text(_results[index]),
                        );
                      },
                    ),
        ),
      ],
    );
  }
}
```

3.12.2.3 Aufgabe 5: Custom Rating Dialog

```
Future<(int, String?)?> showRatingDialog(BuildContext context) async {
  return await showDialog<(int, String?)>(
    context: context,
    builder: (context) => const _RatingDialog(),
  );
}

class _RatingDialog extends StatefulWidget {
  const _RatingDialog();

  @override
  State<_RatingDialog> createState() => _RatingDialogState();
}

class _RatingDialogState extends State<_RatingDialog> {
  int _rating = 0;
  final _commentController = TextEditingController();

  @override
  void dispose() {
    _commentController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: const Text('Bewertung', textAlign: TextAlign.center),
      content: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          // Sterne
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: List.generate(5, (index) {
              return IconButton(
                icon: Icon(
                  index < _rating ? Icons.star : Icons.star_border,
                  color: Colors.amber,
                  size: 36,
                ),
                onPressed: () {
                  setState(() => _rating = index + 1);
                },
              );
            }),
          ),
          const SizedBox(height: 16),
```

```
          // Kommentar
          TextField(
            controller: _commentController,
            decoration: const InputDecoration(
              hintText: 'Dein Kommentar (optional)...',
              border: OutlineInputBorder(),
            ),
            maxLines: 3,
          ),
        ],
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('Abbrechen'),
        ),
        ElevatedButton(
          onPressed: _rating > 0
              ? () {
                  final comment = _commentController.text.trim();
                  Navigator.pop(
                    context,
                    (_rating, comment.isEmpty ? null : comment),
                  );
                }
              : null,
          child: const Text('Bewerten'),
        ),
      ],
    );
  }
}

// Verwendung
void _showRating() async {
  final result = await showRatingDialog(context);
  if (result != null) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Danke für deine ${result.$1}-Sterne Bewertung!'),
      ),
    );
  }
}
```

3.12.2.4 Aufgabe 6: Snackbar Actions

```
class SnackbarExamples {
  static void showSuccess(BuildContext context) {
    ScaffoldMessenger.of(context).showSnackBar(
```

```
    SnackBar(
      content: const Text('Termin gespeichert'),
      backgroundColor: Colors.green,
      duration: const Duration(seconds: 2),
      behavior: SnackBarBehavior.floating,
    ),
  );
}

static void showWarning(BuildContext context) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: const Text('Offline-Modus aktiv'),
      backgroundColor: Colors.orange,
      duration: const Duration(days: 1),  // Persistent
      action: SnackBarAction(
        label: 'Einstellungen',
        textColor: Colors.white,
        onPressed: () {
          // Öffne Einstellungen
        },
      ),
    ),
  );
}

static void showError(BuildContext context, VoidCallback onRetry) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: const Text('Speichern fehlgeschlagen'),
      backgroundColor: Colors.red,
      action: SnackBarAction(
        label: 'Erneut versuchen',
        textColor: Colors.white,
        onPressed: onRetry,
      ),
    ),
  );
}

static void showUndo(
  BuildContext context,
  String message,
  VoidCallback onUndo,
) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(message),
      duration: const Duration(seconds: 5),
      action: SnackBarAction(
        label: 'Rückgängig',
```

```
          onPressed: onUndo,
        ),
      ),
    );
  }
}

// Verwendung
void _deleteAppointment(Appointment apt) {
  // Temporär entfernen
  setState(() => _appointments.remove(apt));

  SnackbarExamples.showUndo(
    context,
    'Termin gelöscht',
    () {
      // Rückgängig machen
      setState(() => _appointments.add(apt));
    },
  );
}
```

3.12.2.5 Aufgabe 2: Bottom Sheet für Aktionen

```
void _showAppointmentActions(Appointment appointment) {
  showModalBottomSheet(
    context: context,
    builder: (context) => SafeArea(
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          ListTile(
            leading: const Icon(Icons.info),
            title: const Text('Details anzeigen'),
            onTap: () {
              Navigator.pop(context);
              _showDetails(appointment);
            },
          ),
          ListTile(
            leading: const Icon(Icons.edit),
            title: const Text('Bearbeiten'),
            onTap: () {
              Navigator.pop(context);
              _editAppointment(appointment);
            },
          ),
          ListTile(
            leading: const Icon(Icons.schedule),
            title: const Text('Verschieben'),
```

```
            onTap: () {
              Navigator.pop(context);
              _rescheduleAppointment(appointment);
            },
          ),
          ListTile(
            leading: const Icon(Icons.delete, color: Colors.red),
            title: const Text('Löschen', style: TextStyle(color: Colors.red)),
            onTap: () {
              Navigator.pop(context);
              _confirmDelete(appointment);
            },
          ),
        ],
      ),
    ),
  );
}

Future<void> _confirmDelete(Appointment appointment) async {
  final confirmed = await showDialog<bool>(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text('Termin löschen?'),
      content: Text('Möchtest du "${appointment.title}" wirklich löschen?'),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context, false),
          child: const Text('Abbrechen'),
        ),
        TextButton(
          onPressed: () => Navigator.pop(context, true),
          style: TextButton.styleFrom(foregroundColor: Colors.red),
          child: const Text('Löschen'),
        ),
      ],
    ),
  );

  if (confirmed == true) {
    _deleteAppointment(appointment);
  }
}
```

3.12.3 Ressourcen

3.12.3.1 Offizielle Dokumentation

- showDatePicker
- showTimePicker
- AlertDialog

- SimpleDialog
- BottomSheet
- SnackBar

3.12.3.2 Flutter Cookbook

- Display a snackbar
- Work with tabs

3.12.3.3 Material Design Guidelines

- Date pickers
- Time pickers
- Dialogs
- Bottom sheets
- Snackbars

3.12.3.4 Packages

- flutter_datetime_picker
- date_time_picker
- intl - Datum/Zeit Formatierung
- another_flushbar - Erweiterte Snackbars
- modal_bottom_sheet - Bessere Bottom Sheets

3.12.3.5 Tutorials

- DatePicker Complete Guide
- Dialogs in Flutter
- Bottom Sheet Guide
- Debounce and Throttle

3.12.3.6 Videos

- DatePicker Widget
- AlertDialog vs SimpleDialog
- Bottom Sheets

3.12.3.7 Zum Vertiefen

- Cupertino Pickers (iOS Style)
- Custom Dialog Animations
- Accessibility for Dialogs

Chapter 4

Block 4: Profi-Themen & Abschlussprojekt

Animationen, Testing, Packages und App-Veröffentlichung.

4.1 Einheit 4.1: Implizite Animationen

4.1.0.1 Lernziele

Nach dieser Einheit kannst du: - Implizite Animationen verstehen und einsetzen - `AnimatedContainer` und andere Animated-Widgets nutzen - `AnimatedSwitcher` für Widget-Übergänge verwenden - `TweenAnimationBuilder` für Custom-Animationen einsetzen

4.1.0.2 1. Was sind implizite Animationen?

Implizit vs. Explizit

Typ	Beschreibung	Kontrolle
Implizit	Animation startet automatisch bei Property-Änderung	Wenig (duration, curve)
Explizit	Volle Kontrolle über Start, Stop, Wiederholung	Voll (AnimationController)

Faustregel: Beginne mit impliziten Animationen. Wechsle zu expliziten nur bei komplexen Anforderungen.

4.1.0.3 2. AnimatedContainer

Basis-Beispiel

```
class AnimatedContainerDemo extends StatefulWidget {
  const AnimatedContainerDemo({super.key});

  @override
  State<AnimatedContainerDemo> createState() => _AnimatedContainerDemoState();
}

class _AnimatedContainerDemoState extends State<AnimatedContainerDemo> {
  bool _expanded = false;

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
```

```
      onTap: () => setState(() => _expanded = !_expanded),
      child: AnimatedContainer(
        duration: const Duration(milliseconds: 300),
        curve: Curves.easeInOut,
        width: _expanded ? 200 : 100,
        height: _expanded ? 200 : 100,
        decoration: BoxDecoration(
          color: _expanded ? Colors.blue : Colors.red,
          borderRadius: BorderRadius.circular(_expanded ? 100 : 8),
        ),
        child: const Center(
          child: Text('Tap me', style: TextStyle(color: Colors.white)),
        ),
      ),
    );
  }
}
```

Animierbare Properties

`AnimatedContainer` animiert automatisch: - `width`, `height` - `color` (über decoration) - `padding`, `margin` - `alignment` - `transform` - `borderRadius` (über decoration)

```
AnimatedContainer(
  duration: const Duration(milliseconds: 500),
  curve: Curves.elasticOut,
  width: _width,
  height: _height,
  padding: EdgeInsets.all(_padding),
  margin: EdgeInsets.all(_margin),
  alignment: _alignment,
  transform: Matrix4.rotationZ(_rotation),
  decoration: BoxDecoration(
    color: _color,
    borderRadius: BorderRadius.circular(_radius),
    boxShadow: [
      BoxShadow(
        color: Colors.black26,
        blurRadius: _shadowBlur,
        offset: Offset(0, _shadowOffset),
      ),
    ],
  ),
  child: child,
)
```

4.1.0.4 3. Weitere Animated-Widgets

AnimatedOpacity

```
AnimatedOpacity(
  duration: const Duration(milliseconds: 300),
```

```
  opacity: _visible ? 1.0 : 0.0,
  child: const Text('Fade in/out'),
)
```

AnimatedPadding

```
AnimatedPadding(
  duration: const Duration(milliseconds: 200),
  padding: EdgeInsets.all(_selected ? 32 : 8),
  child: const Card(child: Text('Padding animiert')),
)
```

AnimatedAlign

```
AnimatedAlign(
  duration: const Duration(milliseconds: 400),
  alignment: _alignRight ? Alignment.centerRight : Alignment.centerLeft,
  child: Container(width: 50, height: 50, color: Colors.blue),
)
```

AnimatedPositioned (in Stack)

```
Stack(
  children: [
    AnimatedPositioned(
      duration: const Duration(milliseconds: 300),
      left: _moved ? 200 : 0,
      top: _moved ? 100 : 0,
      child: Container(width: 50, height: 50, color: Colors.red),
    ),
  ],
)
```

AnimatedDefaultTextStyle

```
AnimatedDefaultTextStyle(
  duration: const Duration(milliseconds: 300),
  style: TextStyle(
    fontSize: _large ? 32 : 16,
    fontWeight: _large ? FontWeight.bold : FontWeight.normal,
    color: _large ? Colors.blue : Colors.black,
  ),
  child: const Text('Animierter Text'),
)
```

AnimatedCrossFade

```
AnimatedCrossFade(
  duration: const Duration(milliseconds: 300),
  crossFadeState: _showFirst
      ? CrossFadeState.showFirst
      : CrossFadeState.showSecond,
  firstChild: const Icon(Icons.favorite, size: 100, color: Colors.red),
```

```
  secondChild: const Icon(Icons.star, size: 100, color: Colors.amber),
)
```

AnimatedPhysicalModel

```
AnimatedPhysicalModel(
  duration: const Duration(milliseconds: 300),
  shape: BoxShape.rectangle,
  elevation: _elevated ? 16 : 0,
  color: _elevated ? Colors.white : Colors.grey[200]!,
  shadowColor: Colors.black,
  borderRadius: BorderRadius.circular(8),
  child: const Padding(
    padding: EdgeInsets.all(16),
    child: Text('Elevation animiert'),
  ),
)
```

4.1.0.5 4. AnimatedSwitcher

Widget-Wechsel animieren

```
class CounterWithAnimation extends StatefulWidget {
  const CounterWithAnimation({super.key});

  @override
  State<CounterWithAnimation> createState() => _CounterWithAnimationState();
}

class _CounterWithAnimationState extends State<CounterWithAnimation> {
  int _count = 0;

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        AnimatedSwitcher(
          duration: const Duration(milliseconds: 300),
          transitionBuilder: (child, animation) {
            return ScaleTransition(scale: animation, child: child);
          },
          child: Text(
            '$_count',
            // Key ist wichtig! Sonst erkennt Flutter den Wechsel nicht
            key: ValueKey<int>(_count),
            style: const TextStyle(fontSize: 48),
          ),
        ),
        const SizedBox(height: 16),
        ElevatedButton(
          onPressed: () => setState(() => _count++),
```

```
          child: const Text('Increment'),
        ),
      ],
    );
  }
}
```

Verschiedene Transitions

```
// Fade (Standard)
transitionBuilder: (child, animation) {
  return FadeTransition(opacity: animation, child: child);
},

// Scale
transitionBuilder: (child, animation) {
  return ScaleTransition(scale: animation, child: child);
},

// Slide von unten
transitionBuilder: (child, animation) {
  return SlideTransition(
    position: Tween<Offset>(
      begin: const Offset(0, 1),
      end: Offset.zero,
    ).animate(animation),
    child: child,
  );
},

// Rotation
transitionBuilder: (child, animation) {
  return RotationTransition(turns: animation, child: child);
},

// Kombiniert
transitionBuilder: (child, animation) {
  return FadeTransition(
    opacity: animation,
    child: ScaleTransition(scale: animation, child: child),
  );
},
```

Layout-Builder für komplexe Wechsel

```
AnimatedSwitcher(
  duration: const Duration(milliseconds: 500),
  switchInCurve: Curves.easeOut,
  switchOutCurve: Curves.easeIn,
  layoutBuilder: (currentChild, previousChildren) {
    return Stack(
      alignment: Alignment.center,
```

```
      children: [
        ...previousChildren,
        if (currentChild != null) currentChild,
      ],
    );
  },
  transitionBuilder: (child, animation) {
    return FadeTransition(opacity: animation, child: child);
  },
  child: _widgets[_currentIndex],
)
```

4.1.0.6 5. TweenAnimationBuilder

Custom-Animationen ohne Controller

```
TweenAnimationBuilder<double>(
  tween: Tween<double>(begin: 0, end: _progress),
  duration: const Duration(milliseconds: 500),
  builder: (context, value, child) {
    return Column(
      children: [
        LinearProgressIndicator(value: value),
        Text('${(value * 100).toInt()}%'),
      ],
    );
  },
)
```

Mit verschiedenen Typen

```
// Color Animation
TweenAnimationBuilder<Color?>(
  tween: ColorTween(begin: Colors.red, end: _targetColor),
  duration: const Duration(milliseconds: 300),
  builder: (context, color, child) {
    return Container(
      width: 100,
      height: 100,
      color: color,
    );
  },
)

// Size Animation
TweenAnimationBuilder<double>(
  tween: Tween<double>(begin: 50, end: _targetSize),
  duration: const Duration(milliseconds: 400),
  curve: Curves.elasticOut,
  builder: (context, size, child) {
    return Container(
      width: size,
```

```
      height: size,
      color: Colors.blue,
      child: child,
    );
  },
  child: const Icon(Icons.star, color: Colors.white),  // child wird nicht     ↪
↪  rebuildet
)

// Offset Animation
TweenAnimationBuilder<Offset>(
  tween: Tween<Offset>(
    begin: Offset.zero,
    end: _targetOffset,
  ),
  duration: const Duration(milliseconds: 300),
  builder: (context, offset, child) {
    return Transform.translate(
      offset: offset,
      child: child,
    );
  },
  child: const FlutterLogo(size: 100),
)
```

onEnd Callback

```
TweenAnimationBuilder<double>(
  tween: Tween<double>(begin: 0, end: 1),
  duration: const Duration(seconds: 2),
  onEnd: () {
    // Animation fertig
    print('Animation completed!');
    // Nächste Animation starten, Navigation, etc.
  },
  builder: (context, value, child) {
    return Opacity(
      opacity: value,
      child: child,
    );
  },
  child: const Text('Fade In Complete'),
)
```

4.1.0.7 6. Curves

Verfügbare Curves

```
// Linear
curve: Curves.linear,

// Ease (Standard)
```

```
curve: Curves.ease,
curve: Curves.easeIn,
curve: Curves.easeOut,
curve: Curves.easeInOut,

// Cubic
curve: Curves.fastOutSlowIn,
curve: Curves.slowMiddle,

// Bounce
curve: Curves.bounceIn,
curve: Curves.bounceOut,
curve: Curves.bounceInOut,

// Elastic
curve: Curves.elasticIn,
curve: Curves.elasticOut,
curve: Curves.elasticInOut,

// Back (überschießt)
curve: Curves.easeInBack,
curve: Curves.easeOutBack,
curve: Curves.easeInOutBack,
```

Curve-Visualisierung

```
class CurveDemo extends StatefulWidget {
  const CurveDemo({super.key});

  @override
  State<CurveDemo> createState() => _CurveDemoState();
}

class _CurveDemoState extends State<CurveDemo> {
  bool _animated = false;
  Curve _selectedCurve = Curves.easeInOut;

  final _curves = {
    'linear': Curves.linear,
    'easeInOut': Curves.easeInOut,
    'bounceOut': Curves.bounceOut,
    'elasticOut': Curves.elasticOut,
    'easeOutBack': Curves.easeOutBack,
  };

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        DropdownButton<String>(
          value: _curves.entries
```

```
              .firstWhere((e) => e.value == _selectedCurve)
              .key,
          items: _curves.keys.map((name) {
            return DropdownMenuItem(value: name, child: Text(name));
          }).toList(),
          onChanged: (name) {
            setState(() => _selectedCurve = _curves[name]!);
          },
        ),
        const SizedBox(height: 32),
        AnimatedAlign(
          duration: const Duration(milliseconds: 1000),
          curve: _selectedCurve,
          alignment: _animated
              ? Alignment.centerRight
              : Alignment.centerLeft,
          child: Container(
            width: 50,
            height: 50,
            decoration: const BoxDecoration(
              color: Colors.blue,
              shape: BoxShape.circle,
            ),
          ),
        ),
        const SizedBox(height: 32),
        ElevatedButton(
          onPressed: () => setState(() => _animated = !_animated),
          child: const Text('Animate'),
        ),
      ],
    );
  }
}
```

4.1.0.8 7. AnimatedList

Dynamische Listen mit Animation

```
class AnimatedListDemo extends StatefulWidget {
  const AnimatedListDemo({super.key});

  @override
  State<AnimatedListDemo> createState() => _AnimatedListDemoState();
}

class _AnimatedListDemoState extends State<AnimatedListDemo> {
  final _listKey = GlobalKey<AnimatedListState>();
  final _items = <String>['Item 1', 'Item 2', 'Item 3'];

  void _addItem() {
```

```
  final index = _items.length;
  _items.add('Item ${index + 1}');
  _listKey.currentState?.insertItem(
    index,
    duration: const Duration(milliseconds: 300),
  );
}

void _removeItem(int index) {
  final removedItem = _items.removeAt(index);
  _listKey.currentState?.removeItem(
    index,
    (context, animation) => _buildItem(removedItem, animation),
    duration: const Duration(milliseconds: 300),
  );
}

Widget _buildItem(String item, Animation<double> animation) {
  return SizeTransition(
    sizeFactor: animation,
    child: FadeTransition(
      opacity: animation,
      child: Card(
        child: ListTile(
          title: Text(item),
          trailing: IconButton(
            icon: const Icon(Icons.delete),
            onPressed: () {
              final index = _items.indexOf(item);
              if (index != -1) _removeItem(index);
            },
          ),
        ),
      ),
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('AnimatedList')),
    body: AnimatedList(
      key: _listKey,
      initialItemCount: _items.length,
      itemBuilder: (context, index, animation) {
        return _buildItem(_items[index], animation);
      },
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: _addItem,
```

```
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

4.1.0.9 8. Praktisches Beispiel: Animated Card

```
class AnimatedProductCard extends StatefulWidget {
  final String title;
  final String price;
  final String imageUrl;

  const AnimatedProductCard({
    super.key,
    required this.title,
    required this.price,
    required this.imageUrl,
  });

  @override
  State<AnimatedProductCard> createState() => _AnimatedProductCardState();
}

class _AnimatedProductCardState extends State<AnimatedProductCard> {
  bool _isHovered = false;
  bool _isFavorite = false;

  @override
  Widget build(BuildContext context) {
    return MouseRegion(
      onEnter: (_) => setState(() => _isHovered = true),
      onExit: (_) => setState(() => _isHovered = false),
      child: AnimatedContainer(
        duration: const Duration(milliseconds: 200),
        curve: Curves.easeOut,
        transform: Matrix4.identity()
          ..scale(_isHovered ? 1.05 : 1.0),
        child: AnimatedPhysicalModel(
          duration: const Duration(milliseconds: 200),
          shape: BoxShape.rectangle,
          elevation: _isHovered ? 12 : 4,
          color: Colors.white,
          shadowColor: Colors.black,
          borderRadius: BorderRadius.circular(12),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Image mit Overlay
              Stack(
```

```
                children: [
                  ClipRRect(
                    borderRadius: const BorderRadius.vertical(
                      top: Radius.circular(12),
                    ),
                    child: Image.network(
                      widget.imageUrl,
                      height: 150,
                      width: double.infinity,
                      fit: BoxFit.cover,
                    ),
                  ),
                  Positioned(
                    top: 8,
                    right: 8,
                    child: GestureDetector(
                      onTap: () => setState(() => _isFavorite = !_isFavorite),
                      child: AnimatedContainer(
                        duration: const Duration(milliseconds: 200),
                        padding: const EdgeInsets.all(8),
                        decoration: BoxDecoration(
                          color: _isFavorite
                              ? Colors.red
                              : Colors.white.withOpacity(0.8),
                          shape: BoxShape.circle,
                        ),
                        child: AnimatedSwitcher(
                          duration: const Duration(milliseconds: 200),
                          transitionBuilder: (child, animation) {
                            return ScaleTransition(
                              scale: animation,
                              child: child,
                            );
                          },
                          child: Icon(
                            _isFavorite
                                ? Icons.favorite
                                : Icons.favorite_border,
                            key: ValueKey(_isFavorite),
                            color: _isFavorite ? Colors.white : Colors.grey,
                            size: 20,
                          ),
                        ),
                      ),
                    ),
                  ),
                ],
              ),
              // Info
              Padding(
                padding: const EdgeInsets.all(12),
```

```
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      widget.title,
                      style: const TextStyle(
                        fontWeight: FontWeight.bold,
                        fontSize: 16,
                      ),
                    ),
                    const SizedBox(height: 4),
                    AnimatedDefaultTextStyle(
                      duration: const Duration(milliseconds: 200),
                      style: TextStyle(
                        fontSize: _isHovered ? 18 : 14,
                        fontWeight: _isHovered
                            ? FontWeight.bold
                            : FontWeight.normal,
                        color: Colors.green[700],
                      ),
                      child: Text(widget.price),
                    ),
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

4.1.0.10 Zusammenfassung

Widget	Animiert
`AnimatedContainer`	Size, Color, Padding, Margin, Decoration
`AnimatedOpacity`	Opacity (0.0 - 1.0)
`AnimatedPadding`	Padding
`AnimatedAlign`	Alignment
`AnimatedPositioned`	Position in Stack
`AnimatedDefaultTextStyle`	TextStyle
`AnimatedCrossFade`	Wechsel zwischen zwei Widgets
`AnimatedSwitcher`	Beliebiger Widget-Wechsel
`AnimatedList`	Listen mit Add/Remove
`TweenAnimationBuilder`	Custom Tween-Animationen

Best Practices: - Implizite Animationen für einfache Übergänge - `duration` von 200-400ms für UI-Feedback - `Curves.easeInOut` als Standard-Curve - `key` bei `AnimatedSwitcher` nicht

vergessen

4.1.1 Übung

4.1.1.1 Ziel

Verschiedene implizite Animationen implementieren und kombinieren.

4.1.1.2 Aufgabe 1: Animated Settings Toggle (20 min)

Erstelle einen animierten Settings-Schalter:

```
+-----------------------------------+
|  Benachrichtigungen              |
|  +----------+                     |
|  | ████▒▒▒▒ |  <- Slider animiert  |
|  +----------+                     |
|                                   |
|  Wenn aktiv:                      |
|  - Hintergrund wird grün          |
|  - Icon wird größer               |
|  - Text wird fett                 |
+-----------------------------------+
```

Anforderungen: - `AnimatedContainer` für Hintergrundfarbe - `AnimatedDefaultTextStyle` für Text - `AnimatedScale` oder `TweenAnimationBuilder` für Icon - Sanfte Übergänge (300ms)

4.1.1.3 Aufgabe 2: Expandable Card (25 min)

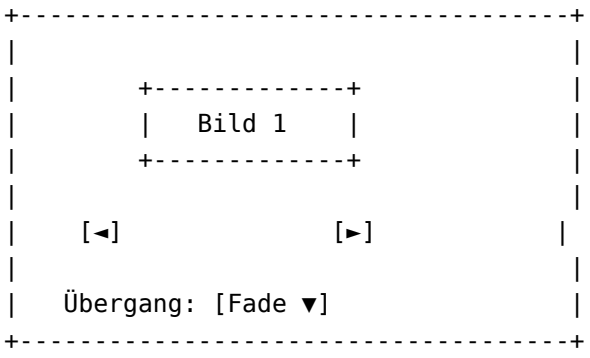
Erstelle eine erweiterbare Info-Karte:

```
Eingeklappt:
+-----------------------------------+
|  Bestellung #12345           ▼  |
| Status: Versendet                 |
+-----------------------------------+

Ausgeklappt:
+-----------------------------------+
|  Bestellung #12345           ▲  |
| Status: Versendet                 |
+-----------------------------------+
| Artikel: Flutter Buch             |
| Preis: 29,99€                     |
| Lieferadresse: Musterstr. 1       |
| Voraussichtlich: 15.03.2024       |
+-----------------------------------+
```

Anforderungen: - Tap auf Karte zum Auf-/Zuklappen - `AnimatedContainer` für Höhe - `AnimatedRotation` für den Pfeil - `AnimatedOpacity` für den Inhalt - Details erscheinen smooth

4.1.1.4 Aufgabe 3: AnimatedSwitcher Gallery (20 min)

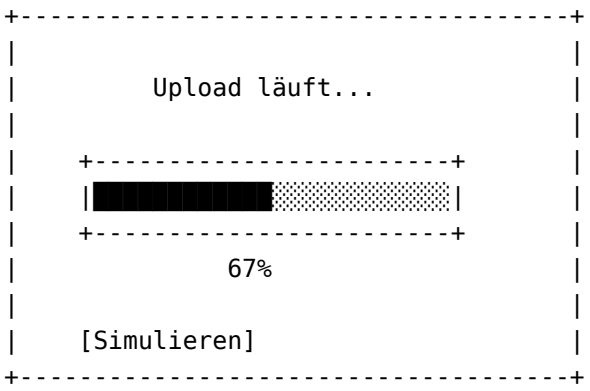
Erstelle eine Bildergalerie mit verschiedenen Übergängen:



Übergänge implementieren: - Fade (Standard) - Scale - Slide (links/rechts) - Rotation

4.1.1.5 Aufgabe 4: Progress Animation (20 min)

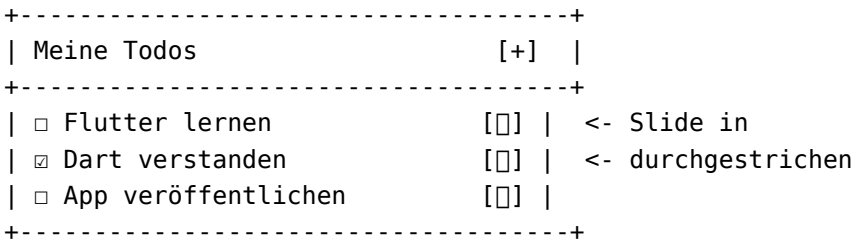
Erstelle einen animierten Fortschrittsanzeiger:



Anforderungen: - `TweenAnimationBuilder` für den Fortschritt - Animierte Prozentanzeige - Farbwechsel: rot -> gelb -> grün - Bei 100%: Erfolgsmeldung mit Animation

4.1.1.6 Aufgabe 5: AnimatedList Todo (25 min)

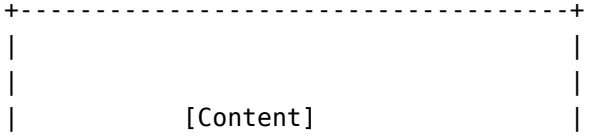
Erstelle eine Todo-Liste mit animierten Einträgen:



Anforderungen: - `AnimatedList` für Add/Remove - Neue Items sliden von rechts rein - Gelöschte Items faden und schrumpfen - Checkbox-Animation beim Abhaken - Durchstreich-Animation für erledigte Items

4.1.1.7 Aufgabe 6: Animated Navigation Bar (20 min)

Erstelle eine animierte Bottom Navigation:



```
|                                    |
|                                    |
+------------------------------------+
|   □      □      ♥      □       |
|  Home   Search  Favs   Profile      |
|   ●                                |
+------------------------------------+
```

Anforderungen: - Aktives Icon wird größer - Indikator (Punkt) bewegt sich animiert - Label erscheint nur bei aktivem Item - Sanfte Farbübergänge

4.1.1.8 Aufgabe 7: Komplette Animated Page (30 min)

Erstelle eine Landing Page mit Stagger-Animationen:

```
+------------------------------------+
|                                    |
|      +---------+                   |  <- Logo faded ein
|      |  LOGO   |                   |
|      +---------+                   |
|                                    |
|      Willkommen!                   |  <- Titel slided
|                                    |
|      Entdecke unsere App           |  <- Subtitle faded
|                                    |
|      +-------------------+         |
|      |    Los geht's     |         |  <- Button scaled
|      +-------------------+         |
|                                    |
+------------------------------------+
```

Anforderungen: - Elemente erscheinen nacheinander (staggered) - Verschiedene Animationstypen - Alle mit `TweenAnimationBuilder` - Verzögerung zwischen Elementen: 200ms - Gesamtdauer: ~1 Sekunde

4.1.1.9 Bonus: Animated Theme Switcher

Erstelle einen Theme-Wechsel mit Animation:

```
// Anforderung:
// - Dark/Light Mode Toggle
// - Sanfter Farbübergang für gesamte App
// - Icon-Animation (Sonne ↔ Mond)
// - Optional: Ripple-Effekt vom Toggle-Button
```

4.1.1.10 Abgabe-Checkliste

- ☐ Settings Toggle mit mehreren Animationen
- ☐ Expandable Card funktioniert smooth
- ☐ AnimatedSwitcher mit 4 Übergängen
- ☐ Progress Animation mit Farbwechsel
- ☐ AnimatedList mit Add/Remove
- ☐ Animated Navigation Bar
- ☐ Staggered Landing Page

☐ Code ist sauber und kommentiert

4.1.2 Lösung

4.1.2.1 Aufgabe 1: Animated Settings Toggle

```
class AnimatedSettingsToggle extends StatefulWidget {
  const AnimatedSettingsToggle({super.key});

  @override
  State<AnimatedSettingsToggle> createState() => _AnimatedSettingsToggleState();
}

class _AnimatedSettingsToggleState extends State<AnimatedSettingsToggle> {
  bool _isEnabled = false;

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => setState(() => _isEnabled = !_isEnabled),
      child: AnimatedContainer(
        duration: const Duration(milliseconds: 300),
        curve: Curves.easeInOut,
        padding: const EdgeInsets.all(16),
        decoration: BoxDecoration(
          color: _isEnabled ? Colors.green[50] : Colors.grey[100],
          borderRadius: BorderRadius.circular(12),
          border: Border.all(
            color: _isEnabled ? Colors.green : Colors.grey[300]!,
            width: 2,
          ),
        ),
        child: Row(
          children: [
            // Animiertes Icon
            TweenAnimationBuilder<double>(
              tween: Tween(begin: 1.0, end: _isEnabled ? 1.3 : 1.0),
              duration: const Duration(milliseconds: 300),
              builder: (context, scale, child) {
                return Transform.scale(
                  scale: scale,
                  child: Icon(
                    Icons.notifications,
                    color: _isEnabled ? Colors.green : Colors.grey,
                    size: 28,
                  ),
                );
              },
            ),
            const SizedBox(width: 16),
            // Animierter Text
```

```
            Expanded(
              child: AnimatedDefaultTextStyle(
                duration: const Duration(milliseconds: 300),
                style: TextStyle(
                  fontSize: 16,
                  fontWeight: _isEnabled ? FontWeight.bold : FontWeight.normal,
                  color: _isEnabled ? Colors.green[800] : Colors.grey[700],
                ),
                child: const Text('Benachrichtigungen'),
              ),
            ),
            // Animierter Switch-Indikator
            AnimatedContainer(
              duration: const Duration(milliseconds: 300),
              width: 50,
              height: 28,
              decoration: BoxDecoration(
                color: _isEnabled ? Colors.green : Colors.grey[400],
                borderRadius: BorderRadius.circular(14),
              ),
              child: AnimatedAlign(
                duration: const Duration(milliseconds: 300),
                curve: Curves.easeInOut,
                alignment:
                    _isEnabled ? Alignment.centerRight : Alignment.centerLeft,
                child: Container(
                  margin: const EdgeInsets.all(3),
                  width: 22,
                  height: 22,
                  decoration: const BoxDecoration(
                    color: Colors.white,
                    shape: BoxShape.circle,
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

4.1.2.2 Aufgabe 2: Expandable Card

```
class ExpandableOrderCard extends StatefulWidget {
  const ExpandableOrderCard({super.key});

  @override
  State<ExpandableOrderCard> createState() => _ExpandableOrderCardState();
```

```
}

class _ExpandableOrderCardState extends State<ExpandableOrderCard> {
  bool _isExpanded = false;

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => setState(() => _isExpanded = !_isExpanded),
      child: AnimatedContainer(
        duration: const Duration(milliseconds: 300),
        curve: Curves.easeInOut,
        decoration: BoxDecoration(
          color: Colors.white,
          borderRadius: BorderRadius.circular(12),
          boxShadow: [
            BoxShadow(
              color: Colors.black.withOpacity(_isExpanded ? 0.15 : 0.08),
              blurRadius: _isExpanded ? 12 : 6,
              offset: Offset(0, _isExpanded ? 4 : 2),
            ),
          ],
        ),
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            // Header (immer sichtbar)
            Padding(
              padding: const EdgeInsets.all(16),
              child: Row(
                children: [
                  const Icon(Icons.inventory_2, color: Colors.blue),
                  const SizedBox(width: 12),
                  const Expanded(
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Text(
                          'Bestellung #12345',
                          style: TextStyle(
                            fontWeight: FontWeight.bold,
                            fontSize: 16,
                          ),
                        ),
                        Text(
                          'Status: Versendet',
                          style: TextStyle(color: Colors.grey),
                        ),
                      ],
                    ),
                  ),
```

```
                AnimatedRotation(
                  turns: _isExpanded ? 0.5 : 0,
                  duration: const Duration(milliseconds: 300),
                  child: const Icon(Icons.keyboard_arrow_down),
                ),
              ],
            ),
          ),

          // Expandable Content
          AnimatedCrossFade(
            duration: const Duration(milliseconds: 300),
            crossFadeState: _isExpanded
                ? CrossFadeState.showSecond
                : CrossFadeState.showFirst,
            firstChild: const SizedBox(width: double.infinity),
            secondChild: Container(
              width: double.infinity,
              padding: const EdgeInsets.fromLTRB(16, 0, 16, 16),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  const Divider(),
                  const SizedBox(height: 8),
                  _buildDetailRow('Artikel:', 'Flutter Buch'),
                  _buildDetailRow('Preis:', '29,99€'),
                  _buildDetailRow('Lieferadresse:', 'Musterstr. 1'),
                  _buildDetailRow('Voraussichtlich:', '15.03.2024'),
                ],
              ),
            ),
          ),
        ],
      ),
    ),
  );
}

Widget _buildDetailRow(String label, String value) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 4),
    child: Row(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        SizedBox(
          width: 120,
          child: Text(
            label,
            style: const TextStyle(color: Colors.grey),
          ),
        ),
```

```
          Expanded(
            child: Text(
              value,
              style: const TextStyle(fontWeight: FontWeight.w500),
            ),
          ),
        ],
      ),
    );
  }
}
```

4.1.2.3 Aufgabe 3: AnimatedSwitcher Gallery

```
class AnimatedGallery extends StatefulWidget {
  const AnimatedGallery({super.key});

  @override
  State<AnimatedGallery> createState() => _AnimatedGalleryState();
}

class _AnimatedGalleryState extends State<AnimatedGallery> {
  int _currentIndex = 0;
  String _transitionType = 'fade';

  final _images = [
    Colors.red,
    Colors.blue,
    Colors.green,
    Colors.orange,
    Colors.purple,
  ];

  final _transitions = ['fade', 'scale', 'slide', 'rotation'];

  Widget _buildTransition(Widget child, Animation<double> animation) {
    switch (_transitionType) {
      case 'scale':
        return ScaleTransition(scale: animation, child: child);
      case 'slide':
        return SlideTransition(
          position: Tween<Offset>(
            begin: const Offset(1, 0),
            end: Offset.zero,
          ).animate(animation),
          child: child,
        );
      case 'rotation':
        return RotationTransition(
          turns: animation,
```

```
        child: FadeTransition(opacity: animation, child: child),
      );
    default:
      return FadeTransition(opacity: animation, child: child);
  }
}

void _previous() {
  setState(() {
    _currentIndex = (_currentIndex - 1 + _images.length) % _images.length;
  });
}

void _next() {
  setState(() {
    _currentIndex = (_currentIndex + 1) % _images.length;
  });
}

@override
Widget build(BuildContext context) {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      // Gallery
      SizedBox(
        height: 200,
        child: AnimatedSwitcher(
          duration: const Duration(milliseconds: 400),
          transitionBuilder: _buildTransition,
          child: Container(
            key: ValueKey(_currentIndex),
            width: 200,
            height: 200,
            decoration: BoxDecoration(
              color: _images[_currentIndex],
              borderRadius: BorderRadius.circular(16),
            ),
            child: Center(
              child: Text(
                'Bild ${_currentIndex + 1}',
                style: const TextStyle(
                  color: Colors.white,
                  fontSize: 24,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
          ),
        ),
      ),
```

```
        const SizedBox(height: 24),

        // Navigation
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            IconButton(
              icon: const Icon(Icons.arrow_back),
              onPressed: _previous,
            ),
            const SizedBox(width: 32),
            IconButton(
              icon: const Icon(Icons.arrow_forward),
              onPressed: _next,
            ),
          ],
        ),
        const SizedBox(height: 24),

        // Transition Selector
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text('Übergang: '),
            DropdownButton<String>(
              value: _transitionType,
              items: _transitions.map((t) {
                return DropdownMenuItem(value: t, child: Text(t));
              }).toList(),
              onChanged: (v) => setState(() => _transitionType = v!),
            ),
          ],
        ),
      ],
    );
  }
}
```

4.1.2.4 Aufgabe 4: Progress Animation

```
class AnimatedProgress extends StatefulWidget {
  const AnimatedProgress({super.key});

  @override
  State<AnimatedProgress> createState() => _AnimatedProgressState();
}

class _AnimatedProgressState extends State<AnimatedProgress> {
  double _progress = 0;
  bool _isComplete = false;
```

```
Color _getColor(double progress) {
  if (progress < 0.3) return Colors.red;
  if (progress < 0.7) return Colors.orange;
  return Colors.green;
}

void _simulate() async {
  setState(() {
    _progress = 0;
    _isComplete = false;
  });

  for (int i = 0; i <= 100; i += 5) {
    await Future.delayed(const Duration(milliseconds: 100));
    if (mounted) {
      setState(() => _progress = i / 100);
    }
  }

  if (mounted) {
    setState(() => _isComplete = true);
  }
}

@override
Widget build(BuildContext context) {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      AnimatedSwitcher(
        duration: const Duration(milliseconds: 300),
        child: _isComplete
            ? const Icon(
                Icons.check_circle,
                key: ValueKey('complete'),
                color: Colors.green,
                size: 64,
              )
            : const Text(
                'Upload läuft...',
                key: ValueKey('uploading'),
                style: TextStyle(fontSize: 18),
              ),
      ),
      const SizedBox(height: 24),

      // Progress Bar
      SizedBox(
        width: 250,
        height: 20,
```

```
          child: TweenAnimationBuilder<double>(
            tween: Tween(begin: 0, end: _progress),
            duration: const Duration(milliseconds: 200),
            builder: (context, value, child) {
              return Stack(
                children: [
                  // Background
                  Container(
                    decoration: BoxDecoration(
                      color: Colors.grey[300],
                      borderRadius: BorderRadius.circular(10),
                    ),
                  ),
                  // Progress
                  FractionallySizedBox(
                    widthFactor: value,
                    child: TweenAnimationBuilder<Color?>(
                      tween: ColorTween(
                        begin: _getColor(value),
                        end: _getColor(value),
                      ),
                      duration: const Duration(milliseconds: 300),
                      builder: (context, color, child) {
                        return Container(
                          decoration: BoxDecoration(
                            color: color,
                            borderRadius: BorderRadius.circular(10),
                          ),
                        );
                      },
                    ),
                  ),
                ],
              );
            },
          ),
        ),
        const SizedBox(height: 16),

        // Percentage
        TweenAnimationBuilder<double>(
          tween: Tween(begin: 0, end: _progress),
          duration: const Duration(milliseconds: 200),
          builder: (context, value, child) {
            return Text(
              '${(value * 100).toInt()}%',
              style: TextStyle(
                fontSize: 24,
                fontWeight: FontWeight.bold,
                color: _getColor(value),
              ),
```

```
            );
          },
        ),
        const SizedBox(height: 24),

        ElevatedButton(
          onPressed: _simulate,
          child: const Text('Simulieren'),
        ),
      ],
    );
  }
}
```

4.1.2.5 Aufgabe 5: AnimatedList Todo

```
class AnimatedTodoList extends StatefulWidget {
  const AnimatedTodoList({super.key});

  @override
  State<AnimatedTodoList> createState() => _AnimatedTodoListState();
}

class _AnimatedTodoListState extends State<AnimatedTodoList> {
  final _listKey = GlobalKey<AnimatedListState>();
  final _todos = <Todo>[
    Todo(text: 'Flutter lernen'),
    Todo(text: 'Dart verstanden', isDone: true),
    Todo(text: 'App veröffentlichen'),
  ];

  void _addTodo() {
    final index = _todos.length;
    _todos.add(Todo(text: 'Neues Todo ${index + 1}'));
    _listKey.currentState?.insertItem(
      index,
      duration: const Duration(milliseconds: 300),
    );
  }

  void _removeTodo(int index) {
    final removed = _todos.removeAt(index);
    _listKey.currentState?.removeItem(
      index,
      (context, animation) => _buildTodoItem(removed, animation, index),
      duration: const Duration(milliseconds: 300),
    );
  }

  void _toggleTodo(int index) {
```

```
    setState(() {
      _todos[index].isDone = !_todos[index].isDone;
    });
  }

  Widget _buildTodoItem(Todo todo, Animation<double> animation, int index) {
    return SlideTransition(
      position: Tween<Offset>(
        begin: const Offset(1, 0),
        end: Offset.zero,
      ).animate(CurvedAnimation(
        parent: animation,
        curve: Curves.easeOut,
      )),
      child: FadeTransition(
        opacity: animation,
        child: SizeTransition(
          sizeFactor: animation,
          child: _TodoTile(
            todo: todo,
            onToggle: () => _toggleTodo(index),
            onDelete: () => _removeTodo(index),
          ),
        ),
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Meine Todos'),
        actions: [
          IconButton(
            icon: const Icon(Icons.add),
            onPressed: _addTodo,
          ),
        ],
      ),
      body: AnimatedList(
        key: _listKey,
        initialItemCount: _todos.length,
        itemBuilder: (context, index, animation) {
          return _buildTodoItem(_todos[index], animation, index);
        },
      ),
    );
  }
}
```

```
class Todo {
  String text;
  bool isDone;

  Todo({required this.text, this.isDone = false});
}

class _TodoTile extends StatelessWidget {
  final Todo todo;
  final VoidCallback onToggle;
  final VoidCallback onDelete;

  const _TodoTile({
    required this.todo,
    required this.onToggle,
    required this.onDelete,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: const EdgeInsets.symmetric(horizontal: 16, vertical: 4),
      child: ListTile(
        leading: GestureDetector(
          onTap: onToggle,
          child: AnimatedContainer(
            duration: const Duration(milliseconds: 200),
            width: 28,
            height: 28,
            decoration: BoxDecoration(
              color: todo.isDone ? Colors.green : Colors.transparent,
              border: Border.all(
                color: todo.isDone ? Colors.green : Colors.grey,
                width: 2,
              ),
              borderRadius: BorderRadius.circular(6),
            ),
            child: todo.isDone
                ? const Icon(Icons.check, color: Colors.white, size: 20)
                : null,
          ),
        ),
        title: AnimatedDefaultTextStyle(
          duration: const Duration(milliseconds: 200),
          style: TextStyle(
            decoration: todo.isDone ? TextDecoration.lineThrough : null,
            color: todo.isDone ? Colors.grey : Colors.black,
            fontSize: 16,
          ),
          child: Text(todo.text),
        ),
```

```
        trailing: IconButton(
          icon: const Icon(Icons.delete_outline),
          onPressed: onDelete,
        ),
      ),
    );
  }
}
```

4.1.2.6 Aufgabe 6: Animated Navigation Bar

```
class AnimatedBottomNav extends StatefulWidget {
  const AnimatedBottomNav({super.key});

  @override
  State<AnimatedBottomNav> createState() => _AnimatedBottomNavState();
}

class _AnimatedBottomNavState extends State<AnimatedBottomNav> {
  int _selectedIndex = 0;

  final _items = [
    (Icons.home, 'Home'),
    (Icons.search, 'Search'),
    (Icons.favorite, 'Favs'),
    (Icons.person, 'Profile'),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: AnimatedSwitcher(
        duration: const Duration(milliseconds: 300),
        child: Center(
          key: ValueKey(_selectedIndex),
          child: Text(
            _items[_selectedIndex].$2,
            style: const TextStyle(fontSize: 32),
          ),
        ),
      ),
      bottomNavigationBar: Container(
        height: 80,
        decoration: BoxDecoration(
          color: Colors.white,
          boxShadow: [
            BoxShadow(
              color: Colors.black.withOpacity(0.1),
              blurRadius: 10,
              offset: const Offset(0, -2),
```

```
          ),
        ],
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        children: List.generate(_items.length, (index) {
          final isSelected = _selectedIndex == index;
          return GestureDetector(
            onTap: () => setState(() => _selectedIndex = index),
            child: AnimatedContainer(
              duration: const Duration(milliseconds: 200),
              padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
              decoration: BoxDecoration(
                color: isSelected
                    ? Colors.blue.withOpacity(0.1)
                    : Colors.transparent,
                borderRadius: BorderRadius.circular(20),
              ),
              child: Column(
                mainAxisSize: MainAxisSize.min,
                children: [
                  TweenAnimationBuilder<double>(
                    tween: Tween(begin: 1, end: isSelected ? 1.2 : 1.0),
                    duration: const Duration(milliseconds: 200),
                    builder: (context, scale, child) {
                      return Transform.scale(
                        scale: scale,
                        child: Icon(
                          _items[index].$1,
                          color: isSelected ? Colors.blue : Colors.grey,
                        ),
                      );
                    },
                  ),
                  AnimatedOpacity(
                    duration: const Duration(milliseconds: 200),
                    opacity: isSelected ? 1.0 : 0.0,
                    child: AnimatedContainer(
                      duration: const Duration(milliseconds: 200),
                      height: isSelected ? 16 : 0,
                      child: Text(
                        _items[index].$2,
                        style: const TextStyle(
                          color: Colors.blue,
                          fontSize: 12,
                          fontWeight: FontWeight.w500,
                        ),
                      ),
                    ),
                  ),
                  AnimatedContainer(
```

```
                    duration: const Duration(milliseconds: 200),
                    height: 4,
                    width: isSelected ? 4 : 0,
                    margin: const EdgeInsets.only(top: 4),
                    decoration: const BoxDecoration(
                      color: Colors.blue,
                      shape: BoxShape.circle,
                    ),
                  ),
                ],
              ),
            ),
          );
        }),
      ),
    ),
  );
  }
}
```

4.1.2.7 Aufgabe 7: Staggered Landing Page

```
class StaggeredLandingPage extends StatefulWidget {
  const StaggeredLandingPage({super.key});

  @override
  State<StaggeredLandingPage> createState() => _StaggeredLandingPageState();
}

class _StaggeredLandingPageState extends State<StaggeredLandingPage> {
  bool _animate = false;

  @override
  void initState() {
    super.initState();
    // Start animation after build
    WidgetsBinding.instance.addPostFrameCallback((_) {
      setState(() => _animate = true);
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            // Logo (delay: 0ms)
            _AnimatedElement(
```

```
              animate: _animate,
              delay: const Duration(milliseconds: 0),
              child: Container(
                width: 100,
                height: 100,
                decoration: BoxDecoration(
                  color: Colors.blue,
                  borderRadius: BorderRadius.circular(20),
                ),
                child: const Icon(
                  Icons.flutter_dash,
                  size: 60,
                  color: Colors.white,
                ),
              ),
            ),
            const SizedBox(height: 32),

            // Title (delay: 200ms)
            _AnimatedElement(
              animate: _animate,
              delay: const Duration(milliseconds: 200),
              slideOffset: const Offset(0, 0.5),
              child: const Text(
                'Willkommen!',
                style: TextStyle(
                  fontSize: 32,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
            const SizedBox(height: 16),

            // Subtitle (delay: 400ms)
            _AnimatedElement(
              animate: _animate,
              delay: const Duration(milliseconds: 400),
              child: Text(
                'Entdecke unsere App',
                style: TextStyle(
                  fontSize: 18,
                  color: Colors.grey[600],
                ),
              ),
            ),
            const SizedBox(height: 48),

            // Button (delay: 600ms)
            _AnimatedElement(
              animate: _animate,
              delay: const Duration(milliseconds: 600),
```

```
              useScale: true,
              child: ElevatedButton(
                onPressed: () {},
                style: ElevatedButton.styleFrom(
                  padding: const EdgeInsets.symmetric(
                    horizontal: 48,
                    vertical: 16,
                  ),
                ),
                child: const Text(
                  "Los geht's",
                  style: TextStyle(fontSize: 18),
                ),
              ),
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(() => _animate = false);
          Future.delayed(const Duration(milliseconds: 100), () {
            setState(() => _animate = true);
          });
        },
        child: const Icon(Icons.replay),
      ),
    );
  }
}

class _AnimatedElement extends StatelessWidget {
  final bool animate;
  final Duration delay;
  final Widget child;
  final Offset slideOffset;
  final bool useScale;

  const _AnimatedElement({
    required this.animate,
    required this.delay,
    required this.child,
    this.slideOffset = Offset.zero,
    this.useScale = false,
  });

  @override
  Widget build(BuildContext context) {
    return TweenAnimationBuilder<double>(
      tween: Tween(begin: 0, end: animate ? 1.0 : 0.0),
      duration: const Duration(milliseconds: 500),
```

```
      curve: Curves.easeOut,
      builder: (context, value, child) {
        // Apply delay
        final delayedValue = animate
            ? ((DateTime.now().millisecondsSinceEpoch / 1000) >
 ↪  delay.inMilliseconds / 1000
                ? value
                : 0.0)
            : 0.0;

        return Opacity(
          opacity: delayedValue.clamp(0.0, 1.0),
          child: Transform.translate(
            offset: Offset(
              slideOffset.dx * (1 - delayedValue) * 50,
              slideOffset.dy * (1 - delayedValue) * 50,
            ),
            child: useScale
                ? Transform.scale(
                    scale: 0.5 + (delayedValue * 0.5),
                    child: child,
                  )
                : child,
          ),
        );
      },
      child: child,
    );
  }
}

// Alternative mit echtem Delay
class StaggeredLandingPageV2 extends StatefulWidget {
  const StaggeredLandingPageV2({super.key});

  @override
  State<StaggeredLandingPageV2> createState() => _StaggeredLandingPageV2State();
}

class _StaggeredLandingPageV2State extends State<StaggeredLandingPageV2> {
  final _visible = [false, false, false, false];

  @override
  void initState() {
    super.initState();
    _startAnimations();
  }

  void _startAnimations() async {
    for (int i = 0; i < _visible.length; i++) {
      await Future.delayed(const Duration(milliseconds: 200));
```

```
    if (mounted) {
      setState(() => _visible[i] = true);
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          // Logo
          AnimatedOpacity(
            duration: const Duration(milliseconds: 500),
            opacity: _visible[0] ? 1.0 : 0.0,
            child: AnimatedScale(
              duration: const Duration(milliseconds: 500),
              scale: _visible[0] ? 1.0 : 0.5,
              child: Container(
                width: 100,
                height: 100,
                decoration: BoxDecoration(
                  color: Colors.blue,
                  borderRadius: BorderRadius.circular(20),
                ),
                child: const Icon(
                  Icons.flutter_dash,
                  size: 60,
                  color: Colors.white,
                ),
              ),
            ),
          ),
          const SizedBox(height: 32),

          // Title
          AnimatedSlide(
            duration: const Duration(milliseconds: 500),
            offset: _visible[1] ? Offset.zero : const Offset(0, 0.5),
            child: AnimatedOpacity(
              duration: const Duration(milliseconds: 500),
              opacity: _visible[1] ? 1.0 : 0.0,
              child: const Text(
                'Willkommen!',
                style: TextStyle(fontSize: 32, fontWeight: FontWeight.bold),
              ),
            ),
          ),
          const SizedBox(height: 16),
```

```
            // Subtitle
            AnimatedOpacity(
              duration: const Duration(milliseconds: 500),
              opacity: _visible[2] ? 1.0 : 0.0,
              child: Text(
                'Entdecke unsere App',
                style: TextStyle(fontSize: 18, color: Colors.grey[600]),
              ),
            ),
            const SizedBox(height: 48),

            // Button
            AnimatedScale(
              duration: const Duration(milliseconds: 500),
              curve: Curves.elasticOut,
              scale: _visible[3] ? 1.0 : 0.0,
              child: ElevatedButton(
                onPressed: () {},
                style: ElevatedButton.styleFrom(
                  padding: const EdgeInsets.symmetric(
                    horizontal: 48,
                    vertical: 16,
                  ),
                ),
                child: const Text("Los geht's", style: TextStyle(fontSize: 18)),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

4.1.3 Ressourcen

4.1.3.1 Offizielle Dokumentation

- Animations Tutorial
- Implicit Animations
- AnimatedContainer
- AnimatedSwitcher
- TweenAnimationBuilder
- AnimatedList
- Curves

4.1.3.2 Flutter Cookbook

- Animate a container
- Fade a widget in and out

4.1.3.3 Videos

- Implicit Animations - Flutter in Focus
- AnimatedContainer Widget of the Week
- AnimatedSwitcher Widget of the Week
- TweenAnimationBuilder Widget of the Week
- AnimatedList Widget of the Week

4.1.3.4 Tutorials & Artikel

- Flutter Animation Guide
- Mastering Implicit Animations
- When to use which animation

4.1.3.5 Packages

- animations - Material Motion Animationen
- flutter_animate - Deklarative Animationen
- animated_text_kit - Text-Animationen

4.1.3.6 Interaktive Tools

- Curve Visualizer - Alle Curves visualisiert
- Flutter Animation Playground

4.1.3.7 Zum Vertiefen

- Custom Implicit Animations
- Physics-based Animations
- Hero Animations

4.2 Einheit 4.2: Explizite Animationen

4.2.0.1 Lernziele

Nach dieser Einheit kannst du: - `AnimationController` erstellen und steuern - `Tween` und `CurvedAnimation` nutzen - `AnimatedBuilder` und `AnimatedWidget` einsetzen - Hero-Animationen implementieren - Lottie-Animationen einbinden

4.2.0.2 1. AnimationController

Grundlagen

```
class ExplicitAnimationDemo extends StatefulWidget {
  const ExplicitAnimationDemo({super.key});

  @override
  State<ExplicitAnimationDemo> createState() => _ExplicitAnimationDemoState();
}

class _ExplicitAnimationDemoState extends State<ExplicitAnimationDemo>
    with SingleTickerProviderStateMixin {

  late AnimationController _controller;
```

```
@override
void initState() {
  super.initState();
  _controller = AnimationController(
    duration: const Duration(seconds: 2),
    vsync: this,  // TickerProvider für Performance
  );
}

@override
void dispose() {
  _controller.dispose();  // Wichtig: Immer disposen!
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      // Animation anzeigen
      AnimatedBuilder(
        animation: _controller,
        builder: (context, child) {
          return Transform.rotate(
            angle: _controller.value * 2 * 3.14159,
            child: child,
          );
        },
        child: const FlutterLogo(size: 100),
      ),
      const SizedBox(height: 32),

      // Steuerung
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ElevatedButton(
            onPressed: () => _controller.forward(),
            child: const Text('Forward'),
          ),
          const SizedBox(width: 8),
          ElevatedButton(
            onPressed: () => _controller.reverse(),
            child: const Text('Reverse'),
          ),
          const SizedBox(width: 8),
          ElevatedButton(
            onPressed: () => _controller.repeat(),
            child: const Text('Repeat'),
```

```
            ),
            const SizedBox(width: 8),
            ElevatedButton(
              onPressed: () => _controller.stop(),
              child: const Text('Stop'),
            ),
          ],
        ),
      ],
    );
  }
}
```

AnimationController Methoden

```
// Vorwärts abspielen (0 -> 1)
_controller.forward();

// Rückwärts abspielen (1 -> 0)
_controller.reverse();

// Von Anfang starten
_controller.forward(from: 0.0);

// Wiederholen
_controller.repeat();

// Hin und her wiederholen
_controller.repeat(reverse: true);

// Stoppen
_controller.stop();

// Zurücksetzen
_controller.reset();

// Direkt auf Wert setzen
_controller.value = 0.5;

// Zu Wert animieren
_controller.animateTo(0.75);

// Status abfragen
_controller.status;  // AnimationStatus.forward, .reverse, .completed, .dismissed

// Listener
_controller.addListener(() {
  print('Value: ${_controller.value}');
});

_controller.addStatusListener((status) {
```

```
  if (status == AnimationStatus.completed) {
    print('Animation fertig!');
  }
});
```

Multiple Controller (TickerProviderStateMixin)

```
class MultipleAnimations extends StatefulWidget {
  const MultipleAnimations({super.key});

  @override
  State<MultipleAnimations> createState() => _MultipleAnimationsState();
}

class _MultipleAnimationsState extends State<MultipleAnimations>
    with TickerProviderStateMixin {  // Nicht SingleTickerProvider!

  late AnimationController _controller1;
  late AnimationController _controller2;

  @override
  void initState() {
    super.initState();
    _controller1 = AnimationController(
      duration: const Duration(seconds: 1),
      vsync: this,
    );
    _controller2 = AnimationController(
      duration: const Duration(milliseconds: 500),
      vsync: this,
    );
  }

  @override
  void dispose() {
    _controller1.dispose();
    _controller2.dispose();
    super.dispose();
  }

  // ...
}
```

4.2.0.3 2. Tween

Grundlagen

```
// Tween definiert Start- und Endwert
final sizeTween = Tween<double>(begin: 50, end: 200);

// Animation erstellen
final sizeAnimation = sizeTween.animate(_controller);
```

```
// Wert abrufen
print(sizeAnimation.value);  // Interpolierter Wert
```

Verschiedene Tween-Typen

```
// Double
Tween<double>(begin: 0, end: 100);

// Color
ColorTween(begin: Colors.red, end: Colors.blue);

// Offset (für Bewegung)
Tween<Offset>(begin: Offset.zero, end: const Offset(100, 50));

// Size
SizeTween(begin: const Size(50, 50), end: const Size(200, 200));

// Rect
RectTween(
  begin: const Rect.fromLTWH(0, 0, 50, 50),
  end: const Rect.fromLTWH(100, 100, 200, 200),
);

// Int
IntTween(begin: 0, end: 100);

// BorderRadius
BorderRadiusTween(
  begin: BorderRadius.circular(0),
  end: BorderRadius.circular(50),
);

// Decoration
DecorationTween(
  begin: const BoxDecoration(color: Colors.red),
  end: const BoxDecoration(color: Colors.blue),
);

// EdgeInsets
EdgeInsetsTween(
  begin: const EdgeInsets.all(0),
  end: const EdgeInsets.all(32),
);
```

Tween-Ketten

```
class ChainedAnimation extends StatefulWidget {
  const ChainedAnimation({super.key});

  @override
  State<ChainedAnimation> createState() => _ChainedAnimationState();
```

```
}

class _ChainedAnimationState extends State<ChainedAnimation>
    with SingleTickerProviderStateMixin {

  late AnimationController _controller;
  late Animation<double> _sizeAnimation;
  late Animation<Color?> _colorAnimation;
  late Animation<double> _rotationAnimation;

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(seconds: 2),
      vsync: this,
    );

    // Verschiedene Tweens mit demselben Controller
    _sizeAnimation = Tween<double>(begin: 50, end: 200).animate(_controller);

    _colorAnimation = ColorTween(
      begin: Colors.red,
      end: Colors.blue,
    ).animate(_controller);

    _rotationAnimation = Tween<double>(
      begin: 0,
      end: 2 * 3.14159,
    ).animate(_controller);
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        if (_controller.status == AnimationStatus.completed) {
          _controller.reverse();
        } else {
          _controller.forward();
        }
      },
      child: AnimatedBuilder(
        animation: _controller,
```

```
        builder: (context, child) {
          return Transform.rotate(
            angle: _rotationAnimation.value,
            child: Container(
              width: _sizeAnimation.value,
              height: _sizeAnimation.value,
              color: _colorAnimation.value,
            ),
          );
        },
      ),
    );
  }
}
```

4.2.0.4 3. CurvedAnimation

Curves anwenden

```
class CurvedAnimationDemo extends StatefulWidget {
  const CurvedAnimationDemo({super.key});

  @override
  State<CurvedAnimationDemo> createState() => _CurvedAnimationDemoState();
}

class _CurvedAnimationDemoState extends State<CurvedAnimationDemo>
    with SingleTickerProviderStateMixin {

  late AnimationController _controller;
  late Animation<double> _animation;

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(seconds: 1),
      vsync: this,
    );

    // CurvedAnimation für nicht-lineare Bewegung
    _animation = CurvedAnimation(
      parent: _controller,
      curve: Curves.elasticOut,
      reverseCurve: Curves.easeIn,  // Andere Curve beim Rückwärts
    );
  }

  @override
  void dispose() {
```

```
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return AnimatedBuilder(
      animation: _animation,
      builder: (context, child) {
        return Transform.scale(
          scale: _animation.value,
          child: child,
        );
      },
      child: const FlutterLogo(size: 100),
    );
  }
}
```

Tween mit Curve kombinieren

```
// Methode 1: CurvedAnimation
final curvedAnimation = CurvedAnimation(
  parent: _controller,
  curve: Curves.bounceOut,
);
final sizeAnimation = Tween<double>(begin: 0, end: 200).animate(curvedAnimation);

// Methode 2: chain()
final sizeAnimation = Tween<double>(begin: 0, end: 200)
    .chain(CurveTween(curve: Curves.bounceOut))
    .animate(_controller);
```

4.2.0.5 4. AnimatedBuilder vs AnimatedWidget

AnimatedBuilder

```
// Gut für einmalige Verwendung
AnimatedBuilder(
  animation: _controller,
  builder: (context, child) {
    return Transform.rotate(
      angle: _controller.value * 2 * 3.14159,
      child: child,  // child wird nicht rebuildet!
    );
  },
  child: const FlutterLogo(size: 100),  // Optimierung
)
```

AnimatedWidget (für Wiederverwendung)

```
// Eigenes animiertes Widget erstellen
class SpinningLogo extends AnimatedWidget {
  const SpinningLogo({
    super.key,
    required Animation<double> animation,
  }) : super(listenable: animation);

  Animation<double> get _animation => listenable as Animation<double>;

  @override
  Widget build(BuildContext context) {
    return Transform.rotate(
      angle: _animation.value * 2 * 3.14159,
      child: const FlutterLogo(size: 100),
    );
  }
}

// Verwendung
SpinningLogo(animation: _controller)
```

Custom Animated Widget mit Tween

```
class PulsingCircle extends AnimatedWidget {
  final Color color;

  const PulsingCircle({
    super.key,
    required Animation<double> animation,
    required this.color,
  }) : super(listenable: animation);

  @override
  Widget build(BuildContext context) {
    final animation = listenable as Animation<double>;
    return Container(
      width: 50 + (animation.value * 20),
      height: 50 + (animation.value * 20),
      decoration: BoxDecoration(
        color: color.withOpacity(1 - animation.value * 0.5),
        shape: BoxShape.circle,
      ),
    );
  }
}
```

4.2.0.6 5. Staggered Animations

Sequentielle Animationen

```
class StaggeredDemo extends StatefulWidget {
  const StaggeredDemo({super.key});
```

```
  @override
  State<StaggeredDemo> createState() => _StaggeredDemoState();
}

class _StaggeredDemoState extends State<StaggeredDemo>
    with SingleTickerProviderStateMixin {

  late AnimationController _controller;
  late Animation<double> _opacity;
  late Animation<double> _width;
  late Animation<double> _height;
  late Animation<EdgeInsets> _padding;
  late Animation<BorderRadius?> _borderRadius;
  late Animation<Color?> _color;

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(milliseconds: 2000),
      vsync: this,
    );

    // Staggered: Verschiedene Intervalle
    _opacity = Tween<double>(begin: 0, end: 1).animate(
      CurvedAnimation(
        parent: _controller,
        curve: const Interval(0.0, 0.2, curve: Curves.easeIn),
      ),
    );

    _width = Tween<double>(begin: 50, end: 200).animate(
      CurvedAnimation(
        parent: _controller,
        curve: const Interval(0.2, 0.4, curve: Curves.easeOut),
      ),
    );

    _height = Tween<double>(begin: 50, end: 200).animate(
      CurvedAnimation(
        parent: _controller,
        curve: const Interval(0.4, 0.6, curve: Curves.easeOut),
      ),
    );

    _padding = EdgeInsetsTween(
      begin: const EdgeInsets.all(0),
      end: const EdgeInsets.all(16),
    ).animate(
```

```
      CurvedAnimation(
        parent: _controller,
        curve: const Interval(0.6, 0.8, curve: Curves.easeOut),
      ),
    );

    _borderRadius = BorderRadiusTween(
      begin: BorderRadius.circular(0),
      end: BorderRadius.circular(50),
    ).animate(
      CurvedAnimation(
        parent: _controller,
        curve: const Interval(0.8, 1.0, curve: Curves.easeOut),
      ),
    );

    _color = ColorTween(begin: Colors.blue, end: Colors.purple).animate(
      CurvedAnimation(
        parent: _controller,
        curve: const Interval(0.0, 1.0),  // Über gesamte Dauer
      ),
    );
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        if (_controller.status == AnimationStatus.completed) {
          _controller.reverse();
        } else {
          _controller.forward();
        }
      },
      child: AnimatedBuilder(
        animation: _controller,
        builder: (context, child) {
          return Opacity(
            opacity: _opacity.value,
            child: Container(
              width: _width.value,
              height: _height.value,
              padding: _padding.value,
              decoration: BoxDecoration(
                color: _color.value,
```

```
              borderRadius: _borderRadius.value,
            ),
            child: const Center(
              child: Text(
                'Tap me',
                style: TextStyle(color: Colors.white),
              ),
            ),
          ),
        );
      },
    ),
  );
 }
}
```

4.2.0.7 6. Hero Animations

Basic Hero

```
// Screen 1 (Liste)
class ProductListScreen extends StatelessWidget {
  final products = ['Product 1', 'Product 2', 'Product 3'];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Products')),
      body: ListView.builder(
        itemCount: products.length,
        itemBuilder: (context, index) {
          return GestureDetector(
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => ProductDetailScreen(
                    productId: index,
                    productName: products[index],
                  ),
                ),
              );
            },
            child: Hero(
              tag: 'product-$index',  // Eindeutiger Tag!
              child: Card(
                child: ListTile(
                  leading: Container(
                    width: 50,
                    height: 50,
                    color: Colors.blue[100 * (index + 1)],
```

```
                ),
                title: Text(products[index]),
              ),
            ),
          ),
        );
      },
    ),
  );
 }
}

// Screen 2 (Detail)
class ProductDetailScreen extends StatelessWidget {
  final int productId;
  final String productName;

  const ProductDetailScreen({
    super.key,
    required this.productId,
    required this.productName,
  });

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(productName)),
      body: Column(
        children: [
          Hero(
            tag: 'product-$productId',  // Gleicher Tag!
            child: Container(
              width: double.infinity,
              height: 300,
              color: Colors.blue[100 * (productId + 1)],
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(16),
            child: Text(
              productName,
              style: const TextStyle(fontSize: 24),
            ),
          ),
        ],
      ),
    );
  }
}
```

Hero mit Custom Flight

```
Hero(
  tag: 'avatar',
  flightShuttleBuilder: (
    flightContext,
    animation,
    flightDirection,
    fromHeroContext,
    toHeroContext,
  ) {
    return AnimatedBuilder(
      animation: animation,
      builder: (context, child) {
        return Transform.rotate(
          angle: animation.value * 2 * 3.14159,
          child: fromHeroContext.widget,
        );
      },
    );
  },
  child: const CircleAvatar(
    radius: 30,
    child: Icon(Icons.person),
  ),
)
```

4.2.0.8 7. Lottie Animationen

Setup

```
# pubspec.yaml
dependencies:
  lottie: ^3.0.0
```

Basis-Verwendung

```
import 'package:lottie/lottie.dart';

// Aus Assets
Lottie.asset('assets/animations/loading.json')

// Aus Netzwerk
Lottie.network('https://example.com/animation.json')

// Mit Optionen
Lottie.asset(
  'assets/animations/success.json',
  width: 200,
  height: 200,
  fit: BoxFit.contain,
  repeat: false,  // Nur einmal abspielen
  reverse: true,  // Rückwärts
  animate: true,  // Auto-start
```

```
)
```

Mit Controller

```
class LottieControllerDemo extends StatefulWidget {
  const LottieControllerDemo({super.key});

  @override
  State<LottieControllerDemo> createState() => _LottieControllerDemoState();
}

class _LottieControllerDemoState extends State<LottieControllerDemo>
    with SingleTickerProviderStateMixin {

  late AnimationController _controller;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(vsync: this);
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Lottie.asset(
          'assets/animations/heart.json',
          controller: _controller,
          onLoaded: (composition) {
            // Duration aus Lottie-Datei übernehmen
            _controller.duration = composition.duration;
          },
        ),
        const SizedBox(height: 16),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            IconButton(
              icon: const Icon(Icons.play_arrow),
              onPressed: () => _controller.forward(),
            ),
            IconButton(
              icon: const Icon(Icons.pause),
```

```
              onPressed: () => _controller.stop(),
            ),
            IconButton(
              icon: const Icon(Icons.replay),
              onPressed: () => _controller.reset(),
            ),
            IconButton(
              icon: const Icon(Icons.repeat),
              onPressed: () => _controller.repeat(),
            ),
          ],
        ),
      ],
    );
  }
}
```

Lottie als Button

```
class LikeButton extends StatefulWidget {
  const LikeButton({super.key});

  @override
  State<LikeButton> createState() => _LikeButtonState();
}

class _LikeButtonState extends State<LikeButton>
    with SingleTickerProviderStateMixin {

  late AnimationController _controller;
  bool _isLiked = false;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(vsync: this);
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  void _toggleLike() {
    setState(() => _isLiked = !_isLiked);

    if (_isLiked) {
      _controller.forward();
    } else {
      _controller.reverse();
```

```
    }
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: _toggleLike,
      child: Lottie.asset(
        'assets/animations/like.json',
        controller: _controller,
        width: 60,
        height: 60,
        onLoaded: (composition) {
          _controller.duration = composition.duration;
        },
      ),
    );
  }
}
```

4.2.0.9 8. Praktisches Beispiel: Animated Drawer

```
class AnimatedDrawer extends StatefulWidget {
  const AnimatedDrawer({super.key});

  @override
  State<AnimatedDrawer> createState() => _AnimatedDrawerState();
}

class _AnimatedDrawerState extends State<AnimatedDrawer>
    with SingleTickerProviderStateMixin {

  late AnimationController _controller;
  late Animation<double> _slideAnimation;
  late Animation<double> _scaleAnimation;
  late Animation<double> _fadeAnimation;

  bool _isOpen = false;

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(milliseconds: 300),
      vsync: this,
    );

    _slideAnimation = Tween<double>(begin: -280, end: 0).animate(
      CurvedAnimation(parent: _controller, curve: Curves.easeOutCubic),
```

```
    );

    _scaleAnimation = Tween<double>(begin: 1, end: 0.85).animate(
      CurvedAnimation(parent: _controller, curve: Curves.easeOutCubic),
    );

    _fadeAnimation = Tween<double>(begin: 0, end: 0.5).animate(
      CurvedAnimation(parent: _controller, curve: Curves.easeOutCubic),
    );
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  void _toggleDrawer() {
    setState(() => _isOpen = !_isOpen);
    if (_isOpen) {
      _controller.forward();
    } else {
      _controller.reverse();
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          // Drawer
          AnimatedBuilder(
            animation: _controller,
            builder: (context, child) {
              return Transform.translate(
                offset: Offset(_slideAnimation.value, 0),
                child: child,
              );
            },
            child: Container(
              width: 280,
              color: Colors.blue[800],
              child: SafeArea(
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    const Padding(
                      padding: EdgeInsets.all(16),
                      child: Text(
                        'Menu',
```

```
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 24,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
                _menuItem(Icons.home, 'Home'),
                _menuItem(Icons.person, 'Profile'),
                _menuItem(Icons.settings, 'Settings'),
                _menuItem(Icons.logout, 'Logout'),
              ],
            ),
          ),
        ),
      ),

      // Main Content
      AnimatedBuilder(
        animation: _controller,
        builder: (context, child) {
          return Transform.translate(
            offset: Offset(_slideAnimation.value + 280, 0),
            child: Transform.scale(
              scale: _scaleAnimation.value,
              alignment: Alignment.centerLeft,
              child: child,
            ),
          );
        },
        child: GestureDetector(
          onTap: _isOpen ? _toggleDrawer : null,
          onHorizontalDragEnd: (details) {
            if (details.primaryVelocity! > 0 && !_isOpen) {
              _toggleDrawer();
            } else if (details.primaryVelocity! < 0 && _isOpen) {
              _toggleDrawer();
            }
          },
          child: Container(
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.circular(_isOpen ? 20 : 0),
              boxShadow: [
                BoxShadow(
                  color: Colors.black.withOpacity(0.2),
                  blurRadius: 20,
                ),
              ],
            ),
            child: Scaffold(
```

```
                appBar: AppBar(
                  leading: IconButton(
                    icon: AnimatedIcon(
                      icon: AnimatedIcons.menu_close,
                      progress: _controller,
                    ),
                    onPressed: _toggleDrawer,
                  ),
                  title: const Text('Animated Drawer'),
                ),
                body: const Center(
                  child: Text('Main Content'),
                ),
              ),
            ),
          ),
        ),

        // Overlay
        AnimatedBuilder(
          animation: _controller,
          builder: (context, child) {
            return IgnorePointer(
              ignoring: !_isOpen,
              child: Container(
                color: Colors.black.withOpacity(_fadeAnimation.value),
              ),
            );
          },
        ),
      ],
    ),
  );
 }

 Widget _menuItem(IconData icon, String title) {
   return ListTile(
     leading: Icon(icon, color: Colors.white70),
     title: Text(title, style: const TextStyle(color: Colors.white)),
     onTap: _toggleDrawer,
   );
 }
}
```

4.2.0.10 Zusammenfassung

Konzept	Verwendung
`AnimationController`	Steuert Animation (start, stop, repeat)
`Tween`	Definiert Wertebereich (begin -> end)
`CurvedAnimation`	Fügt Easing/Curves hinzu

Konzept	Verwendung
`AnimatedBuilder`	Rebuildet nur animierten Teil
`AnimatedWidget`	Wiederverwendbare animierte Widgets
`Interval`	Staggered Animations
`Hero`	Shared Element Transitions
`Lottie`	After Effects Animationen

Best Practices: - Immer `dispose()` für Controller - `SingleTickerProviderStateMixin` für einen Controller - `TickerProviderStateMixin` für mehrere Controller - `child`-Parameter in AnimatedBuilder für Performance - Hero-Tags müssen eindeutig sein

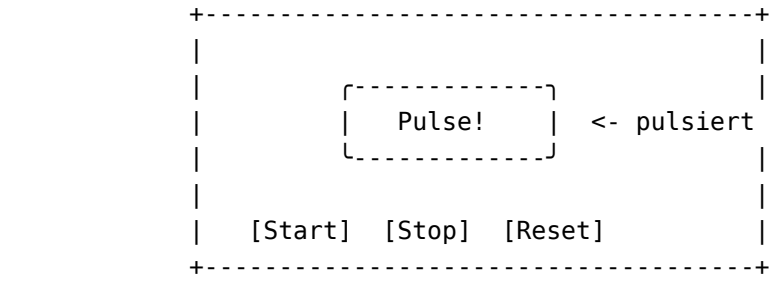
4.2.1 Übung

4.2.1.1 Ziel

AnimationController beherrschen und komplexe Animationen erstellen.

4.2.1.2 Aufgabe 1: Pulsing Button (20 min)

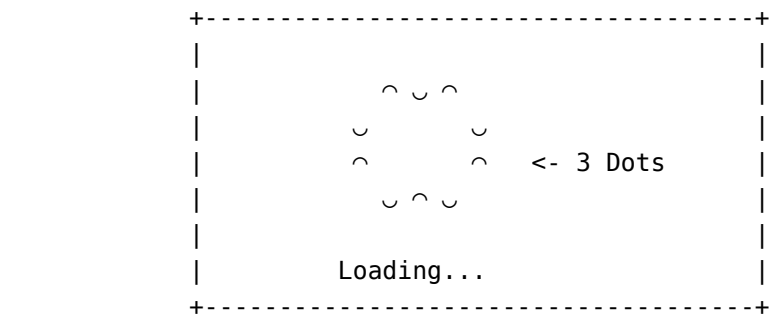
Erstelle einen Button mit Pulsier-Animation:



Anforderungen: - Button pulsiert kontinuierlich (scale 1.0 -> 1.2 -> 1.0) - Zusätzlich: Schatten pulsiert mit - Steuerung über Start/Stop/Reset Buttons - `repeat(reverse: true)` verwenden

4.2.1.3 Aufgabe 2: Loading Spinner (25 min)

Erstelle einen custom Loading-Spinner:



Anforderungen: - 3 Kreise die nacheinander hüpfen (staggered) - Jeder Kreis bewegt sich vertikal - Verschiedene Delays (0ms, 200ms, 400ms) - Smooth loop

4.2.1.4 Aufgabe 3: Flip Card (25 min)

Erstelle eine Karte die sich umdreht:

```
Vorderseite:           Rückseite (nach Flip):
+-------------+        +-------------+
|             |        |             |
|     ?       |   ->   |   Antwort   |
|  Frage...   |        |   Text...   |
|             |        |             |
+-------------+        +-------------+
```

Anforderungen: - 3D-Flip Animation (um Y-Achse) - Vorderseite zeigt Frage - Rückseite zeigt Antwort - Bei 90° Inhalt wechseln - Tap zum Umdrehen

4.2.1.5 Aufgabe 4: Staggered List (20 min)

Erstelle eine Liste mit gestaffelter Einblend-Animation:

```
+-----------------------------------+
| Notifications                     |
+-----------------------------------+
| +-------------------------------+ |  <- Slide in 0ms
| | ? Neue Nachricht            | |
| +-------------------------------+ |
| +-------------------------------+ |  <- Slide in 100ms
| | ? Erinnerung                | |
| +-------------------------------+ |
| +-------------------------------+ |  <- Slide in 200ms
| | ? Paket versendet           | |
| +-------------------------------+ |
+-----------------------------------+
```

Anforderungen: - Items sliden von rechts rein - Gestaffelt mit 100ms Verzögerung - Fade + Slide kombiniert - Button zum “Replay”

4.2.1.6 Aufgabe 5: Hero Gallery (25 min)

Erstelle eine Bildergalerie mit Hero-Animationen:

```
Grid-Ansicht:             Detail-Ansicht:
+---+---+---+             +-----------------+
| 1 | 2 | 3 |             |                 |
+---+---+---+    tap ->    |    Bild 1       |
| 4 | 5 | 6 |             |   (fullscreen)  |
+---+---+---+             |                 |
| 7 | 8 | 9 |             |  Beschreibung   |
+---+---+---+             +-----------------+
```

Anforderungen: - GridView mit Vorschaubildern - Tap öffnet Detail-Screen - Hero-Animation für das Bild - Zurück mit Back-Button/Gesture

4.2.1.7 Aufgabe 6: Animated Counter (20 min)

Erstelle einen Counter mit rollenden Zahlen:

```
+-----------------------------------+
|                                   |
|           +---------+             |
|           |  0042   |  <- Zahlen  |
```

```
|           +---------+    rollen    |
|                                    |
|       [-]           [+]            |
+------------------------------------+
```

Anforderungen: - Zahlen “rollen” wie bei einem Zähler - Jede Ziffer animiert einzeln - Slide-Transition nach oben/unten - Min: 0, Max: 9999

4.2.1.8 Aufgabe 7: Lottie Integration (20 min)

Integriere Lottie-Animationen:

```
+------------------------------------+
|                                    |
|         [Lottie Animation]         |
|                                    |
|   ▶ Play    Pause    Loop      |
|                                    |
|   Progress: ═══════▒▒ 70%     |
|                                    |
|   Speed: [0.5x] [1x] [2x]          |
+------------------------------------+
```

Anforderungen: - Lottie-Animation von LottieFiles laden - Play/Pause/Reset Controls - Progress-Anzeige - Geschwindigkeit änderbar - Loop toggle

4.2.1.9 Aufgabe 8: Animated Menu (30 min)

```
Geschlossen:              Geöffnet:
                                    
                                 
                          
     [+]           ->           [×]
                          ✏           
                                 
```

Anforderungen: - FAB öffnet/schließt Menu - Items fliegen radial nach außen - Staggered Animation - Rotation des Haupt-Buttons (+ -> ×) - Tap auf Item führt Aktion aus

4.2.1.10 Bonus: Page Transition

Erstelle eine Custom Page Transition:

```
// Ziel: Eigene Übergangsanimation zwischen Screens
// - Slide von unten + Fade
// - Scale-Effekt
// - Alte Page faded/scaled out
```

```
Screen 1                Transition              Screen 2
+---------+            +---------+            +---------+
|         |            | ▲▲▲▲▲▲▲ |            |         |
|  Page   |     ->      | Screen 2|      ->      |  Page   |
|    1    |            | ▲▲▲▲▲▲▲ |            |    2    |
|         |            |         |            |         |
+---------+            +---------+            +---------+
```

4.2.1.11 Abgabe-Checkliste

- ☐ Pulsing Button mit Controller
- ☐ Custom Loading Spinner (staggered)
- ☐ 3D Flip Card Animation
- ☐ Staggered List mit Interval
- ☐ Hero Gallery funktioniert
- ☐ Animated Counter mit rollenden Ziffern
- ☐ Lottie mit Controls
- ☐ Animated Radial Menu
- ☐ Alle Controller werden disposed
- ☐ Code ist sauber strukturiert

4.2.2 Lösung

4.2.2.1 Aufgabe 1: Pulsing Button

```
class PulsingButton extends StatefulWidget {
  const PulsingButton({super.key});

  @override
  State<PulsingButton> createState() => _PulsingButtonState();
}

class _PulsingButtonState extends State<PulsingButton>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _scaleAnimation;
  late Animation<double> _shadowAnimation;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(milliseconds: 800),
      vsync: this,
    );

    _scaleAnimation = Tween<double>(begin: 1.0, end: 1.2).animate(
      CurvedAnimation(parent: _controller, curve: Curves.easeInOut),
    );

    _shadowAnimation = Tween<double>(begin: 4, end: 12).animate(
      CurvedAnimation(parent: _controller, curve: Curves.easeInOut),
    );
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }
```

```
@override
Widget build(BuildContext context) {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      AnimatedBuilder(
        animation: _controller,
        builder: (context, child) {
          return Transform.scale(
            scale: _scaleAnimation.value,
            child: Container(
              padding: const EdgeInsets.symmetric(
                horizontal: 32,
                vertical: 16,
              ),
              decoration: BoxDecoration(
                color: Colors.blue,
                borderRadius: BorderRadius.circular(30),
                boxShadow: [
                  BoxShadow(
                    color: Colors.blue.withOpacity(0.5),
                    blurRadius: _shadowAnimation.value,
                    spreadRadius: _shadowAnimation.value / 4,
                  ),
                ],
              ),
              child: const Text(
                'Pulse!',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 18,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
          );
        },
      ),
      const SizedBox(height: 48),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ElevatedButton(
            onPressed: () => _controller.repeat(reverse: true),
            child: const Text('Start'),
          ),
          const SizedBox(width: 8),
          ElevatedButton(
            onPressed: () => _controller.stop(),
            child: const Text('Stop'),
```

```
            ),
            const SizedBox(width: 8),
            ElevatedButton(
              onPressed: () => _controller.reset(),
              child: const Text('Reset'),
            ),
          ],
        ),
      ],
    );
  }
}
```

4.2.2.2 Aufgabe 2: Loading Spinner

```
class BouncingDotsLoader extends StatefulWidget {
  const BouncingDotsLoader({super.key});

  @override
  State<BouncingDotsLoader> createState() => _BouncingDotsLoaderState();
}

class _BouncingDotsLoaderState extends State<BouncingDotsLoader>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  final List<Animation<double>> _animations = [];

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(milliseconds: 1200),
      vsync: this,
    )..repeat();

    // Staggered animations für 3 Dots
    for (int i = 0; i < 3; i++) {
      final start = i * 0.15;
      final end = start + 0.4;
      _animations.add(
        Tween<double>(begin: 0, end: -20).animate(
          CurvedAnimation(
            parent: _controller,
            curve: Interval(start, end.clamp(0, 1), curve: Curves.easeInOut),
          ),
        ),
      );
    }
  }
```

```
  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: List.generate(3, (index) {
            return AnimatedBuilder(
              animation: _controller,
              builder: (context, child) {
                return Container(
                  margin: const EdgeInsets.symmetric(horizontal: 4),
                  child: Transform.translate(
                    offset: Offset(0, _animations[index].value),
                    child: Container(
                      width: 16,
                      height: 16,
                      decoration: BoxDecoration(
                        color: Colors.blue[400 + (index * 200)],
                        shape: BoxShape.circle,
                      ),
                    ),
                  ),
                );
              },
            );
          }),
        ),
        const SizedBox(height: 24),
        const Text(
          'Loading...',
          style: TextStyle(fontSize: 16, color: Colors.grey),
        ),
      ],
    );
  }
}
```

4.2.2.3 Aufgabe 3: Flip Card

```
class FlipCard extends StatefulWidget {
  final String question;
  final String answer;
```

```
  const FlipCard({
    super.key,
    required this.question,
    required this.answer,
  });

  @override
  State<FlipCard> createState() => _FlipCardState();
}

class _FlipCardState extends State<FlipCard>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _animation;
  bool _showFront = true;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(milliseconds: 600),
      vsync: this,
    );

    _animation = Tween<double>(begin: 0, end: 1).animate(
      CurvedAnimation(parent: _controller, curve: Curves.easeInOut),
    );

    _controller.addListener(() {
      if (_controller.value >= 0.5 && _showFront) {
        setState(() => _showFront = false);
      } else if (_controller.value < 0.5 && !_showFront) {
        setState(() => _showFront = true);
      }
    });
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  void _flip() {
    if (_controller.status == AnimationStatus.completed) {
      _controller.reverse();
    } else {
      _controller.forward();
    }
  }
```

```
@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: _flip,
    child: AnimatedBuilder(
      animation: _animation,
      builder: (context, child) {
        // Berechne Rotation (0 bis pi)
        final angle = _animation.value * 3.14159;
        // Spiegle die Rückseite
        final transform = Matrix4.identity()
          ..setEntry(3, 2, 0.001)  // Perspektive
          ..rotateY(angle);

        return Transform(
          transform: transform,
          alignment: Alignment.center,
          child: _showFront
              ? _buildCardFace(
                  widget.question,
                  Icons.help_outline,
                  Colors.blue,
                )
              : Transform(
                  transform: Matrix4.identity()..rotateY(3.14159),
                  alignment: Alignment.center,
                  child: _buildCardFace(
                    widget.answer,
                    Icons.lightbulb_outline,
                    Colors.green,
                  ),
                ),
        );
      },
    ),
  );
}

Widget _buildCardFace(String text, IconData icon, Color color) {
  return Container(
    width: 250,
    height: 350,
    decoration: BoxDecoration(
      color: color,
      borderRadius: BorderRadius.circular(16),
      boxShadow: [
        BoxShadow(
          color: color.withOpacity(0.3),
          blurRadius: 12,
          offset: const Offset(0, 6),
```

```
          ),
        ],
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(icon, size: 48, color: Colors.white),
          const SizedBox(height: 24),
          Padding(
            padding: const EdgeInsets.all(16),
            child: Text(
              text,
              textAlign: TextAlign.center,
              style: const TextStyle(
                color: Colors.white,
                fontSize: 18,
              ),
            ),
          ),
          const SizedBox(height: 24),
          const Text(
            'Tap to flip',
            style: TextStyle(color: Colors.white70, fontSize: 12),
          ),
        ],
      ),
    );
  }
}
```

4.2.2.4 Aufgabe 4: Staggered List

```
class StaggeredNotificationList extends StatefulWidget {
  const StaggeredNotificationList({super.key});

  @override
  State<StaggeredNotificationList> createState() =>
      _StaggeredNotificationListState();
}

class _StaggeredNotificationListState extends State<StaggeredNotificationList>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  final _items = [
    ('□', 'Neue Nachricht', 'Von Max Mustermann'),
    ('□', 'Erinnerung', 'Meeting in 30 Minuten'),
    ('□', 'Paket versendet', 'Ankunft morgen'),
    ('□', 'Kommentar', 'Auf deinen Beitrag'),
    ('❤', 'Gefällt mir', '5 neue Likes'),
  ];
```

```
@override
void initState() {
  super.initState();
  _controller = AnimationController(
    duration: const Duration(milliseconds: 1500),
    vsync: this,
  )..forward();
}

@override
void dispose() {
  _controller.dispose();
  super.dispose();
}

void _replay() {
  _controller.reset();
  _controller.forward();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Notifications'),
      actions: [
        IconButton(
          icon: const Icon(Icons.replay),
          onPressed: _replay,
        ),
      ],
    ),
    body: ListView.builder(
      padding: const EdgeInsets.all(16),
      itemCount: _items.length,
      itemBuilder: (context, index) {
        final intervalStart = index * 0.1;
        final intervalEnd = (intervalStart + 0.4).clamp(0.0, 1.0);

        final slideAnimation = Tween<Offset>(
          begin: const Offset(1, 0),
          end: Offset.zero,
        ).animate(
          CurvedAnimation(
            parent: _controller,
            curve: Interval(intervalStart, intervalEnd,
                curve: Curves.easeOutCubic),
          ),
        );
```

```
          final fadeAnimation = Tween<double>(begin: 0, end: 1).animate(
            CurvedAnimation(
              parent: _controller,
              curve: Interval(intervalStart, intervalEnd),
            ),
          );

          return AnimatedBuilder(
            animation: _controller,
            builder: (context, child) {
              return FadeTransition(
                opacity: fadeAnimation,
                child: SlideTransition(
                  position: slideAnimation,
                  child: child,
                ),
              );
            },
            child: Card(
              margin: const EdgeInsets.only(bottom: 12),
              child: ListTile(
                leading: Text(_items[index].$1, style: const
 ↪  TextStyle(fontSize: 24)),
                title: Text(_items[index].$2),
                subtitle: Text(_items[index].$3),
                trailing: const Icon(Icons.chevron_right),
              ),
            ),
          );
        },
      ),
    );
  }
}
```

4.2.2.5 Aufgabe 5: Hero Gallery

```
// Grid Screen
class HeroGalleryGrid extends StatelessWidget {
  const HeroGalleryGrid({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Gallery')),
      body: GridView.builder(
        padding: const EdgeInsets.all(8),
        gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
          crossAxisCount: 3,
          crossAxisSpacing: 8,
```

```
          mainAxisSpacing: 8,
        ),
        itemCount: 9,
        itemBuilder: (context, index) {
          final color = Colors.primaries[index % Colors.primaries.length];
          return GestureDetector(
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => HeroGalleryDetail(
                    index: index,
                    color: color,
                  ),
                ),
              );
            },
            child: Hero(
              tag: 'image-$index',
              child: Container(
                decoration: BoxDecoration(
                  color: color,
                  borderRadius: BorderRadius.circular(8),
                ),
                child: Center(
                  child: Text(
                    '${index + 1}',
                    style: const TextStyle(
                      color: Colors.white,
                      fontSize: 24,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
              ),
            ),
          );
        },
      ),
    );
  }
}

// Detail Screen
class HeroGalleryDetail extends StatelessWidget {
  final int index;
  final Color color;

  const HeroGalleryDetail({
    super.key,
    required this.index,
```

```
    required this.color,
  });

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.transparent,
        elevation: 0,
        title: Text('Bild ${index + 1}'),
      ),
      body: Column(
        children: [
          Expanded(
            child: Center(
              child: Hero(
                tag: 'image-$index',
                child: Container(
                  width: double.infinity,
                  margin: const EdgeInsets.all(16),
                  decoration: BoxDecoration(
                    color: color,
                    borderRadius: BorderRadius.circular(16),
                  ),
                  child: Center(
                    child: Text(
                      '${index + 1}',
                      style: const TextStyle(
                        color: Colors.white,
                        fontSize: 72,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ),
                ),
              ),
            ),
          ),
          Container(
            padding: const EdgeInsets.all(24),
            child: Text(
              'Beschreibung für Bild ${index + 1}',
              style: const TextStyle(color: Colors.white, fontSize: 16),
            ),
          ),
        ],
      ),
    );
  }
}
```

4.2.2.6 Aufgabe 6: Animated Counter

```
class AnimatedCounter extends StatefulWidget {
  const AnimatedCounter({super.key});

  @override
  State<AnimatedCounter> createState() => _AnimatedCounterState();
}

class _AnimatedCounterState extends State<AnimatedCounter> {
  int _count = 0;

  void _increment() {
    if (_count < 9999) {
      setState(() => _count++);
    }
  }

  void _decrement() {
    if (_count > 0) {
      setState(() => _count--);
    }
  }

  @override
  Widget build(BuildContext context) {
    final digits = _count.toString().padLeft(4, '0').split('');

    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Container(
          padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 16),
          decoration: BoxDecoration(
            color: Colors.grey[900],
            borderRadius: BorderRadius.circular(12),
          ),
          child: Row(
            mainAxisSize: MainAxisSize.min,
            children: digits.map((digit) {
              return _AnimatedDigit(digit: int.parse(digit));
            }).toList(),
          ),
        ),
        const SizedBox(height: 32),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            FloatingActionButton(
```

```
              onPressed: _decrement,
              heroTag: 'minus',
              child: const Icon(Icons.remove),
            ),
            const SizedBox(width: 32),
            FloatingActionButton(
              onPressed: _increment,
              heroTag: 'plus',
              child: const Icon(Icons.add),
            ),
          ],
        ),
      ],
    );
  }
}

class _AnimatedDigit extends StatelessWidget {
  final int digit;

  const _AnimatedDigit({required this.digit});

  @override
  Widget build(BuildContext context) {
    return Container(
      width: 50,
      height: 70,
      margin: const EdgeInsets.symmetric(horizontal: 4),
      decoration: BoxDecoration(
        color: Colors.grey[800],
        borderRadius: BorderRadius.circular(8),
      ),
      child: AnimatedSwitcher(
        duration: const Duration(milliseconds: 300),
        transitionBuilder: (child, animation) {
          final inAnimation = Tween<Offset>(
            begin: const Offset(0, -1),
            end: Offset.zero,
          ).animate(animation);

          return ClipRect(
            child: SlideTransition(
              position: inAnimation,
              child: child,
            ),
          );
        },
        child: Text(
          '$digit',
          key: ValueKey(digit),
          style: const TextStyle(
```

```
            color: Colors.white,
            fontSize: 40,
            fontWeight: FontWeight.bold,
            fontFamily: 'monospace',
          ),
        ),
      ),
    );
  }
}
```

4.2.2.7 Aufgabe 7: Lottie Integration

```
import 'package:lottie/lottie.dart';

class LottieDemo extends StatefulWidget {
  const LottieDemo({super.key});

  @override
  State<LottieDemo> createState() => _LottieDemoState();
}

class _LottieDemoState extends State<LottieDemo>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  bool _isLooping = false;
  double _speed = 1.0;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(vsync: this);
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  void _toggleLoop() {
    setState(() => _isLooping = !_isLooping);
    if (_isLooping) {
      _controller.repeat();
    } else {
      _controller.stop();
    }
  }

  void _setSpeed(double speed) {
```

```
    setState(() => _speed = speed);
    // Duration anpassen
    if (_controller.duration != null) {
      _controller.duration = Duration(
        milliseconds: (_controller.duration!.inMilliseconds / speed).round(),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Lottie.network(
          'https://assets5.lottiefiles.com/packages/lf20_V9t630.json',
          controller: _controller,
          width: 200,
          height: 200,
          onLoaded: (composition) {
            _controller.duration = composition.duration;
          },
        ),
        const SizedBox(height: 24),

        // Controls
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            IconButton(
              icon: const Icon(Icons.play_arrow),
              onPressed: () => _controller.forward(),
              tooltip: 'Play',
            ),
            IconButton(
              icon: const Icon(Icons.pause),
              onPressed: () => _controller.stop(),
              tooltip: 'Pause',
            ),
            IconButton(
              icon: const Icon(Icons.replay),
              onPressed: () {
                _controller.reset();
                _controller.forward();
              },
              tooltip: 'Replay',
            ),
            IconButton(
              icon: Icon(_isLooping ? Icons.repeat_on : Icons.repeat),
              onPressed: _toggleLoop,
              tooltip: 'Loop',
```

```
            ),
          ],
        ),
        const SizedBox(height: 16),

        // Progress
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 32),
          child: AnimatedBuilder(
            animation: _controller,
            builder: (context, child) {
              return Column(
                children: [
                  LinearProgressIndicator(value: _controller.value),
                  const SizedBox(height: 8),
                  Text('${(_controller.value * 100).toInt()}%'),
                ],
              );
            },
          ),
        ),
        const SizedBox(height: 24),

        // Speed Controls
        const Text('Speed:'),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [0.5, 1.0, 2.0].map((speed) {
            return Padding(
              padding: const EdgeInsets.symmetric(horizontal: 4),
              child: ChoiceChip(
                label: Text('${speed}x'),
                selected: _speed == speed,
                onSelected: (_) => _setSpeed(speed),
              ),
            );
          }).toList(),
        ),
      ],
    );
  }
}
```

4.2.2.8 Aufgabe 8: Animated Menu

```
class RadialMenu extends StatefulWidget {
  const RadialMenu({super.key});

  @override
  State<RadialMenu> createState() => _RadialMenuState();
```

```
}

class _RadialMenuState extends State<RadialMenu>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  bool _isOpen = false;

  final _menuItems = [
    (Icons.camera_alt, Colors.red),
    (Icons.folder, Colors.orange),
    (Icons.link, Colors.yellow),
    (Icons.edit, Colors.green),
    (Icons.save, Colors.blue),
    (Icons.share, Colors.purple),
  ];

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(milliseconds: 400),
      vsync: this,
    );
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  void _toggle() {
    setState(() => _isOpen = !_isOpen);
    if (_isOpen) {
      _controller.forward();
    } else {
      _controller.reverse();
    }
  }

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      width: 250,
      height: 250,
      child: Stack(
        alignment: Alignment.center,
        children: [
          // Menu Items
          ...List.generate(_menuItems.length, (index) {
            final angle = (index * 60 - 90) * 3.14159 / 180;
```

```
            final radius = 80.0;

            final intervalStart = index * 0.1;
            final intervalEnd = (intervalStart + 0.4).clamp(0.0, 1.0);

            final animation = CurvedAnimation(
              parent: _controller,
              curve: Interval(intervalStart, intervalEnd,
                  curve: Curves.easeOutBack),
            );

            return AnimatedBuilder(
              animation: animation,
              builder: (context, child) {
                final x = animation.value * radius * cos(angle);
                final y = animation.value * radius * sin(angle);

                return Transform.translate(
                  offset: Offset(x, y),
                  child: Transform.scale(
                    scale: animation.value,
                    child: child,
                  ),
                );
              },
              child: FloatingActionButton(
                heroTag: 'menu-$index',
                mini: true,
                backgroundColor: _menuItems[index].$2,
                onPressed: () {
                  _toggle();
                  // Handle action
                },
                child: Icon(_menuItems[index].$1, size: 20),
              ),
            );
          }),

          // Main Button
          FloatingActionButton(
            onPressed: _toggle,
            child: AnimatedBuilder(
              animation: _controller,
              builder: (context, child) {
                return Transform.rotate(
                  angle: _controller.value * 0.75 * 3.14159,
                  child: Icon(_isOpen ? Icons.close : Icons.add),
                );
              },
            ),
          ),
```

```
        ],
      ),
    );
  }
}

// Import für cos/sin
import 'dart:math';
```

4.2.3 Ressourcen

4.2.3.1 Offizielle Dokumentation

- Explicit Animations
- AnimationController
- Tween
- CurvedAnimation
- AnimatedBuilder
- Hero
- AnimatedIcon

4.2.3.2 Flutter Cookbook

- Animate a widget using a physics simulation
- Animate the properties of a container
- Create a staggered menu animation

4.2.3.3 Videos

- Animation Deep Dive
- AnimationController - Flutter in Focus
- Hero Widget of the Week
- Staggered Animations

4.2.3.4 Lottie

- lottie Package
- LottieFiles - Kostenlose Animationen
- Lottie Documentation

4.2.3.5 Tutorials & Artikel

- Flutter Animation Complete Guide
- Staggered Animations Tutorial
- Hero Animations Guide

4.2.3.6 Packages

- lottie - After Effects Animationen
- rive - Interaktive Animationen
- simple_animations - Vereinfachte Animation API
- spring - Physics-based Animationen
- animator - Deklarative Animationen

4.2.3.7 Tools

- Rive - Animation Editor
- LottieFiles - Animation Bibliothek
- Flow - Animation Design Tool

4.2.3.8 Zum Vertiefen

- Custom Painter Animations
- Physics-based Animations
- Route Transitions

4.3 Einheit 4.3: Unit Tests

4.3.0.1 Lernziele

Nach dieser Einheit kannst du: - Unit Tests für Dart-Code schreiben - Assertions und Matchers richtig einsetzen - Mit `mocktail` Mocks erstellen - Test-Driven Development (TDD) anwenden

4.3.0.2 1. Test-Grundlagen

Setup

```
# pubspec.yaml
dev_dependencies:
  test: ^1.24.0
  mocktail: ^1.0.0
```

Erster Test

```
// test/calculator_test.dart
import 'package:test/test.dart';
import 'package:my_app/calculator.dart';

void main() {
  test('adds two numbers', () {
    final calculator = Calculator();

    final result = calculator.add(2, 3);

    expect(result, equals(5));
  });
}
```

Tests ausführen

```
# Alle Tests
flutter test

# Bestimmte Datei
flutter test test/calculator_test.dart

# Mit Coverage
flutter test --coverage
```

4.3.0.3 2. Test-Struktur

Gruppierung mit `group`

```
void main() {
  group('Calculator', () {
    late Calculator calculator;

    setUp(() {
      calculator = Calculator();
    });

    group('add', () {
      test('returns sum of two positive numbers', () {
        expect(calculator.add(2, 3), equals(5));
      });

      test('returns sum with negative numbers', () {
        expect(calculator.add(-2, 3), equals(1));
      });

      test('returns 0 when both are 0', () {
        expect(calculator.add(0, 0), equals(0));
      });
    });

    group('divide', () {
      test('returns quotient of two numbers', () {
        expect(calculator.divide(6, 2), equals(3));
      });

      test('throws when dividing by zero', () {
        expect(
          () => calculator.divide(6, 0),
          throwsA(isA<ArgumentError>()),
        );
      });
    });
  });
}
```

setUp und tearDown

```
void main() {
  late Database db;
  late UserRepository repository;

  // Einmal vor allen Tests
  setUpAll(() async {
    db = await Database.open(':memory:');
  });

  // Vor jedem Test
```

```
  setUp(() {
    repository = UserRepository(db);
  });

  // Nach jedem Test
  tearDown(() async {
    await db.clear();
  });

  // Einmal nach allen Tests
  tearDownAll(() async {
    await db.close();
  });

  test('creates a user', () async {
    await repository.createUser('John');
    final users = await repository.getAllUsers();
    expect(users, hasLength(1));
  });
}
```

4.3.0.4 3. Assertions & Matchers

Basis-Matchers

```
// Gleichheit
expect(result, equals(5));
expect(result, 5);  // Kurzform

// Ungleichheit
expect(result, isNot(equals(3)));

// Null-Checks
expect(value, isNull);
expect(value, isNotNull);

// Boolean
expect(flag, isTrue);
expect(flag, isFalse);

// Typen
expect(value, isA<String>());
expect(value, isA<int>());
```

Numerische Matchers

```
// Vergleiche
expect(value, greaterThan(5));
expect(value, lessThan(10));
expect(value, greaterThanOrEqualTo(5));
expect(value, lessThanOrEqualTo(10));
```

```
// Bereich
expect(value, inInclusiveRange(1, 10));
expect(value, inExclusiveRange(0, 11));

// Ungefähr gleich (für Floats)
expect(3.14159, closeTo(3.14, 0.01));
```

Collection-Matchers

```
final list = [1, 2, 3, 4, 5];

// Länge
expect(list, hasLength(5));
expect(list, isEmpty);
expect(list, isNotEmpty);

// Enthält
expect(list, contains(3));
expect(list, containsAll([1, 2, 3]));
expect(list, containsAllInOrder([1, 2, 3]));

// Jedes Element
expect(list, everyElement(greaterThan(0)));
expect(list, everyElement(isA<int>()));

// Mindestens ein Element
expect(list, anyElement(equals(3)));

// Reihenfolge
expect(list, orderedEquals([1, 2, 3, 4, 5]));
expect([1, 2, 3], unorderedEquals([3, 1, 2]));
```

String-Matchers

```
final text = 'Hello World';

expect(text, startsWith('Hello'));
expect(text, endsWith('World'));
expect(text, contains('lo Wo'));
expect(text, matches(RegExp(r'Hello \w+')));

// Case-insensitive
expect(text, equalsIgnoringCase('hello world'));
expect(text, equalsIgnoringWhitespace('Hello  World'));
```

Exception-Matchers

```
// Wirft irgendeine Exception
expect(() => throw Exception(), throwsException);

// Wirft bestimmten Typ
expect(() => throw ArgumentError(), throwsArgumentError);
```

```
expect(() => throw StateError(''), throwsStateError);
expect(() => throw FormatException(), throwsFormatException);

// Custom Exception
expect(
  () => throw CustomException('message'),
  throwsA(isA<CustomException>()),
);

// Exception mit bestimmter Message
expect(
  () => throw Exception('invalid input'),
  throwsA(
    predicate<Exception>(
      (e) => e.toString().contains('invalid input'),
    ),
  ),
);

// Präziser mit having
expect(
  () => throw CustomException('error', code: 404),
  throwsA(
    isA<CustomException>()
        .having((e) => e.message, 'message', 'error')
        .having((e) => e.code, 'code', 404),
  ),
);
```

Custom Matchers

```
// Mit predicate
expect(
  user,
  predicate<User>(
    (u) => u.age >= 18 && u.isVerified,
    'is an adult and verified',
  ),
);

// Mit having (für bessere Fehlermeldungen)
expect(
  user,
  isA<User>()
      .having((u) => u.name, 'name', 'John')
      .having((u) => u.age, 'age', greaterThan(18))
      .having((u) => u.email, 'email', contains('@')),
);
```

4.3.0.5 4. Async Tests

Futures testen

```
test('fetches data from API', () async {
  final service = ApiService();

  final result = await service.fetchData();

  expect(result, isNotEmpty);
});

// Oder mit completion Matcher
test('completes successfully', () {
  final future = service.fetchData();
  expect(future, completes);
});

// Future wirft Exception
test('throws on invalid input', () {
  final future = service.fetchData('invalid');
  expect(future, throwsA(isA<ApiException>()));
});
```

Streams testen

```
test('emits values', () {
  final stream = Stream.fromIterable([1, 2, 3]);

  expect(stream, emitsInOrder([1, 2, 3]));
});

test('emits error', () {
  final stream = Stream.error(Exception('error'));

  expect(stream, emitsError(isA<Exception>()));
});

test('completes after values', () {
  final stream = Stream.fromIterable([1, 2, 3]);

  expect(
    stream,
    emitsInOrder([1, 2, 3, emitsDone]),
  );
});

// Komplexere Patterns
test('emits values matching pattern', () {
  final stream = controller.stream;

  expect(
    stream,
```

```
    emitsInOrder([
      emits(equals(1)),
      emits(greaterThan(5)),
      emitsAnyOf([equals(10), equals(20)]),
      mayEmit(anything),
      emitsDone,
    ]),
  );
});
```

4.3.0.6 5. Mocking mit mocktail

Setup

```
import 'package:mocktail/mocktail.dart';

// Mock-Klasse erstellen
class MockUserRepository extends Mock implements UserRepository {}

class MockHttpClient extends Mock implements HttpClient {}
```

Basis-Mocking

```
void main() {
  late MockUserRepository mockRepository;
  late UserService userService;

  setUp(() {
    mockRepository = MockUserRepository();
    userService = UserService(mockRepository);
  });

  test('returns user by id', () async {
    // Arrange: Mock-Verhalten definieren
    when(() => mockRepository.findById(1))
        .thenAnswer((_) async => User(id: 1, name: 'John'));

    // Act
    final user = await userService.getUser(1);

    // Assert
    expect(user.name, equals('John'));
    verify(() => mockRepository.findById(1)).called(1);
  });
}
```

when() Patterns

```
// Synchroner Return
when(() => mock.getValue()).thenReturn(42);

// Async Return
```

```
when(() => mock.fetchData()).thenAnswer((_) async => 'data');

// Exception werfen
when(() => mock.riskyOperation()).thenThrow(Exception());

// Async Exception
when(() => mock.fetchData()).thenAnswer((_) async => throw ApiException());

// Mehrere Aufrufe unterschiedlich
when(() => mock.getValue())
    .thenReturn(1)
    .thenReturn(2)
    .thenReturn(3);

// Mit Argumenten
when(() => mock.findById(any())).thenAnswer(
  (invocation) async {
    final id = invocation.positionalArguments[0] as int;
    return User(id: id, name: 'User $id');
  },
);
```

verify() Patterns

```
// Wurde aufgerufen
verify(() => mock.save(any())).called(1);

// Mehrfach aufgerufen
verify(() => mock.log(any())).called(3);

// Mindestens einmal
verify(() => mock.connect()).called(greaterThan(0));

// Nie aufgerufen
verifyNever(() => mock.delete(any()));

// Reihenfolge prüfen
verifyInOrder([
  () => mock.open(),
  () => mock.write(any()),
  () => mock.close(),
]);

// Keine weiteren Aufrufe
verifyNoMoreInteractions(mock);
```

Argument Matchers

```
// any() - beliebiger Wert
when(() => mock.findById(any())).thenReturn(user);

// any() mit Typ
```

```
when(() => mock.findById(any<int>())).thenReturn(user);

// captureAny() - Wert erfassen
verify(() => mock.save(captureAny())).captured;

// Komplexe Bedingungen
when(() => mock.findByAge(any(that: greaterThan(18))))
    .thenReturn([adult1, adult2]);
```

Argument Capturing

```
test('saves user with correct data', () async {
  when(() => mockRepository.save(any())).thenAnswer((_) async {});

  await userService.createUser('John', 25);

  final captured = verify(() => mockRepository.save(captureAny())).captured;
  final savedUser = captured.first as User;

  expect(savedUser.name, equals('John'));
  expect(savedUser.age, equals(25));
});
```

Fallback Values registrieren

```
// Für Custom-Typen bei any()
setUpAll(() {
  registerFallbackValue(User(id: 0, name: ''));
  registerFallbackValue(Uri.parse('https://example.com'));
});
```

4.3.0.7 6. Test-Driven Development (TDD)

Der TDD-Zyklus

```
1. RED:   Schreibe einen fehlschlagenden Test
2. GREEN: Schreibe minimalen Code um den Test zu bestehen
3. REFACTOR: Verbessere den Code ohne die Tests zu brechen
```

TDD Beispiel

```
// 1. RED - Test schreiben (wird fehlschlagen)
test('validates email format', () {
  final validator = EmailValidator();

  expect(validator.isValid('test@example.com'), isTrue);
  expect(validator.isValid('invalid'), isFalse);
  expect(validator.isValid(''), isFalse);
});

// 2. GREEN - Minimale Implementierung
class EmailValidator {
  bool isValid(String email) {
```

```
    if (email.isEmpty) return false;
    return email.contains('@');
  }
}

// 3. REFACTOR - Verbessern
class EmailValidator {
  static final _emailRegex = RegExp(
    r'^[a-zA-Z0-9.]+@[a-zA-Z0-9]+\.[a-zA-Z]+$',
  );

  bool isValid(String email) {
    if (email.isEmpty) return false;
    return _emailRegex.hasMatch(email);
  }
}
```

TDD für eine TodoList

```
// Schritt 1: Test für leere Liste
test('starts with empty list', () {
  final todoList = TodoList();
  expect(todoList.items, isEmpty);
});

// Schritt 2: Test für Hinzufügen
test('adds item to list', () {
  final todoList = TodoList();

  todoList.add('Buy milk');

  expect(todoList.items, hasLength(1));
  expect(todoList.items.first.text, equals('Buy milk'));
});

// Schritt 3: Test für Abhaken
test('marks item as done', () {
  final todoList = TodoList();
  todoList.add('Buy milk');

  todoList.markDone(0);

  expect(todoList.items.first.isDone, isTrue);
});

// Schritt 4: Test für Löschen
test('removes item from list', () {
  final todoList = TodoList();
  todoList.add('Buy milk');

  todoList.remove(0);
```

```
  expect(todoList.items, isEmpty);
});

// Schritt 5: Test für Filtern
test('filters completed items', () {
  final todoList = TodoList();
  todoList.add('Task 1');
  todoList.add('Task 2');
  todoList.markDone(0);

  final pending = todoList.pending;

  expect(pending, hasLength(1));
  expect(pending.first.text, equals('Task 2'));
});
```

4.3.0.8 7. Best Practices

Test-Benennung

```
// Muster: should_expectedBehavior_when_condition

test('should return null when user not found', () {});
test('should throw exception when input is invalid', () {});
test('should emit loading state when fetch starts', () {});

// Oder: Given-When-Then
test('given valid email, when validate, then returns true', () {});
```

AAA-Pattern (Arrange-Act-Assert)

```
test('calculates total with discount', () {
  // Arrange
  final cart = ShoppingCart();
  cart.add(Product('Book', 20.0));
  cart.add(Product('Pen', 5.0));
  cart.applyDiscount(0.1);  // 10%

  // Act
  final total = cart.calculateTotal();

  // Assert
  expect(total, equals(22.5));
});
```

Ein Konzept pro Test

```
// SCHLECHT: Mehrere Konzepte
test('user operations', () {
  final user = User.create('John');
  expect(user.name, equals('John'));
```

```
  user.updateAge(25);
  expect(user.age, equals(25));

  user.delete();
  expect(user.isDeleted, isTrue);
});

// GUT: Einzelne Konzepte
test('creates user with name', () {
  final user = User.create('John');
  expect(user.name, equals('John'));
});

test('updates user age', () {
  final user = User.create('John');
  user.updateAge(25);
  expect(user.age, equals(25));
});

test('deletes user', () {
  final user = User.create('John');
  user.delete();
  expect(user.isDeleted, isTrue);
});
```

Test-Doubles

Typ	Beschreibung
Dummy	Wird übergeben, aber nicht benutzt
Stub	Gibt vordefinierten Wert zurück
Mock	Prüft Interaktionen
Fake	Funktionierende Implementierung (z.B. InMemory-DB)
Spy	Echter Aufruf + Aufzeichnung

```
// Dummy
final dummyLogger = DummyLogger();
final service = Service(logger: dummyLogger);

// Stub
when(() => mockRepo.getAll()).thenReturn([user1, user2]);

// Mock
verify(() => mockRepo.save(any())).called(1);

// Fake
class FakeUserRepository implements UserRepository {
  final _users = <User>[];

  @override
```

```
  Future<List<User>> getAll() async => _users;

  @override
  Future<void> save(User user) async => _users.add(user);
}
```

4.3.0.9 8. Praktisches Beispiel: UserService Tests

```
// lib/services/user_service.dart
class UserService {
  final UserRepository _repository;
  final EmailService _emailService;

  UserService(this._repository, this._emailService);

  Future<User> createUser(String name, String email) async {
    if (name.isEmpty) throw ArgumentError('Name cannot be empty');
    if (!email.contains('@')) throw ArgumentError('Invalid email');

    final user = User(
      id: DateTime.now().millisecondsSinceEpoch,
      name: name,
      email: email,
    );

    await _repository.save(user);
    await _emailService.sendWelcome(email);

    return user;
  }

  Future<User?> findByEmail(String email) async {
    return _repository.findByEmail(email);
  }
}

// test/services/user_service_test.dart
import 'package:test/test.dart';
import 'package:mocktail/mocktail.dart';

class MockUserRepository extends Mock implements UserRepository {}
class MockEmailService extends Mock implements EmailService {}

void main() {
  late MockUserRepository mockRepository;
  late MockEmailService mockEmailService;
  late UserService userService;

  setUpAll(() {
    registerFallbackValue(User(id: 0, name: '', email: ''));
```

```
});

setUp(() {
  mockRepository = MockUserRepository();
  mockEmailService = MockEmailService();
  userService = UserService(mockRepository, mockEmailService);
});

group('createUser', () {
  test('creates user and sends welcome email', () async {
    // Arrange
    when(() => mockRepository.save(any())).thenAnswer((_) async {});
    when(() => mockEmailService.sendWelcome(any())).thenAnswer((_) async {});

    // Act
    final user = await userService.createUser('John', 'john@example.com');

    // Assert
    expect(user.name, equals('John'));
    expect(user.email, equals('john@example.com'));
    verify(() => mockRepository.save(any())).called(1);
    verify(() => mockEmailService.sendWelcome('john@example.com')).called(1);
  });

  test('throws when name is empty', () {
    expect(
      () => userService.createUser('', 'test@example.com'),
      throwsArgumentError,
    );
    verifyNever(() => mockRepository.save(any()));
  });

  test('throws when email is invalid', () {
    expect(
      () => userService.createUser('John', 'invalid'),
      throwsArgumentError,
    );
  });
});

group('findByEmail', () {
  test('returns user when found', () async {
    final expectedUser = User(id: 1, name: 'John', email: 'john@example.com');
    when(() => mockRepository.findByEmail('john@example.com'))
        .thenAnswer((_) async => expectedUser);

    final result = await userService.findByEmail('john@example.com');

    expect(result, equals(expectedUser));
  });
```

```
    test('returns null when not found', () async {
      when(() => mockRepository.findByEmail(any()))
          .thenAnswer((_) async => null);

      final result = await userService.findByEmail('unknown@example.com');

      expect(result, isNull);
    });
  });
}
```

4.3.0.10 Zusammenfassung

Konzept	Beschreibung
`test()`	Einzelner Testfall
`group()`	Tests gruppieren
`setUp()` / `tearDown()`	Vor/Nach jedem Test
`expect()`	Assertion
Matchers	`equals`, `isA`, `throwsA`, etc.
`Mock`	Fake-Objekt mit mocktail
`when()`	Mock-Verhalten definieren
`verify()`	Aufrufe prüfen

Best Practices: - Ein Konzept pro Test - AAA-Pattern (Arrange-Act-Assert) - Aussagekräftige Testnamen - Tests sollten isoliert und wiederholbar sein - TDD für robusteren Code

4.3.1 Übung

4.3.1.1 Ziel

Unit Tests für verschiedene Szenarien schreiben und TDD anwenden.

4.3.1.2 Aufgabe 1: Calculator Tests (20 min)

Schreibe Tests für eine Calculator-Klasse:

```
class Calculator {
  double add(double a, double b) => a + b;
  double subtract(double a, double b) => a - b;
  double multiply(double a, double b) => a * b;
  double divide(double a, double b) {
    if (b == 0) throw ArgumentError('Cannot divide by zero');
    return a / b;
  }
  double power(double base, int exponent) {
    if (exponent < 0) throw ArgumentError('Negative exponent not supported');
    return pow(base, exponent).toDouble();
  }
}
```

Teste: - Alle Operationen mit positiven Zahlen - Operationen mit negativen Zahlen - Division durch Null - Negativer Exponent - Edge Cases (0, sehr große Zahlen)

4.3.1.3 Aufgabe 2: Validator Tests (20 min)

Erstelle Tests für Validatoren:

```
class Validators {
  static bool isValidEmail(String email);
  static bool isValidPassword(String password);  // min 8 chars, 1 upper, 1 digit
  static bool isValidPhone(String phone);  // +49...
  static bool isValidIBAN(String iban);
}
```

Teste verschiedene Szenarien: - Gültige Eingaben - Ungültige Eingaben - Edge Cases (leer, nur Leerzeichen, Unicode) - Grenzfälle (genau 8 Zeichen, etc.)

4.3.1.4 Aufgabe 3: TDD - ShoppingCart (30 min)

Entwickle mit TDD einen Warenkorb:

```
// Schreibe ZUERST die Tests, dann die Implementierung!

// Anforderungen:
// - Produkte hinzufügen/entfernen
// - Menge ändern
// - Gesamtpreis berechnen
// - Rabatt anwenden (prozentual)
// - Mindestbestellwert prüfen
// - Versandkosten berechnen (frei ab 50€)
```

Folge dem TDD-Zyklus: 1. Schreibe einen fehlschlagenden Test 2. Implementiere minimal 3. Refactore

4.3.1.5 Aufgabe 4: Async Tests (20 min)

Teste asynchrone Operationen:

```
class WeatherService {
  final HttpClient _client;

  WeatherService(this._client);

  Future<Weather> fetchWeather(String city) async {
    final response = await _client.get('api/weather/$city');
    if (response.statusCode != 200) {
      throw ApiException('Failed to fetch weather');
    }
    return Weather.fromJson(response.body);
  }

  Stream<Weather> watchWeather(String city, Duration interval) async* {
    while (true) {
      yield await fetchWeather(city);
```

```
      await Future.delayed(interval);
    }
  }
}
```

Teste: - Erfolgreicher API-Call - Fehler-Handling - Stream emittiert korrekte Werte

4.3.1.6 Aufgabe 5: Mocking (25 min)

Schreibe Tests mit Mocks für einen AuthService:

```
abstract class AuthRepository {
  Future<User?> signIn(String email, String password);
  Future<void> signOut();
  Future<User?> getCurrentUser();
  Future<void> resetPassword(String email);
}

abstract class TokenStorage {
  Future<void> saveToken(String token);
  Future<String?> getToken();
  Future<void> deleteToken();
}

class AuthService {
  final AuthRepository _repository;
  final TokenStorage _storage;

  AuthService(this._repository, this._storage);

  Future<User> login(String email, String password) async {
    final user = await _repository.signIn(email, password);
    if (user == null) throw AuthException('Invalid credentials');
    await _storage.saveToken(user.token);
    return user;
  }

  Future<void> logout() async {
    await _repository.signOut();
    await _storage.deleteToken();
  }

  Future<bool> isLoggedIn() async {
    final token = await _storage.getToken();
    return token != null;
  }
}
```

Teste mit mocktail: - Login speichert Token - Login wirft Exception bei ungültigen Credentials - Logout löscht Token - isLoggedIn prüft Token

4.3.1.7 Aufgabe 6: Exception Tests (15 min)

Teste verschiedene Exception-Szenarien:

```
class UserService {
  void validateUser(User user) {
    if (user.name.isEmpty) {
      throw ValidationException('Name required', field: 'name');
    }
    if (user.age < 0) {
      throw ValidationException('Invalid age', field: 'age');
    }
    if (user.age < 18) {
      throw UnderageException(user.age);
    }
  }
}

class ValidationException implements Exception {
  final String message;
  final String field;
  ValidationException(this.message, {required this.field});
}

class UnderageException implements Exception {
  final int age;
  UnderageException(this.age);
}
```

Teste: - Korrekte Exception-Typen - Exception-Properties - Kein Fehler bei gültigen Daten

4.3.1.8 Aufgabe 7: Collection Tests (15 min)

Teste eine TaskList-Klasse:

```
class TaskList {
  List<Task> getAll();
  List<Task> getByStatus(TaskStatus status);
  List<Task> getOverdue();
  List<Task> getSortedByPriority();
  Map<TaskStatus, int> getStatistics();
}
```

Verwende Collection-Matchers: - `hasLength` - `contains` - `everyElement` - `orderedEquals`

4.3.1.9 Aufgabe 8: Test Coverage (20 min)

Erreiche 100% Coverage für diese Klasse:

```
class PriceCalculator {
  double calculatePrice({
    required double basePrice,
    int quantity = 1,
    double? discountPercent,
```

```
    bool isMember = false,
    String? couponCode,
  }) {
    var price = basePrice * quantity;

    if (discountPercent != null && discountPercent > 0) {
      price -= price * (discountPercent / 100);
    }

    if (isMember) {
      price *= 0.95;  // 5% Mitgliederrabatt
    }

    if (couponCode != null) {
      switch (couponCode) {
        case 'SAVE10':
          price -= 10;
          break;
        case 'HALF':
          price *= 0.5;
          break;
        case 'FREE':
          price = 0;
          break;
      }
    }

    return price < 0 ? 0 : price;
  }
}
```

Führe aus:

```
flutter test --coverage
genhtml coverage/lcov.info -o coverage/html
```

4.3.1.10 Bonus: Parameterized Tests

Schreibe parametrisierte Tests:

```
// Teste mehrere Eingabe/Ausgabe-Kombinationen elegant

void main() {
  final testCases = [
    ('test@example.com', true),
    ('invalid', false),
    ('test@test', false),
    ('a@b.c', true),
    ('', false),
  ];

  for (final (input, expected) in testCases) {
```

```
    test('isValidEmail("$input") should be $expected', () {
      expect(Validators.isValidEmail(input), equals(expected));
    });
  }
}
```

4.3.1.11 Abgabe-Checkliste

- ☐ Calculator mit allen Edge Cases getestet
- ☐ Validator-Tests für alle Szenarien
- ☐ ShoppingCart mit TDD entwickelt
- ☐ Async-Tests für Futures und Streams
- ☐ AuthService mit Mocks getestet
- ☐ Exception-Tests mit having()
- ☐ Collection-Tests mit Matchers
- ☐ 100% Coverage für PriceCalculator
- ☐ Alle Tests sind grün
- ☐ Code ist gut strukturiert

4.3.2 Lösung

4.3.2.1 Aufgabe 1: Calculator Tests

```
import 'dart:math';
import 'package:test/test.dart';

class Calculator {
  double add(double a, double b) => a + b;
  double subtract(double a, double b) => a - b;
  double multiply(double a, double b) => a * b;
  double divide(double a, double b) {
    if (b == 0) throw ArgumentError('Cannot divide by zero');
    return a / b;
  }
  double power(double base, int exponent) {
    if (exponent < 0) throw ArgumentError('Negative exponent not supported');
    return pow(base, exponent).toDouble();
  }
}

void main() {
  late Calculator calculator;

  setUp(() {
    calculator = Calculator();
  });

  group('Calculator', () {
    group('add', () {
      test('adds two positive numbers', () {
        expect(calculator.add(2, 3), equals(5));
```

```
  });

  test('adds negative numbers', () {
    expect(calculator.add(-2, -3), equals(-5));
  });

  test('adds positive and negative', () {
    expect(calculator.add(5, -3), equals(2));
  });

  test('adds zero', () {
    expect(calculator.add(5, 0), equals(5));
  });

  test('handles large numbers', () {
    expect(calculator.add(1e10, 1e10), equals(2e10));
  });
});

group('subtract', () {
  test('subtracts two positive numbers', () {
    expect(calculator.subtract(5, 3), equals(2));
  });

  test('subtracts resulting in negative', () {
    expect(calculator.subtract(3, 5), equals(-2));
  });
});

group('multiply', () {
  test('multiplies two positive numbers', () {
    expect(calculator.multiply(4, 3), equals(12));
  });

  test('multiplies with zero', () {
    expect(calculator.multiply(5, 0), equals(0));
  });

  test('multiplies negative numbers', () {
    expect(calculator.multiply(-2, -3), equals(6));
  });
});

group('divide', () {
  test('divides two numbers', () {
    expect(calculator.divide(6, 2), equals(3));
  });

  test('divides with decimal result', () {
    expect(calculator.divide(5, 2), equals(2.5));
  });
```

```
    test('throws when dividing by zero', () {
      expect(
        () => calculator.divide(5, 0),
        throwsA(isA<ArgumentError>().having(
          (e) => e.message,
          'message',
          'Cannot divide by zero',
        )),
      );
    });
  });

  group('power', () {
    test('calculates power', () {
      expect(calculator.power(2, 3), equals(8));
    });

    test('power of zero', () {
      expect(calculator.power(5, 0), equals(1));
    });

    test('throws for negative exponent', () {
      expect(
        () => calculator.power(2, -1),
        throwsArgumentError,
      );
    });
  });
});
}
```

4.3.2.2 Aufgabe 2: Validator Tests

```
import 'package:test/test.dart';

class Validators {
  static bool isValidEmail(String email) {
    if (email.isEmpty) return false;
    final regex = RegExp(r'^[\w\.-]+@[\w\.-]+\.\w{2,}$');
    return regex.hasMatch(email);
  }

  static bool isValidPassword(String password) {
    if (password.length < 8) return false;
    if (!password.contains(RegExp(r'[A-Z]'))) return false;
    if (!password.contains(RegExp(r'[0-9]'))) return false;
    return true;
  }
```

```
  static bool isValidPhone(String phone) {
    final regex = RegExp(r'^\+49\d{10,11}$');
    return regex.hasMatch(phone);
  }

  static bool isValidIBAN(String iban) {
    final cleaned = iban.replaceAll(' ', '').toUpperCase();
    if (cleaned.length < 15 || cleaned.length > 34) return false;
    if (!cleaned.startsWith(RegExp(r'^[A-Z]{2}'))) return false;
    return true;
  }
}

void main() {
  group('Validators', () {
    group('isValidEmail', () {
      test('accepts valid email', () {
        expect(Validators.isValidEmail('test@example.com'), isTrue);
        expect(Validators.isValidEmail('user.name@domain.co.uk'), isTrue);
      });

      test('rejects invalid email', () {
        expect(Validators.isValidEmail('invalid'), isFalse);
        expect(Validators.isValidEmail('test@'), isFalse);
        expect(Validators.isValidEmail('@domain.com'), isFalse);
      });

      test('rejects empty string', () {
        expect(Validators.isValidEmail(''), isFalse);
      });

      test('rejects whitespace only', () {
        expect(Validators.isValidEmail('   '), isFalse);
      });
    });

    group('isValidPassword', () {
      test('accepts valid password', () {
        expect(Validators.isValidPassword('Password1'), isTrue);
        expect(Validators.isValidPassword('MyP@ssw0rd'), isTrue);
      });

      test('rejects too short password', () {
        expect(Validators.isValidPassword('Pass1'), isFalse);
      });

      test('rejects password without uppercase', () {
        expect(Validators.isValidPassword('password1'), isFalse);
      });

      test('rejects password without digit', () {
```

```
        expect(Validators.isValidPassword('Password'), isFalse);
      });

      test('accepts exactly 8 characters', () {
        expect(Validators.isValidPassword('Passw0rd'), isTrue);
      });
    });

    group('isValidPhone', () {
      test('accepts valid German phone', () {
        expect(Validators.isValidPhone('+4917612345678'), isTrue);
      });

      test('rejects without country code', () {
        expect(Validators.isValidPhone('017612345678'), isFalse);
      });

      test('rejects wrong country code', () {
        expect(Validators.isValidPhone('+1234567890123'), isFalse);
      });
    });

    group('isValidIBAN', () {
      test('accepts valid IBAN', () {
        expect(Validators.isValidIBAN('DE89370400440532013000'), isTrue);
      });

      test('accepts IBAN with spaces', () {
        expect(Validators.isValidIBAN('DE89 3704 0044 0532 0130 00'), isTrue);
      });

      test('rejects too short IBAN', () {
        expect(Validators.isValidIBAN('DE8937040044'), isFalse);
      });
    });
  });
}
```

4.3.2.3 Aufgabe 3: TDD - ShoppingCart

```
import 'package:test/test.dart';

// Model
class Product {
  final String id;
  final String name;
  final double price;

  Product({required this.id, required this.name, required this.price});
}
```

```
class CartItem {
  final Product product;
  int quantity;

  CartItem({required this.product, this.quantity = 1});

  double get total => product.price * quantity;
}

// Implementation (nach TDD entwickelt)
class ShoppingCart {
  final List<CartItem> _items = [];
  double _discountPercent = 0;

  List<CartItem> get items => List.unmodifiable(_items);

  void addProduct(Product product, [int quantity = 1]) {
    final existing = _items.where((i) => i.product.id == product.id).firstOrNull;
    if (existing != null) {
      existing.quantity += quantity;
    } else {
      _items.add(CartItem(product: product, quantity: quantity));
    }
  }

  void removeProduct(String productId) {
    _items.removeWhere((i) => i.product.id == productId);
  }

  void updateQuantity(String productId, int quantity) {
    if (quantity <= 0) {
      removeProduct(productId);
      return;
    }
    final item = _items.where((i) => i.product.id == productId).firstOrNull;
    item?.quantity = quantity;
  }

  void applyDiscount(double percent) {
    _discountPercent = percent.clamp(0, 100);
  }

  double get subtotal => _items.fold(0, (sum, item) => sum + item.total);

  double get discount => subtotal * (_discountPercent / 100);

  double get total => subtotal - discount;

  bool get meetsMinimumOrder => subtotal >= 10;
```

```
  double get shippingCost => subtotal >= 50 ? 0 : 4.99;

  double get grandTotal => total + shippingCost;
}

void main() {
  late ShoppingCart cart;
  late Product book;
  late Product pen;

  setUp(() {
    cart = ShoppingCart();
    book = Product(id: '1', name: 'Book', price: 20.0);
    pen = Product(id: '2', name: 'Pen', price: 5.0);
  });

  group('ShoppingCart', () {
    test('starts empty', () {
      expect(cart.items, isEmpty);
      expect(cart.subtotal, equals(0));
    });

    group('addProduct', () {
      test('adds product to cart', () {
        cart.addProduct(book);

        expect(cart.items, hasLength(1));
        expect(cart.items.first.product.name, equals('Book'));
      });

      test('increases quantity for existing product', () {
        cart.addProduct(book);
        cart.addProduct(book);

        expect(cart.items, hasLength(1));
        expect(cart.items.first.quantity, equals(2));
      });

      test('adds with custom quantity', () {
        cart.addProduct(book, 3);

        expect(cart.items.first.quantity, equals(3));
      });
    });

    group('removeProduct', () {
      test('removes product from cart', () {
        cart.addProduct(book);
        cart.removeProduct('1');

        expect(cart.items, isEmpty);
```

```
    });

    test('does nothing for non-existent product', () {
      cart.addProduct(book);
      cart.removeProduct('999');

      expect(cart.items, hasLength(1));
    });
  });

  group('updateQuantity', () {
    test('updates quantity', () {
      cart.addProduct(book);
      cart.updateQuantity('1', 5);

      expect(cart.items.first.quantity, equals(5));
    });

    test('removes product when quantity is 0', () {
      cart.addProduct(book);
      cart.updateQuantity('1', 0);

      expect(cart.items, isEmpty);
    });
  });

  group('pricing', () {
    test('calculates subtotal', () {
      cart.addProduct(book);
      cart.addProduct(pen, 2);

      expect(cart.subtotal, equals(30.0));  // 20 + 2*5
    });

    test('applies discount', () {
      cart.addProduct(book);  // 20.0
      cart.applyDiscount(10);  // 10%

      expect(cart.discount, equals(2.0));
      expect(cart.total, equals(18.0));
    });

    test('clamps discount to valid range', () {
      cart.addProduct(book);
      cart.applyDiscount(150);  // Over 100%

      expect(cart.total, equals(0));  // Clamped to 100%
    });
  });

  group('minimumOrder', () {
```

```
      test('returns false below minimum', () {
        cart.addProduct(pen);  // 5.0

        expect(cart.meetsMinimumOrder, isFalse);
      });

      test('returns true at minimum', () {
        cart.addProduct(pen, 2);  // 10.0

        expect(cart.meetsMinimumOrder, isTrue);
      });
    });

    group('shipping', () {
      test('charges shipping below 50', () {
        cart.addProduct(book);  // 20.0

        expect(cart.shippingCost, equals(4.99));
      });

      test('free shipping at 50 or above', () {
        cart.addProduct(book, 3);  // 60.0

        expect(cart.shippingCost, equals(0));
      });

      test('calculates grand total', () {
        cart.addProduct(book);  // 20.0

        expect(cart.grandTotal, equals(24.99));  // 20 + 4.99
      });
    });
  });
}
```

4.3.2.4 Aufgabe 4: Async Tests

```
import 'package:test/test.dart';
import 'package:mocktail/mocktail.dart';

class HttpResponse {
  final int statusCode;
  final String body;
  HttpResponse(this.statusCode, this.body);
}

abstract class HttpClient {
  Future<HttpResponse> get(String url);
}
```

```
class Weather {
  final String city;
  final double temperature;

  Weather({required this.city, required this.temperature});

  factory Weather.fromJson(String json) {
    return Weather(city: 'Berlin', temperature: 20.0);
  }
}

class ApiException implements Exception {
  final String message;
  ApiException(this.message);
}

class WeatherService {
  final HttpClient _client;

  WeatherService(this._client);

  Future<Weather> fetchWeather(String city) async {
    final response = await _client.get('api/weather/$city');
    if (response.statusCode != 200) {
      throw ApiException('Failed to fetch weather');
    }
    return Weather.fromJson(response.body);
  }
}

class MockHttpClient extends Mock implements HttpClient {}

void main() {
  late MockHttpClient mockClient;
  late WeatherService service;

  setUp(() {
    mockClient = MockHttpClient();
    service = WeatherService(mockClient);
  });

  group('WeatherService', () {
    group('fetchWeather', () {
      test('returns weather on success', () async {
        when(() => mockClient.get(any())).thenAnswer(
          (_) async => HttpResponse(200, '{"city":"Berlin","temp":20}'),
        );

        final weather = await service.fetchWeather('Berlin');

        expect(weather.city, equals('Berlin'));
```

```
        verify(() => mockClient.get('api/weather/Berlin')).called(1);
      });

      test('throws ApiException on error', () async {
        when(() => mockClient.get(any())).thenAnswer(
          (_) async => HttpResponse(500, 'Server Error'),
        );

        expect(
          () => service.fetchWeather('Berlin'),
          throwsA(isA<ApiException>()),
        );
      });

      test('completes successfully', () {
        when(() => mockClient.get(any())).thenAnswer(
          (_) async => HttpResponse(200, '{}'),
        );

        expect(service.fetchWeather('Berlin'), completes);
      });
    });
  });
}
```

4.3.2.5 Aufgabe 5: Mocking AuthService

```
import 'package:test/test.dart';
import 'package:mocktail/mocktail.dart';

class User {
  final String id;
  final String email;
  final String token;

  User({required this.id, required this.email, required this.token});
}

abstract class AuthRepository {
  Future<User?> signIn(String email, String password);
  Future<void> signOut();
}

abstract class TokenStorage {
  Future<void> saveToken(String token);
  Future<String?> getToken();
  Future<void> deleteToken();
}

class AuthException implements Exception {
```

```
  final String message;
  AuthException(this.message);
}

class AuthService {
  final AuthRepository _repository;
  final TokenStorage _storage;

  AuthService(this._repository, this._storage);

  Future<User> login(String email, String password) async {
    final user = await _repository.signIn(email, password);
    if (user == null) throw AuthException('Invalid credentials');
    await _storage.saveToken(user.token);
    return user;
  }

  Future<void> logout() async {
    await _repository.signOut();
    await _storage.deleteToken();
  }

  Future<bool> isLoggedIn() async {
    final token = await _storage.getToken();
    return token != null;
  }
}

class MockAuthRepository extends Mock implements AuthRepository {}
class MockTokenStorage extends Mock implements TokenStorage {}

void main() {
  late MockAuthRepository mockRepository;
  late MockTokenStorage mockStorage;
  late AuthService authService;

  setUp(() {
    mockRepository = MockAuthRepository();
    mockStorage = MockTokenStorage();
    authService = AuthService(mockRepository, mockStorage);
  });

  group('AuthService', () {
    group('login', () {
      test('saves token on successful login', () async {
        final user = User(id: '1', email: 'test@test.com', token: 'abc123');
        when(() => mockRepository.signIn(any(), any()))
            .thenAnswer((_) async => user);
        when(() => mockStorage.saveToken(any()))
            .thenAnswer((_) async {});
```

```
      final result = await authService.login('test@test.com', 'password');

      expect(result.email, equals('test@test.com'));
      verify(() => mockStorage.saveToken('abc123')).called(1);
    });

    test('throws AuthException on invalid credentials', () async {
      when(() => mockRepository.signIn(any(), any()))
          .thenAnswer((_) async => null);

      expect(
        () => authService.login('test@test.com', 'wrong'),
        throwsA(isA<AuthException>().having(
          (e) => e.message,
          'message',
          'Invalid credentials',
        )),
      );

      verifyNever(() => mockStorage.saveToken(any()));
    });
  });

  group('logout', () {
    test('signs out and deletes token', () async {
      when(() => mockRepository.signOut()).thenAnswer((_) async {});
      when(() => mockStorage.deleteToken()).thenAnswer((_) async {});

      await authService.logout();

      verifyInOrder([
        () => mockRepository.signOut(),
        () => mockStorage.deleteToken(),
      ]);
    });
  });

  group('isLoggedIn', () {
    test('returns true when token exists', () async {
      when(() => mockStorage.getToken()).thenAnswer((_) async => 'token');

      final result = await authService.isLoggedIn();

      expect(result, isTrue);
    });

    test('returns false when no token', () async {
      when(() => mockStorage.getToken()).thenAnswer((_) async => null);

      final result = await authService.isLoggedIn();
```

```
        expect(result, isFalse);
      });
    });
  });
}
```

4.3.2.6 Aufgabe 6: Exception Tests

```
import 'package:test/test.dart';

class User {
  final String name;
  final int age;

  User({required this.name, required this.age});
}

class ValidationException implements Exception {
  final String message;
  final String field;
  ValidationException(this.message, {required this.field});
}

class UnderageException implements Exception {
  final int age;
  UnderageException(this.age);
}

class UserService {
  void validateUser(User user) {
    if (user.name.isEmpty) {
      throw ValidationException('Name required', field: 'name');
    }
    if (user.age < 0) {
      throw ValidationException('Invalid age', field: 'age');
    }
    if (user.age < 18) {
      throw UnderageException(user.age);
    }
  }
}

void main() {
  late UserService service;

  setUp(() {
    service = UserService();
  });

  group('UserService.validateUser', () {
```

```
    test('throws ValidationException for empty name', () {
      final user = User(name: '', age: 25);

      expect(
        () => service.validateUser(user),
        throwsA(
          isA<ValidationException>()
              .having((e) => e.message, 'message', 'Name required')
              .having((e) => e.field, 'field', 'name'),
        ),
      );
    });

    test('throws ValidationException for negative age', () {
      final user = User(name: 'John', age: -1);

      expect(
        () => service.validateUser(user),
        throwsA(
          isA<ValidationException>()
              .having((e) => e.field, 'field', 'age'),
        ),
      );
    });

    test('throws UnderageException for age under 18', () {
      final user = User(name: 'John', age: 16);

      expect(
        () => service.validateUser(user),
        throwsA(
          isA<UnderageException>()
              .having((e) => e.age, 'age', 16),
        ),
      );
    });

    test('does not throw for valid user', () {
      final user = User(name: 'John', age: 25);

      expect(() => service.validateUser(user), returnsNormally);
    });

    test('validates exactly 18 years old', () {
      final user = User(name: 'John', age: 18);

      expect(() => service.validateUser(user), returnsNormally);
    });
  });
}
```

4.3.2.7 Aufgabe 8: 100% Coverage

```
import 'package:test/test.dart';

class PriceCalculator {
  double calculatePrice({
    required double basePrice,
    int quantity = 1,
    double? discountPercent,
    bool isMember = false,
    String? couponCode,
  }) {
    var price = basePrice * quantity;

    if (discountPercent != null && discountPercent > 0) {
      price -= price * (discountPercent / 100);
    }

    if (isMember) {
      price *= 0.95;
    }

    if (couponCode != null) {
      switch (couponCode) {
        case 'SAVE10':
          price -= 10;
          break;
        case 'HALF':
          price *= 0.5;
          break;
        case 'FREE':
          price = 0;
          break;
      }
    }

    return price < 0 ? 0 : price;
  }
}

void main() {
  late PriceCalculator calculator;

  setUp(() {
    calculator = PriceCalculator();
  });

  group('PriceCalculator', () {
    test('calculates base price', () {
      final price = calculator.calculatePrice(basePrice: 100);
      expect(price, equals(100));
```

```
  });

  test('multiplies by quantity', () {
    final price = calculator.calculatePrice(basePrice: 10, quantity: 5);
    expect(price, equals(50));
  });

  test('applies discount percent', () {
    final price = calculator.calculatePrice(
      basePrice: 100,
      discountPercent: 20,
    );
    expect(price, equals(80));
  });

  test('ignores null discount', () {
    final price = calculator.calculatePrice(
      basePrice: 100,
      discountPercent: null,
    );
    expect(price, equals(100));
  });

  test('ignores zero discount', () {
    final price = calculator.calculatePrice(
      basePrice: 100,
      discountPercent: 0,
    );
    expect(price, equals(100));
  });

  test('applies member discount', () {
    final price = calculator.calculatePrice(
      basePrice: 100,
      isMember: true,
    );
    expect(price, equals(95));
  });

  test('applies SAVE10 coupon', () {
    final price = calculator.calculatePrice(
      basePrice: 100,
      couponCode: 'SAVE10',
    );
    expect(price, equals(90));
  });

  test('applies HALF coupon', () {
    final price = calculator.calculatePrice(
      basePrice: 100,
      couponCode: 'HALF',
```

```
      );
      expect(price, equals(50));
    });

    test('applies FREE coupon', () {
      final price = calculator.calculatePrice(
        basePrice: 100,
        couponCode: 'FREE',
      );
      expect(price, equals(0));
    });

    test('ignores unknown coupon', () {
      final price = calculator.calculatePrice(
        basePrice: 100,
        couponCode: 'INVALID',
      );
      expect(price, equals(100));
    });

    test('returns 0 for negative result', () {
      final price = calculator.calculatePrice(
        basePrice: 5,
        couponCode: 'SAVE10',
      );
      expect(price, equals(0));
    });

    test('combines all discounts', () {
      final price = calculator.calculatePrice(
        basePrice: 100,
        quantity: 2,
        discountPercent: 10,
        isMember: true,
        couponCode: 'SAVE10',
      );
      // 200 - 20 (10%) = 180
      // 180 * 0.95 (member) = 171
      // 171 - 10 (coupon) = 161
      expect(price, equals(161));
    });
  });
}
```

4.3.3 Ressourcen

4.3.3.1 Offizielle Dokumentation

- Testing Flutter Apps
- An introduction to unit testing
- test Package

- Dart Testing

4.3.3.2 Packages

- test - Dart Test Framework
- mocktail - Mocking (empfohlen)
- mockito - Klassisches Mocking
- fake_async - Zeit-Kontrolle in Tests
- clock - Testbare Zeit

4.3.3.3 Tutorials & Artikel

- Flutter Testing Guide
- TDD in Flutter
- Mocktail Tutorial
- Testing Best Practices

4.3.3.4 Videos

- Flutter Testing for Beginners
- Unit Testing in Flutter
- TDD in Flutter

4.3.3.5 Bücher

- Test-Driven Development by Example - Kent Beck
- Clean Code - Robert C. Martin

4.3.3.6 Tools

- Very Good CLI - Projekt-Templates mit Tests
- Coverage - Code Coverage
- lcov - Coverage Reports

4.3.3.7 Code Coverage

```
# Coverage generieren
flutter test --coverage

# HTML-Report (benötigt lcov)
genhtml coverage/lcov.info -o coverage/html

# Report öffnen
open coverage/html/index.html
```

4.3.3.8 Zum Vertiefen

- Property-based Testing
- Golden Tests
- Integration Tests

4.4 Einheit 4.4: Widget & Integration Tests

4.4.0.1 Lernziele

Nach dieser Einheit kannst du: - Widget Tests mit `flutter_test` schreiben - `WidgetTester` und Finder verwenden - Integration Tests erstellen - Code Coverage analysieren

4.4.0.2 1. Widget Test Grundlagen

Setup

```
# pubspec.yaml (bereits enthalten)
dev_dependencies:
  flutter_test:
    sdk: flutter
```

Erster Widget Test

```
// test/widget_test.dart
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:my_app/counter_widget.dart';

void main() {
  testWidgets('Counter increments', (WidgetTester tester) async {
    // Widget aufbauen
    await tester.pumpWidget(const MaterialApp(
      home: CounterWidget(),
    ));

    // Initialen Zustand prüfen
    expect(find.text('0'), findsOneWidget);

    // Button drücken
    await tester.tap(find.byIcon(Icons.add));

    // Frame rebuilden
    await tester.pump();

    // Neuen Zustand prüfen
    expect(find.text('1'), findsOneWidget);
  });
}
```

pump vs pumpAndSettle

```
// pump() - Ein einzelner Frame
await tester.pump();

// pump(duration) - Ein Frame nach Duration
await tester.pump(const Duration(milliseconds: 100));

// pumpAndSettle() - Alle Frames bis keine Animationen mehr laufen
```

```
await tester.pumpAndSettle();

// pumpAndSettle mit Timeout
await tester.pumpAndSettle(const Duration(seconds: 5));
```

4.4.0.3 2. Finder

Text finden

```
// Exakter Text
find.text('Hello')

// Text enthält
find.textContaining('Hello')

// Mit Regex
find.textContaining(RegExp(r'Hello \w+'))

// Rich Text
find.richText('Hello')
```

Widget-Typen finden

```
// Nach Typ
find.byType(ElevatedButton)
find.byType(TextField)
find.byType(CircularProgressIndicator)

// Nach Subtyp
find.bySubtype<StatelessWidget>()
```

Icons und Keys finden

```
// Icon
find.byIcon(Icons.add)
find.byIcon(Icons.delete)

// Key
find.byKey(const Key('submit_button'))
find.byKey(const ValueKey('item_1'))
```

Widgets finden

```
// Nach Widget-Instanz
find.byWidget(myWidget)

// Nach Semantics Label
find.bySemanticsLabel('Add item')

// Nach Tooltip
find.byTooltip('Delete')
```

Kombinierte Finder

```
// Descendant (Kind von)
find.descendant(
  of: find.byType(Card),
  matching: find.text('Title'),
)

// Ancestor (Eltern von)
find.ancestor(
  of: find.text('Title'),
  matching: find.byType(ListTile),
)

// Erstes/Letztes von mehreren
find.byType(ListTile).first
find.byType(ListTile).last
find.byType(ListTile).at(2)
```

Finder Matchers

```
// Genau ein Widget
expect(find.text('Hello'), findsOneWidget);

// Kein Widget
expect(find.text('Error'), findsNothing);

// Mindestens ein Widget
expect(find.byType(ListTile), findsWidgets);

// Genau N Widgets
expect(find.byType(ListTile), findsNWidgets(3));

// Mindestens N Widgets
expect(find.byType(ListTile), findsAtLeastNWidgets(2));
```

4.4.0.4 3. Interaktionen

Tap

```
// Einfacher Tap
await tester.tap(find.byType(ElevatedButton));
await tester.pump();

// Tap an Position
await tester.tapAt(const Offset(100, 200));

// Double Tap
await tester.tap(find.text('Item'));
await tester.pump(const Duration(milliseconds: 100));
await tester.tap(find.text('Item'));

// Long Press
await tester.longPress(find.text('Item'));
```

```
await tester.pump();
```

Text eingeben

```
// Text eingeben
await tester.enterText(find.byType(TextField), 'Hello World');
await tester.pump();

// Text in bestimmtes Feld
await tester.enterText(
  find.byKey(const Key('email_field')),
  'test@example.com',
);
```

Scrollen

```
// Drag/Scroll
await tester.drag(find.byType(ListView), const Offset(0, -300));
await tester.pumpAndSettle();

// Fling (schnelles Scrollen)
await tester.fling(find.byType(ListView), const Offset(0, -500), 1000);
await tester.pumpAndSettle();

// Zu Widget scrollen
await tester.scrollUntilVisible(
  find.text('Item 50'),
  500.0,
  scrollable: find.byType(Scrollable),
);
```

Gesten

```
// Swipe
await tester.drag(find.byType(Dismissible), const Offset(500, 0));
await tester.pumpAndSettle();

// Pinch/Zoom (zwei Finger)
final center = tester.getCenter(find.byType(InteractiveViewer));
await tester.startGesture(center - const Offset(50, 0));
final gesture2 = await tester.startGesture(center + const Offset(50, 0));
await gesture2.moveBy(const Offset(50, 0));
await tester.pumpAndSettle();
```

4.4.0.5 4. Widget-State prüfen

State auslesen

```
// State eines StatefulWidget
final state = tester.state<CounterWidgetState>(find.byType(CounterWidget));
expect(state.counter, equals(5));

// Widget-Properties
```

```
final widget = tester.widget<Text>(find.text('Hello'));
expect(widget.style?.color, equals(Colors.red));

// Element
final element = tester.element(find.byType(MyWidget));
```

Form-Werte prüfen

```
testWidgets('form validation', (tester) async {
  await tester.pumpWidget(MaterialApp(home: MyForm()));

  // Leeres Feld submitten
  await tester.tap(find.text('Submit'));
  await tester.pump();

  // Fehlermeldung prüfen
  expect(find.text('Required field'), findsOneWidget);

  // Wert eingeben
  await tester.enterText(find.byType(TextField), 'test@email.com');
  await tester.tap(find.text('Submit'));
  await tester.pump();

  // Keine Fehlermeldung mehr
  expect(find.text('Required field'), findsNothing);
});
```

Checkbox/Switch prüfen

```
// Checkbox-Wert
final checkbox = tester.widget<Checkbox>(find.byType(Checkbox));
expect(checkbox.value, isTrue);

// Switch-Wert
final switchWidget = tester.widget<Switch>(find.byType(Switch));
expect(switchWidget.value, isFalse);
```

4.4.0.6 5. Async und Loading States

FutureBuilder testen

```
testWidgets('shows loading then data', (tester) async {
  await tester.pumpWidget(MaterialApp(
    home: FutureBuilder<String>(
      future: Future.delayed(
        const Duration(seconds: 1),
        () => 'Data loaded',
      ),
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const CircularProgressIndicator();
        }
```

```
        return Text(snapshot.data ?? '');
      },
    ),
  ));

  // Loading State
  expect(find.byType(CircularProgressIndicator), findsOneWidget);

  // Zeit vorspulen
  await tester.pump(const Duration(seconds: 1));

  // Daten angezeigt
  expect(find.text('Data loaded'), findsOneWidget);
  expect(find.byType(CircularProgressIndicator), findsNothing);
});
```

Mit Mocks

```
class MockApiService extends Mock implements ApiService {}

testWidgets('displays user list', (tester) async {
  final mockService = MockApiService();
  when(() => mockService.getUsers()).thenAnswer(
    (_) async => [User(name: 'John'), User(name: 'Jane')],
  );

  await tester.pumpWidget(
    MaterialApp(
      home: Provider<ApiService>.value(
        value: mockService,
        child: const UserListScreen(),
      ),
    ),
  );

  // Warte auf Future
  await tester.pumpAndSettle();

  expect(find.text('John'), findsOneWidget);
  expect(find.text('Jane'), findsOneWidget);
});
```

4.4.0.7 6. Navigation testen

Navigator.push

```
testWidgets('navigates to detail screen', (tester) async {
  await tester.pumpWidget(MaterialApp(
    home: const HomeScreen(),
    routes: {
      '/detail': (context) => const DetailScreen(),
    },
```

```
  ));

  // Tap auf Item
  await tester.tap(find.text('View Details'));
  await tester.pumpAndSettle();

  // Prüfe Navigation
  expect(find.byType(DetailScreen), findsOneWidget);
  expect(find.byType(HomeScreen), findsNothing);
});
```

Navigator.pop mit Ergebnis

```
testWidgets('returns result from dialog', (tester) async {
  String? result;

  await tester.pumpWidget(MaterialApp(
    home: Builder(
      builder: (context) => ElevatedButton(
        onPressed: () async {
          result = await Navigator.push<String>(
            context,
            MaterialPageRoute(
              builder: (context) => SelectionScreen(),
            ),
          );
        },
        child: const Text('Open'),
      ),
    ),
  ));

  await tester.tap(find.text('Open'));
  await tester.pumpAndSettle();

  await tester.tap(find.text('Option A'));
  await tester.pumpAndSettle();

  expect(result, equals('Option A'));
});
```

4.4.0.8 7. Golden Tests

Snapshot-Vergleiche

```
testWidgets('matches golden file', (tester) async {
  await tester.pumpWidget(const MaterialApp(
    home: MyWidget(),
  ));

  await expectLater(
    find.byType(MyWidget),
```

```
    matchesGoldenFile('goldens/my_widget.png'),
  );
});
```

Golden Tests ausführen

```
# Goldens erstellen/aktualisieren
flutter test --update-goldens

# Goldens prüfen
flutter test
```

Best Practices für Goldens

```
testWidgets('card golden test', (tester) async {
  // Feste Größe für konsistente Screenshots
  tester.binding.window.physicalSizeTestValue = const Size(400, 300);
  tester.binding.window.devicePixelRatioTestValue = 1.0;

  await tester.pumpWidget(const MaterialApp(
    home: Scaffold(
      body: Center(child: ProductCard()),
    ),
  ));

  await expectLater(
    find.byType(ProductCard),
    matchesGoldenFile('goldens/product_card.png'),
  );

  // Cleanup
  addTearDown(tester.binding.window.clearPhysicalSizeTestValue);
});
```

4.4.0.9 8. Integration Tests

Setup

```
# pubspec.yaml
dev_dependencies:
  integration_test:
    sdk: flutter
```

Test erstellen

```
// integration_test/app_test.dart
import 'package:flutter_test/flutter_test.dart';
import 'package:integration_test/integration_test.dart';
import 'package:my_app/main.dart' as app;

void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();
```

```
  group('end-to-end test', () {
    testWidgets('complete user flow', (tester) async {
      app.main();
      await tester.pumpAndSettle();

      // Login
      await tester.enterText(
        find.byKey(const Key('email')),
        'test@example.com',
      );
      await tester.enterText(
        find.byKey(const Key('password')),
        'password123',
      );
      await tester.tap(find.text('Login'));
      await tester.pumpAndSettle();

      // Verify home screen
      expect(find.text('Welcome'), findsOneWidget);

      // Navigate to profile
      await tester.tap(find.byIcon(Icons.person));
      await tester.pumpAndSettle();

      expect(find.text('test@example.com'), findsOneWidget);
    });
  });
}
```

Integration Tests ausführen

```
# Auf verbundenem Gerät/Emulator
flutter test integration_test/app_test.dart

# Auf spezifischem Gerät
flutter test integration_test/app_test.dart -d <device_id>

# Mit Screenshot bei Fehler
flutter drive --driver=test_driver/integration_test.dart                    ↪
 ↪  --target=integration_test/app_test.dart
```

4.4.0.10 9. Code Coverage

Coverage generieren

```
# Tests mit Coverage
flutter test --coverage

# HTML-Report erstellen (benötigt lcov)
genhtml coverage/lcov.info -o coverage/html

# Report öffnen
```

```
open coverage/html/index.html  # macOS
xdg-open coverage/html/index.html  # Linux
```

Coverage in CI

```
# .github/workflows/test.yml
name: Tests

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: subosito/flutter-action@v2
      - run: flutter pub get
      - run: flutter test --coverage
      - uses: codecov/codecov-action@v3
        with:
          file: coverage/lcov.info
```

Minimum Coverage erzwingen

```
// test/coverage_test.dart
import 'dart:io';

void main() {
  test('coverage is above threshold', () {
    final lcov = File('coverage/lcov.info').readAsStringSync();
    final lines = lcov.split('\n');

    int linesFound = 0;
    int linesHit = 0;

    for (final line in lines) {
      if (line.startsWith('LF:')) {
        linesFound += int.parse(line.substring(3));
      }
      if (line.startsWith('LH:')) {
        linesHit += int.parse(line.substring(3));
      }
    }

    final coverage = linesHit / linesFound * 100;
    expect(coverage, greaterThanOrEqualTo(80));
  });
}
```

4.4.0.11 10. Praktisches Beispiel: Login Screen Tests

```
// lib/screens/login_screen.dart
class LoginScreen extends StatefulWidget {
  final AuthService authService;

  const LoginScreen({super.key, required this.authService});

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  bool _isLoading = false;
  String? _error;

  Future<void> _login() async {
    if (!_formKey.currentState!.validate()) return;

    setState(() {
      _isLoading = true;
      _error = null;
    });

    try {
      await widget.authService.login(
        _emailController.text,
        _passwordController.text,
      );
      if (mounted) {
        Navigator.pushReplacementNamed(context, '/home');
      }
    } catch (e) {
      setState(() => _error = e.toString());
    } finally {
      if (mounted) {
        setState(() => _isLoading = false);
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Login')),
      body: Form(
        key: _formKey,
        child: Padding(
```

```
          padding: const EdgeInsets.all(16),
          child: Column(
            children: [
              TextFormField(
                key: const Key('email_field'),
                controller: _emailController,
                decoration: const InputDecoration(labelText: 'Email'),
                validator: (v) =>
                    v?.contains('@') == true ? null : 'Invalid email',
              ),
              TextFormField(
                key: const Key('password_field'),
                controller: _passwordController,
                obscureText: true,
                decoration: const InputDecoration(labelText: 'Password'),
                validator: (v) =>
                    (v?.length ?? 0) >= 6 ? null : 'Min 6 characters',
              ),
              if (_error != null)
                Text(_error!, style: const TextStyle(color: Colors.red)),
              const SizedBox(height: 16),
              _isLoading
                  ? const CircularProgressIndicator()
                  : ElevatedButton(
                      key: const Key('login_button'),
                      onPressed: _login,
                      child: const Text('Login'),
                    ),
            ],
          ),
        ),
      ),
    );
  }
}

// test/screens/login_screen_test.dart
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:mocktail/mocktail.dart';

class MockAuthService extends Mock implements AuthService {}
class MockNavigatorObserver extends Mock implements NavigatorObserver {}

void main() {
  late MockAuthService mockAuthService;
  late MockNavigatorObserver mockObserver;

  setUp(() {
    mockAuthService = MockAuthService();
    mockObserver = MockNavigatorObserver();
```

```
});

Widget createWidget() {
  return MaterialApp(
    home: LoginScreen(authService: mockAuthService),
    routes: {
      '/home': (context) => const Scaffold(body: Text('Home')),
    },
    navigatorObservers: [mockObserver],
  );
}

group('LoginScreen', () {
  testWidgets('renders login form', (tester) async {
    await tester.pumpWidget(createWidget());

    expect(find.text('Email'), findsOneWidget);
    expect(find.text('Password'), findsOneWidget);
    expect(find.text('Login'), findsOneWidget);
  });

  testWidgets('shows validation errors', (tester) async {
    await tester.pumpWidget(createWidget());

    await tester.tap(find.byKey(const Key('login_button')));
    await tester.pump();

    expect(find.text('Invalid email'), findsOneWidget);
    expect(find.text('Min 6 characters'), findsOneWidget);
  });

  testWidgets('shows loading indicator during login', (tester) async {
    when(() => mockAuthService.login(any(), any()))
        .thenAnswer((_) => Future.delayed(const Duration(seconds: 1)));

    await tester.pumpWidget(createWidget());

    await tester.enterText(
      find.byKey(const Key('email_field')),
      'test@test.com',
    );
    await tester.enterText(
      find.byKey(const Key('password_field')),
      'password123',
    );
    await tester.tap(find.byKey(const Key('login_button')));
    await tester.pump();

    expect(find.byType(CircularProgressIndicator), findsOneWidget);
  });
```

```
    testWidgets('navigates to home on success', (tester) async {
      when(() => mockAuthService.login(any(), any()))
          .thenAnswer((_) async {});

      await tester.pumpWidget(createWidget());

      await tester.enterText(
        find.byKey(const Key('email_field')),
        'test@test.com',
      );
      await tester.enterText(
        find.byKey(const Key('password_field')),
        'password123',
      );
      await tester.tap(find.byKey(const Key('login_button')));
      await tester.pumpAndSettle();

      expect(find.text('Home'), findsOneWidget);
    });

    testWidgets('shows error on failed login', (tester) async {
      when(() => mockAuthService.login(any(), any()))
          .thenThrow(Exception('Invalid credentials'));

      await tester.pumpWidget(createWidget());

      await tester.enterText(
        find.byKey(const Key('email_field')),
        'test@test.com',
      );
      await tester.enterText(
        find.byKey(const Key('password_field')),
        'wrongpassword',
      );
      await tester.tap(find.byKey(const Key('login_button')));
      await tester.pumpAndSettle();

      expect(find.textContaining('Invalid credentials'), findsOneWidget);
    });
  });
}
```

4.4.0.12 Zusammenfassung

Konzept	Beschreibung
`testWidgets`	Widget Test definieren
`pumpWidget`	Widget aufbauen
`pump`	Einen Frame rendern
`pumpAndSettle`	Alle Animationen abwarten
`find.text()`	Text-Finder

Konzept	Beschreibung
`find.byType()`	Typ-Finder
`find.byKey()`	Key-Finder
`findsOneWidget`	Genau ein Match
`tester.tap()`	Tap simulieren
`tester.enterText()`	Text eingeben
`matchesGoldenFile`	Screenshot-Vergleich
Integration Tests	End-to-End Tests

Best Practices: - Keys für wichtige Widgets verwenden - Mocks für externe Abhängigkeiten - `pumpAndSettle` für Animationen - Golden Tests für UI-Regression - CI-Integration für automatische Tests

4.4.1 Übung

4.4.1.1 Ziel

Widget Tests für verschiedene UI-Komponenten schreiben und Integration Tests erstellen.

4.4.1.2 Aufgabe 1: Counter Widget Tests (20 min)

Teste ein Counter-Widget vollständig:

```
class CounterWidget extends StatefulWidget {
  final int initialValue;
  final int minValue;
  final int maxValue;
  final void Function(int)? onChanged;

  const CounterWidget({
    super.key,
    this.initialValue = 0,
    this.minValue = 0,
    this.maxValue = 100,
    this.onChanged,
  });

  @override
  State<CounterWidget> createState() => _CounterWidgetState();
}
```

Teste: - Initial value wird angezeigt - Increment-Button erhöht Wert - Decrement-Button verringert Wert - Min-/Max-Grenzen werden eingehalten - Buttons werden deaktiviert an Grenzen - onChanged Callback wird aufgerufen

4.4.1.3 Aufgabe 2: Form Widget Tests (25 min)

Teste ein Registrierungsformular:

```
+-------------------------------------+
| Registrierung                       |
+-------------------------------------+
| Name: [_________________]           |
```

```
| Email: [________________]           |
| Passwort: [________________]        |
| Passwort bestätigen: [________]     |
|                                     |
| [ ] AGB akzeptieren                 |
|                                     |
| [      Registrieren      ]          |
+-------------------------------------+
```

Teste: - Alle Felder sind vorhanden - Validierung bei leerem Submit - Email-Format Validierung - Passwort-Übereinstimmung - AGB müssen akzeptiert werden - Submit-Button wird bei Loading deaktiviert

4.4.1.4 Aufgabe 3: Liste mit Interaktionen (25 min)

Teste eine Todo-Liste:

```
class TodoList extends StatefulWidget {
  const TodoList({super.key});

  @override
  State<TodoList> createState() => _TodoListState();
}

// Features:
// - Liste von Todos anzeigen
// - Checkbox zum Abhaken
// - Swipe zum Löschen (Dismissible)
// - FAB zum Hinzufügen (öffnet Dialog)
// - Filter: Alle/Offen/Erledigt
```

Teste: - Todos werden angezeigt - Checkbox toggeln funktioniert - Swipe-to-Delete - Neues Todo hinzufügen über Dialog - Filter wechseln

4.4.1.5 Aufgabe 4: Navigation Tests (20 min)

Teste die Navigation zwischen Screens:

```
// App mit folgenden Screens:
// 1. Home Screen mit Liste
// 2. Detail Screen (bei Tap auf Item)
// 3. Edit Screen (bei Tap auf Edit-Button)
// 4. Settings Screen (über Drawer)
```

Teste: - Navigation zu Detail bei Item-Tap - Back-Navigation von Detail - Navigation zu Edit mit korrekten Daten - Rückgabe-Wert von Edit Screen - Drawer öffnen und zu Settings navigieren

4.4.1.6 Aufgabe 5: Async Widget Tests (20 min)

Teste ein Widget mit asynchronen Daten:

```
class UserProfile extends StatelessWidget {
  final UserRepository repository;
```

```
  const UserProfile({super.key, required this.repository});

  @override
  Widget build(BuildContext context) {
    return FutureBuilder<User>(
      future: repository.getCurrentUser(),
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const CircularProgressIndicator(key: Key('loading'));
        }
        if (snapshot.hasError) {
          return Text('Error: ${snapshot.error}', key: const Key('error'));
        }
        final user = snapshot.data!;
        return Column(
          children: [
            Text(user.name, key: const Key('name')),
            Text(user.email, key: const Key('email')),
          ],
        );
      },
    );
  }
}
```

Teste: - Loading-State wird angezeigt - Daten werden nach Laden angezeigt - Fehler wird bei Exception angezeigt

4.4.1.7 Aufgabe 6: Golden Tests (20 min)

Erstelle Golden Tests für UI-Komponenten:

1. **ProductCard** - Produktkarte mit Bild, Titel, Preis
2. **RatingStars** - 1-5 Sterne Bewertung
3. **StatusBadge** - Badge mit verschiedenen Farben (success, warning, error)

```
// Erstelle Golden Files für:
// - ProductCard in verschiedenen Zuständen
// - RatingStars mit 0-5 Sternen
// - StatusBadge für jeden Status
```

4.4.1.8 Aufgabe 7: Integration Test (30 min)

Erstelle einen vollständigen Integration Test für einen User Flow:

```
User Flow: Produkt zum Warenkorb hinzufügen

1. App starten -> Home Screen
2. Produkt suchen (Suchfeld)
3. Ergebnisse anzeigen
4. Auf Produkt tippen -> Detail Screen
5. Menge auswählen
6. "In den Warenkorb" tippen
```

```
7. Snackbar bestätigen
8. Zum Warenkorb navigieren
9. Produkt ist im Warenkorb
```

Schreibe einen Integration Test der den gesamten Flow durchläuft.

4.4.1.9 Aufgabe 8: Test Coverage Report (15 min)

1. Führe alle Tests mit Coverage aus
2. Generiere HTML-Report
3. Identifiziere ungetestete Bereiche
4. Schreibe Tests für mindestens einen ungetesteten Bereich

```
# Commands
flutter test --coverage
genhtml coverage/lcov.info -o coverage/html
```

Ziel: Erreiche mindestens 80% Coverage.

4.4.1.10 Bonus: Custom Finder

Erstelle eigene Finder für häufige Patterns:

```
// Beispiel: Finder für Formularfeld mit Label
Finder findFieldByLabel(String label) {
  return find.ancestor(
    of: find.text(label),
    matching: find.byType(TextFormField),
  );
}

// Erstelle Finder für:
// 1. Card mit bestimmtem Titel
// 2. Button mit Icon und Text
// 3. ListTile mit Subtitle
```

4.4.1.11 Abgabe-Checkliste

- ☐ Counter Widget vollständig getestet
- ☐ Form mit Validierung getestet
- ☐ Todo-Liste mit Interaktionen getestet
- ☐ Navigation Tests funktionieren
- ☐ Async Widget Tests mit Mocks
- ☐ Golden Tests für 3 Komponenten
- ☐ Integration Test für User Flow
- ☐ Coverage Report generiert
- ☐ Mindestens 80% Coverage erreicht
- ☐ Alle Tests sind grün

4.4.2 Lösung

4.4.2.1 Aufgabe 1: Counter Widget Tests

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

class CounterWidget extends StatefulWidget {
  final int initialValue;
  final int minValue;
  final int maxValue;
  final void Function(int)? onChanged;

  const CounterWidget({
    super.key,
    this.initialValue = 0,
    this.minValue = 0,
    this.maxValue = 100,
    this.onChanged,
  });

  @override
  State<CounterWidget> createState() => _CounterWidgetState();
}

class _CounterWidgetState extends State<CounterWidget> {
  late int _value;

  @override
  void initState() {
    super.initState();
    _value = widget.initialValue;
  }

  void _increment() {
    if (_value < widget.maxValue) {
      setState(() => _value++);
      widget.onChanged?.call(_value);
    }
  }

  void _decrement() {
    if (_value > widget.minValue) {
      setState(() => _value--);
      widget.onChanged?.call(_value);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisSize: MainAxisSize.min,
```

```
      children: [
        IconButton(
          key: const Key('decrement'),
          icon: const Icon(Icons.remove),
          onPressed: _value > widget.minValue ? _decrement : null,
        ),
        Text('$_value', key: const Key('value')),
        IconButton(
          key: const Key('increment'),
          icon: const Icon(Icons.add),
          onPressed: _value < widget.maxValue ? _increment : null,
        ),
      ],
    );
  }
}

void main() {
  group('CounterWidget', () {
    testWidgets('displays initial value', (tester) async {
      await tester.pumpWidget(const MaterialApp(
        home: CounterWidget(initialValue: 5),
      ));

      expect(find.text('5'), findsOneWidget);
    });

    testWidgets('increments value on tap', (tester) async {
      await tester.pumpWidget(const MaterialApp(
        home: CounterWidget(),
      ));

      expect(find.text('0'), findsOneWidget);

      await tester.tap(find.byKey(const Key('increment')));
      await tester.pump();

      expect(find.text('1'), findsOneWidget);
    });

    testWidgets('decrements value on tap', (tester) async {
      await tester.pumpWidget(const MaterialApp(
        home: CounterWidget(initialValue: 5),
      ));

      await tester.tap(find.byKey(const Key('decrement')));
      await tester.pump();

      expect(find.text('4'), findsOneWidget);
    });
```

```
    testWidgets('respects maxValue', (tester) async {
      await tester.pumpWidget(const MaterialApp(
        home: CounterWidget(initialValue: 10, maxValue: 10),
      ));

      // Increment button sollte deaktiviert sein
      final incrementButton = tester.widget<IconButton>(
        find.byKey(const Key('increment')),
      );
      expect(incrementButton.onPressed, isNull);
    });

    testWidgets('respects minValue', (tester) async {
      await tester.pumpWidget(const MaterialApp(
        home: CounterWidget(initialValue: 0, minValue: 0),
      ));

      final decrementButton = tester.widget<IconButton>(
        find.byKey(const Key('decrement')),
      );
      expect(decrementButton.onPressed, isNull);
    });

    testWidgets('calls onChanged callback', (tester) async {
      int? changedValue;
      await tester.pumpWidget(MaterialApp(
        home: CounterWidget(
          onChanged: (value) => changedValue = value,
        ),
      ));

      await tester.tap(find.byKey(const Key('increment')));
      await tester.pump();

      expect(changedValue, equals(1));
    });
  });
}
```

4.4.2.2 Aufgabe 2: Form Widget Tests

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

void main() {
  group('RegistrationForm', () {
    Widget createWidget({VoidCallback? onSubmit}) {
      return MaterialApp(
        home: Scaffold(
          body: RegistrationForm(onSubmit: onSubmit ?? () {}),
```

```
      ),
    );
  }

  testWidgets('renders all fields', (tester) async {
    await tester.pumpWidget(createWidget());

    expect(find.byKey(const Key('name_field')), findsOneWidget);
    expect(find.byKey(const Key('email_field')), findsOneWidget);
    expect(find.byKey(const Key('password_field')), findsOneWidget);
    expect(find.byKey(const Key('confirm_password_field')), findsOneWidget);
    expect(find.byKey(const Key('terms_checkbox')), findsOneWidget);
    expect(find.text('Registrieren'), findsOneWidget);
  });

  testWidgets('shows validation errors on empty submit', (tester) async {
    await tester.pumpWidget(createWidget());

    await tester.tap(find.text('Registrieren'));
    await tester.pump();

    expect(find.text('Name ist erforderlich'), findsOneWidget);
    expect(find.text('Email ist erforderlich'), findsOneWidget);
    expect(find.text('Passwort ist erforderlich'), findsOneWidget);
  });

  testWidgets('validates email format', (tester) async {
    await tester.pumpWidget(createWidget());

    await tester.enterText(
      find.byKey(const Key('email_field')),
      'invalid-email',
    );
    await tester.tap(find.text('Registrieren'));
    await tester.pump();

    expect(find.text('Ungültige Email'), findsOneWidget);
  });

  testWidgets('validates password match', (tester) async {
    await tester.pumpWidget(createWidget());

    await tester.enterText(
      find.byKey(const Key('password_field')),
      'password123',
    );
    await tester.enterText(
      find.byKey(const Key('confirm_password_field')),
      'different',
    );
    await tester.tap(find.text('Registrieren'));
```

```
    await tester.pump();

    expect(find.text('Passwörter stimmen nicht überein'), findsOneWidget);
  });

  testWidgets('requires terms acceptance', (tester) async {
    await tester.pumpWidget(createWidget());

    // Fülle alle Felder korrekt aus
    await tester.enterText(find.byKey(const Key('name_field')), 'John');
    await tester.enterText(
      find.byKey(const Key('email_field')),
      'john@test.com',
    );
    await tester.enterText(
      find.byKey(const Key('password_field')),
      'password123',
    );
    await tester.enterText(
      find.byKey(const Key('confirm_password_field')),
      'password123',
    );

    await tester.tap(find.text('Registrieren'));
    await tester.pump();

    expect(find.text('AGB müssen akzeptiert werden'), findsOneWidget);
  });

  testWidgets('submits valid form', (tester) async {
    bool submitted = false;
    await tester.pumpWidget(createWidget(onSubmit: () => submitted = true));

    await tester.enterText(find.byKey(const Key('name_field')), 'John');
    await tester.enterText(
      find.byKey(const Key('email_field')),
      'john@test.com',
    );
    await tester.enterText(
      find.byKey(const Key('password_field')),
      'password123',
    );
    await tester.enterText(
      find.byKey(const Key('confirm_password_field')),
      'password123',
    );
    await tester.tap(find.byKey(const Key('terms_checkbox')));
    await tester.pump();

    await tester.tap(find.text('Registrieren'));
    await tester.pumpAndSettle();
```

```
      expect(submitted, isTrue);
    });
  });
}
```

4.4.2.3 Aufgabe 3: Todo-Liste Tests

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

void main() {
  group('TodoList', () {
    testWidgets('displays todos', (tester) async {
      await tester.pumpWidget(const MaterialApp(
        home: TodoListScreen(
          initialTodos: [
            Todo(id: '1', title: 'Task 1'),
            Todo(id: '2', title: 'Task 2'),
          ],
        ),
      ));

      expect(find.text('Task 1'), findsOneWidget);
      expect(find.text('Task 2'), findsOneWidget);
    });

    testWidgets('toggles checkbox', (tester) async {
      await tester.pumpWidget(const MaterialApp(
        home: TodoListScreen(
          initialTodos: [Todo(id: '1', title: 'Task 1')],
        ),
      ));

      // Initial unchecked
      var checkbox = tester.widget<Checkbox>(find.byType(Checkbox));
      expect(checkbox.value, isFalse);

      // Toggle
      await tester.tap(find.byType(Checkbox));
      await tester.pump();

      // Now checked
      checkbox = tester.widget<Checkbox>(find.byType(Checkbox));
      expect(checkbox.value, isTrue);
    });

    testWidgets('deletes todo on swipe', (tester) async {
      await tester.pumpWidget(const MaterialApp(
        home: TodoListScreen(
```

```
        initialTodos: [Todo(id: '1', title: 'Task 1')],
      ),
    ));

    expect(find.text('Task 1'), findsOneWidget);

    // Swipe to delete
    await tester.drag(find.text('Task 1'), const Offset(500, 0));
    await tester.pumpAndSettle();

    expect(find.text('Task 1'), findsNothing);
  });

  testWidgets('adds new todo via dialog', (tester) async {
    await tester.pumpWidget(const MaterialApp(
      home: TodoListScreen(initialTodos: []),
    ));

    // Open dialog
    await tester.tap(find.byType(FloatingActionButton));
    await tester.pumpAndSettle();

    expect(find.byType(AlertDialog), findsOneWidget);

    // Enter todo
    await tester.enterText(find.byType(TextField), 'New Task');
    await tester.tap(find.text('Hinzufügen'));
    await tester.pumpAndSettle();

    expect(find.text('New Task'), findsOneWidget);
  });

  testWidgets('filters todos', (tester) async {
    await tester.pumpWidget(const MaterialApp(
      home: TodoListScreen(
        initialTodos: [
          Todo(id: '1', title: 'Open Task', isDone: false),
          Todo(id: '2', title: 'Done Task', isDone: true),
        ],
      ),
    ));

    // Default: All
    expect(find.text('Open Task'), findsOneWidget);
    expect(find.text('Done Task'), findsOneWidget);

    // Filter: Open
    await tester.tap(find.text('Offen'));
    await tester.pump();

    expect(find.text('Open Task'), findsOneWidget);
```

```
      expect(find.text('Done Task'), findsNothing);

      // Filter: Done
      await tester.tap(find.text('Erledigt'));
      await tester.pump();

      expect(find.text('Open Task'), findsNothing);
      expect(find.text('Done Task'), findsOneWidget);
    });
  });
}
```

4.4.2.4 Aufgabe 4: Navigation Tests

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

void main() {
  group('Navigation', () {
    Widget createApp() {
      return MaterialApp(
        initialRoute: '/',
        routes: {
          '/': (context) => const HomeScreen(),
          '/detail': (context) => const DetailScreen(),
          '/edit': (context) => const EditScreen(),
          '/settings': (context) => const SettingsScreen(),
        },
      );
    }

    testWidgets('navigates to detail on item tap', (tester) async {
      await tester.pumpWidget(createApp());

      await tester.tap(find.text('Item 1'));
      await tester.pumpAndSettle();

      expect(find.byType(DetailScreen), findsOneWidget);
      expect(find.byType(HomeScreen), findsNothing);
    });

    testWidgets('navigates back from detail', (tester) async {
      await tester.pumpWidget(createApp());

      // Go to detail
      await tester.tap(find.text('Item 1'));
      await tester.pumpAndSettle();

      // Go back
      await tester.tap(find.byTooltip('Back'));
```

```
    await tester.pumpAndSettle();

    expect(find.byType(HomeScreen), findsOneWidget);
  });

  testWidgets('navigates to edit with data', (tester) async {
    await tester.pumpWidget(createApp());

    await tester.tap(find.text('Item 1'));
    await tester.pumpAndSettle();

    await tester.tap(find.byIcon(Icons.edit));
    await tester.pumpAndSettle();

    expect(find.byType(EditScreen), findsOneWidget);
    // Verify data was passed
    expect(find.text('Editing: Item 1'), findsOneWidget);
  });

  testWidgets('receives result from edit screen', (tester) async {
    await tester.pumpWidget(createApp());

    await tester.tap(find.text('Item 1'));
    await tester.pumpAndSettle();

    await tester.tap(find.byIcon(Icons.edit));
    await tester.pumpAndSettle();

    // Edit and save
    await tester.enterText(find.byType(TextField), 'Updated Item');
    await tester.tap(find.text('Speichern'));
    await tester.pumpAndSettle();

    // Back on detail with updated data
    expect(find.byType(DetailScreen), findsOneWidget);
    expect(find.text('Updated Item'), findsOneWidget);
  });

  testWidgets('opens drawer and navigates to settings', (tester) async {
    await tester.pumpWidget(createApp());

    // Open drawer
    await tester.tap(find.byIcon(Icons.menu));
    await tester.pumpAndSettle();

    expect(find.byType(Drawer), findsOneWidget);

    // Navigate to settings
    await tester.tap(find.text('Einstellungen'));
    await tester.pumpAndSettle();
```

```
      expect(find.byType(SettingsScreen), findsOneWidget);
    });
  });
}
```

4.4.2.5 Aufgabe 5: Async Widget Tests

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:mocktail/mocktail.dart';

class User {
  final String name;
  final String email;
  User({required this.name, required this.email});
}

abstract class UserRepository {
  Future<User> getCurrentUser();
}

class MockUserRepository extends Mock implements UserRepository {}

void main() {
  late MockUserRepository mockRepository;

  setUp(() {
    mockRepository = MockUserRepository();
  });

  group('UserProfile', () {
    testWidgets('shows loading state', (tester) async {
      when(() => mockRepository.getCurrentUser()).thenAnswer(
        (_) => Future.delayed(
          const Duration(seconds: 1),
          () => User(name: 'John', email: 'john@test.com'),
        ),
      );

      await tester.pumpWidget(MaterialApp(
        home: UserProfile(repository: mockRepository),
      ));

      // Loading indicator visible
      expect(find.byKey(const Key('loading')), findsOneWidget);
    });

    testWidgets('shows user data after loading', (tester) async {
      when(() => mockRepository.getCurrentUser()).thenAnswer(
        (_) async => User(name: 'John Doe', email: 'john@example.com'),
```

```
      );

      await tester.pumpWidget(MaterialApp(
        home: UserProfile(repository: mockRepository),
      ));

      // Wait for future
      await tester.pumpAndSettle();

      // Data displayed
      expect(find.byKey(const Key('name')), findsOneWidget);
      expect(find.text('John Doe'), findsOneWidget);
      expect(find.text('john@example.com'), findsOneWidget);
      expect(find.byKey(const Key('loading')), findsNothing);
    });

    testWidgets('shows error on exception', (tester) async {
      when(() => mockRepository.getCurrentUser())
          .thenThrow(Exception('Network error'));

      await tester.pumpWidget(MaterialApp(
        home: UserProfile(repository: mockRepository),
      ));

      await tester.pumpAndSettle();

      expect(find.byKey(const Key('error')), findsOneWidget);
      expect(find.textContaining('Network error'), findsOneWidget);
    });
  });
}
```

4.4.2.6 Aufgabe 6: Golden Tests

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

void main() {
  group('Golden Tests', () {
    testWidgets('ProductCard matches golden', (tester) async {
      tester.binding.window.physicalSizeTestValue = const Size(400, 250);
      tester.binding.window.devicePixelRatioTestValue = 1.0;

      await tester.pumpWidget(const MaterialApp(
        home: Scaffold(
          body: Center(
            child: ProductCard(
              title: 'Flutter Book',
              price: 29.99,
              imageUrl: 'assets/book.png',
```

```
          ),
        ),
      ),
    ));

    await expectLater(
      find.byType(ProductCard),
      matchesGoldenFile('goldens/product_card.png'),
    );

    addTearDown(tester.binding.window.clearPhysicalSizeTestValue);
  });

  testWidgets('RatingStars matches golden for each rating', (tester) async {
    for (int rating = 0; rating <= 5; rating++) {
      tester.binding.window.physicalSizeTestValue = const Size(200, 50);
      tester.binding.window.devicePixelRatioTestValue = 1.0;

      await tester.pumpWidget(MaterialApp(
        home: Scaffold(
          body: Center(
            child: RatingStars(rating: rating),
          ),
        ),
      ));

      await expectLater(
        find.byType(RatingStars),
        matchesGoldenFile('goldens/rating_stars_$rating.png'),
      );
    }

    addTearDown(tester.binding.window.clearPhysicalSizeTestValue);
  });

  testWidgets('StatusBadge matches golden for each status', (tester) async {
    final statuses = ['success', 'warning', 'error'];

    for (final status in statuses) {
      tester.binding.window.physicalSizeTestValue = const Size(150, 50);
      tester.binding.window.devicePixelRatioTestValue = 1.0;

      await tester.pumpWidget(MaterialApp(
        home: Scaffold(
          body: Center(
            child: StatusBadge(status: status),
          ),
        ),
      ));

      await expectLater(
```

```
          find.byType(StatusBadge),
          matchesGoldenFile('goldens/status_badge_$status.png'),
        );
      }

      addTearDown(tester.binding.window.clearPhysicalSizeTestValue);
    });
  });
}
```

4.4.2.7 Aufgabe 7: Integration Test

```
// integration_test/shopping_flow_test.dart
import 'package:flutter_test/flutter_test.dart';
import 'package:integration_test/integration_test.dart';
import 'package:my_app/main.dart' as app;

void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  group('Shopping Flow', () {
    testWidgets('add product to cart', (tester) async {
      app.main();
      await tester.pumpAndSettle();

      // 1. Verify Home Screen
      expect(find.text('Home'), findsOneWidget);

      // 2. Search for product
      await tester.tap(find.byIcon(Icons.search));
      await tester.pumpAndSettle();

      await tester.enterText(
        find.byType(TextField),
        'Flutter Book',
      );
      await tester.testTextInput.receiveAction(TextInputAction.search);
      await tester.pumpAndSettle();

      // 3. Verify search results
      expect(find.text('Flutter Book'), findsOneWidget);

      // 4. Navigate to detail
      await tester.tap(find.text('Flutter Book'));
      await tester.pumpAndSettle();

      expect(find.text('Produktdetails'), findsOneWidget);

      // 5. Select quantity
      await tester.tap(find.byIcon(Icons.add));
```

```
      await tester.pump();
      expect(find.text('2'), findsOneWidget);

      // 6. Add to cart
      await tester.tap(find.text('In den Warenkorb'));
      await tester.pump();

      // 7. Verify snackbar
      expect(find.text('Zum Warenkorb hinzugefügt'), findsOneWidget);

      // 8. Navigate to cart
      await tester.tap(find.byIcon(Icons.shopping_cart));
      await tester.pumpAndSettle();

      // 9. Verify product in cart
      expect(find.text('Warenkorb'), findsOneWidget);
      expect(find.text('Flutter Book'), findsOneWidget);
      expect(find.text('Menge: 2'), findsOneWidget);
    });
  });
}
```

4.4.2.8 Aufgabe 8: Bonus - Custom Finder

```
// test/helpers/custom_finders.dart
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

extension CustomFinders on CommonFinders {
  /// Findet ein Formularfeld anhand seines Labels
  Finder fieldByLabel(String label) {
    return find.ancestor(
      of: find.text(label),
      matching: find.byWidgetPredicate(
        (widget) => widget is TextField || widget is TextFormField,
      ),
    );
  }

  /// Findet eine Card mit einem bestimmten Titel
  Finder cardWithTitle(String title) {
    return find.ancestor(
      of: find.text(title),
      matching: find.byType(Card),
    );
  }

  /// Findet einen Button mit Icon und Text
  Finder buttonWithIconAndText(IconData icon, String text) {
    return find.byWidgetPredicate((widget) {
```

```
      if (widget is ElevatedButton || widget is TextButton) {
        final buttonChild = (widget as dynamic).child;
        if (buttonChild is Row) {
          final hasIcon = buttonChild.children.any(
            (child) => child is Icon && child.icon == icon,
          );
          final hasText = buttonChild.children.any(
            (child) => child is Text && child.data == text,
          );
          return hasIcon && hasText;
        }
      }
      return false;
    });
  }

  /// Findet ein ListTile mit einem bestimmten Subtitle
  Finder listTileWithSubtitle(String subtitle) {
    return find.byWidgetPredicate((widget) {
      if (widget is ListTile && widget.subtitle is Text) {
        return (widget.subtitle as Text).data == subtitle;
      }
      return false;
    });
  }
}

// Verwendung in Tests
void main() {
  testWidgets('uses custom finders', (tester) async {
    await tester.pumpWidget(const MaterialApp(home: MyWidget()));

    // Formularfeld finden
    await tester.enterText(find.fieldByLabel('Email'), 'test@test.com');

    // Card mit Titel finden
    expect(find.cardWithTitle('Produkt A'), findsOneWidget);

    // Button mit Icon und Text
    await tester.tap(find.buttonWithIconAndText(Icons.save, 'Speichern'));

    // ListTile mit Subtitle
    expect(find.listTileWithSubtitle('Details'), findsOneWidget);
  });
}
```

4.4.3 Ressourcen

4.4.3.1 Offizielle Dokumentation

- Widget Testing

- Integration Testing
- flutter_test Package
- WidgetTester
- Finder

4.4.3.2 Flutter Cookbook

- An introduction to widget testing
- Find widgets
- Tap, drag, and enter text

4.4.3.3 Videos

- Widget Testing Flutter
- Integration Testing
- Golden Tests

4.4.3.4 Tutorials & Artikel

- Complete Guide to Widget Testing
- Flutter Testing for Beginners
- Golden Tests Tutorial

4.4.3.5 Packages

- integration_test - Integration Tests
- golden_toolkit - Erweiterte Golden Tests
- patrol - Native Integration Tests
- flutter_test_robots - Test Utilities

4.4.3.6 Tools

- Coverage - Code Coverage
- lcov - Coverage Reports
- Codecov - Coverage in CI

4.4.3.7 Golden Test Tools

```
# Goldens aktualisieren
flutter test --update-goldens

# Nur Golden Tests
flutter test --tags golden
```

4.4.3.8 Zum Vertiefen

- Accessibility Testing
- Performance Testing
- Patrol Native Testing

4.5 Einheit 4.5: Packages & Plugins

4.5.0.1 Lernziele

Nach dieser Einheit kannst du: - Packages von pub.dev nutzen und bewerten - Eigene Packages erstellen und veröffentlichen - Platform Channels (MethodChannel) verstehen - Plattform-spezifischen Code schreiben

4.5.0.2 1. Packages verwenden

Package hinzufügen

```yaml
# pubspec.yaml
dependencies:
  http: ^1.1.0
  provider: ^6.1.0
  shared_preferences: ^2.2.0

dev_dependencies:
  flutter_test:
    sdk: flutter
  mocktail: ^1.0.0
```

```bash
# Package hinzufügen
flutter pub add http
flutter pub add --dev mocktail

# Dependencies installieren
flutter pub get

# Veraltete Dependencies prüfen
flutter pub outdated

# Dependencies aktualisieren
flutter pub upgrade
```

Package-Qualität bewerten

Auf pub.dev achten auf: - **Likes** - Community-Beliebtheit - **Pub Points** - Qualitätsmetriken (max 140) - **Popularity** - Nutzungshäufigkeit - **Null Safety** - Dart 3 kompatibel - **Platform Support** - iOS, Android, Web, Desktop - **Maintenance** - Letzte Updates, offene Issues

Wichtige Packages

Kategorie	Package	Beschreibung
State	`provider`, `riverpod`, `bloc`	State Management
HTTP	`http`, `dio`	Netzwerk-Requests
Storage	`shared_preferences`, `hive`, `sqflite`	Lokale Daten
UI	`flutter_svg`, `cached_network_image`	Erweiterte Widgets
Utils	`intl`, `path_provider`, `url_launcher`	Hilfsfunktionen
Testing	`mocktail`, `bloc_test`	Test-Utilities

4.5.0.3 2. Eigene Packages erstellen

Package-Projekt anlegen

```
# Dart Package (nur Dart-Code)
flutter create --template=package my_package

# Flutter Plugin (mit Plattform-Code)
flutter create --template=plugin --platforms=android,ios my_plugin
```

Package-Struktur

```
my_package/
+-- lib/
|   +-- my_package.dart        # Haupt-Export
|   +-- src/
|       +-- models/
|       |   +-- user.dart
|       +-- services/
|           +-- api_service.dart
+-- test/
|   +-- my_package_test.dart
+-- example/                    # Beispiel-App
|   +-- lib/
|       +-- main.dart
+-- pubspec.yaml
+-- README.md
+-- CHANGELOG.md
+-- LICENSE
```

Haupt-Export-Datei

```
// lib/my_package.dart

/// A useful package for doing things.
library my_package;

// Public exports
export 'src/models/user.dart';
export 'src/services/api_service.dart' show ApiService;
export 'src/utils/validators.dart' hide internalHelper;
```

pubspec.yaml für Package

```
name: my_package
description: A useful Flutter package for managing users.
version: 1.0.0
homepage: https://github.com/username/my_package
repository: https://github.com/username/my_package
issue_tracker: https://github.com/username/my_package/issues

environment:
  sdk: '>=3.0.0 <4.0.0'
  flutter: '>=3.10.0'
```

```yaml
dependencies:
  flutter:
    sdk: flutter

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^3.0.0

# Für reine Dart-Packages (ohne Flutter):
# environment:
#   sdk: '>=3.0.0 <4.0.0'
#
# dependencies:
#   http: ^1.1.0
```

Beispiel: Validatoren-Package

```dart
// lib/src/validators.dart

/// Email validator
class EmailValidator {
  static final _emailRegex = RegExp(
    r'^[a-zA-Z0-9.]+@[a-zA-Z0-9]+\.[a-zA-Z]+$',
  );

  /// Validates an email address.
  ///
  /// Returns `true` if [email] is a valid email format.
  ///
  /// Example:
  /// ```dart
  /// EmailValidator.isValid('test@example.com'); // true
  /// EmailValidator.isValid('invalid'); // false
  /// ```
  static bool isValid(String email) {
    return _emailRegex.hasMatch(email);
  }
}

/// Password validator with configurable rules.
class PasswordValidator {
  final int minLength;
  final bool requireUppercase;
  final bool requireDigit;
  final bool requireSpecial;

  const PasswordValidator({
    this.minLength = 8,
    this.requireUppercase = true,
```

```
    this.requireDigit = true,
    this.requireSpecial = false,
  });

  /// Validates a password against configured rules.
  ValidationResult validate(String password) {
    final errors = <String>[];

    if (password.length < minLength) {
      errors.add('Minimum $minLength characters required');
    }
    if (requireUppercase && !password.contains(RegExp(r'[A-Z]'))) {
      errors.add('Uppercase letter required');
    }
    if (requireDigit && !password.contains(RegExp(r'[0-9]'))) {
      errors.add('Digit required');
    }
    if (requireSpecial && !password.contains(RegExp(r'[!@#$%^&*]'))) {
      errors.add('Special character required');
    }

    return ValidationResult(
      isValid: errors.isEmpty,
      errors: errors,
    );
  }
}

class ValidationResult {
  final bool isValid;
  final List<String> errors;

  const ValidationResult({
    required this.isValid,
    required this.errors,
  });
}
```

4.5.0.4 3. Package veröffentlichen

Vorbereitung

```
# Package analysieren
flutter pub publish --dry-run

# Typische Probleme beheben:
# - README.md hinzufügen
# - CHANGELOG.md pflegen
# - LICENSE-Datei hinzufügen
# - Description in pubspec.yaml
# - Dartdoc-Kommentare
```

README.md Template

```
# my_package

A Flutter package for user validation.

#### Features

- Email validation
- Password validation with configurable rules
- Phone number validation

#### Getting started

Add to your `pubspec.yaml`:

```yaml
dependencies:
  my_package: ^1.0.0
```

4.5.0.5 Usage

```
import 'package:my_package/my_package.dart';

// Email validation
if (EmailValidator.isValid('test@example.com')) {
  print('Valid email!');
}

// Password validation
final validator = PasswordValidator(
  minLength: 8,
  requireUppercase: true,
);
final result = validator.validate('MyPassword1');
print(result.isValid); // true
```

4.5.0.6 Additional information

For more examples, see the `/example` folder.

```
####### CHANGELOG.md

```markdown
#### 1.0.0

- Initial release
- Email validation
- Password validation

#### 1.0.1
```

```
- Added phone number validation
- Bug fix for special characters

#### 1.1.0

- Added German locale support
- Breaking: Renamed `ValidationResult.valid` to `isValid`
```

Veröffentlichen

```
# Auf pub.dev veröffentlichen
flutter pub publish

# Bei erstem Publish: Authentifizierung erforderlich
# Folge den Anweisungen im Terminal
```

4.5.0.7 4. Platform Channels

MethodChannel Grundlagen

```
// Dart-Seite
import 'package:flutter/services.dart';

class BatteryService {
  static const _channel = MethodChannel('com.example/battery');

  Future<int> getBatteryLevel() async {
    try {
      final level = await _channel.invokeMethod<int>('getBatteryLevel');
      return level ?? -1;
    } on PlatformException catch (e) {
      print('Failed to get battery level: ${e.message}');
      return -1;
    }
  }

  Future<bool> isCharging() async {
    final charging = await _channel.invokeMethod<bool>('isCharging');
    return charging ?? false;
  }
}
```

Android Implementation (Kotlin)

```
// android/app/src/main/kotlin/.../MainActivity.kt
package com.example.my_app

import android.content.Context
import android.content.Intent
import android.content.IntentFilter
import android.os.BatteryManager
import android.os.Build
import io.flutter.embedding.android.FlutterActivity
```

```
import io.flutter.embedding.engine.FlutterEngine
import io.flutter.plugin.common.MethodChannel

class MainActivity: FlutterActivity() {
    private val CHANNEL = "com.example/battery"

    override fun configureFlutterEngine(flutterEngine: FlutterEngine) {
        super.configureFlutterEngine(flutterEngine)

        MethodChannel(flutterEngine.dartExecutor.binaryMessenger, CHANNEL)
            .setMethodCallHandler { call, result ->
                when (call.method) {
                    "getBatteryLevel" -> {
                        val batteryLevel = getBatteryLevel()
                        if (batteryLevel != -1) {
                            result.success(batteryLevel)
                        } else {
                            result.error("UNAVAILABLE", "Battery level not
 ↪  available", null)
                        }
                    }
                    "isCharging" -> {
                        result.success(isCharging())
                    }
                    else -> {
                        result.notImplemented()
                    }
                }
            }
    }

    private fun getBatteryLevel(): Int {
        val batteryManager = getSystemService(Context.BATTERY_SERVICE) as
         ↪  BatteryManager
        return batteryManager.getIntProperty(BatteryManager.
         ↪  BATTERY_PROPERTY_CAPACITY)
    }

    private fun isCharging(): Boolean {
        val intent = registerReceiver(null,
         ↪  IntentFilter(Intent.ACTION_BATTERY_CHANGED))
        val status = intent?.getIntExtra(BatteryManager.EXTRA_STATUS, -1) ?: -1
        return status == BatteryManager.BATTERY_STATUS_CHARGING ||
               status == BatteryManager.BATTERY_STATUS_FULL
    }
}
```

iOS Implementation (Swift)

```
// ios/Runner/AppDelegate.swift
import UIKit
```

```
import Flutter

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
    override func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions:
        ↪ [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        let controller = window?.rootViewController as! FlutterViewController
        let batteryChannel = FlutterMethodChannel(
            name: "com.example/battery",
            binaryMessenger: controller.binaryMessenger
        )

        batteryChannel.setMethodCallHandler { [weak self] (call, result) in
            switch call.method {
            case "getBatteryLevel":
                result(self?.getBatteryLevel())
            case "isCharging":
                result(self?.isCharging())
            default:
                result(FlutterMethodNotImplemented)
            }
        }

        GeneratedPluginRegistrant.register(with: self)
        return super.application(application, didFinishLaunchingWithOptions:
        ↪ launchOptions)
    }

    private func getBatteryLevel() -> Int {
        UIDevice.current.isBatteryMonitoringEnabled = true
        return Int(UIDevice.current.batteryLevel * 100)
    }

    private func isCharging() -> Bool {
        UIDevice.current.isBatteryMonitoringEnabled = true
        return UIDevice.current.batteryState == .charging ||
               UIDevice.current.batteryState == .full
    }
}
```

4.5.0.8 5. EventChannel für Streams

Dart-Seite

```
class BatteryMonitor {
  static const _eventChannel = EventChannel('com.example/battery_events');

  Stream<int> get batteryLevelStream {
```

```
    return _eventChannel.receiveBroadcastStream().map((level) => level as int);
  }
}

// Verwendung
class BatteryWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StreamBuilder<int>(
      stream: BatteryMonitor().batteryLevelStream,
      builder: (context, snapshot) {
        if (snapshot.hasData) {
          return Text('Battery: ${snapshot.data}%');
        }
        return const CircularProgressIndicator();
      },
    );
  }
}
```

Android EventChannel (Kotlin)

```
class MainActivity: FlutterActivity() {
    private val EVENT_CHANNEL = "com.example/battery_events"

    override fun configureFlutterEngine(flutterEngine: FlutterEngine) {
        super.configureFlutterEngine(flutterEngine)

        EventChannel(flutterEngine.dartExecutor.binaryMessenger, EVENT_CHANNEL)
            .setStreamHandler(object : EventChannel.StreamHandler {
                private var receiver: BroadcastReceiver? = null

                override fun onListen(arguments: Any?, events:
                ↪  EventChannel.EventSink?) {
                    receiver = object : BroadcastReceiver() {
                        override fun onReceive(context: Context?, intent:
                        ↪  Intent?) {
                            val level = intent?.getIntExtra(BatteryManager.
                            ↪  EXTRA_LEVEL,
                            ↪  -1)
                            events?.success(level)
                        }
                    }
                    registerReceiver(receiver,
↪  IntentFilter(Intent.ACTION_BATTERY_CHANGED))
                }

                override fun onCancel(arguments: Any?) {
                    unregisterReceiver(receiver)
                    receiver = null
                }
```

```
            })
    }
}
```

4.5.0.9 6. Plattform-spezifischer Code

Platform Check

```
import 'dart:io' show Platform;
import 'package:flutter/foundation.dart' show kIsWeb;

class PlatformService {
  static String get platformName {
    if (kIsWeb) return 'Web';
    if (Platform.isAndroid) return 'Android';
    if (Platform.isIOS) return 'iOS';
    if (Platform.isMacOS) return 'macOS';
    if (Platform.isWindows) return 'Windows';
    if (Platform.isLinux) return 'Linux';
    return 'Unknown';
  }

  static bool get isMobile => !kIsWeb && (Platform.isAndroid || Platform.isIOS);
  static bool get isDesktop => !kIsWeb && (Platform.isMacOS ||                ↪
 ↪  Platform.isWindows || Platform.isLinux);
}
```

Conditional Imports

```
// lib/storage/storage.dart
export 'storage_stub.dart'
    if (dart.library.io) 'storage_io.dart'
    if (dart.library.html) 'storage_web.dart';

// lib/storage/storage_stub.dart
class Storage {
  Future<void> save(String key, String value) {
    throw UnimplementedError();
  }
}

// lib/storage/storage_io.dart
import 'package:shared_preferences/shared_preferences.dart';

class Storage {
  Future<void> save(String key, String value) async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.setString(key, value);
  }
}

// lib/storage/storage_web.dart
```

```
import 'dart:html' as html;

class Storage {
  Future<void> save(String key, String value) async {
    html.window.localStorage[key] = value;
  }
}
```

Plattform-spezifische Widgets

```
import 'dart:io';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

class AdaptiveButton extends StatelessWidget {
  final String text;
  final VoidCallback onPressed;

  const AdaptiveButton({
    super.key,
    required this.text,
    required this.onPressed,
  });

  @override
  Widget build(BuildContext context) {
    if (Platform.isIOS) {
      return CupertinoButton(
        onPressed: onPressed,
        child: Text(text),
      );
    }
    return ElevatedButton(
      onPressed: onPressed,
      child: Text(text),
    );
  }
}

class AdaptiveDialog {
  static Future<bool?> show(
    BuildContext context, {
    required String title,
    required String content,
  }) {
    if (Platform.isIOS) {
      return showCupertinoDialog<bool>(
        context: context,
        builder: (context) => CupertinoAlertDialog(
          title: Text(title),
          content: Text(content),
```

```
          actions: [
            CupertinoDialogAction(
              isDestructiveAction: true,
              onPressed: () => Navigator.pop(context, false),
              child: const Text('Abbrechen'),
            ),
            CupertinoDialogAction(
              onPressed: () => Navigator.pop(context, true),
              child: const Text('OK'),
            ),
          ],
        ),
      );
    }

    return showDialog<bool>(
      context: context,
      builder: (context) => AlertDialog(
        title: Text(title),
        content: Text(content),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context, false),
            child: const Text('Abbrechen'),
          ),
          TextButton(
            onPressed: () => Navigator.pop(context, true),
            child: const Text('OK'),
          ),
        ],
      ),
    );
  }
}
```

4.5.0.10 7. Flutter Plugin erstellen

Plugin-Struktur

```
my_plugin/
+-- lib/
|   +-- my_plugin.dart
+-- android/
|   +-- src/main/kotlin/.../MyPlugin.kt
+-- ios/
|   +-- Classes/
|       +-- MyPlugin.swift
+-- example/
+-- pubspec.yaml
+-- README.md
```

Plugin pubspec.yaml

```yaml
name: my_plugin
description: A Flutter plugin for native features.
version: 1.0.0

environment:
  sdk: '>=3.0.0 <4.0.0'
  flutter: '>=3.10.0'

dependencies:
  flutter:
    sdk: flutter

flutter:
  plugin:
    platforms:
      android:
        package: com.example.my_plugin
        pluginClass: MyPlugin
      ios:
        pluginClass: MyPlugin
```

4.5.0.11 Zusammenfassung

Konzept	Beschreibung
Package	Wiederverwendbarer Dart/Flutter Code
Plugin	Package mit nativem Plattform-Code
pub.dev	Zentrales Package-Repository
MethodChannel	Synchrone Kommunikation mit Native
EventChannel	Stream-basierte Kommunikation
Platform Check	Plattform-spezifische Logik

Best Practices: - Packages gut dokumentieren (README, Dartdoc) - CHANGELOG.md pflegen - Semantic Versioning verwenden - Platform Channels nur wenn nötig - Beispiel-App im `/example` Ordner

4.5.1 Übung

4.5.1.1 Ziel

Eigene Packages erstellen und Platform Channels nutzen.

4.5.1.2 Aufgabe 1: Package-Recherche (15 min)

Recherchiere auf pub.dev:

1. Finde 3 State Management Packages und vergleiche:
   - Pub Points
   - Likes
   - Letzte Aktualisierung
   - Dokumentationsqualität
2. Finde ein Package für:

- PDF-Generierung
- Biometrie-Authentifizierung
- Push Notifications

Dokumentiere deine Entscheidungskriterien.

4.5.1.3 Aufgabe 2: Validator Package (30 min)

Erstelle ein eigenes Validierungs-Package:

```
my_validators/
+-- lib/
|   +-- my_validators.dart
|   +-- src/
|       +-- email_validator.dart
|       +-- password_validator.dart
|       +-- phone_validator.dart
|       +-- iban_validator.dart
+-- test/
+-- example/
+-- pubspec.yaml
+-- README.md
```

Anforderungen: - Alle Validatoren als static methods - Konfigurierbare Regeln für Passwort - Deutsche und internationale Telefonnummern - IBAN-Validierung (vereinfacht) - Vollständige Dartdoc-Kommentare - Unit Tests für alle Validatoren - Beispiel-App

4.5.1.4 Aufgabe 3: Widget Package (25 min)

Erstelle ein Package mit wiederverwendbaren Widgets:

```
// Widgets die das Package enthalten soll:

// 1. LoadingButton
// - Zeigt CircularProgressIndicator während isLoading
// - Deaktiviert während Loading
LoadingButton(
  isLoading: _isLoading,
  onPressed: _submit,
  child: Text('Submit'),
)

// 2. RatingBar
// - Interaktive Sterne-Bewertung (1-5)
// - Callback für Änderungen
RatingBar(
  rating: 3,
  onRatingChanged: (rating) {},
)

// 3. ExpandableText
// - Text mit "mehr anzeigen" wenn zu lang
ExpandableText(
  text: longText,
```

```
  maxLines: 3,
)
```

4.5.1.5 Aufgabe 4: Platform Channel - Device Info (30 min)

Implementiere einen Platform Channel für Geräteinformationen:

```
class DeviceInfoService {
  // Implementiere:
  Future<String> getDeviceModel();
  Future<String> getOsVersion();
  Future<int> getAvailableStorage();  // in MB
  Future<bool> isEmulator();
}
```

Implementiere für: - Android (Kotlin) - iOS (Swift) - optional

4.5.1.6 Aufgabe 5: EventChannel - Connectivity (25 min)

Erstelle einen EventChannel für Netzwerk-Status:

```
class ConnectivityService {
  Stream<ConnectivityStatus> get statusStream;
}

enum ConnectivityStatus {
  wifi,
  mobile,
  none,
}
```

Der Stream soll: - Initial den aktuellen Status senden - Bei Änderungen updaten - Korrekt disposen

4.5.1.7 Aufgabe 6: Plattform-spezifisches UI (20 min)

Erstelle adaptive Widgets:

```
// 1. AdaptiveScaffold
// - iOS: CupertinoPageScaffold
// - Android: Scaffold mit AppBar

// 2. AdaptiveListTile
// - iOS: CupertinoListTile style
// - Android: Material ListTile

// 3. AdaptiveSwitch
// - iOS: CupertinoSwitch
// - Android: Material Switch

// 4. AdaptiveProgressIndicator
// - iOS: CupertinoActivityIndicator
// - Android: CircularProgressIndicator
```

4.5.1.8 Aufgabe 7: Package veröffentlichen (Dry-Run) (20 min)

Bereite dein Validator-Package zur Veröffentlichung vor:

1. Vervollständige README.md:
   - Features-Liste
   - Installation
   - Usage mit Code-Beispielen
   - API-Dokumentation
2. Erstelle CHANGELOG.md
3. Füge LICENSE hinzu (MIT)
4. Führe aus:

```
flutter pub publish --dry-run
```

5. Behebe alle gemeldeten Probleme

4.5.1.9 Aufgabe 8: Plugin-Struktur verstehen (15 min)

Analysiere ein bestehendes Plugin:

```
# Clone ein einfaches Plugin
git clone https://github.com/miguelpruivo/flutter_file_picker

# Oder schaue dir an:
# - shared_preferences
# - url_launcher
# - image_picker
```

Dokumentiere: - Wie ist der Code organisiert? - Wo ist der Dart-Code? - Wo ist der Android/iOS Code? - Wie kommunizieren sie?

4.5.1.10 Bonus: Pigeon für Type-Safe Channels

Nutze das `pigeon` Package für typsichere Platform Channels:

```
// pigeons/messages.dart
import 'package:pigeon/pigeon.dart';

class DeviceInfo {
  String? model;
  String? osVersion;
}

@HostApi()
abstract class DeviceInfoApi {
  DeviceInfo getDeviceInfo();
}
```

```
# Code generieren
flutter pub run pigeon --input pigeons/messages.dart
```

4.5.1.11 Abgabe-Checkliste

- ☐ Package-Vergleich dokumentiert
- ☐ Validator Package erstellt
- ☐ Alle Validatoren getestet
- ☐ Widget Package mit 3 Widgets
- ☐ Platform Channel für Device Info
- ☐ EventChannel für Connectivity
- ☐ Adaptive Widgets implementiert
- ☐ Package publish –dry-run erfolgreich
- ☐ Plugin-Struktur dokumentiert

4.5.2 Lösung

4.5.2.1 Aufgabe 2: Validator Package

Struktur

```
my_validators/
+-- lib/
|   +-- my_validators.dart
|   +-- src/
|       +-- email_validator.dart
|       +-- password_validator.dart
|       +-- phone_validator.dart
|       +-- iban_validator.dart
+-- test/
|   +-- validators_test.dart
+-- example/
|   +-- lib/main.dart
+-- pubspec.yaml
+-- README.md
```

lib/my_validators.dart

```
/// A collection of validators for common use cases.
library my_validators;

export 'src/email_validator.dart';
export 'src/password_validator.dart';
export 'src/phone_validator.dart';
export 'src/iban_validator.dart';
```

lib/src/email_validator.dart

```
/// Validates email addresses.
class EmailValidator {
  static final _emailRegex = RegExp(
    r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$',
  );

  /// Returns `true` if [email] is a valid email address.
  ///
  /// Example:
```

```
  /// ```dart
  /// EmailValidator.isValid('test@example.com'); // true
  /// EmailValidator.isValid('invalid'); // false
  /// ```
  static bool isValid(String email) {
    if (email.isEmpty) return false;
    return _emailRegex.hasMatch(email.trim());
  }

  /// Returns an error message if invalid, `null` if valid.
  static String? validate(String? email) {
    if (email == null || email.isEmpty) {
      return 'Email ist erforderlich';
    }
    if (!isValid(email)) {
      return 'Ungültige Email-Adresse';
    }
    return null;
  }
}
```

lib/src/password_validator.dart

```
/// Configuration for password validation.
class PasswordConfig {
  final int minLength;
  final bool requireUppercase;
  final bool requireLowercase;
  final bool requireDigit;
  final bool requireSpecialChar;

  const PasswordConfig({
    this.minLength = 8,
    this.requireUppercase = true,
    this.requireLowercase = true,
    this.requireDigit = true,
    this.requireSpecialChar = false,
  });

  static const standard = PasswordConfig();
  static const strong = PasswordConfig(
    minLength: 12,
    requireSpecialChar: true,
  );
}

/// Result of password validation.
class PasswordValidationResult {
  final bool isValid;
  final List<String> errors;
```

```
  const PasswordValidationResult({
    required this.isValid,
    this.errors = const [],
  });
}

/// Validates passwords with configurable rules.
class PasswordValidator {
  final PasswordConfig config;

  const PasswordValidator({this.config = PasswordConfig.standard});

  /// Validates [password] against configured rules.
  PasswordValidationResult validate(String password) {
    final errors = <String>[];

    if (password.length < config.minLength) {
      errors.add('Mindestens ${config.minLength} Zeichen erforderlich');
    }
    if (config.requireUppercase && !password.contains(RegExp(r'[A-Z]'))) {
      errors.add('Großbuchstabe erforderlich');
    }
    if (config.requireLowercase && !password.contains(RegExp(r'[a-z]'))) {
      errors.add('Kleinbuchstabe erforderlich');
    }
    if (config.requireDigit && !password.contains(RegExp(r'[0-9]'))) {
      errors.add('Ziffer erforderlich');
    }
    if (config.requireSpecialChar &&                                    ↪
↪   !password.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]'))) {
      errors.add('Sonderzeichen erforderlich');
    }

    return PasswordValidationResult(
      isValid: errors.isEmpty,
      errors: errors,
    );
  }

  /// Convenience method for form validation.
  String? validateForForm(String? password) {
    if (password == null || password.isEmpty) {
      return 'Passwort ist erforderlich';
    }
    final result = validate(password);
    return result.isValid ? null : result.errors.first;
  }
}
```

lib/src/phone_validator.dart

```
/// Validates phone numbers.
class PhoneValidator {
  static final _germanMobile = RegExp(r'^\+49\s?1[5-7]\d{1,2}\s?\d{6,8}$');
  static final _germanLandline = RegExp(r'^\+49\s?\d{2,5}\s?\d{4,10}$');
  static final _international = RegExp(r'^\+\d{1,3}\s?\d{4,14}$');

  /// Validates a German mobile phone number.
  static bool isValidGermanMobile(String phone) {
    final cleaned = phone.replaceAll(RegExp(r'[\s\-()]'), '');
    return _germanMobile.hasMatch(cleaned);
  }

  /// Validates a German landline number.
  static bool isValidGermanLandline(String phone) {
    final cleaned = phone.replaceAll(RegExp(r'[\s\-()]'), '');
    return _germanLandline.hasMatch(cleaned);
  }

  /// Validates any German phone number.
  static bool isValidGerman(String phone) {
    return isValidGermanMobile(phone) || isValidGermanLandline(phone);
  }

  /// Validates an international phone number.
  static bool isValidInternational(String phone) {
    final cleaned = phone.replaceAll(RegExp(r'[\s\-()]'), '');
    return _international.hasMatch(cleaned);
  }

  /// Form validator for German phone numbers.
  static String? validateGerman(String? phone) {
    if (phone == null || phone.isEmpty) {
      return 'Telefonnummer ist erforderlich';
    }
    if (!isValidGerman(phone)) {
      return 'Ungültige deutsche Telefonnummer';
    }
    return null;
  }
}
```

lib/src/iban_validator.dart

```
/// Validates IBAN numbers.
class IbanValidator {
  static final _ibanRegex = RegExp(r'^[A-Z]{2}\d{2}[A-Z0-9]{4,30}$');

  /// Validates an IBAN (basic format check).
  static bool isValid(String iban) {
    final cleaned = iban.replaceAll(RegExp(r'\s'), '').toUpperCase();
```

```
    if (!_ibanRegex.hasMatch(cleaned)) return false;
    if (cleaned.length < 15 || cleaned.length > 34) return false;

    // Checksum validation
    return _validateChecksum(cleaned);
  }

  static bool _validateChecksum(String iban) {
    // Move first 4 chars to end
    final rearranged = iban.substring(4) + iban.substring(0, 4);

    // Convert letters to numbers (A=10, B=11, etc.)
    final numericString = rearranged.split('').map((char) {
      final code = char.codeUnitAt(0);
      if (code >= 65 && code <= 90) {
        return (code - 55).toString();
      }
      return char;
    }).join();

    // Calculate mod 97
    return _mod97(numericString) == 1;
  }

  static int _mod97(String numericString) {
    var remainder = 0;
    for (var i = 0; i < numericString.length; i++) {
      final digit = int.parse(numericString[i]);
      remainder = (remainder * 10 + digit) % 97;
    }
    return remainder;
  }

  /// Form validator for IBAN.
  static String? validate(String? iban) {
    if (iban == null || iban.isEmpty) {
      return 'IBAN ist erforderlich';
    }
    if (!isValid(iban)) {
      return 'Ungültige IBAN';
    }
    return null;
  }
}
```

test/validators_test.dart

```
import 'package:test/test.dart';
import 'package:my_validators/my_validators.dart';

void main() {
```

```
  group('EmailValidator', () {
    test('validates correct emails', () {
      expect(EmailValidator.isValid('test@example.com'), isTrue);
      expect(EmailValidator.isValid('user.name@domain.co.uk'), isTrue);
    });

    test('rejects invalid emails', () {
      expect(EmailValidator.isValid('invalid'), isFalse);
      expect(EmailValidator.isValid('test@'), isFalse);
      expect(EmailValidator.isValid(''), isFalse);
    });
  });

  group('PasswordValidator', () {
    test('validates with default config', () {
      final validator = PasswordValidator();
      expect(validator.validate('Password1').isValid, isTrue);
      expect(validator.validate('weak').isValid, isFalse);
    });

    test('validates with strong config', () {
      final validator = PasswordValidator(config: PasswordConfig.strong);
      expect(validator.validate('Password1!@#').isValid, isTrue);
      expect(validator.validate('Password1').isValid, isFalse);
    });
  });

  group('PhoneValidator', () {
    test('validates German mobile', () {
      expect(PhoneValidator.isValidGermanMobile('+49 151 12345678'), isTrue);
      expect(PhoneValidator.isValidGermanMobile('invalid'), isFalse);
    });
  });

  group('IbanValidator', () {
    test('validates correct IBAN', () {
      expect(IbanValidator.isValid('DE89 3704 0044 0532 0130 00'), isTrue);
    });

    test('rejects invalid IBAN', () {
      expect(IbanValidator.isValid('DE00 0000 0000 0000 0000 00'), isFalse);
    });
  });
}
```

4.5.2.2 Aufgabe 3: Widget Package

```
// lib/src/loading_button.dart
import 'package:flutter/material.dart';
```

```
/// A button that shows a loading indicator while processing.
class LoadingButton extends StatelessWidget {
  final bool isLoading;
  final VoidCallback? onPressed;
  final Widget child;
  final ButtonStyle? style;

  const LoadingButton({
    super.key,
    required this.isLoading,
    required this.onPressed,
    required this.child,
    this.style,
  });

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      style: style,
      onPressed: isLoading ? null : onPressed,
      child: isLoading
          ? const SizedBox(
              width: 20,
              height: 20,
              child: CircularProgressIndicator(strokeWidth: 2),
            )
          : child,
    );
  }
}

// lib/src/rating_bar.dart
import 'package:flutter/material.dart';

/// An interactive star rating widget.
class RatingBar extends StatelessWidget {
  final int rating;
  final int maxRating;
  final ValueChanged<int>? onRatingChanged;
  final double size;
  final Color activeColor;
  final Color inactiveColor;

  const RatingBar({
    super.key,
    required this.rating,
    this.maxRating = 5,
    this.onRatingChanged,
    this.size = 32,
    this.activeColor = Colors.amber,
    this.inactiveColor = Colors.grey,
```

```
  });

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisSize: MainAxisSize.min,
      children: List.generate(maxRating, (index) {
        final isActive = index < rating;
        return GestureDetector(
          onTap: onRatingChanged != null
              ? () => onRatingChanged!(index + 1)
              : null,
          child: Icon(
            isActive ? Icons.star : Icons.star_border,
            size: size,
            color: isActive ? activeColor : inactiveColor,
          ),
        );
      }),
    );
  }
}

// lib/src/expandable_text.dart
import 'package:flutter/material.dart';

/// A text widget that can be expanded to show full content.
class ExpandableText extends StatefulWidget {
  final String text;
  final int maxLines;
  final TextStyle? style;
  final String expandText;
  final String collapseText;

  const ExpandableText({
    super.key,
    required this.text,
    this.maxLines = 3,
    this.style,
    this.expandText = 'mehr anzeigen',
    this.collapseText = 'weniger anzeigen',
  });

  @override
  State<ExpandableText> createState() => _ExpandableTextState();
}

class _ExpandableTextState extends State<ExpandableText> {
  bool _isExpanded = false;

  @override
```

```
  Widget build(BuildContext context) {
    return LayoutBuilder(
      builder: (context, constraints) {
        final textSpan = TextSpan(text: widget.text, style: widget.style);
        final textPainter = TextPainter(
          text: textSpan,
          maxLines: widget.maxLines,
          textDirection: TextDirection.ltr,
        )..layout(maxWidth: constraints.maxWidth);

        final isOverflowing = textPainter.didExceedMaxLines;

        return Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              widget.text,
              style: widget.style,
              maxLines: _isExpanded ? null : widget.maxLines,
              overflow: _isExpanded ? null : TextOverflow.ellipsis,
            ),
            if (isOverflowing)
              GestureDetector(
                onTap: () => setState(() => _isExpanded = !_isExpanded),
                child: Padding(
                  padding: const EdgeInsets.only(top: 4),
                  child: Text(
                    _isExpanded ? widget.collapseText : widget.expandText,
                    style: TextStyle(
                      color: Theme.of(context).primaryColor,
                      fontWeight: FontWeight.w500,
                    ),
                  ),
                ),
              ),
          ],
        );
      },
    );
  }
}
```

4.5.2.3 Aufgabe 4: Platform Channel - Device Info

Dart

```
import 'package:flutter/services.dart';

class DeviceInfoService {
  static const _channel = MethodChannel('com.example/device_info');
```

```
  Future<String> getDeviceModel() async {
    try {
      return await _channel.invokeMethod<String>('getDeviceModel') ?? 'Unknown';
    } on PlatformException {
      return 'Unknown';
    }
  }

  Future<String> getOsVersion() async {
    try {
      return await _channel.invokeMethod<String>('getOsVersion') ?? 'Unknown';
    } on PlatformException {
      return 'Unknown';
    }
  }

  Future<int> getAvailableStorage() async {
    try {
      return await _channel.invokeMethod<int>('getAvailableStorage') ?? -1;
    } on PlatformException {
      return -1;
    }
  }

  Future<bool> isEmulator() async {
    try {
      return await _channel.invokeMethod<bool>('isEmulator') ?? false;
    } on PlatformException {
      return false;
    }
  }
}
```

Android (Kotlin)

```
package com.example.my_app

import android.os.Build
import android.os.Environment
import android.os.StatFs
import io.flutter.embedding.android.FlutterActivity
import io.flutter.embedding.engine.FlutterEngine
import io.flutter.plugin.common.MethodChannel

class MainActivity: FlutterActivity() {
    private val CHANNEL = "com.example/device_info"

    override fun configureFlutterEngine(flutterEngine: FlutterEngine) {
        super.configureFlutterEngine(flutterEngine)

        MethodChannel(flutterEngine.dartExecutor.binaryMessenger, CHANNEL)
```

```
            .setMethodCallHandler { call, result ->
                when (call.method) {
                    "getDeviceModel" ->                                    ↪
                     ↪  result.success("${Build.MANUFACTURER}              ↪
                     ↪  ${Build.MODEL}")
                    "getOsVersion" -> result.success("Android              ↪
                     ↪  ${Build.VERSION.RELEASE}")
                    "getAvailableStorage" ->                               ↪
                     ↪  result.success(getAvailableStorageMB())
                    "isEmulator" -> result.success(isEmulator())
                    else -> result.notImplemented()
                }
            }
    }

    private fun getAvailableStorageMB(): Int {
        val stat = StatFs(Environment.getDataDirectory().path)
        val bytesAvailable = stat.blockSizeLong * stat.availableBlocksLong
        return (bytesAvailable / (1024 * 1024)).toInt()
    }

    private fun isEmulator(): Boolean {
        return Build.FINGERPRINT.contains("generic") ||
               Build.FINGERPRINT.contains("emulator") ||
               Build.MODEL.contains("Emulator") ||
               Build.MANUFACTURER.contains("Genymotion")
    }
}
```

4.5.2.4 Aufgabe 6: Plattform-spezifisches UI

```
import 'dart:io';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

/// Adaptive progress indicator.
class AdaptiveProgressIndicator extends StatelessWidget {
  const AdaptiveProgressIndicator({super.key});

  @override
  Widget build(BuildContext context) {
    if (Platform.isIOS) {
      return const CupertinoActivityIndicator();
    }
    return const CircularProgressIndicator();
  }
}

/// Adaptive switch.
class AdaptiveSwitch extends StatelessWidget {
```

```
  final bool value;
  final ValueChanged<bool>? onChanged;

  const AdaptiveSwitch({
    super.key,
    required this.value,
    this.onChanged,
  });

  @override
  Widget build(BuildContext context) {
    if (Platform.isIOS) {
      return CupertinoSwitch(value: value, onChanged: onChanged);
    }
    return Switch(value: value, onChanged: onChanged);
  }
}

/// Adaptive list tile.
class AdaptiveListTile extends StatelessWidget {
  final Widget? leading;
  final Widget title;
  final Widget? subtitle;
  final Widget? trailing;
  final VoidCallback? onTap;

  const AdaptiveListTile({
    super.key,
    this.leading,
    required this.title,
    this.subtitle,
    this.trailing,
    this.onTap,
  });

  @override
  Widget build(BuildContext context) {
    if (Platform.isIOS) {
      return CupertinoListTile(
        leading: leading,
        title: title,
        subtitle: subtitle,
        trailing: trailing ?? const CupertinoListTileChevron(),
        onTap: onTap,
      );
    }
    return ListTile(
      leading: leading,
      title: title,
      subtitle: subtitle,
      trailing: trailing,
```

```
      onTap: onTap,
    );
  }
}

/// Adaptive scaffold with navigation bar.
class AdaptiveScaffold extends StatelessWidget {
  final String title;
  final Widget body;
  final List<Widget>? actions;

  const AdaptiveScaffold({
    super.key,
    required this.title,
    required this.body,
    this.actions,
  });

  @override
  Widget build(BuildContext context) {
    if (Platform.isIOS) {
      return CupertinoPageScaffold(
        navigationBar: CupertinoNavigationBar(
          middle: Text(title),
          trailing: actions != null && actions!.isNotEmpty
              ? Row(mainAxisSize: MainAxisSize.min, children: actions!)
              : null,
        ),
        child: SafeArea(child: body),
      );
    }
    return Scaffold(
      appBar: AppBar(title: Text(title), actions: actions),
      body: body,
    );
  }
}
```

4.5.3 Ressourcen

4.5.3.1 Offizielle Dokumentation

- Using packages
- Developing packages & plugins
- Writing platform-specific code
- pub.dev
- Dart package layout

4.5.3.2 pub.dev Guides

- Publishing packages
- Package scoring

- Verified publishers

4.5.3.3 Platform Channels

- MethodChannel
- EventChannel
- Pigeon - Type-safe Platform Channels

4.5.3.4 Videos

- How to Create Flutter Packages
- Platform Channels
- Writing a Flutter Plugin

4.5.3.5 Tutorials & Artikel

- Creating packages guide
- Platform Channels Guide
- FFI for native code

4.5.3.6 Nützliche Packages

- pigeon - Code-Generator für Platform Channels
- ffi - Foreign Function Interface
- flutter_native_splash - Native Splash Screens
- permission_handler - Berechtigungen

4.5.3.7 Tools

- pana - Package Analyzer (pub.dev scoring)
- dart_doc - Dokumentation generieren
- very_good_cli - Package Templates

4.5.3.8 Zum Vertiefen

- FFI (Foreign Function Interface)
- Federated Plugins
- Background Isolates

4.6 Einheit 4.6: Build & Release

4.6.0.1 Lernziele

Nach dieser Einheit kannst du: - App-Icons und Splash Screens konfigurieren - Berechtigungen für Android und iOS setzen - Release-Builds erstellen - Apps für den Store vorbereiten

4.6.0.2 1. App-Icons

flutter_launcher_icons Package

```yaml
# pubspec.yaml
dev_dependencies:
  flutter_launcher_icons: ^0.13.1
```

```yaml
flutter_launcher_icons:
  android: true
  ios: true
  image_path: "assets/icon/app_icon.png"
  min_sdk_android: 21

  # Adaptive Icon für Android
  adaptive_icon_background: "#FFFFFF"
  adaptive_icon_foreground: "assets/icon/app_icon_foreground.png"

  # Web
  web:
    generate: true
    image_path: "assets/icon/app_icon.png"
    background_color: "#FFFFFF"
    theme_color: "#2196F3"
```

```
# Icons generieren
dart run flutter_launcher_icons
```

Manuell (Android)

```
android/app/src/main/res/
+-- mipmap-hdpi/
|   +-- ic_launcher.png       # 72x72
+-- mipmap-mdpi/
|   +-- ic_launcher.png       # 48x48
+-- mipmap-xhdpi/
|   +-- ic_launcher.png       # 96x96
+-- mipmap-xxhdpi/
|   +-- ic_launcher.png       # 144x144
+-- mipmap-xxxhdpi/
    +-- ic_launcher.png       # 192x192
```

Manuell (iOS)

```
ios/Runner/Assets.xcassets/AppIcon.appiconset/
+-- Icon-App-20x20@1x.png
+-- Icon-App-20x20@2x.png
+-- Icon-App-20x20@3x.png
+-- Icon-App-29x29@1x.png
+-- ... (alle Größen)
+-- Contents.json
```

4.6.0.3 2. Splash Screens

flutter_native_splash Package

```yaml
# pubspec.yaml
dev_dependencies:
  flutter_native_splash: ^2.3.0

flutter_native_splash:
```

```
  color: "#FFFFFF"
  image: assets/splash/logo.png

  # Android 12+ Splash
  android_12:
    color: "#FFFFFF"
    icon_background_color: "#FFFFFF"
    image: assets/splash/logo.png

  # iOS
  ios: true

  # Web
  web: true
  web_image_mode: center

  # Fullscreen (keine Statusbar)
  fullscreen: false
```

```
# Splash Screen generieren
dart run flutter_native_splash:create

# Splash entfernen (nach App-Start)
dart run flutter_native_splash:remove
```

Im Code entfernen

```
import 'package:flutter_native_splash/flutter_native_splash.dart';

void main() {
  // Splash behalten während Initialisierung
  WidgetsBinding widgetsBinding = WidgetsFlutterBinding.ensureInitialized();
  FlutterNativeSplash.preserve(widgetsBinding: widgetsBinding);

  // Initialisierung...
  await initializeApp();

  // Splash entfernen
  FlutterNativeSplash.remove();

  runApp(const MyApp());
}
```

4.6.0.4 3. App-Name & Bundle ID

Android

```
<!-- android/app/src/main/AndroidManifest.xml -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.my_app">

    <application
```

```
        android:label="Meine App"
        android:icon="@mipmap/ic_launcher">
        ...
    </application>
</manifest>
```

```
// android/app/build.gradle
android {
    defaultConfig {
        applicationId "com.mycompany.myapp"
        minSdkVersion 21
        targetSdkVersion 34
        versionCode 1
        versionName "1.0.0"
    }
}
```

iOS

```
<!-- ios/Runner/Info.plist -->
<dict>
    <key>CFBundleName</key>
    <string>Meine App</string>

    <key>CFBundleIdentifier</key>
    <string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>

    <key>CFBundleShortVersionString</key>
    <string>$(FLUTTER_BUILD_NAME)</string>

    <key>CFBundleVersion</key>
    <string>$(FLUTTER_BUILD_NUMBER)</string>
</dict>
```

In Xcode: - Runner -> General -> Identity -> Bundle Identifier

4.6.0.5 4. Berechtigungen

Android Berechtigungen

```
<!-- android/app/src/main/AndroidManifest.xml -->
<manifest>
    <!-- Internet -->
    <uses-permission android:name="android.permission.INTERNET"/>

    <!-- Kamera -->
    <uses-permission android:name="android.permission.CAMERA"/>

    <!-- Speicher -->
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

```
    <!-- Standort -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

    <!-- Mikrofon -->
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>

    <!-- Benachrichtigungen -->
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>

    <application ...>
</manifest>
```

iOS Berechtigungen

```
<!-- ios/Runner/Info.plist -->
<dict>
    <!-- Kamera -->
    <key>NSCameraUsageDescription</key>
    <string>Diese App benötigt Zugriff auf die Kamera für Fotos.</string>

    <!-- Fotobibliothek -->
    <key>NSPhotoLibraryUsageDescription</key>
    <string>Diese App benötigt Zugriff auf Fotos.</string>

    <!-- Standort -->
    <key>NSLocationWhenInUseUsageDescription</key>
    <string>Diese App benötigt Ihren Standort für...</string>

    <key>NSLocationAlwaysUsageDescription</key>
    <string>Diese App benötigt Standortzugriff im Hintergrund.</string>

    <!-- Mikrofon -->
    <key>NSMicrophoneUsageDescription</key>
    <string>Diese App benötigt Zugriff auf das Mikrofon.</string>

    <!-- Kontakte -->
    <key>NSContactsUsageDescription</key>
    <string>Diese App benötigt Zugriff auf Ihre Kontakte.</string>

    <!-- Face ID -->
    <key>NSFaceIDUsageDescription</key>
    <string>Diese App nutzt Face ID zur Authentifizierung.</string>
</dict>
```

Permission Handler Package

```
dependencies:
  permission_handler: ^11.0.0
```

```
import 'package:permission_handler/permission_handler.dart';

```

```
Future<void> requestCameraPermission() async {
  final status = await Permission.camera.request();

  if (status.isGranted) {
    // Berechtigung erteilt
  } else if (status.isDenied) {
    // Berechtigung abgelehnt
  } else if (status.isPermanentlyDenied) {
    // Permanent abgelehnt -> Einstellungen öffnen
    await openAppSettings();
  }
}

// Mehrere Berechtigungen
Future<void> requestPermissions() async {
  final statuses = await [
    Permission.camera,
    Permission.microphone,
    Permission.storage,
  ].request();

  if (statuses[Permission.camera]!.isGranted) {
    // Kamera OK
  }
}
```

4.6.0.6 5. Build-Konfiguration

Flavor/Build Variants

```
// android/app/build.gradle
android {
    flavorDimensions "environment"

    productFlavors {
        dev {
            dimension "environment"
            applicationIdSuffix ".dev"
            versionNameSuffix "-dev"
            resValue "string", "app_name", "MyApp Dev"
        }
        staging {
            dimension "environment"
            applicationIdSuffix ".staging"
            versionNameSuffix "-staging"
            resValue "string", "app_name", "MyApp Staging"
        }
        prod {
            dimension "environment"
            resValue "string", "app_name", "MyApp"
        }
```

```
    }
}
```

```
# Build mit Flavor
flutter build apk --flavor dev
flutter build apk --flavor prod
```

Environment Variables

```
// lib/config/environment.dart
class Environment {
  static const String apiUrl = String.fromEnvironment(
    'API_URL',
    defaultValue: 'https://api.example.com',
  );

  static const bool isProduction = bool.fromEnvironment(
    'PRODUCTION',
    defaultValue: false,
  );
}
```

```
# Build mit Variablen
flutter build apk --dart-define=API_URL=https://prod.api.com
 ↪  --dart-define=PRODUCTION=true
```

4.6.0.7 6. Release Builds

Android Release

```
# APK (Universal)
flutter build apk --release

# APK pro Architektur (kleiner)
flutter build apk --split-per-abi

# App Bundle (für Play Store)
flutter build appbundle --release
```

Keystore erstellen (Android)

```
# Keystore generieren
keytool -genkey -v -keystore ~/upload-keystore.jks -keyalg RSA -keysize 2048
 ↪  -validity 10000 -alias upload
```

```
# android/key.properties (nicht committen!)
storePassword=<password>
keyPassword=<password>
keyAlias=upload
storeFile=/path/to/upload-keystore.jks
```

```
// android/app/build.gradle
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
    keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
}

android {
    signingConfigs {
        release {
            keyAlias keystoreProperties['keyAlias']
            keyPassword keystoreProperties['keyPassword']
            storeFile keystoreProperties['storeFile'] ?
 ↪  file(keystoreProperties['storeFile']) : null
            storePassword keystoreProperties['storePassword']
        }
    }

    buildTypes {
        release {
            signingConfig signingConfigs.release
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
 ↪  'proguard-rules.pro'
        }
    }
}
```

iOS Release

```
# iOS Build (Archiv)
flutter build ios --release

# IPA für Distribution
flutter build ipa --release
```

In Xcode: 1. Product -> Archive 2. Distribute App -> App Store Connect

4.6.0.8 7. App Store Vorbereitung

Google Play Store

Benötigt: - App Bundle (.aab) - App-Icon (512x512) - Feature Graphic (1024x500) - Screenshots (mind. 2) - Beschreibung (kurz & lang) - Datenschutzerklärung URL - Content Rating (Fragebogen)

Apple App Store

Benötigt: - Build über Xcode/Transporter - App-Icon (1024x1024, ohne Alpha) - Screenshots für alle Geräte - App-Beschreibung - Keywords - Support URL - Privacy Policy URL - App Store Connect Konfiguration

App Store Screenshots

```
# Größen für iOS
- iPhone 6.7" (1290 x 2796)
- iPhone 6.5" (1284 x 2778)
- iPhone 5.5" (1242 x 2208)
- iPad Pro 12.9" (2048 x 2732)

# Größen für Android
- Phone (1080 x 1920)
- 7" Tablet (1200 x 1920)
- 10" Tablet (1800 x 2560)
```

4.6.0.9 8. ProGuard & Code Shrinking

ProGuard Rules (Android)

```
# android/app/proguard-rules.pro

# Flutter
-keep class io.flutter.** { *; }
-keep class io.flutter.plugins.** { *; }

# Firebase (falls verwendet)
-keep class com.google.firebase.** { *; }

# JSON Serialization (falls reflection)
-keepattributes *Annotation*
-keep class * extends com.google.gson.TypeAdapter
```

Build-Optimierungen

```
// android/app/build.gradle
android {
    buildTypes {
        release {
            minifyEnabled true     // Code Shrinking
            shrinkResources true   // Resource Shrinking
            proguardFiles
 ↪  getDefaultProguardFile('proguard-android-optimize.txt'),
                          'proguard-rules.pro'
        }
    }
}
```

4.6.0.10 9. CI/CD Basics

GitHub Actions

```
# .github/workflows/build.yml
name: Build

on:
  push:
```

```
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  build-android:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - uses: subosito/flutter-action@v2
        with:
          flutter-version: '3.16.0'

      - run: flutter pub get
      - run: flutter test
      - run: flutter build apk --release

      - uses: actions/upload-artifact@v3
        with:
          name: release-apk
          path: build/app/outputs/flutter-apk/app-release.apk

  build-ios:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v3

      - uses: subosito/flutter-action@v2
        with:
          flutter-version: '3.16.0'

      - run: flutter pub get
      - run: flutter test
      - run: flutter build ios --release --no-codesign
```

Fastlane (Optional)

```
# ios/fastlane/Fastfile
default_platform(:ios)

platform :ios do
  desc "Deploy to TestFlight"
  lane :beta do
    build_app(scheme: "Runner")
    upload_to_testflight
  end
end
```

4.6.0.11 10. Debugging Release Builds

Symbolicate Crash Reports

```
# Android - Mapping-Datei
build/app/outputs/mapping/release/mapping.txt

# iOS - dSYM Dateien
build/ios/archive/Runner.xcarchive/dSYMs/
```

Release mit Debug-Info

```
# Android
flutter build apk --release --obfuscate --split-debug-info=build/debug-info

# iOS
flutter build ios --release --obfuscate --split-debug-info=build/debug-info
```

4.6.0.12 Zusammenfassung

Thema	Tool/Methode
App Icons	`flutter_launcher_icons`
Splash Screen	`flutter_native_splash`
Berechtigungen	`permission_handler`
Android Build	`flutter build apk/appbundle`
iOS Build	`flutter build ios/ipa`
Signing	Keystore (Android), Xcode (iOS)
CI/CD	GitHub Actions, Fastlane

Checkliste vor Release: - [ ] App-Icon in allen Größen - [ ] Splash Screen konfiguriert - [ ] Alle Berechtigungen mit Beschreibung - [ ] Release-Signatur eingerichtet - [ ] ProGuard konfiguriert - [ ] Tests bestanden - [ ] Store-Beschreibung & Screenshots - [ ] Datenschutzerklärung verlinkt

4.6.1 Übung

4.6.1.1 Ziel

Eine App für den Release vorbereiten und Store-tauglich machen.

4.6.1.2 Aufgabe 1: App-Icon konfigurieren (20 min)

1. Erstelle ein App-Icon (mindestens 1024x1024 px)
   - Verwende ein Design-Tool oder einen Generator
2. Konfiguriere `flutter_launcher_icons`:
   - Standard-Icon für iOS und Android
   - Adaptive Icon für Android
   - Foreground und Background
3. Generiere die Icons:

```
dart run flutter_launcher_icons
```

4. Überprüfe die generierten Dateien.

4.6.1.3 Aufgabe 2: Splash Screen einrichten (20 min)

1. Erstelle ein Splash-Logo (PNG mit Transparenz)
2. Konfiguriere `flutter_native_splash`:
    - Hintergrundfarbe passend zum Design
    - Logo zentriert
    - Android 12+ Konfiguration
    - Fullscreen-Option testen
3. Implementiere das Entfernen des Splash nach App-Start:

```
// Nach Initialisierung:
FlutterNativeSplash.remove();
```

4.6.1.4 Aufgabe 3: App-Metadaten konfigurieren (15 min)

Setze folgende Werte für deine App:

Android

- Application ID: `com.deinname.appname`
- App-Name (Label)
- Version: 1.0.0
- Build Number: 1

iOS

- Bundle Identifier
- Display Name
- Version & Build

4.6.1.5 Aufgabe 4: Berechtigungen einrichten (25 min)

Implementiere folgende Berechtigungen:

1. **Internet** (Standard)
2. **Kamera**
    - Android: Manifest
    - iOS: Info.plist mit deutscher Beschreibung
3. **Standort**
    - “When in use” Berechtigung
    - Sinnvolle Beschreibung
4. Implementiere Permission-Request im Code:

```
// Beim ersten Kamera-Zugriff
Future<bool> requestCameraPermission() async {
  // ...
}
```

5. Handle alle Fälle:
    - Granted
    - Denied
    - Permanently Denied -> Settings öffnen

4.6.1.6 Aufgabe 5: Build-Varianten (20 min)

Erstelle zwei Build-Varianten:

Development

- App ID: `com.example.app.dev`
- App Name: "MyApp Dev"
- API URL: `https://dev-api.example.com`

Production

- App ID: `com.example.app`
- App Name: "MyApp"
- API URL: `https://api.example.com`

Nutze: - Android Flavors - `--dart-define` für Variablen

4.6.1.7 Aufgabe 6: Release Build erstellen (25 min)

Android

1. Erstelle einen Keystore:

```
keytool -genkey -v -keystore my-release-key.jks ...
```

2. Konfiguriere `key.properties`
3. Konfiguriere `build.gradle` für Signing
4. Erstelle Release-Builds:

```
flutter build apk --release
flutter build appbundle --release
```

5. Notiere die Dateigrößen

iOS (falls Mac verfügbar)

1. Öffne Xcode und konfiguriere Signing
2. Erstelle einen Archive-Build
3. Exportiere für Ad-Hoc Distribution

4.6.1.8 Aufgabe 7: Store-Assets vorbereiten (20 min)

Erstelle folgende Assets:

Screenshots

- Mindestens 3 Screenshots
- In korrekten Größen
- Mit App-Inhalt (nicht Splash)

Beschreibungstexte

- Kurzbeschreibung (80 Zeichen)
- Vollständige Beschreibung (4000 Zeichen)
- Keywords/Tags

Grafiken

- Feature Graphic für Play Store (1024x500)

- Promotional Text

4.6.1.9 Aufgabe 8: Pre-Release Checkliste (15 min)

Gehe die folgende Checkliste durch:

Code-Qualität

- ☐ Keine Debug-Prints im Code
- ☐ Keine hardcodierten Test-URLs
- ☐ Keine Test-Accounts
- ☐ Error Handling vollständig

Konfiguration

- ☐ Korrekter App-Name
- ☐ Korrekte Bundle/Package ID
- ☐ Version und Build Number gesetzt
- ☐ Berechtigungen minimiert

Assets

- ☐ App-Icon in allen Größen
- ☐ Splash Screen konfiguriert
- ☐ Screenshots erstellt

Testing

- ☐ Release-Build getestet
- ☐ Auf echtem Gerät getestet
- ☐ Kritische Flows manuell geprüft

Store

- ☐ Datenschutzerklärung URL
- ☐ Support-Kontakt
- ☐ Beschreibungstexte

4.6.1.10 Bonus: CI/CD Pipeline

Erstelle eine GitHub Actions Workflow-Datei:

```
# .github/workflows/release.yml

# Trigger: Bei Tag mit v* (z.B. v1.0.0)
# Jobs:
# 1. Tests ausführen
# 2. Android APK bauen
# 3. Artifacts hochladen
```

4.6.1.11 Abgabe-Checkliste

- ☐ App-Icon generiert und sichtbar
- ☐ Splash Screen funktioniert
- ☐ App-Name und ID konfiguriert
- ☐ Berechtigungen eingerichtet
- ☐ Permission-Request implementiert

- ☐ Build-Varianten konfiguriert
- ☐ Release-APK erstellt
- ☐ Store-Assets vorbereitet
- ☐ Pre-Release Checkliste durchgegangen

4.6.2 Lösung

4.6.2.1 Aufgabe 1 & 2: App-Icon und Splash Screen

pubspec.yaml

```yaml
dev_dependencies:
  flutter_launcher_icons: ^0.13.1
  flutter_native_splash: ^2.3.0

# App Icon Konfiguration
flutter_launcher_icons:
  android: true
  ios: true
  image_path: "assets/icon/app_icon.png"
  min_sdk_android: 21

  # Adaptive Icon (Android)
  adaptive_icon_background: "#2196F3"
  adaptive_icon_foreground: "assets/icon/app_icon_foreground.png"

  # Web
  web:
    generate: true
    image_path: "assets/icon/app_icon.png"
    background_color: "#2196F3"
    theme_color: "#2196F3"

# Splash Screen Konfiguration
flutter_native_splash:
  color: "#2196F3"
  image: assets/splash/logo.png

  android_12:
    color: "#2196F3"
    icon_background_color: "#2196F3"
    image: assets/splash/logo.png

  ios: true
  web: true
  fullscreen: false
```

Icons und Splash generieren

```
# Icons generieren
dart run flutter_launcher_icons

# Splash Screen generieren
```

```
dart run flutter_native_splash:create
```

Splash im Code entfernen

```
// lib/main.dart
import 'package:flutter_native_splash/flutter_native_splash.dart';

void main() async {
  // Splash behalten
  WidgetsBinding widgetsBinding = WidgetsFlutterBinding.ensureInitialized();
  FlutterNativeSplash.preserve(widgetsBinding: widgetsBinding);

  // App initialisieren
  await _initializeApp();

  // Splash entfernen
  FlutterNativeSplash.remove();

  runApp(const MyApp());
}

Future<void> _initializeApp() async {
  // SharedPreferences laden
  // Firebase initialisieren
  // etc.
  await Future.delayed(const Duration(seconds: 1));
}
```

4.6.2.2 Aufgabe 3: App-Metadaten

Android - android/app/build.gradle

```
android {
    namespace "com.deinname.appname"
    compileSdkVersion 34

    defaultConfig {
        applicationId "com.deinname.appname"
        minSdkVersion 21
        targetSdkVersion 34
        versionCode 1
        versionName "1.0.0"
    }
}
```

Android - AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <application
        android:label="Meine App"
        android:icon="@mipmap/ic_launcher"
        android:roundIcon="@mipmap/ic_launcher_round">
```

```
        ...
    </application>
</manifest>
```

iOS - Info.plist

```
<dict>
    <key>CFBundleDisplayName</key>
    <string>Meine App</string>

    <key>CFBundleIdentifier</key>
    <string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>

    <key>CFBundleShortVersionString</key>
    <string>$(FLUTTER_BUILD_NAME)</string>

    <key>CFBundleVersion</key>
    <string>$(FLUTTER_BUILD_NUMBER)</string>
</dict>
```

4.6.2.3 Aufgabe 4: Berechtigungen

Android - AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- Internet -->
    <uses-permission android:name="android.permission.INTERNET"/>

    <!-- Kamera -->
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-feature android:name="android.hardware.camera"
    android:required="false"/>

    <!-- Standort -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

    <application ...>
    </application>
</manifest>
```

iOS - Info.plist

```
<dict>
    <!-- Kamera -->
    <key>NSCameraUsageDescription</key>
    <string>Diese App benötigt Zugriff auf die Kamera, um Fotos
    aufzunehmen.</string>

    <!-- Standort -->
    <key>NSLocationWhenInUseUsageDescription</key>
```

```
    <string>Diese App benötigt Ihren Standort, um Ihnen relevante Inhalte in
↪   Ihrer Nähe anzuzeigen.</string>

    <key>NSLocationAlwaysAndWhenInUseUsageDescription</key>
    <string>Diese App benötigt Ihren Standort auch im Hintergrund für
↪   Benachrichtigungen.</string>
</dict>
```

Permission Handler Implementation

```
// lib/services/permission_service.dart
import 'package:permission_handler/permission_handler.dart';

class PermissionService {
  /// Fordert Kamera-Berechtigung an
  Future<bool> requestCameraPermission() async {
    final status = await Permission.camera.status;

    if (status.isGranted) {
      return true;
    }

    if (status.isDenied) {
      final result = await Permission.camera.request();
      return result.isGranted;
    }

    if (status.isPermanentlyDenied) {
      // Dialog anzeigen und zu Einstellungen leiten
      final opened = await openAppSettings();
      if (opened) {
        // Warten bis User zurückkommt und erneut prüfen
        await Future.delayed(const Duration(seconds: 1));
        return await Permission.camera.isGranted;
      }
    }

    return false;
  }

  /// Fordert Standort-Berechtigung an
  Future<bool> requestLocationPermission() async {
    // Zuerst prüfen ob Location Services aktiviert
    final serviceEnabled = await Permission.location.serviceStatus.isEnabled;
    if (!serviceEnabled) {
      // User auffordern Location Services zu aktivieren
      return false;
    }

    var status = await Permission.locationWhenInUse.status;
```

```
    if (status.isDenied) {
      status = await Permission.locationWhenInUse.request();
    }

    if (status.isPermanentlyDenied) {
      await openAppSettings();
      return false;
    }

    return status.isGranted;
  }

  /// Prüft alle benötigten Berechtigungen
  Future<Map<Permission, PermissionStatus>> checkAllPermissions() async {
    return await [
      Permission.camera,
      Permission.locationWhenInUse,
    ].request();
  }
}

// Verwendung im Widget
class CameraButton extends StatelessWidget {
  final PermissionService _permissionService = PermissionService();

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: () async {
        final granted = await _permissionService.requestCameraPermission();
        if (granted) {
          // Kamera öffnen
          _openCamera(context);
        } else {
          // Fehlermeldung
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
              content: Text('Kamera-Berechtigung erforderlich'),
            ),
          );
        }
      },
      child: const Text('Kamera öffnen'),
    );
  }

  void _openCamera(BuildContext context) {
    // Kamera-Logik
  }
}
```

4.6.2.4 Aufgabe 5: Build-Varianten

android/app/build.gradle

```
android {
    flavorDimensions "environment"

    productFlavors {
        dev {
            dimension "environment"
            applicationIdSuffix ".dev"
            versionNameSuffix "-dev"
            resValue "string", "app_name", "MyApp Dev"
        }
        prod {
            dimension "environment"
            resValue "string", "app_name", "MyApp"
        }
    }
}
```

Environment Config

```
// lib/config/environment.dart
class Environment {
  static const String apiUrl = String.fromEnvironment(
    'API_URL',
    defaultValue: 'https://dev-api.example.com',
  );

  static const bool isProduction = bool.fromEnvironment(
    'PRODUCTION',
    defaultValue: false,
  );

  static const String appName = String.fromEnvironment(
    'APP_NAME',
    defaultValue: 'MyApp Dev',
  );
}

// Verwendung
class ApiService {
  final _baseUrl = Environment.apiUrl;

  Future<Response> get(String path) {
    return http.get(Uri.parse('$_baseUrl/$path'));
  }
}
```

Build-Befehle

```
# Development
flutter build apk --flavor dev \
  --dart-define=API_URL=https://dev-api.example.com \
  --dart-define=PRODUCTION=false

# Production
flutter build apk --flavor prod \
  --dart-define=API_URL=https://api.example.com \
  --dart-define=PRODUCTION=true
```

4.6.2.5 Aufgabe 6: Release Build

Keystore erstellen

```
keytool -genkey -v \
  -keystore ~/my-release-key.jks \
  -keyalg RSA \
  -keysize 2048 \
  -validity 10000 \
  -alias my-key-alias
```

android/key.properties

```
storePassword=<dein-store-passwort>
keyPassword=<dein-key-passwort>
keyAlias=my-key-alias
storeFile=/Users/dein-name/my-release-key.jks
```

android/app/build.gradle

```
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
    keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
}

android {
    signingConfigs {
        release {
            keyAlias keystoreProperties['keyAlias']
            keyPassword keystoreProperties['keyPassword']
            storeFile keystoreProperties['storeFile'] ?
 ↪  file(keystoreProperties['storeFile']) : null
            storePassword keystoreProperties['storePassword']
        }
    }

    buildTypes {
        release {
            signingConfig signingConfigs.release
            minifyEnabled true
            shrinkResources true
```

```
            proguardFiles                                                    ↪
↪   getDefaultProguardFile('proguard-android-optimize.txt'),                 ↪
↪   'proguard-rules.pro'
        }
    }
}
```

Build ausführen

```
# Release APK
flutter build apk --release

# Release App Bundle (für Play Store)
flutter build appbundle --release

# Mit Obfuscation
flutter build apk --release --obfuscate --split-debug-info=build/debug-info
```

4.6.2.6 Bonus: GitHub Actions

.github/workflows/release.yml

```
name: Release Build

on:
  push:
    tags:
      - 'v*'

jobs:
  build-android:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Java
        uses: actions/setup-java@v3
        with:
          distribution: 'temurin'
          java-version: '17'

      - name: Setup Flutter
        uses: subosito/flutter-action@v2
        with:
          flutter-version: '3.16.0'
          channel: 'stable'

      - name: Get dependencies
        run: flutter pub get

      - name: Run tests
        run: flutter test
```

```
      - name: Decode Keystore
        run: |
          echo "${{ secrets.KEYSTORE_BASE64 }}" | base64 --decode >
↪  android/app/keystore.jks

      - name: Create key.properties
        run: |
          echo "storePassword=${{ secrets.KEYSTORE_PASSWORD }}" >
↪  android/key.properties
          echo "keyPassword=${{ secrets.KEY_PASSWORD }}" >>
↪  android/key.properties
          echo "keyAlias=${{ secrets.KEY_ALIAS }}" >> android/key.properties
          echo "storeFile=keystore.jks" >> android/key.properties

      - name: Build APK
        run: flutter build apk --release

      - name: Build App Bundle
        run: flutter build appbundle --release

      - name: Upload APK
        uses: actions/upload-artifact@v3
        with:
          name: release-apk
          path: build/app/outputs/flutter-apk/app-release.apk

      - name: Upload AAB
        uses: actions/upload-artifact@v3
        with:
          name: release-aab
          path: build/app/outputs/bundle/release/app-release.aab

  build-ios:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Flutter
        uses: subosito/flutter-action@v2
        with:
          flutter-version: '3.16.0'

      - name: Get dependencies
        run: flutter pub get

      - name: Build iOS (no codesign)
        run: flutter build ios --release --no-codesign

      # Für echtes Signing: Certificates und Provisioning Profiles einrichten
```

4.6.3 Ressourcen

4.6.3.1 Offizielle Dokumentation

- Build and release Android
- Build and release iOS
- Obfuscating Dart code
- Flavors

4.6.3.2 Packages

- flutter_launcher_icons - App Icons
- flutter_native_splash - Splash Screens
- permission_handler - Berechtigungen
- package_info_plus - App-Version auslesen

4.6.3.3 App Stores

- Google Play Console
- Apple App Store Connect
- Play Store Guidelines
- App Store Guidelines

4.6.3.4 Videos

- Flutter Release Build
- Publish to Play Store
- Publish to App Store
- Flutter Flavors

4.6.3.5 Tutorials & Artikel

- Complete Guide to Deployment
- Android Signing
- iOS Certificates

4.6.3.6 CI/CD

- GitHub Actions for Flutter
- Fastlane
- Codemagic
- Bitrise

4.6.3.7 Tools

- App Icon Generator
- Android Asset Studio
- Screenshot Generator

4.6.3.8 Zum Vertiefen

- Google Play App Signing
- App Store Distribution
- Firebase App Distribution

4.7 Einheit 4.7: Abschlussprojekt Frontend

4.7.0.1 Projektübersicht

In diesem Abschlussprojekt entwickelst du eine vollständige Notiz-App, die alle gelernten Konzepte aus dem gesamten Kurs vereint. Das Projekt erstreckt sich über 6 Einheiten (12 Stunden).

4.7.0.2 App-Beschreibung

NoteFlow - Eine moderne Notiz-App mit folgenden Features:

- Notizen erstellen, bearbeiten, löschen
- Kategorien/Tags für Organisation
- Suche und Filter
- Lokale Speicherung (offline-first)
- Cloud-Synchronisation (optional)
- Dark/Light Mode
- Animierte UI-Übergänge

4.7.0.3 Projektphasen

Phase 1: Projektplanung & Setup (Einheit 4.7)

Ziele: - Projektstruktur anlegen - Dependencies definieren - Architektur planen - Datenmodelle entwerfen

Aufgaben: 1. Flutter-Projekt erstellen 2. Ordnerstruktur nach Clean Architecture 3. pubspec.yaml mit allen Dependencies 4. Grundlegende Datenmodelle 5. App-Theme definieren

Phase 2: UI & Navigation (Einheit 4.8)

Ziele: - Alle Screens implementieren - Navigation einrichten - Responsive Layouts - Wiederverwendbare Widgets

Screens: - Home (Notizliste) - Note Detail/Editor - Settings - Search - Category Management

Phase 3: State Management & Datenmodelle (Einheit 4.9)

Ziele: - Provider/Riverpod Setup - State-Klassen implementieren - Business-Logik - Repository-Pattern

Komponenten: - NotesNotifier/Provider - CategoriesNotifier - SettingsNotifier - SearchState

Phase 4: API-Anbindung & lokale Speicherung (Einheit 4.10)

Ziele: - Lokale Datenbank (Hive/SQLite) - REST API Service (optional) - Offline-First Strategie - Sync-Logik

Implementierung: - LocalStorageService - ApiService (Mock oder real) - SyncService - Error Handling

Phase 5: Formulare & Validierung (Einheit 4.11)

Ziele: - Note Editor mit Validierung - Settings-Formular - Kategorie-Dialog - Input-Handling

Features: - Titel-Validierung (nicht leer) - Auto-Save (Debouncing) - Unsaved Changes Detection - Confirmation Dialogs

Phase 6: Animationen, Tests & Feinschliff (Einheit 4.12)

Ziele: - UI-Animationen - Unit & Widget Tests - Performance-Optimierung - Release-Vorbereitung

Aufgaben: - Hero-Animationen für Notizen - List-Animationen - Staggered Einblendungen - Mindestens 80% Test-Coverage

4.7.0.4 Architektur

Ordnerstruktur

```
lib/
+-- main.dart
+-- app.dart
+-- config/
|   +-- routes.dart
|   +-- theme.dart
|   +-- constants.dart
+-- models/
|   +-- note.dart
|   +-- category.dart
|   +-- user_settings.dart
+-- providers/
|   +-- notes_provider.dart
|   +-- categories_provider.dart
|   +-- settings_provider.dart
+-- services/
|   +-- storage_service.dart
|   +-- api_service.dart
|   +-- sync_service.dart
+-- repositories/
|   +-- notes_repository.dart
|   +-- categories_repository.dart
+-- screens/
|   +-- home/
|   |   +-- home_screen.dart
|   |   +-- widgets/
|   +-- editor/
|   |   +-- editor_screen.dart
|   |   +-- widgets/
|   +-- settings/
|   |   +-- settings_screen.dart
|   +-- search/
|       +-- search_screen.dart
+-- widgets/
|   +-- note_card.dart
|   +-- category_chip.dart
|   +-- empty_state.dart
|   +-- loading_indicator.dart
+-- utils/
    +-- date_formatter.dart
    +-- validators.dart
```

Datenfluss

```
UI (Screens/Widgets)
        ↓ ↑
    Providers
        ↓ ↑
   Repositories
        ↓ ↑
    Services
   ↙       ↘
Local DB    API
```

4.7.0.5 Datenmodelle

Note

```
@freezed
class Note with _$Note {
  const factory Note({
    required String id,
    required String title,
    required String content,
    required DateTime createdAt,
    required DateTime updatedAt,
    String? categoryId,
    @Default(false) bool isPinned,
    @Default(false) bool isArchived,
    @Default([]) List<String> tags,
  }) = _Note;

  factory Note.fromJson(Map<String, dynamic> json) => _$NoteFromJson(json);
}
```

Category

```
@freezed
class Category with _$Category {
  const factory Category({
    required String id,
    required String name,
    required int color,
    @Default(0) int noteCount,
  }) = _Category;

  factory Category.fromJson(Map<String, dynamic> json) =>
      _$CategoryFromJson(json);
}
```

UserSettings

```
@freezed
class UserSettings with _$UserSettings {
  const factory UserSettings({
    @Default(ThemeMode.system) ThemeMode themeMode,
    @Default(SortOrder.updatedDesc) SortOrder defaultSortOrder,
```

```
    @Default(true) bool showPinnedFirst,
    @Default(true) bool confirmDelete,
    @Default(false) bool syncEnabled,
  }) = _UserSettings;
}
```

4.7.0.6 Bewertungskriterien

Funktionalität (40%)

- ☐ CRUD für Notizen funktioniert
- ☐ Kategorien können verwaltet werden
- ☐ Suche/Filter funktioniert
- ☐ Offline-Speicherung funktioniert
- ☐ Settings werden persistiert

Code-Qualität (25%)

- ☐ Clean Architecture eingehalten
- ☐ Separation of Concerns
- ☐ Keine Code-Duplikation
- ☐ Sinnvolle Benennung
- ☐ Kommentare wo nötig

UI/UX (20%)

- ☐ Konsistentes Design
- ☐ Responsive Layouts
- ☐ Sinnvolle Animationen
- ☐ Gute Error States
- ☐ Loading States

Testing (15%)

- ☐ Unit Tests für Models
- ☐ Unit Tests für Repositories
- ☐ Widget Tests für wichtige Screens
- ☐ Mindestens 70% Coverage

4.7.0.7 Zeitplan

Einheit	Fokus	Dauer
4.7	Projektplanung & Setup	2h
4.8	UI & Navigation	2h
4.9	State Management	2h
4.10	Daten & API	2h
4.11	Formulare	2h
4.12	Animationen & Tests	2h

4.7.0.8 Tipps

1. **Iterativ arbeiten** - Erst Grundfunktion, dann erweitern
2. **Commit oft** - Nach jeder funktionierenden Feature

3. **Tests parallel** - Nicht alles am Ende
4. **Keep it simple** - Lieber weniger Features, dafür robust
5. **Dokumentieren** - Für dich selbst in der Zukunft

4.7.1 Übung

4.7.1.1 Projektbeschreibung

Entwickle eine vollständige Notiz-App "NoteFlow" mit den unten beschriebenen Features. Das Projekt erstreckt sich über 6 Einheiten (12 Stunden Arbeitszeit).

4.7.1.2 Einheit 4.7: Projektplanung & Setup (2h)

Aufgaben

1. **Projekt erstellen**

```
flutter create --org com.deinname noteflow
```

2. **Ordnerstruktur anlegen**
   - Erstelle die Verzeichnisse gemäß der Architektur
   - `lib/models/`, `lib/providers/`, `lib/services/`, etc.
3. **Dependencies hinzufügen**

```
# Füge hinzu:
# - State Management (provider oder riverpod)
# - Routing (go_router)
# - Lokale DB (hive_flutter)
# - Utilities (uuid, intl)
# - Testing (mocktail)
```

4. **Datenmodelle erstellen**
   - `Note` Modell mit allen Properties
   - `Category` Modell
   - `UserSettings` Modell
   - JSON-Serialisierung
5. **Theme definieren**
   - Light Theme
   - Dark Theme
   - Gemeinsame Farben/Styles

Abgabe

☐ Lauffähiges Projekt
☐ Alle Dependencies installiert
☐ Ordnerstruktur vorhanden
☐ Models implementiert
☐ Theme konfiguriert

4.7.1.3 Einheit 4.8: UI & Navigation (2h)

Aufgaben

1. **Navigation einrichten**
   - go_router konfigurieren
   - Routes definieren
   - Deep Links vorbereiten
2. **Home Screen**

```
+---------------------------------+
| NoteFlow              □  ⚙  |
+---------------------------------+
| [Alle] [Arbeit] [Privat] [+]   |
+---------------------------------+
| □ Wichtige Notiz              |
|    Letzte Änderung: Heute      |
+---------------------------------+
| Meeting Notes                   |
|    Letzte Änderung: Gestern    |
+---------------------------------+
| Einkaufsliste                  |
|    Letzte Änderung: 12.03.     |
+---------------------------------+
                            [+]
```

3. **Note Editor Screen**

```
+---------------------------------+
| <-  Bearbeiten           □  □ |
+---------------------------------+
| Titel                          |
| +----------------------------+  |
| | Meine Notiz              |  |
| +----------------------------+  |
|                                 |
| Inhalt                         |
| +----------------------------+  |
| | Lorem ipsum dolor sit    |  |
| | amet, consectetur...     |  |
| |                          |  |
| +----------------------------+  |
|                                 |
| Kategorie: [Arbeit ▼]          |
| Tags: [meeting] [wichtig] [+]  |
+---------------------------------+
```

4. **Settings Screen**
   - Theme-Auswahl
   - Sortierung
   - Sync-Option (UI only)
5. **Wiederverwendbare Widgets**
   - NoteCard
   - CategoryChip
   - EmptyState

- LoadingIndicator

Abgabe

- ☐ Alle Screens implementiert
- ☐ Navigation funktioniert
- ☐ Responsive Layout
- ☐ Widgets extrahiert

4.7.1.4 Einheit 4.9: State Management (2h)

Aufgaben

1. **Provider Setup**
   - ProviderScope in main.dart
   - Alle Provider definieren
2. **NotesProvider**

```
// Funktionen:
- loadNotes()
- addNote(Note)
- updateNote(Note)
- deleteNote(String id)
- togglePin(String id)
- archiveNote(String id)
- searchNotes(String query)
- filterByCategory(String? categoryId)
```

3. **CategoriesProvider**

```
// Funktionen:
- loadCategories()
- addCategory(Category)
- updateCategory(Category)
- deleteCategory(String id)
```

4. **SettingsProvider**

```
// Funktionen:
- loadSettings()
- updateThemeMode(ThemeMode)
- updateSortOrder(SortOrder)
- toggleConfirmDelete()
```

5. **Repository-Pattern**
   - NotesRepository Interface
   - CategoriesRepository Interface
   - Implementierungen

Abgabe

- ☐ Alle Provider implementiert
- ☐ UI reagiert auf State-Änderungen
- ☐ Repository-Pattern umgesetzt

☐ Separation of Concerns

4.7.1.5 Einheit 4.10: Daten & API (2h)

Aufgaben

1. **Hive Setup**

```
// In main.dart:
await Hive.initFlutter();
Hive.registerAdapter(NoteAdapter());
Hive.registerAdapter(CategoryAdapter());
```

2. **LocalStorageService**

```
class LocalStorageService {
  // Notes
  Future<List<Note>> getAllNotes();
  Future<void> saveNote(Note note);
  Future<void> deleteNote(String id);

  // Categories
  Future<List<Category>> getAllCategories();
  Future<void> saveCategory(Category category);

  // Settings
  Future<UserSettings> getSettings();
  Future<void> saveSettings(UserSettings settings);
}
```

3. **ApiService (Mock)**

```
class ApiService {
  // Simuliere API-Calls mit Delay
  Future<List<Note>> fetchNotes();
  Future<Note> createNote(Note note);
  Future<Note> updateNote(Note note);
  Future<void> deleteNote(String id);
}
```

4. **Repository Implementation**
   - Lokale Daten als primäre Quelle
   - API als sekundäre Quelle (optional)
   - Offline-Erkennung

5. **Error Handling**
   - Try/Catch in Services
   - Error States in Providern
   - User-Feedback bei Fehlern

Abgabe

☐ Daten werden lokal gespeichert
☐ Daten überleben App-Neustart

- ☐ API-Service (Mock) funktioniert
- ☐ Error Handling implementiert

4.7.1.6 Einheit 4.11: Formulare & Validierung (2h)

Aufgaben

1. **Note Editor Form**

```
// Validierung:
- Titel: Nicht leer, max 100 Zeichen
- Inhalt: Optional, max 10000 Zeichen
- Kategorie: Optional
```

2. **Auto-Save mit Debouncing**

```
// Nach 2 Sekunden ohne Eingabe speichern
// Indikator anzeigen beim Speichern
```

3. **Unsaved Changes Detection**

```
// Bei Zurück-Navigation:
// - Prüfen ob Änderungen vorhanden
// - Dialog "Änderungen verwerfen?"
```

4. **Category Dialog**

```
+---------------------------------+
| Neue Kategorie                  |
+---------------------------------+
| Name: [_________________]       |
|                                 |
| Farbe: □ □ □ □ □ □      |
|                                 |
|    [Abbrechen]  [Speichern]    |
+---------------------------------+
```

5. **Delete Confirmation**
   - Einstellung beachten
   - Dialog mit Notiz-Titel
   - Undo via Snackbar (optional)

Abgabe

- ☐ Formular mit Validierung
- ☐ Auto-Save funktioniert
- ☐ Unsaved Changes Dialog
- ☐ Category CRUD mit Dialog
- ☐ Delete Confirmation

4.7.1.7 Einheit 4.12: Animationen, Tests & Feinschliff (2h)

Aufgaben

1. **Animationen**
   - Hero-Animation für Note-Karten

- AnimatedList für Notizliste
- Staggered Einblendung beim Laden
- Page Transitions

2. **Unit Tests**

```
// Teste:
- Note Model (fromJson, toJson)
- NotesRepository
- Validators
```

3. **Widget Tests**

```
// Teste:
- NoteCard Widget
- Home Screen (mit Mock-Daten)
- Editor Screen (Formular)
```

4. **Feinschliff**

- Loading States überall
- Empty States
- Error States
- Pull-to-Refresh

5. **Release-Vorbereitung**

- App-Icon
- Splash Screen
- Release Build testen

Abgabe

☐ Animationen implementiert
☐ Unit Tests (min. 5)
☐ Widget Tests (min. 3)
☐ Code Coverage > 70%
☐ Release Build funktioniert

4.7.1.8 Gesamtabgabe

Pflicht-Features

☐ Notizen erstellen/bearbeiten/löschen
☐ Kategorien verwalten
☐ Suche funktioniert
☐ Offline-Speicherung
☐ Dark/Light Mode
☐ Grundlegende Animationen
☐ Tests vorhanden

Bonus-Features (optional)

☐ Cloud-Sync mit Firebase/Supabase
☐ Markdown-Unterstützung
☐ Bilder in Notizen
☐ Export als PDF

- ☐ Widgets auf Homescreen
- ☐ Share-Funktion

Dokumentation

- ☐ README.md mit Setup-Anleitung
- ☐ Screenshots der App
- ☐ Architektur-Beschreibung

4.7.1.9 Hinweise

- Arbeite in kleinen Commits
- Teste regelmäßig auf echtem Gerät
- Bei Problemen: Dokumentation konsultieren
- Code Reviews mit Kollegen (falls möglich)
- Zeitmanagement beachten

4.7.2 Lösung

Diese Lösung zeigt die wichtigsten Code-Abschnitte. Eine vollständige Implementierung findest du im begleitenden Repository.

4.7.2.1 Projektstruktur

```
noteflow/
+-- lib/
|   +-- main.dart
|   +-- app.dart
|   +-- config/
|   |   +-- routes.dart
|   |   +-- theme.dart
|   +-- models/
|   |   +-- note.dart
|   |   +-- category.dart
|   +-- providers/
|   |   +-- notes_provider.dart
|   |   +-- settings_provider.dart
|   +-- services/
|   |   +-- storage_service.dart
|   +-- screens/
|   |   +-- home/
|   |   +-- editor/
|   +-- widgets/
|       +-- note_card.dart
+-- test/
    +-- ...
```

4.7.2.2 Models

lib/models/note.dart

```
import 'package:hive/hive.dart';

part 'note.g.dart';
```

```
@HiveType(typeId: 0)
class Note extends HiveObject {
  @HiveField(0)
  final String id;

  @HiveField(1)
  String title;

  @HiveField(2)
  String content;

  @HiveField(3)
  final DateTime createdAt;

  @HiveField(4)
  DateTime updatedAt;

  @HiveField(5)
  String? categoryId;

  @HiveField(6)
  bool isPinned;

  @HiveField(7)
  bool isArchived;

  Note({
    required this.id,
    required this.title,
    this.content = '',
    DateTime? createdAt,
    DateTime? updatedAt,
    this.categoryId,
    this.isPinned = false,
    this.isArchived = false,
  })  : createdAt = createdAt ?? DateTime.now(),
        updatedAt = updatedAt ?? DateTime.now();

  Note copyWith({
    String? title,
    String? content,
    String? categoryId,
    bool? isPinned,
    bool? isArchived,
  }) {
    return Note(
      id: id,
      title: title ?? this.title,
      content: content ?? this.content,
      createdAt: createdAt,
```

```
      updatedAt: DateTime.now(),
      categoryId: categoryId ?? this.categoryId,
      isPinned: isPinned ?? this.isPinned,
      isArchived: isArchived ?? this.isArchived,
    );
  }

  Map<String, dynamic> toJson() => {
        'id': id,
        'title': title,
        'content': content,
        'createdAt': createdAt.toIso8601String(),
        'updatedAt': updatedAt.toIso8601String(),
        'categoryId': categoryId,
        'isPinned': isPinned,
        'isArchived': isArchived,
      };

  factory Note.fromJson(Map<String, dynamic> json) => Note(
        id: json['id'],
        title: json['title'],
        content: json['content'] ?? '',
        createdAt: DateTime.parse(json['createdAt']),
        updatedAt: DateTime.parse(json['updatedAt']),
        categoryId: json['categoryId'],
        isPinned: json['isPinned'] ?? false,
        isArchived: json['isArchived'] ?? false,
      );
}
```

4.7.2.3 Services

lib/services/storage_service.dart

```
import 'package:hive_flutter/hive_flutter.dart';
import '../models/note.dart';
import '../models/category.dart';

class StorageService {
  static const String _notesBox = 'notes';
  static const String _categoriesBox = 'categories';
  static const String _settingsBox = 'settings';

  late Box<Note> _notesBoxInstance;
  late Box<Category> _categoriesBoxInstance;
  late Box _settingsBoxInstance;

  Future<void> init() async {
    await Hive.initFlutter();
    Hive.registerAdapter(NoteAdapter());
    Hive.registerAdapter(CategoryAdapter());
```

```
    _notesBoxInstance = await Hive.openBox<Note>(_notesBox);
    _categoriesBoxInstance = await Hive.openBox<Category>(_categoriesBox);
    _settingsBoxInstance = await Hive.openBox(_settingsBox);
  }

  // Notes
  List<Note> getAllNotes() {
    return _notesBoxInstance.values.toList();
  }

  Future<void> saveNote(Note note) async {
    await _notesBoxInstance.put(note.id, note);
  }

  Future<void> deleteNote(String id) async {
    await _notesBoxInstance.delete(id);
  }

  Note? getNote(String id) {
    return _notesBoxInstance.get(id);
  }

  // Categories
  List<Category> getAllCategories() {
    return _categoriesBoxInstance.values.toList();
  }

  Future<void> saveCategory(Category category) async {
    await _categoriesBoxInstance.put(category.id, category);
  }

  Future<void> deleteCategory(String id) async {
    await _categoriesBoxInstance.delete(id);
  }

  // Settings
  T? getSetting<T>(String key) {
    return _settingsBoxInstance.get(key) as T?;
  }

  Future<void> setSetting(String key, dynamic value) async {
    await _settingsBoxInstance.put(key, value);
  }
}
```

4.7.2.4 Providers

lib/providers/notes_provider.dart

```
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:uuid/uuid.dart';
import '../models/note.dart';
import '../services/storage_service.dart';

final storageServiceProvider = Provider((ref) => StorageService());

final notesProvider =
    StateNotifierProvider<NotesNotifier, AsyncValue<List<Note>>>((ref) {
  return NotesNotifier(ref.read(storageServiceProvider));
});

class NotesNotifier extends StateNotifier<AsyncValue<List<Note>>> {
  final StorageService _storage;
  final _uuid = const Uuid();

  String? _searchQuery;
  String? _categoryFilter;

  NotesNotifier(this._storage) : super(const AsyncValue.loading()) {
    loadNotes();
  }

  List<Note> _allNotes = [];

  Future<void> loadNotes() async {
    state = const AsyncValue.loading();
    try {
      _allNotes = _storage.getAllNotes();
      _applyFilters();
    } catch (e, st) {
      state = AsyncValue.error(e, st);
    }
  }

  void _applyFilters() {
    var filtered = _allNotes.where((n) => !n.isArchived).toList();

    // Search filter
    if (_searchQuery != null && _searchQuery!.isNotEmpty) {
      final query = _searchQuery!.toLowerCase();
      filtered = filtered.where((n) {
        return n.title.toLowerCase().contains(query) ||
            n.content.toLowerCase().contains(query);
      }).toList();
    }

    // Category filter
    if (_categoryFilter != null) {
      filtered = filtered.where((n) => n.categoryId == _categoryFilter).toList();
    }
```

```
  // Sort: pinned first, then by updatedAt
  filtered.sort((a, b) {
    if (a.isPinned && !b.isPinned) return -1;
    if (!a.isPinned && b.isPinned) return 1;
    return b.updatedAt.compareTo(a.updatedAt);
  });

  state = AsyncValue.data(filtered);
}

Future<Note> addNote({String? title, String? content}) async {
  final note = Note(
    id: _uuid.v4(),
    title: title ?? 'Neue Notiz',
    content: content ?? '',
  );

  await _storage.saveNote(note);
  _allNotes.add(note);
  _applyFilters();

  return note;
}

Future<void> updateNote(Note note) async {
  final updated = note.copyWith();
  await _storage.saveNote(updated);

  final index = _allNotes.indexWhere((n) => n.id == note.id);
  if (index != -1) {
    _allNotes[index] = updated;
  }
  _applyFilters();
}

Future<void> deleteNote(String id) async {
  await _storage.deleteNote(id);
  _allNotes.removeWhere((n) => n.id == id);
  _applyFilters();
}

void togglePin(String id) {
  final index = _allNotes.indexWhere((n) => n.id == id);
  if (index != -1) {
    final note = _allNotes[index];
    updateNote(note.copyWith(isPinned: !note.isPinned));
  }
}

void search(String query) {
```

```
    _searchQuery = query;
    _applyFilters();
  }

  void filterByCategory(String? categoryId) {
    _categoryFilter = categoryId;
    _applyFilters();
  }
}
```

4.7.2.5 Screens

lib/screens/home/home_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';
import '../../providers/notes_provider.dart';
import '../../widgets/note_card.dart';

class HomeScreen extends ConsumerWidget {
  const HomeScreen({super.key});

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final notesAsync = ref.watch(notesProvider);

    return Scaffold(
      appBar: AppBar(
        title: const Text('NoteFlow'),
        actions: [
          IconButton(
            icon: const Icon(Icons.search),
            onPressed: () => context.push('/search'),
          ),
          IconButton(
            icon: const Icon(Icons.settings),
            onPressed: () => context.push('/settings'),
          ),
        ],
      ),
      body: notesAsync.when(
        loading: () => const Center(child: CircularProgressIndicator()),
        error: (error, _) => Center(child: Text('Fehler: $error')),
        data: (notes) {
          if (notes.isEmpty) {
            return const _EmptyState();
          }

          return RefreshIndicator(
            onRefresh: () => ref.read(notesProvider.notifier).loadNotes(),
```

```
            child: ListView.builder(
              padding: const EdgeInsets.all(16),
              itemCount: notes.length,
              itemBuilder: (context, index) {
                final note = notes[index];
                return Padding(
                  padding: const EdgeInsets.only(bottom: 12),
                  child: NoteCard(
                    note: note,
                    onTap: () => context.push('/note/${note.id}'),
                    onTogglePin: () =>
                        ref.read(notesProvider.notifier).togglePin(note.id),
                  ),
                );
              },
            ),
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          final note = await ref.read(notesProvider.notifier).addNote();
          if (context.mounted) {
            context.push('/note/${note.id}');
          }
        },
        child: const Icon(Icons.add),
      ),
    );
  }
}

class _EmptyState extends StatelessWidget {
  const _EmptyState();

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(
            Icons.note_outlined,
            size: 80,
            color: Colors.grey[400],
          ),
          const SizedBox(height: 16),
          Text(
            'Keine Notizen',
            style: Theme.of(context).textTheme.titleLarge?.copyWith(
                  color: Colors.grey[600],
```

```
                ),
            ),
            const SizedBox(height: 8),
            Text(
              'Tippe auf + um eine neue Notiz zu erstellen',
              style: Theme.of(context).textTheme.bodyMedium?.copyWith(
                    color: Colors.grey[500],
                ),
            ),
          ],
        ),
    );
  }
}
```

lib/screens/editor/editor_screen.dart

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';
import '../../models/note.dart';
import '../../providers/notes_provider.dart';

class EditorScreen extends ConsumerStatefulWidget {
  final String noteId;

  const EditorScreen({super.key, required this.noteId});

  @override
  ConsumerState<EditorScreen> createState() => _EditorScreenState();
}

class _EditorScreenState extends ConsumerState<EditorScreen> {
  final _titleController = TextEditingController();
  final _contentController = TextEditingController();
  final _formKey = GlobalKey<FormState>();

  Timer? _debounce;
  Note? _note;
  bool _hasUnsavedChanges = false;
  bool _isSaving = false;

  @override
  void initState() {
    super.initState();
    _loadNote();
  }

  void _loadNote() {
    final storage = ref.read(storageServiceProvider);
```

```
    _note = storage.getNote(widget.noteId);

    if (_note != null) {
      _titleController.text = _note!.title;
      _contentController.text = _note!.content;
    }
  }

  void _onChanged() {
    if (!_hasUnsavedChanges) {
      setState(() => _hasUnsavedChanges = true);
    }

    _debounce?.cancel();
    _debounce = Timer(const Duration(seconds: 2), _save);
  }

  Future<void> _save() async {
    if (_note == null || !_formKey.currentState!.validate()) return;

    setState(() => _isSaving = true);

    final updated = _note!.copyWith(
      title: _titleController.text,
      content: _contentController.text,
    );

    await ref.read(notesProvider.notifier).updateNote(updated);

    setState(() {
      _isSaving = false;
      _hasUnsavedChanges = false;
      _note = updated;
    });
  }

  Future<bool> _onWillPop() async {
    if (!_hasUnsavedChanges) return true;

    final result = await showDialog<bool>(
      context: context,
      builder: (context) => AlertDialog(
        title: const Text('Änderungen verwerfen?'),
        content: const Text(
          'Du hast ungespeicherte Änderungen. Möchtest du sie verwerfen?',
        ),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context, false),
            child: const Text('Abbrechen'),
          ),
```

```
        TextButton(
          onPressed: () => Navigator.pop(context, true),
          child: const Text('Verwerfen'),
        ),
      ],
    ),
  );

  return result ?? false;
}

Future<void> _delete() async {
  final confirmed = await showDialog<bool>(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text('Notiz löschen?'),
      content: Text('Möchtest du "${_note?.title}" wirklich löschen?'),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context, false),
          child: const Text('Abbrechen'),
        ),
        TextButton(
          onPressed: () => Navigator.pop(context, true),
          style: TextButton.styleFrom(foregroundColor: Colors.red),
          child: const Text('Löschen'),
        ),
      ],
    ),
  );

  if (confirmed == true && mounted) {
    await ref.read(notesProvider.notifier).deleteNote(widget.noteId);
    if (mounted) context.pop();
  }
}

@override
void dispose() {
  _debounce?.cancel();
  _titleController.dispose();
  _contentController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return PopScope(
    canPop: !_hasUnsavedChanges,
    onPopInvoked: (didPop) async {
      if (!didPop) {
```

```
          final shouldPop = await _onWillPop();
          if (shouldPop && mounted) {
            context.pop();
          }
        }
      },
      child: Scaffold(
        appBar: AppBar(
          title: Text(_note == null ? 'Neue Notiz' : 'Bearbeiten'),
          actions: [
            if (_isSaving)
              const Padding(
                padding: EdgeInsets.all(16),
                child: SizedBox(
                  width: 20,
                  height: 20,
                  child: CircularProgressIndicator(strokeWidth: 2),
                ),
              )
            else if (_hasUnsavedChanges)
              IconButton(
                icon: const Icon(Icons.save),
                onPressed: _save,
              ),
            IconButton(
              icon: const Icon(Icons.delete),
              onPressed: _delete,
            ),
          ],
        ),
        body: Form(
          key: _formKey,
          child: ListView(
            padding: const EdgeInsets.all(16),
            children: [
              TextFormField(
                controller: _titleController,
                decoration: const InputDecoration(
                  labelText: 'Titel',
                  border: OutlineInputBorder(),
                ),
                style: const TextStyle(
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Titel ist erforderlich';
                  }
                  if (value.length > 100) {
                    return 'Maximal 100 Zeichen';
```

```
                }
                return null;
              },
              onChanged: (_) => _onChanged(),
            ),
            const SizedBox(height: 16),
            TextFormField(
              controller: _contentController,
              decoration: const InputDecoration(
                labelText: 'Inhalt',
                border: OutlineInputBorder(),
                alignLabelWithHint: true,
              ),
              maxLines: null,
              minLines: 10,
              onChanged: (_) => _onChanged(),
            ),
          ],
        ),
      ),
    ),
  );
  }
}
```

4.7.2.6 Widgets

lib/widgets/note_card.dart

```
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import '../models/note.dart';

class NoteCard extends StatelessWidget {
  final Note note;
  final VoidCallback onTap;
  final VoidCallback onTogglePin;

  const NoteCard({
    super.key,
    required this.note,
    required this.onTap,
    required this.onTogglePin,
  });

  String _formatDate(DateTime date) {
    final now = DateTime.now();
    final diff = now.difference(date);

    if (diff.inDays == 0) {
      return 'Heute, ${DateFormat.Hm().format(date)}';
```

```
    } else if (diff.inDays == 1) {
      return 'Gestern';
    } else if (diff.inDays < 7) {
      return DateFormat.EEEE('de').format(date);
    } else {
      return DateFormat.yMd('de').format(date);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: note.isPinned ? 4 : 1,
      child: InkWell(
        onTap: onTap,
        borderRadius: BorderRadius.circular(12),
        child: Padding(
          padding: const EdgeInsets.all(16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Row(
                children: [
                  if (note.isPinned)
                    Padding(
                      padding: const EdgeInsets.only(right: 8),
                      child: Icon(
                        Icons.push_pin,
                        size: 16,
                        color: Theme.of(context).colorScheme.primary,
                      ),
                    ),
                  Expanded(
                    child: Text(
                      note.title,
                      style: const TextStyle(
                        fontSize: 16,
                        fontWeight: FontWeight.bold,
                      ),
                      maxLines: 1,
                      overflow: TextOverflow.ellipsis,
                    ),
                  ),
                  IconButton(
                    icon: Icon(
                      note.isPinned ? Icons.push_pin : Icons.push_pin_outlined,
                      size: 20,
                    ),
                    onPressed: onTogglePin,
                    visualDensity: VisualDensity.compact,
                  ),
```

```
                ],
              ),
              if (note.content.isNotEmpty) ...[
                const SizedBox(height: 8),
                Text(
                  note.content,
                  maxLines: 2,
                  overflow: TextOverflow.ellipsis,
                  style: TextStyle(
                    color: Colors.grey[600],
                  ),
                ),
              ],
              const SizedBox(height: 8),
              Text(
                _formatDate(note.updatedAt),
                style: TextStyle(
                  fontSize: 12,
                  color: Colors.grey[500],
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

4.7.2.7 Tests

test/models/note_test.dart

```
import 'package:flutter_test/flutter_test.dart';
import 'package:noteflow/models/note.dart';

void main() {
  group('Note', () {
    test('creates with required fields', () {
      final note = Note(id: '1', title: 'Test');

      expect(note.id, '1');
      expect(note.title, 'Test');
      expect(note.content, '');
      expect(note.isPinned, false);
    });

    test('copyWith updates fields', () {
      final note = Note(id: '1', title: 'Original');
      final updated = note.copyWith(title: 'Updated', isPinned: true);
```

```
      expect(updated.title, 'Updated');
      expect(updated.isPinned, true);
      expect(updated.id, '1');
    });

    test('toJson and fromJson', () {
      final note = Note(
        id: '1',
        title: 'Test',
        content: 'Content',
        isPinned: true,
      );

      final json = note.toJson();
      final restored = Note.fromJson(json);

      expect(restored.id, note.id);
      expect(restored.title, note.title);
      expect(restored.isPinned, note.isPinned);
    });
  });
}
```

test/widgets/note_card_test.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:noteflow/models/note.dart';
import 'package:noteflow/widgets/note_card.dart';

void main() {
  group('NoteCard', () {
    testWidgets('displays note title', (tester) async {
      final note = Note(id: '1', title: 'Test Title');

      await tester.pumpWidget(MaterialApp(
        home: Scaffold(
          body: NoteCard(
            note: note,
            onTap: () {},
            onTogglePin: () {},
          ),
        ),
      ));

      expect(find.text('Test Title'), findsOneWidget);
    });

    testWidgets('shows pin icon when pinned', (tester) async {
      final note = Note(id: '1', title: 'Test', isPinned: true);
```

```
      await tester.pumpWidget(MaterialApp(
        home: Scaffold(
          body: NoteCard(
            note: note,
            onTap: () {},
            onTogglePin: () {},
          ),
        ),
      ));

      expect(find.byIcon(Icons.push_pin), findsOneWidget);
    });

    testWidgets('calls onTap when tapped', (tester) async {
      bool tapped = false;
      final note = Note(id: '1', title: 'Test');

      await tester.pumpWidget(MaterialApp(
        home: Scaffold(
          body: NoteCard(
            note: note,
            onTap: () => tapped = true,
            onTogglePin: () {},
          ),
        ),
      ));

      await tester.tap(find.byType(NoteCard));
      expect(tapped, true);
    });
  });
}
```

4.7.3 Ressourcen

4.7.3.1 Architektur & Patterns

- Flutter App Architecture
- Clean Architecture in Flutter
- Repository Pattern

4.7.3.2 State Management

- Riverpod Documentation
- Provider Documentation
- Flutter State Management Guide

4.7.3.3 Empfohlene Packages

Core

```yaml
dependencies:
  flutter_riverpod: ^2.4.0  # oder provider: ^6.1.0
  go_router: ^12.0.0
  hive_flutter: ^1.1.0
  uuid: ^4.2.0
  intl: ^0.18.0
```

UI

```yaml
dependencies:
  flutter_staggered_animations: ^1.1.0
  shimmer: ^3.0.0
  flutter_slidable: ^3.0.0
```

Utilities

```yaml
dependencies:
  freezed_annotation: ^2.4.0
  json_annotation: ^4.8.0

dev_dependencies:
  freezed: ^2.4.0
  json_serializable: ^6.7.0
  build_runner: ^2.4.0
```

4.7.3.4 Beispiel-Apps zur Inspiration

- Flutter Notes App
- Taskist
- Flutter Todos

4.7.3.5 Design-Ressourcen

- Material Design 3
- Dribbble - Notes App
- Figma Community

4.7.3.6 API (Optional)

Falls du eine echte API nutzen möchtest: - JSONPlaceholder - Firebase - Supabase

4.7.3.7 Testing

- Flutter Testing Guide
- Mocktail Package
- Widget Testing

4.7.3.8 Hilfreiche Tutorials

- Build a Notes App
- Offline-First Flutter
- Flutter Animations Deep Dive

4.7.3.9 Checklisten

Pre-Development

- ☐ User Stories definiert
- ☐ Wireframes/Mockups erstellt
- ☐ Datenmodelle entworfen
- ☐ Tech Stack entschieden

Development

- ☐ Git Repository initialisiert
- ☐ CI/CD eingerichtet (optional)
- ☐ Regelmäßige Commits
- ☐ Code Reviews (bei Team)

Pre-Release

- ☐ Alle Features getestet
- ☐ Performance geprüft
- ☐ Error Handling vollständig
- ☐ Dokumentation aktuell