

Flutter

- Widgets, Layoutsystem, State, Navigation, ...



Agenda



- Was sind Widgets?
- Wie funktioniert das Layouting?
- Stateless vs. Stateful

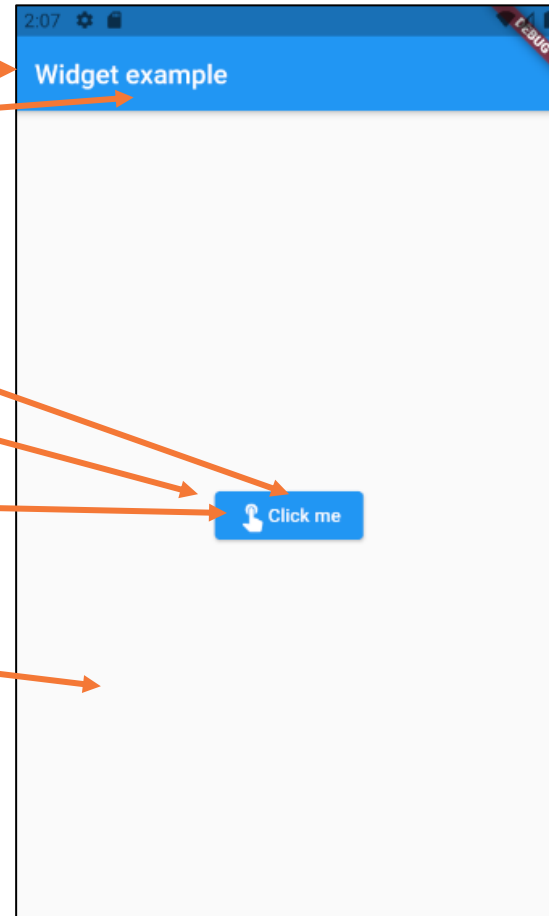
Alles ist ein Widget

- Die Bedienoberfläche setzt sich aus Widgets zusammen
- Jedes Widget besitzt eine Aufgabe:
 - Etwas anzeigen (Text, Bild, ...)
 - Layout beeinflussen (Position, Größe, ...)

```
return Scaffold(  
  appBar: AppBar(  
    title: Text("Widget example"),  
  ), // AppBar  
  body: Column(  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
      Placeholder(  
        fallbackHeight: 200,  
      ), // Placeholder  
      Padding(  
        padding: const EdgeInsets.fromLTRB(16, 16, 16, 0),  
        child: Row(  
          mainAxisAlignment: MainAxisAlignment.spaceBetween,  
          children: [  
            Text(  
              "Lorem ipsum",  
              style: TextStyle(fontSize: 35),  
            ), // Text  
            IconButton(  
              onPressed: () {},  
              icon: Icon(Icons.share),  
            ) // IconButton  
          ],  
        ), // Row  
      ), // Padding  
    ],  
  ),  
);
```

Alles ist ein Widget

- AppBar
- Text
- Button
- Image
- Icon
- Center

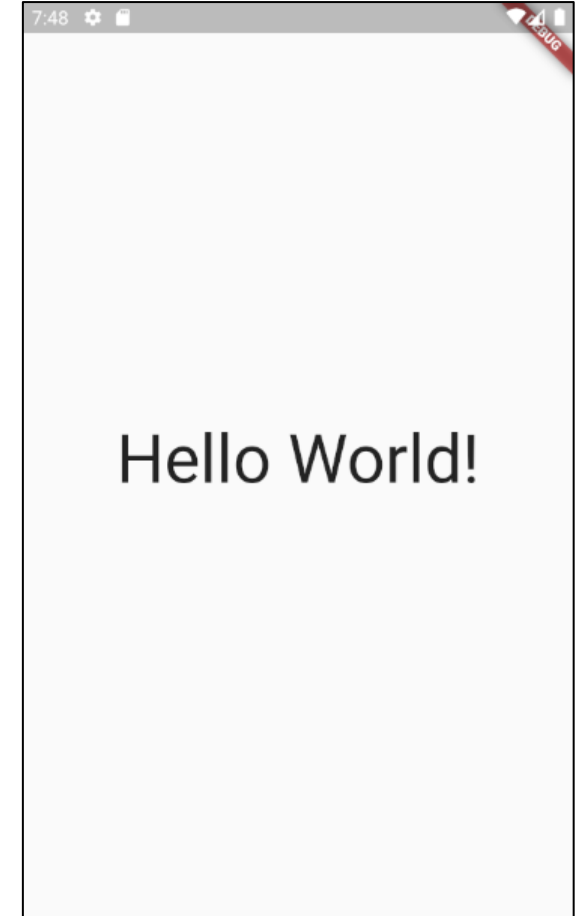


Widgets

– Text und Center

```
@override
Widget build(BuildContext context) {
  return Center(
    child: Text(
      "Hello World!",
      style: TextStyle(fontSize: 50),
    ), // Text
  ); // Center
}
```

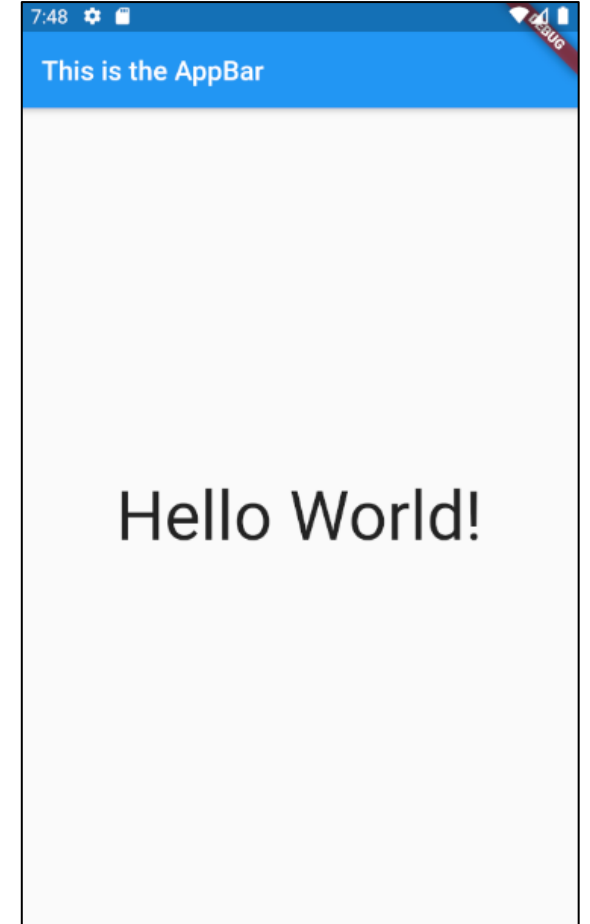
- Text ist ein Widget
 - Zeigt Text an
 - „style“ bietet die Möglichkeit den Text zu formatieren
- Center ist ein Widget
 - Platziert das „child“ Widget in der Mitte von sich selbst



Widgets

– Scaffold und AppBar

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("This is the AppBar"),
    ), // AppBar
    body: Center(
      child: Text("Hello World!", style: TextStyle(fontSize: 50)),
    ), // Center
  ); // Scaffold
}
```



- Scaffold
 - Skelett für gängige App Designs
- AppBar
 - Leiste am oberen Bildschirmrand

Arbeiten mit Android Studio

– Neues Widget erstellen

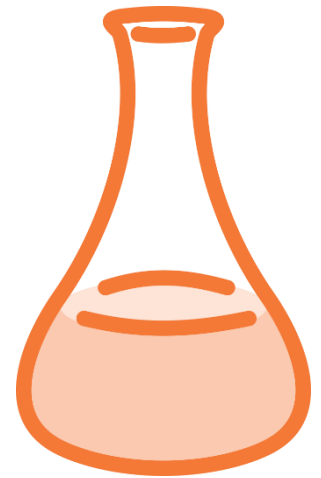
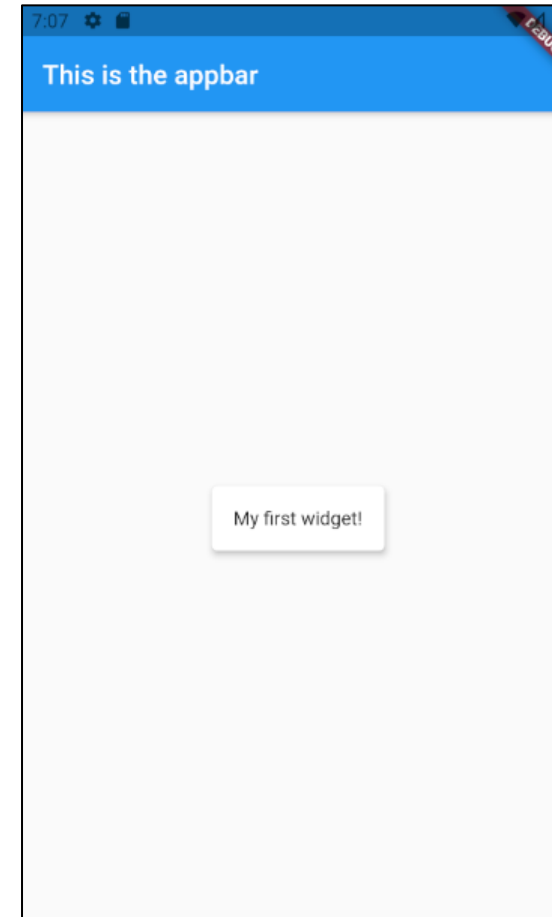
- „stless“ tippen
- Mit Tab/Enter bestätigen
- Name des neuen Widget eingeben
- Mit Tab/Enter bestätigen

```
27     }
28   }
29
30   class ThisIsANewWidget extends StatelessWidget {
31     const ThisIsANewWidget({Key? key}) : super(key: key);
32
33     @override
34     Widget build(BuildContext context) {
35       return Container();
36     }
37   }
38
```

Übung

– Erste Schritte mit Flutter

- Übung:
Baue dein erstes eigenes Widget
- Schritte:
 - Neues Flutter Projekt erstellen
 - MyHomePage und _MyHomePageState löschen
 - Neues Widget erstellen
 - Das Beispiel rechts nachbauen

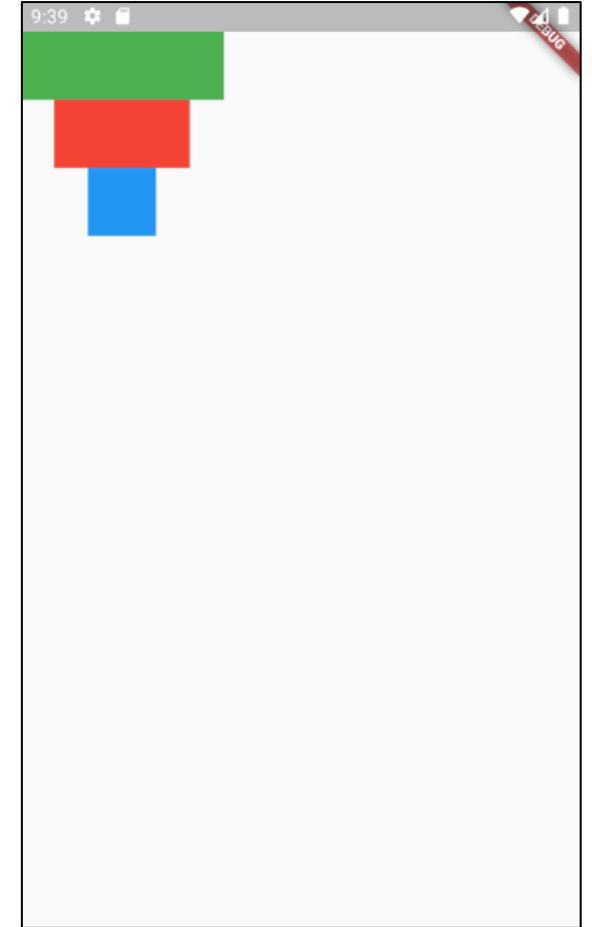


Layout

– Column

- Column = Spalte
- Zeigt Kindelemente in einer Spalte (untereinander) an

```
@override
Widget build(BuildContext context) {
  return Column(
    children: [
      Container(
        color: Colors.green,
        width: 150,
        height: 50,
      ), // Container
      Container(
        color: Colors.red,
        width: 100,
        height: 50,
      ), // Container
      Container(
        color: Colors.blue,
        width: 50,
        height: 50,
      ), // Container
    ],
  ); // Column
}
```

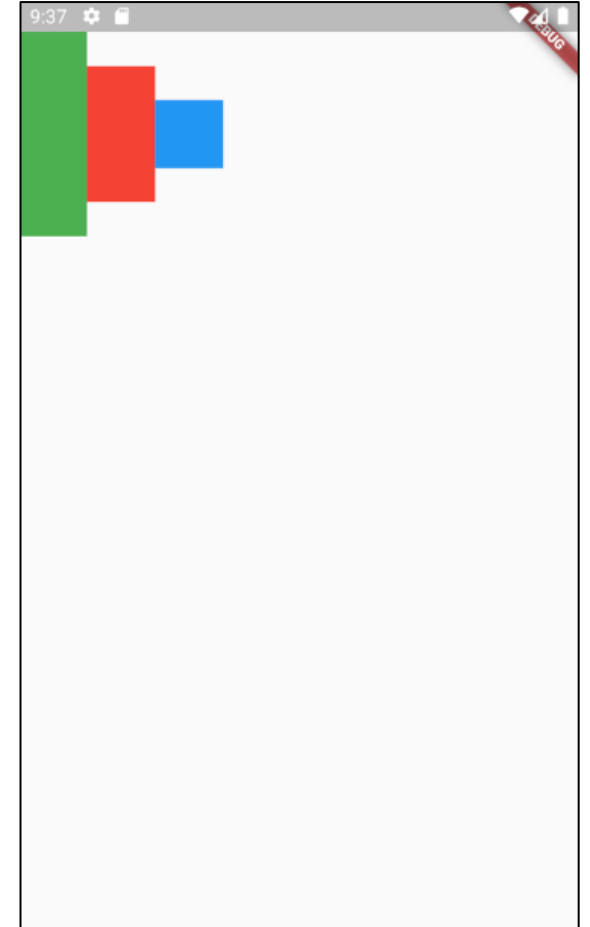


Layout

– Row

- Row = Zeile
- Zeigt Kindelemente in einer Zeile (nebeneinander) an

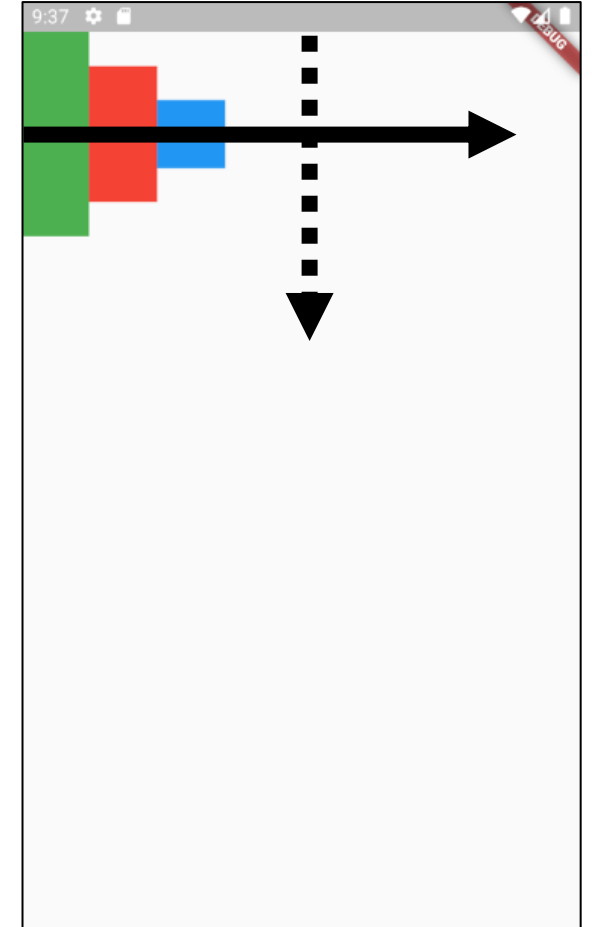
```
@override
Widget build(BuildContext context) {
  return Row(
    children: [
      Container(
        color: Colors.green,
        width: 50,
        height: 150,
      ), // Container
      Container(
        color: Colors.red,
        width: 50,
        height: 100,
      ), // Container
      Container(
        color: Colors.blue,
        width: 50,
        height: 50,
      ), // Container
    ],
  ); // Row
}
```



Layout

– Main Axis vs. Cross Axis

- Main Axis verläuft entlang der Hauptachse
- Cross Axis verläuft rechtwinklig zur Hauptachse
- Beispiel: Row
 - Main Axis: von links nach rechts
 - Cross Axis: von oben nach unten

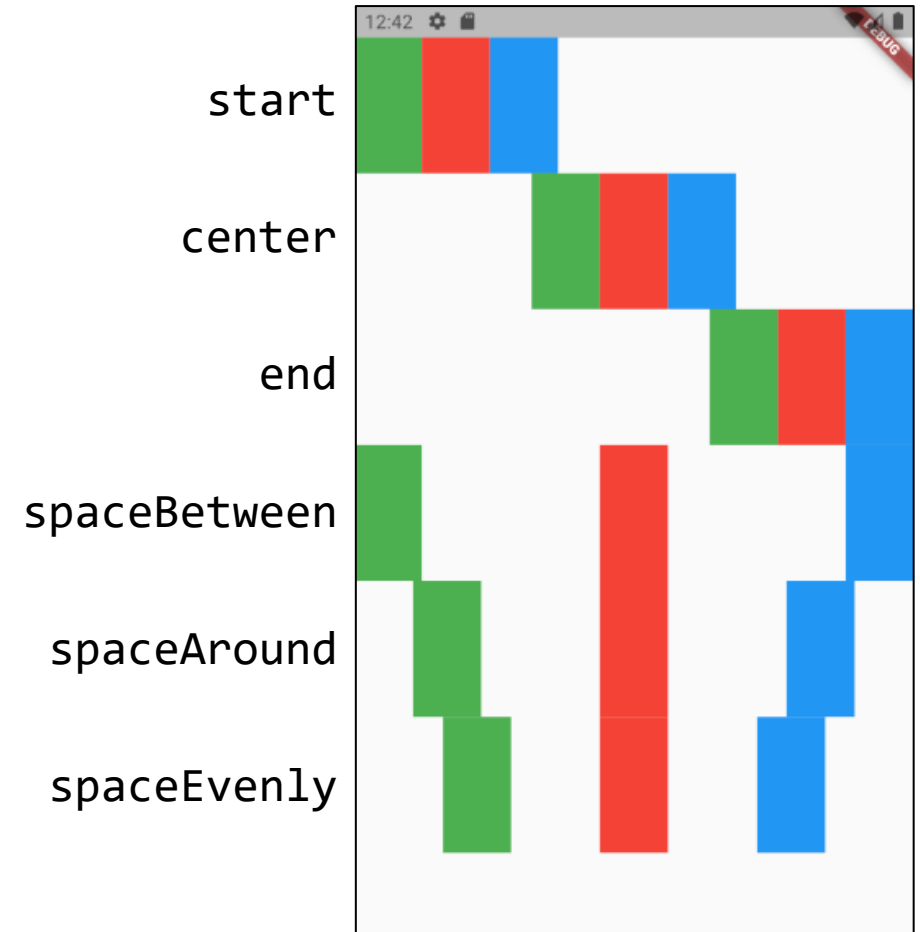


Layout

– Optionen der Main Axis

- Verteilung der Elemente auf der Hauptachse kann eingestellt werden

```
Row(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: [  
    Container(  
      color: Colors.green,  
      width: 50,  
      height: 100,  
    ), // Container  
    Container(  
      color: Colors.red,  
      width: 50,  
      height: 100,  
    ), // Container  
    Container(  
      color: Colors.blue,  
      width: 50,  
      height: 100,  
    ), // Container  
  ],  
)
```

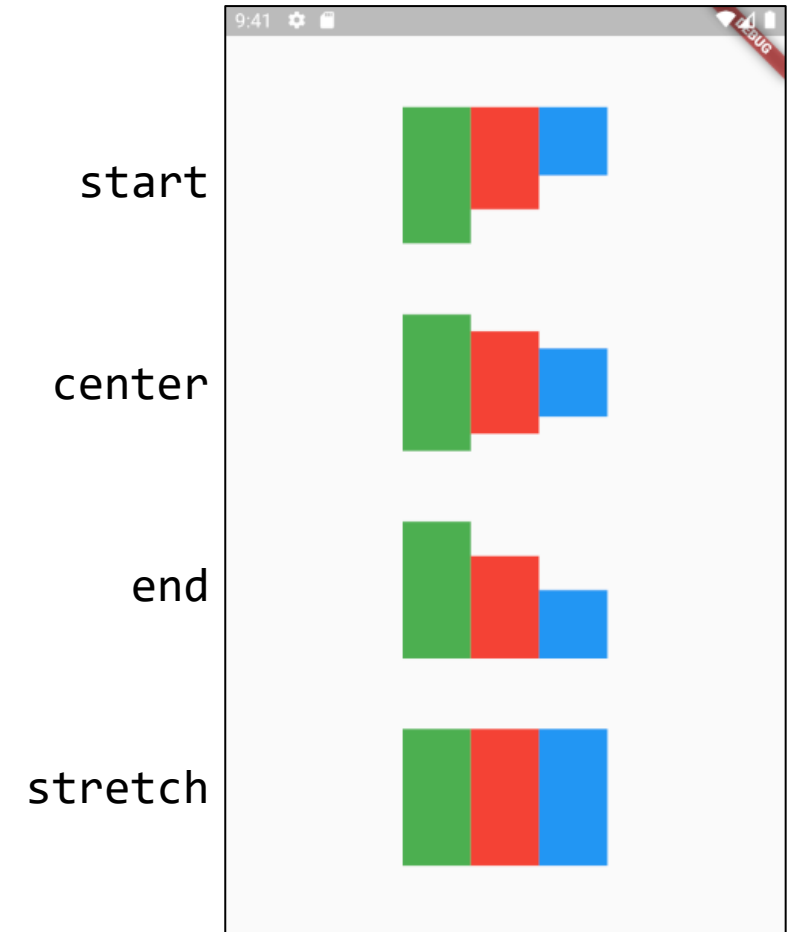


Layout

– Optionen der Cross Axis

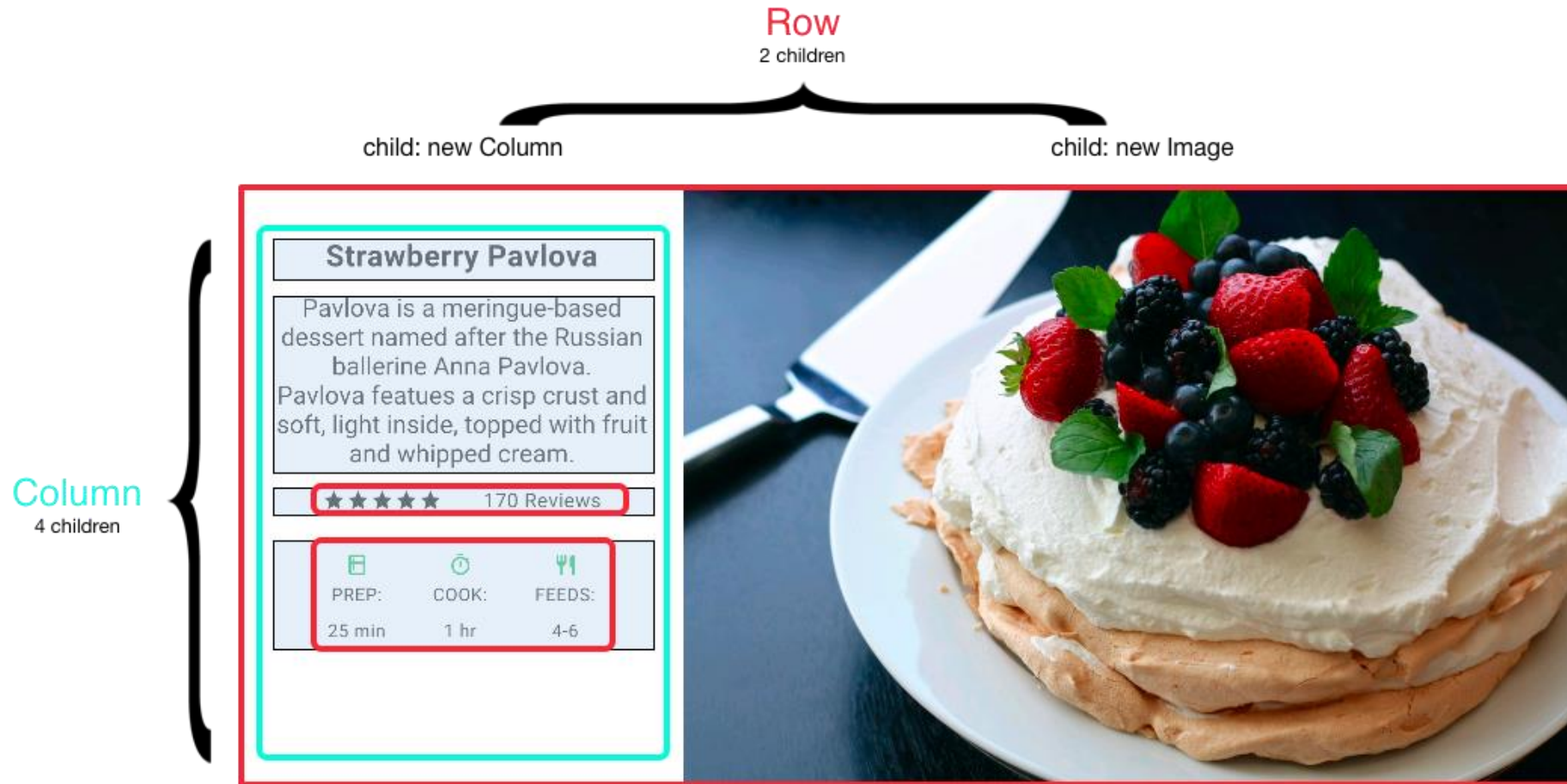
- Verteilung der Elemente auf der Achse vertikal zur Hauptachse kann eingestellt werden

```
Row(  
  mainAxisAlignment: MainAxisAlignment.center,  
  crossAxisAlignment: CrossAxisAlignment.stretch,  
  children: [  
    Container(  
      color: Colors.green,  
      width: 50,  
    ), // Container  
    Container(  
      color: Colors.red,  
      width: 50,  
    ),  
  ],  
)
```



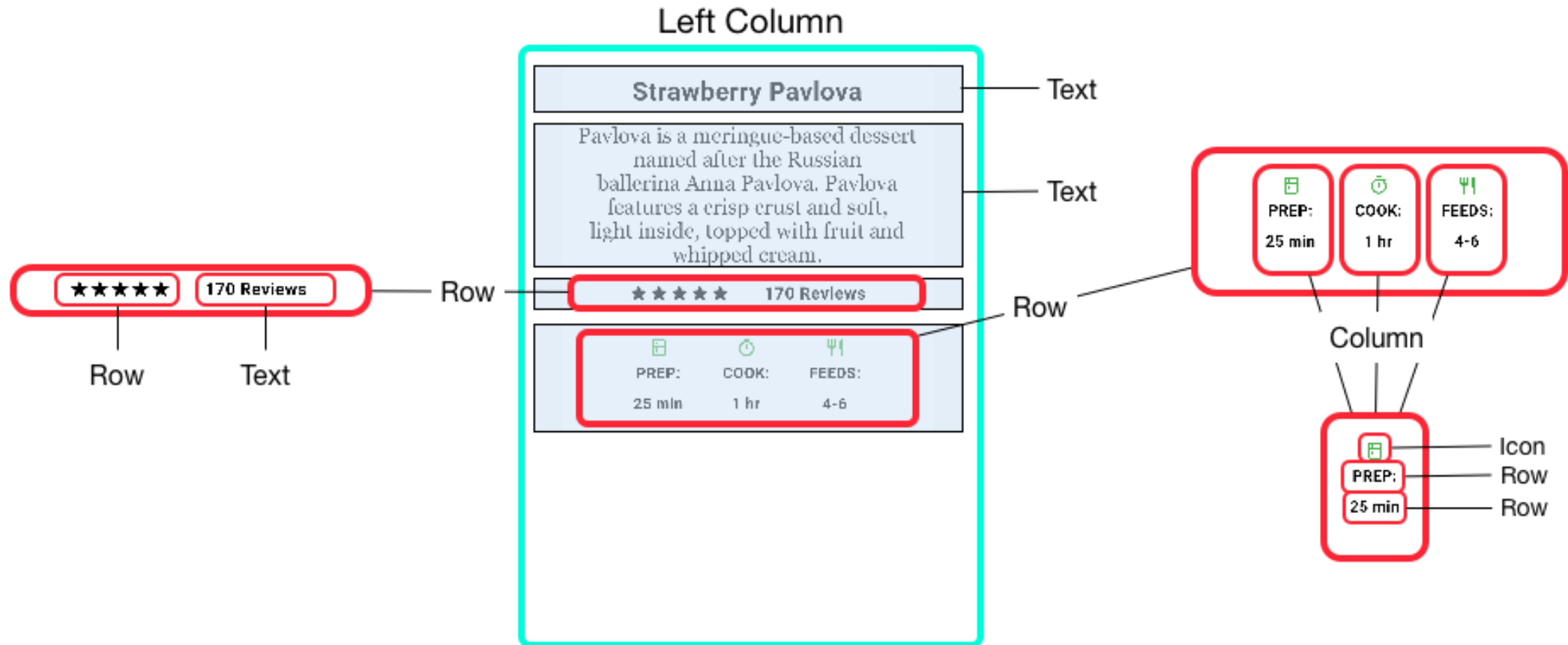
Layout

– Beispiele



Layout

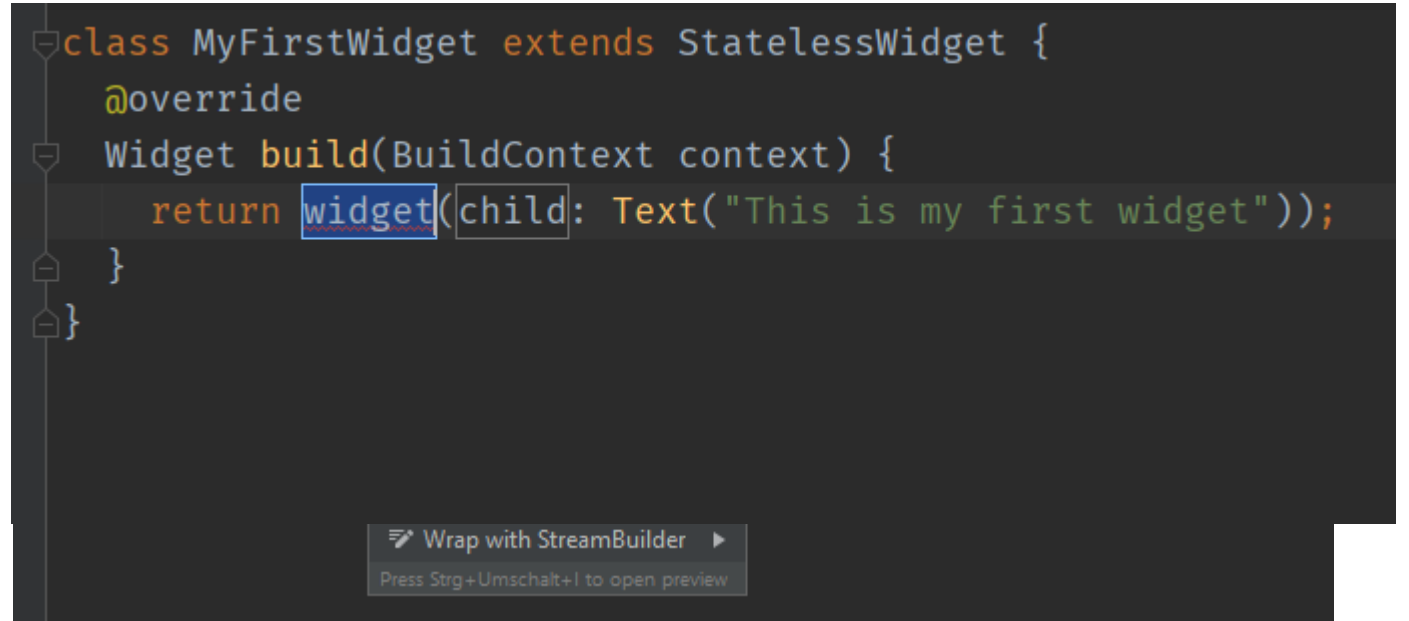
– Beispiele



Arbeiten mit Android Studio

– Widget einfügen

- Cursor in `build()` auf einem UI Element platzieren
- Alt + Enter
- Wrap with ...
- Ggf. Widgetname eingeben
- Mit Tab/Enter bestätigen

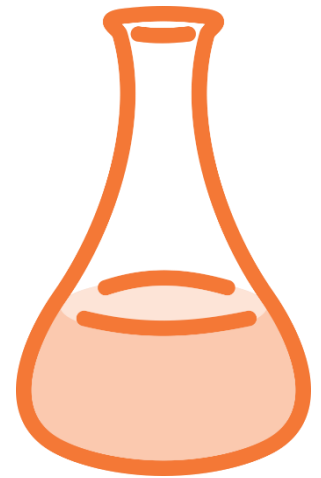
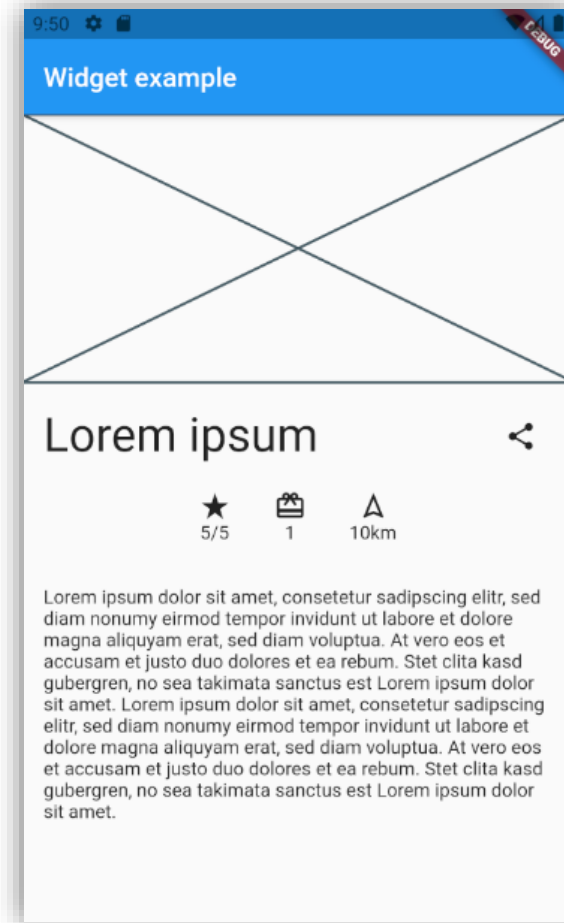


```
class MyFirstWidget extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return widget(child: Text("This is my first widget"));  
  }  
}
```

Wrap with StreamBuilder ▶
Press Strg+Umschalt+I to open preview

Übung - Layout

- Übung:
Baue dieses Layout nach
- Tipps:
 - Verwendete Widgets:
 - Scaffold
 - AppBar
 - Text
 - Placeholder
 - Padding
 - IconButton
 - Icon
 - Row
 - Column
 - Googeln ist erlaubt ;)



Stateless vs. Stateful

- Bisher kann sich die Benutzeroberfläche nicht verändern
- Ziel: Benutzeroberfläche, die sich durch Benutzeraktionen verändert



Besonderheiten eines StatefulWidget

- StatefulWidget führt die „build(...)“ Methode aus, wenn sich die Daten ändern
- Daten können mit „setState((){ ... });“ verändert werden

Aufbau eines StatefulWidget

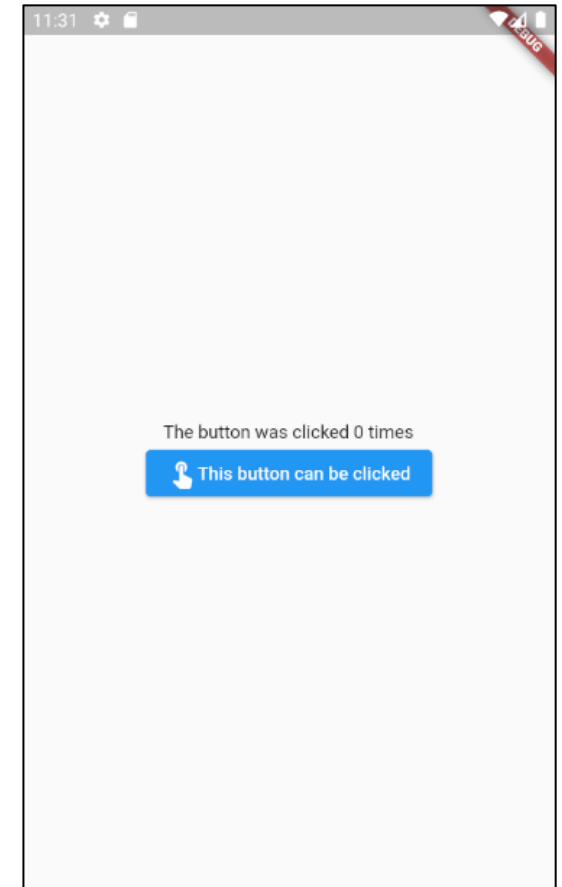
Das eigentliche Widget

Der „State“, der zu dem Widget gehört

Funktion, die beim Knopfdruck aufgerufen wird

```
class ThisIsAWidget extends StatefulWidget {  
  const ThisIsAWidget({Key? key}) : super(key: key);  
  
  @override  
  _ThisIsAWidgetState createState() => _ThisIsAWidgetState();  
}
```

```
class _ThisIsAWidgetState extends State<ThisIsAWidget> {  
  @override  
  Widget build(BuildContext context) {  
    return Center(  
      child: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
          Text("The button was clicked 0 times"),  
          ElevatedButton(  
            onPressed: () {},  
            child: Row(  
              mainAxisAlignment: MainAxisAlignment.min,  
              children: [  
                Icon(Icons.touch_app),  
                Text("This button can be clicked"),  
              ],  
            ), // Row  
          ), // ElevatedButton  
        ],  
      ), // Column  
    ); // Center  
  }  
}
```



Aufbau eines StatefulWidget

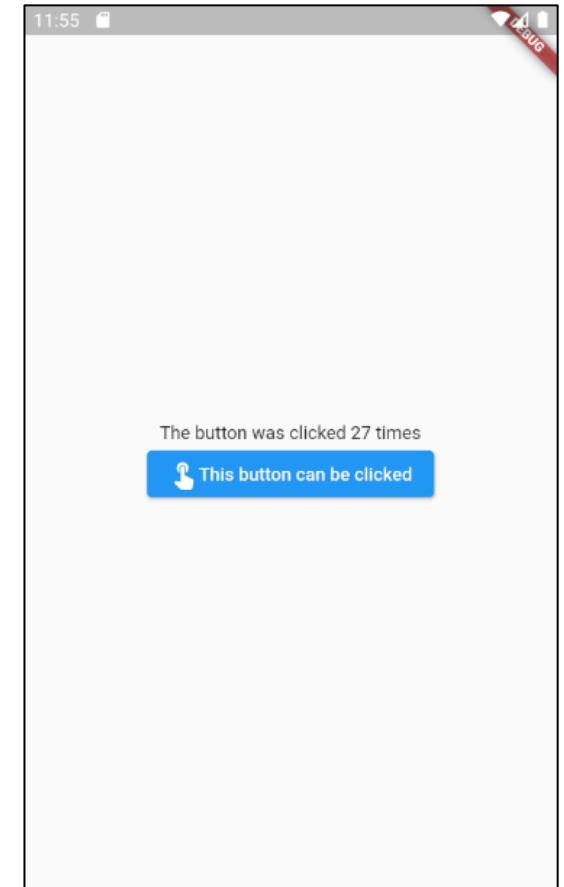
Es ist eine Variable dazugekommen

Der Text wird anhand der Variable erzeugt

„onPressed“ verweist auf die unten definierte Methode

Neue Methode, die „setState({})“ aufruft und den Zähler erhöht

```
class ThisIsAWidgetState extends State<ThisIsAWidget> {  
  int counter = 0;  
  
  @override  
  Widget build(BuildContext context) {  
    return Center(  
      child: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
          Text("The button was clicked $counter times"),  
          ElevatedButton(  
            onPressed: onPressed,  
            child: Row(  
              mainAxisAlignment: MainAxisAlignment.min,  
              children: [  
                Icon(Icons.touch_app),  
                Text("This button can be clicked"),  
              ],  
            ), // Row  
          ), // ElevatedButton  
        ],  
      ), // Column  
    ); // Center  
  }  
  
  void onPressed() {  
    setState(() {  
      counter = counter + 1  
    });  
  }  
}
```



Arbeiten mit Android Studio

– Neues StatefulWidget erstellen

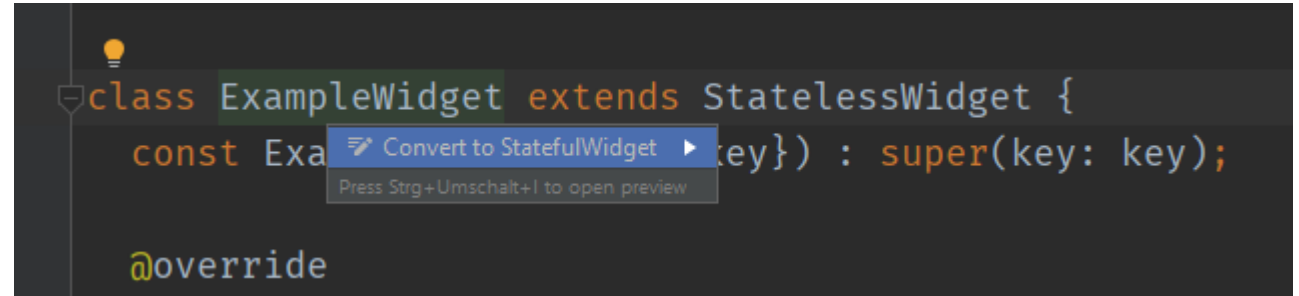
- „stful“ tippen
- Mit Tab/Enter bestätigen
- Name des neuen Widget eingeben
- Mit Tab/Enter bestätigen

```
88
89 class ThisIsAStatefulWidget extends StatefulWidget {
90     const ThisIsAStatefulWidget({Key? key}) : super(key: key);
91
92     @override
93     _ThisIsAStatefulWidgetState createState() => _ThisIsAStatefulWidgetState();
94 }
95
96 class _ThisIsAStatefulWidgetState extends State<ThisIsAStatefulWidget> {
97     @override
98     Widget build(BuildContext context) {
99         return Container();
100     }
101 }
```

Arbeiten mit Android Studio

– Stateless zu Stateful Widget ändern

- Cursor auf Widgetklasse platzieren
- Alt + Enter
- „Convert to StatefulWidget“ auswählen

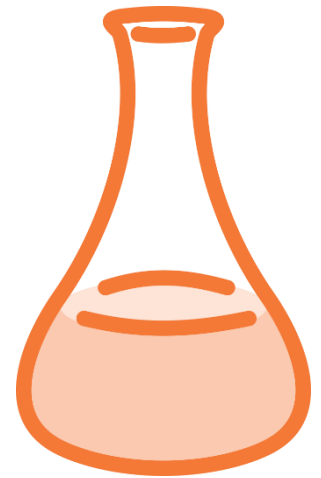
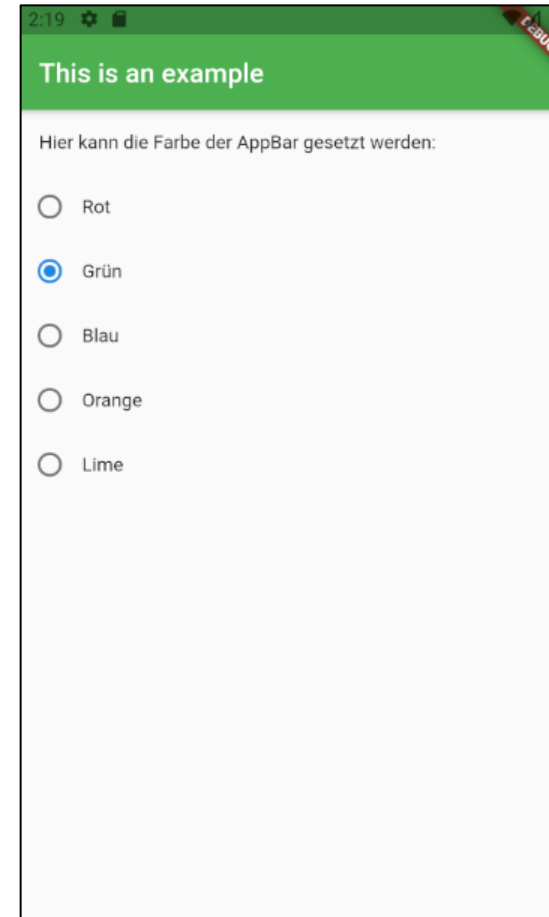


```
class ExampleWidget extends StatelessWidget {  
  const ExampleWidget({Key? key}) : super(key: key);  
  
  @override  
  Widget build(BuildContext context) {  
    // ...  
  }  
}
```

The screenshot shows the Android Studio IDE with a code editor. A class named `ExampleWidget` is shown, which extends `StatelessWidget`. A context menu is open over the class name, with the option `Convert to StatefulWidget` highlighted. Below this option, a smaller tooltip says `Press Strg+Umschalt+I to open preview`. The code editor has a dark theme.

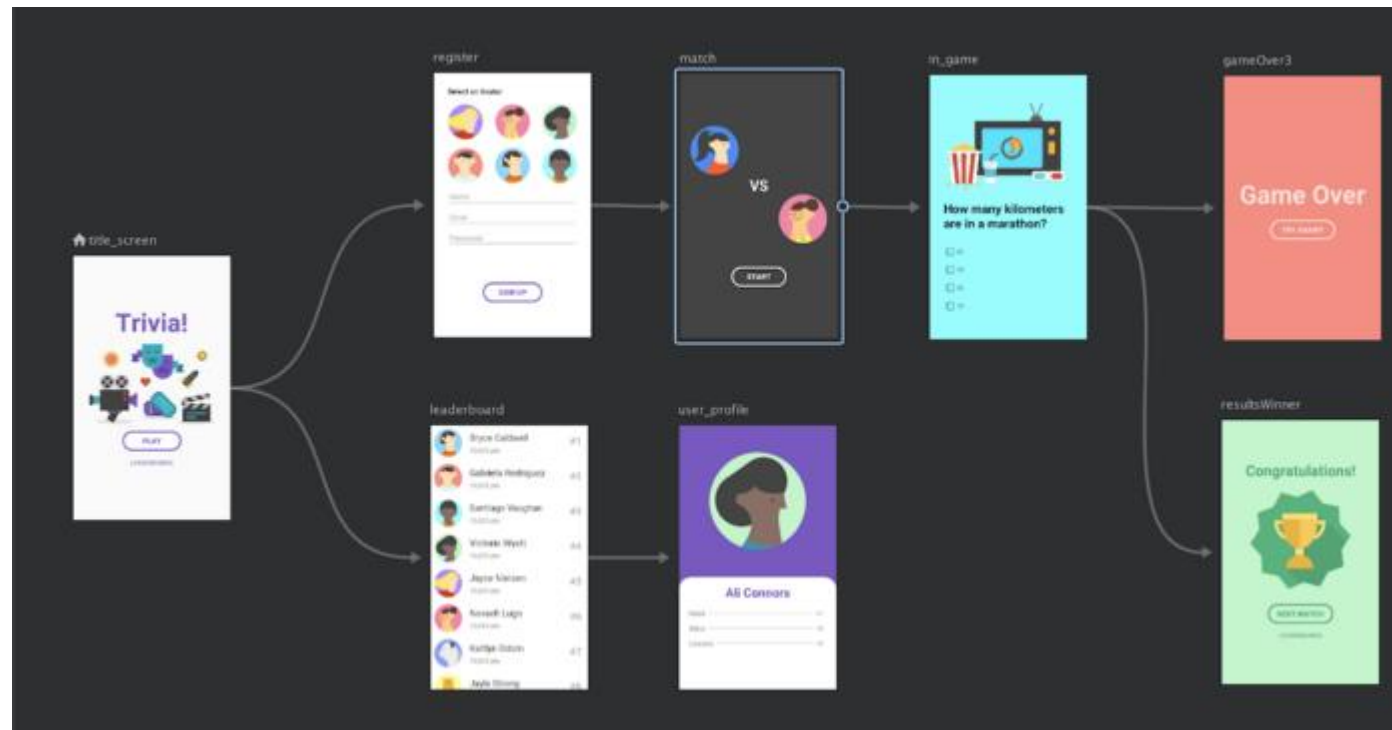
Übung – Status

- Übung:
 - Baue dieses Layout nach
 - Die AppBar sollte die Hintergrundfarbe je nach Auswahl verändern
- Tipps:
 - Verwendete Widgets:
 - Scaffold
 - AppBar
 - Text
 - Padding
 - Radio
 - Row
 - Column
 - Zuerst das Layout, dann die Interaktion
 - Code formatieren mit STRG + ALT + L



Navigation

- Eine App benötigt meist mehr als nur eine Seite → Navigation



Navigation

- Zu einer anderen Seite navigieren:

```
Navigator.of(context).push(  
  MaterialPageRoute(builder: (context) => StartExecutingChecklistPage()));
```

Navigationsziel

→ StartExecutingChecklistPage ist ein „normales“ Widget

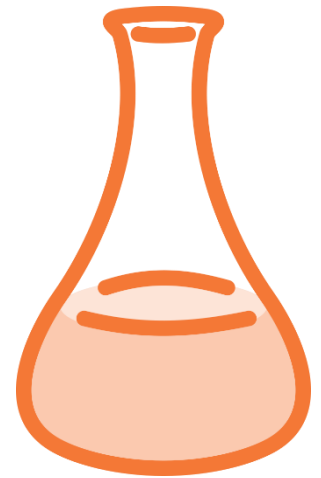
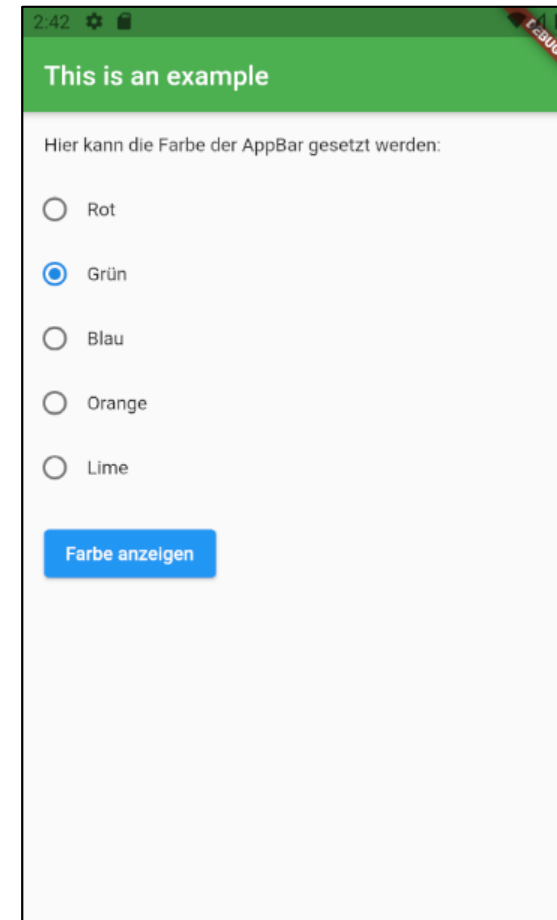
Navigation

- Zurück zur vorherigen Seite:

```
Navigator.of(context).pop();
```

Übung – Navigation

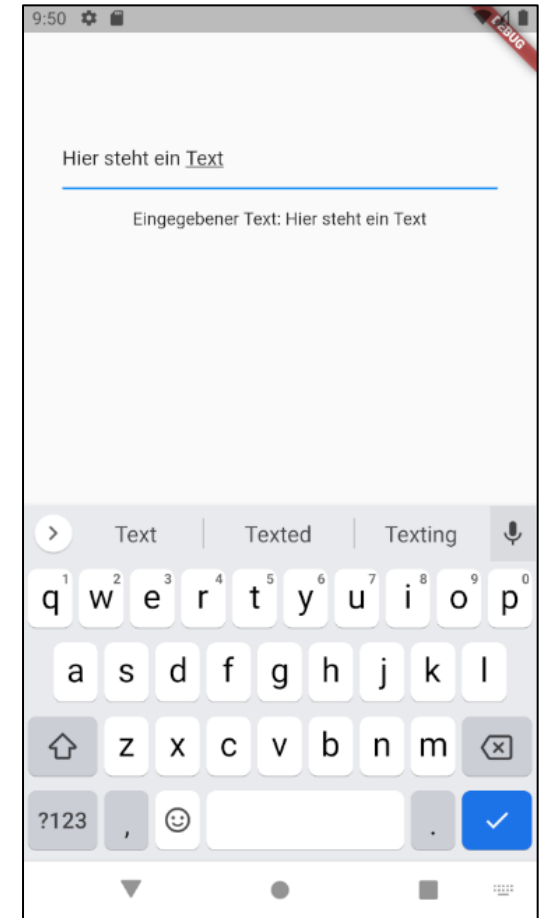
- Nun wird die vorherige Übung erweitert
- Übung:
 - Füge einen Knopf hinzu, der zu einer neuen Seite navigiert.
 - Die neue Seite zeigt ein Quadrat mit der ausgewählten Farbe.



Widget

– TextField

- Ermöglicht eine Texteingabe
- Benötigt ein TextEditingController



Widget

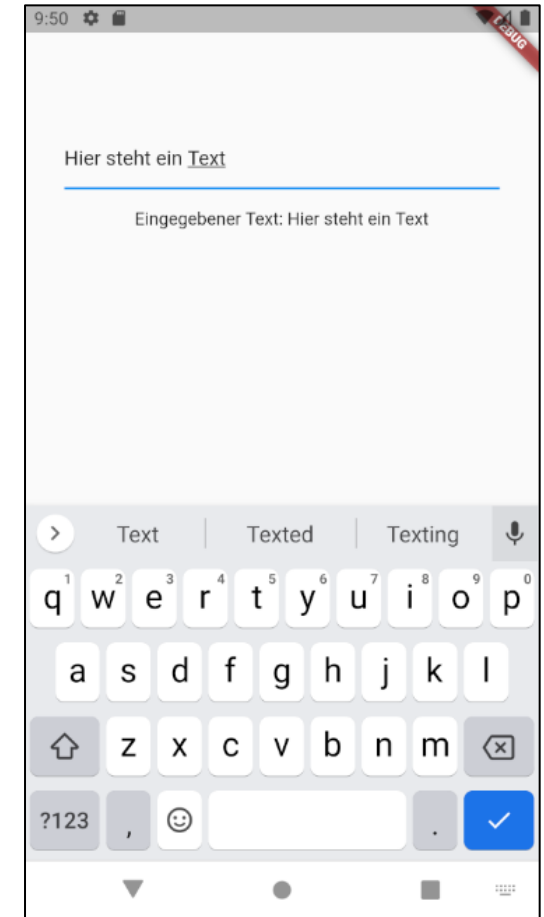
– TextField

TextEditingController und eine Variable, die den Text beinhaltet

Hier wird der TextEditingController und die Variable verbunden

Das TextField benötigt den TextEditingController

```
class TextFieldExampleState extends State<TextFieldExample> {  
  final TextEditingController _controller = TextEditingController();  
  String _text = "Das ist ein Text";  
  
  @override  
  void initState() {  
    super.initState();  
  
    _controller.text = _text;  
    _controller.addListener(() {  
      setState(() {  
        _text = _controller.text;  
      });  
    });  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      body: Column(  
        children: [  
          Padding(  
            padding: const EdgeInsets.fromLTRB(32, 100, 32, 16),  
            child: TextField(  
              controller: _controller,  
            ), // TextField  
          ), // Padding  
          Text("Eingegebener Text: $_text")  
        ],  
      ), // Column  
    ); // Scaffold  
  }  
}
```

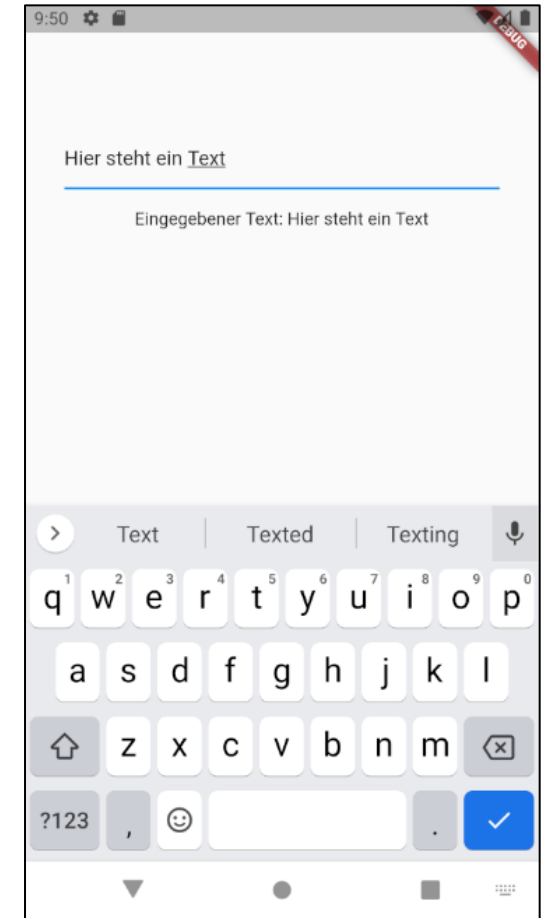


Widget

– TextField

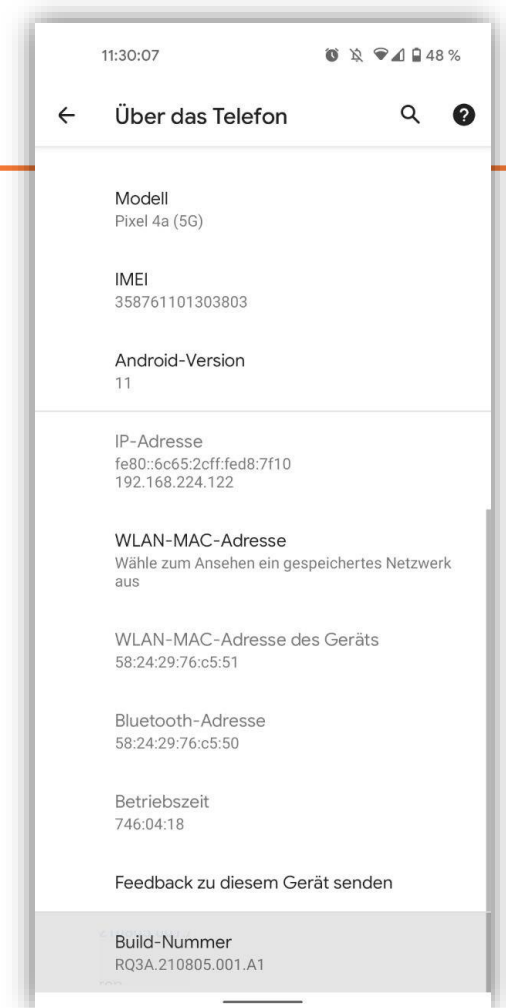
Ein TextEditingController
sollte disposed werden

```
void initState() {  
  super.initState();  
  
  _controller.text = _text;  
  _controller.addListener(() {  
    setState(() {  
      _text = _controller.text;  
    });  
  });  
}  
  
@override  
void dispose() {  
  super.dispose();  
  _controller.dispose();  
}  
  
@override  
Widget build(BuildContext context) {  
  return Scaffold(  
    body: Column(  
      children: [  
        Padding(  
          padding: const EdgeInsets.fromLTRB(32, 16, 32, 16),  
          child: TextField(  
            controller: _controller,  
          ), // TextField  
        ), // Padding  
        Text("Eingegebener Text: $text")  
      ],  
    ),  
  );  
}
```



Android – Entwickleroptionen aktivieren

- Einstellungen öffnen
 - „Über das Telefon“
 - Ein paar mal hintereinander auf „Build-Nummer“ klicken
- Es werden die Entwickleroptionen freigeschaltet
 - Falls noch nicht aktiviert:
 - Entwickleroptionen aktivieren
 - „USB-Debugging“ aktivieren



Fragen

