

# Softwarepraktikum

Prof. Dr. Bernhard Standl

SoSe 23

BAS-Inf-6A; ErMAS-Inf-7A

# Kommunikation

- E-Mail
- Sprechstunde
- Seminar
- Mittwoch Vormittag



# Vorbesprechung

- 1) Rahmen
- 2) Thema
- 3) Ablauf
- 4) Hintergrund
- 5) Grundlagen
- 6) Erste Schritte

# Zeitplan

16:30 – 17:30 Grundlagen

17:30 – 17:45 Pause

18:00 – 19:00 Einrichtung Github, Flutter, VS Code

# Kompetenzziele

Nach Abschluss des Moduls können die Studierenden...

- verschiedene Strategien zur gemeinsamen Entwicklung von Programmierprojekten verwenden.
- Tests zur Qualitätssicherung formulieren und anwenden.
- Werkzeuge zur Entwicklung, zur Analyse, zum Test und zur Dokumentation von SoftwareProjekten anwenden.
- Konzepte der objektorientierten Programmierung in ihren Programmen anwenden.
- Probleme mit Hilfe selbst geschriebener Programme analysieren.

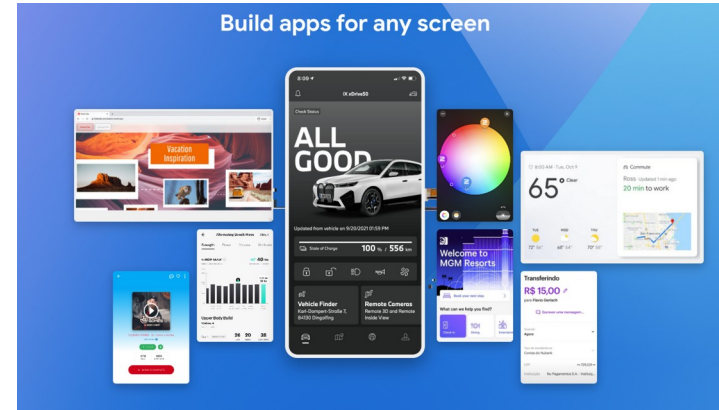
# Inhalte

- Vorgehensmodelle für den Entwurf großer Softwaresysteme
- Software-Testmethoden
- Methoden und Sprachen für den objektorientierten Entwurf
- Programmierumgebungen und -werkzeuge
- Projektplanung und -durchführung
- Qualitätssicherung bei Programmierprojekten
- Arbeiten im Team bei Programmierprojekten

# Arbeitsleistung

- 4 SWS
  - 180 Minuten / Woche Kontaktzeit / 13 Wochen = 39 Stunden
- 8 CP
  - Ein CP entspricht einer Gesamtarbeitsleistung der Studierenden im Präsenz- und Selbststudium von 30 Zeitstunden.
  - $8 * 30 = 240$  Stunden
  - $\Rightarrow 240 - 39 = 201$  Stunden Selbststudium
- Leistungsfeststellung: Studienleistung

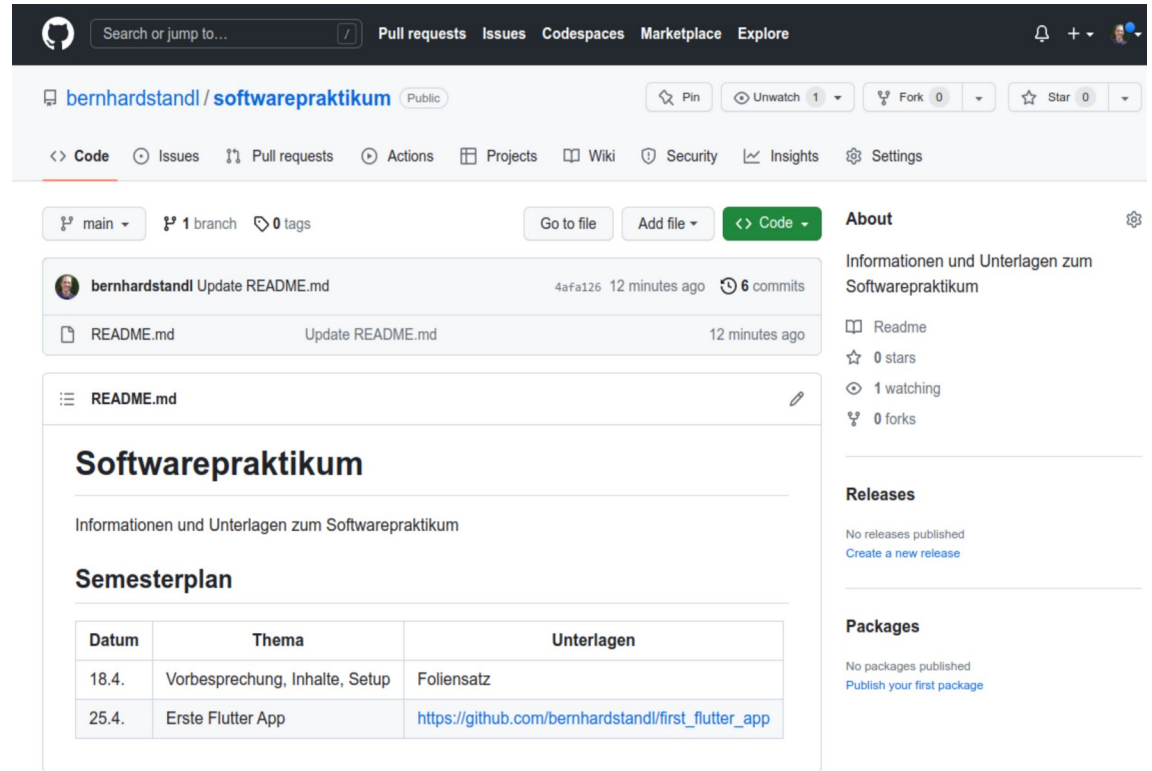
# Ziel der Lehrveranstaltung



- **Planung und Entwicklung einer nativen mobilen Cross-Plattform App**



# Zeitplan

A screenshot of a GitHub repository page for 'bernhardstandl / softwarepraktikum'. The page shows the repository name, a search bar, and navigation tabs like 'Code', 'Issues', 'Pull requests', etc. The main content area displays the 'README.md' file, which includes a title 'Softwarepraktikum', a subtitle 'Semesterplan', and a table with dates, topics, and documents. The right sidebar contains 'About', 'Releases', and 'Packages' sections.

Search or jump to... Pull requests Issues Codespaces Marketplace Explore

bernhardstandl / softwarepraktikum Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file <> Code

bernhardstandl Update README.md 4afa126 12 minutes ago 6 commits

README.md Update README.md 12 minutes ago

README.md

## Softwarepraktikum

Informationen und Unterlagen zum Softwarepraktikum

### Semesterplan

Datum	Thema	Unterlagen
18.4.	Vorbesprechung, Inhalte, Setup	Foliensatz
25.4.	Erste Flutter App	<a href="https://github.com/bernhardstandl/first_flutter_app">https://github.com/bernhardstandl/first_flutter_app</a>

#### About

Informationen und Unterlagen zum Softwarepraktikum

Readme

0 stars

1 watching

0 forks

#### Releases

No releases published

[Create a new release](#)

#### Packages

No packages published

[Publish your first package](#)

<https://github.com/bernhardstandl/softwarepraktikum>

# It's all widgets!



**IT'S ALL Widgets!**

Apps Developers Resources Streams Podcast Twitter

MADE WITH ♥ BY THE FLUTTER COMMUNITY

**An open list of apps built with Flutter**

Feel free to add an app in progress and update it when it goes live

SUBMIT APP SUBMIT EVENT

Search 3,208 apps... OPEN SOURCE TEMPLATE ZOOM SORT FEATURED

Mobile Desktop Web

**Instaboard**

Free Instant Collaborative Whiteboard

GOOGLE PLAY APP STORE

Collaborate with others in real time.

**BlueBubbles**

BlueBubbles is a cross-platform app ecosystem, bringing iMessage to...

GOOGLE PLAY APP STORE

iMessage on Android

**Loop Messenger**

The ultimate solution for group messaging that's less noisy and...

GOOGLE PLAY APP STORE

More organized, less noise

**Haiku Lens**

Turn your photos into haikus: Take a photo and get a video with a haik...

GOOGLE PLAY APP STORE

Take a Photo Get a Poem

# Ablauf

Phase	Grundlagen	Projektideen	Entwicklung	Test & Share
Seminar	Setup (Github, Flutter, VS Code) Agile Methoden Debugging, SW Dev Teamorganization Tutorials	Brainstorming GitHub Support Flutter Tutorials  Identification of existing Solutions	Ausarbeiten des Projektes Knowledge Base Wiki Support	Test Roll out
Team	Einarbeiten  Abstimmen	Spezifizieren Projektideen  Agiles Entwickeln Eines Projektes / User Stories	Agiles Entwickeln Eines Projektes  Coding Dev	Agiles Entwickeln Eines Projektes  Code Dev



# Didaktischer Ansatz

## Student-centered Approach

Motschnig-Pitrik, R., & Standl, B. (2013). **Person-centered technology enhanced learning: Dimensions of added value.** Computers in human behavior, 29(2), 401-409.

## Agile Approach

Romeike, R., & Göttel, T. (2012). **Agile projects in high school computing education: emphasizing a learners' perspective.** In Proceedings of the 7th Workshop in Primary and Secondary Computing Education (pp. 48-57).



## Constructionist Approach

Monga, M., Lodi, M., Malchiodi, D., Morpurgo, A., & Spieler, B. (2018). **Learning to program in a constructionist way.** In Proceedings of Constructionism 2018.

## Problem-based Learning

Wijnia, L., Loyens, S. M., & Rikers, R. M. (2019). **The problem-based learning process: An overview of different models.** The Wiley handbook of problem-based learning, 273-295.

# Didaktischer Ansatz

Student-centered Approach

Constructionist Approach

Agile Approach

Problem-based Learning

- Learning on three levels: Attitudes, Skills, Knowledge (Standl 2016)
- Positive Climate and interpersonal attitudes: Authenticity, Unconditional Positive Regards, Empathy (Motschnig & Standl 2013, Rogers 1994)
- Freedom to Learn innerhalb gemeinsam definierter Grenzen (Rogers 1994)



# Didaktischer Ansatz

Student-centered Approach

Constructionist Approach

Agile Approach

Problem-based Learning

- Das Konzept der Konstruktion von Wissen durch die Herstellung von konkreten und öffentlichen Artefakten.
- If art interprets our dreams, the computer executes them in the guise of programs! (Abelson et al. 1996)
- Programs are a join point between our mind and the computer. (Monga et al. 2018)
- Introduce coding in this order: Motivation → Logical Aspects → Syntax → Personal Projects
- 4P: Projects, Peers, Passion, Play (Resinick 2014)



# Didaktischer Ansatz

Student-centered Approach

Constructionist Approach

Agile Approach

Problem-based Learning

- Wichtiger als Vorträge und Bücher ist die Zusammenarbeit und Kommunikation unter Studierenden
- Wichtiger als viel Zeit in Dokumentation zu stecken ist funktionierende Software zu entwickeln
- Wichtiger als alle Vorgaben des (curricularen) Rahmens zu erfüllen ist es, die gemeinsamen Ziele zu erreichen
- Wichtiger als dem ursprünglichen Plan hart zu folgen ist es, bereit für Anpassungen zu sein



# Didaktischer Ansatz

Student-centered Approach

Constructionist Approach

Agile Approach

Problem-based Learning

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan





# Didaktischer Ansatz

Student-centered Approach

Constructionist Approach

Agile Approach

Problem-based Learning

- 1) Klären unklarer Begriffe
- 2) Definition und Eingrenzung des Problems
- 3) Wissenssammlung im Team und Brainstorming
- 4) Konkretisierung des geplanten Vorhabens
- 5) Formulierung von (Teil)Zielen
- 6) Bearbeitung der Aufgaben / Ziele
- 7) Zusammenfassung der Aufgabenergebnisse



# Didaktischer Ansatz

## Student-centered Approach

Grundlage unserer Arbeitsatmosphäre mit authentischem, wertschätzendem und empathischen Verhalten und dem Bewusstsein des Lernens auf drei Ebenen

## Agile Approach

Grundlage unserer Arbeitsatmosphäre mit authentischem, wertschätzendem und empathischen Verhalten und dem Bewusstsein des Lernens auf drei Ebenen



## Constructionist Approach

Die Lernenden arbeiten mit konkreten Gegenständen an Aufgaben mit persönlichen Zielen und mit Bedeutung und das in einem Rahmen von „Projects, Passion, Peers, Play“

## Problem-based Learning

Systematische gemeinsame Bearbeitung eines Problems / Vorhabens im Team geteilt und die Ergebnisse zu einem Gesamtergebnis zusammengeführt

# Ablauf

Phase	Grundlagen	Projektideen	Entwicklung	Test & Share
Seminar	Setup (Github, Flutter, VS Code) Agile Methoden Debugging, SW Dev Teamorganization Tutorials	Brainstorming GitHub Support Flutter Tutorials  Identification of existing Solutions	Ausarbeiten des Projektes Knowledge Base Wiki Support	Test Roll out
Team	Einarbeiten  Abstimmen	Spezifizieren Projektideen  Agiles Entwickeln Eines Projektes / User Stories	Agiles Entwickeln Eines Projektes  Coding Dev	Agiles Entwickeln Eines Projektes  Code Dev

# Didaktischer Ansatz

Construct-  
ionist  
Approach

Student-  
centered  
Approach

Leitprinzipien



Agile  
Approach

Problem-  
based  
Learning

Durchführungsprinzipien

# Was ist Flutter?

## FLUTTER Features



Flutter is based on Dart



Quicker & Simple  
Code Writing



Less Testing



Smooth Performance



Open Source

## Top Mobile Apps Developed Using Flutter

Google

Square

PHILIPS



The New York Times

ny bank

ebay

SONOS

reflectly

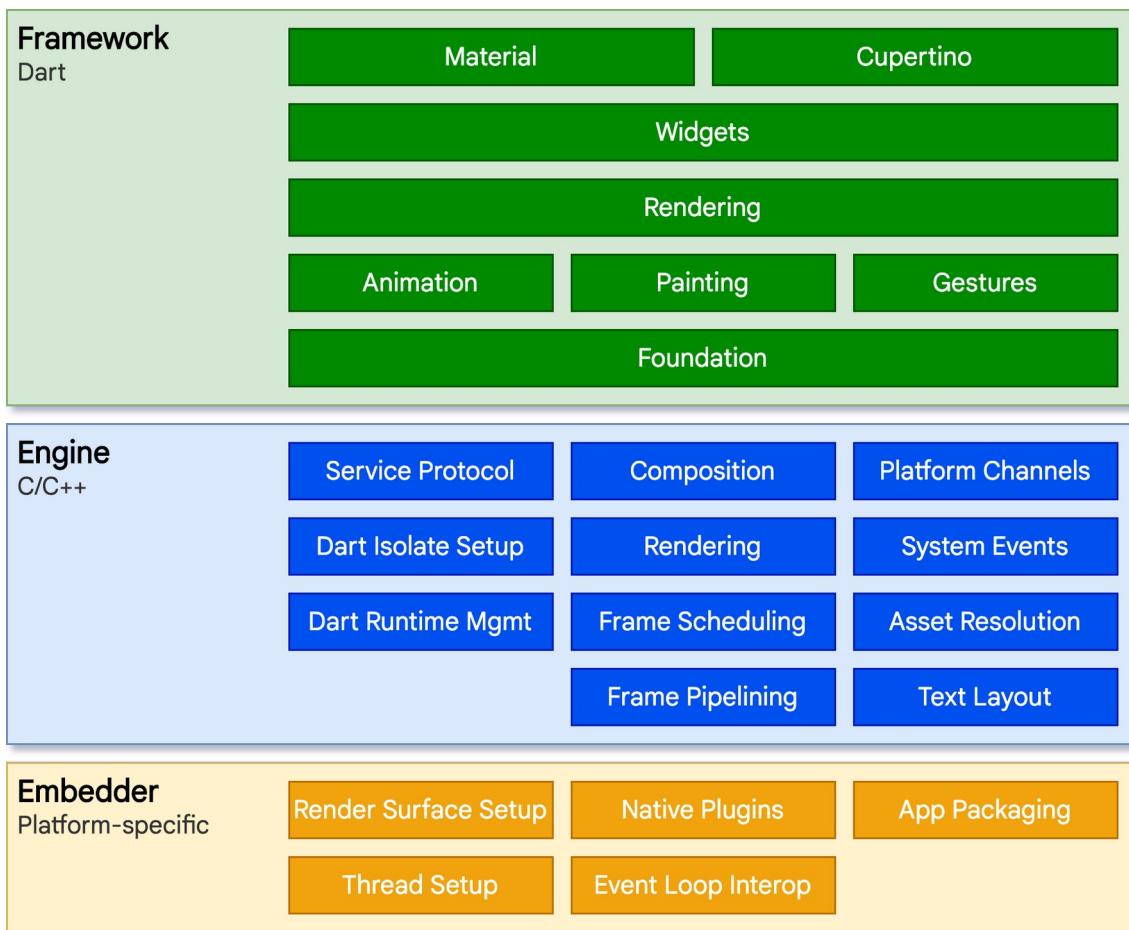
Xianyu  
by Alibaba

Google Pay

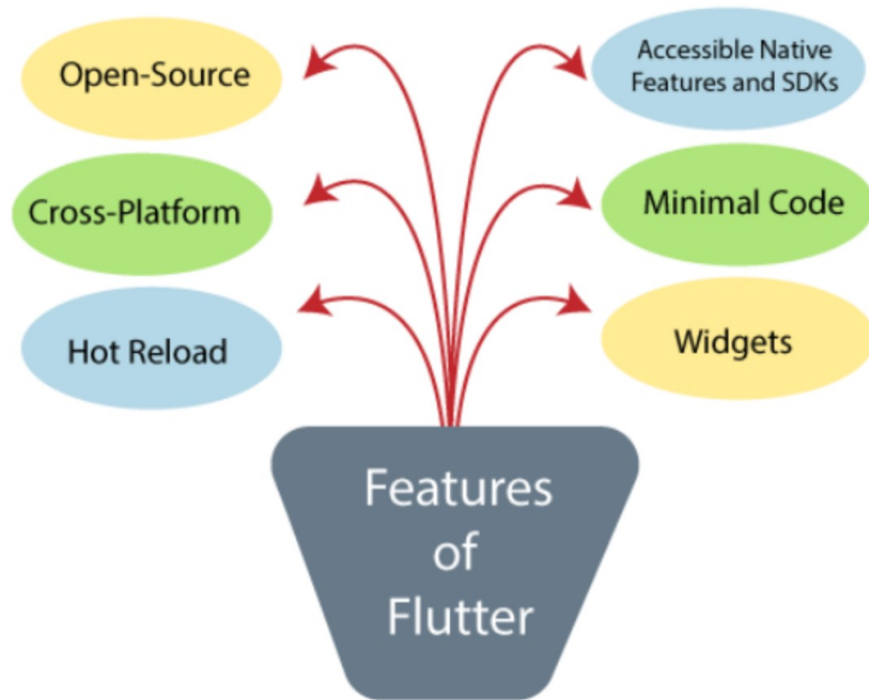
# Was ist Flutter?

- Flutter ist ein von Google entwickeltes Open-Source-Framework für die Erstellung von plattformübergreifenden mobilen Anwendungen (iOS und Android) sowie Webanwendungen und Desktop-Anwendungen.
- Das Framework ermöglicht die schnelle Erstellung von nativen und performanten Anwendungen durch die Verwendung der Programmiersprache Dart und einer umfangreichen Bibliothek an Widgets und Tools.
- Flutter verwendet eine sogenannte "Widget-basierte" Architektur, bei der alle visuellen Elemente als Widgets dargestellt werden, um eine einfache Erstellung und Wiederverwendbarkeit von UI-Elementen zu ermöglichen.
- Flutter wird von einer wachsenden Community von Entwicklern unterstützt und eignet sich sowohl für die Entwicklung von kleinen als auch großen Anwendungen, einschließlich von anspruchsvollen Anwendungen wie E-Commerce-Plattformen, sozialen Netzwerken und Spielen.

# Flutter Framework



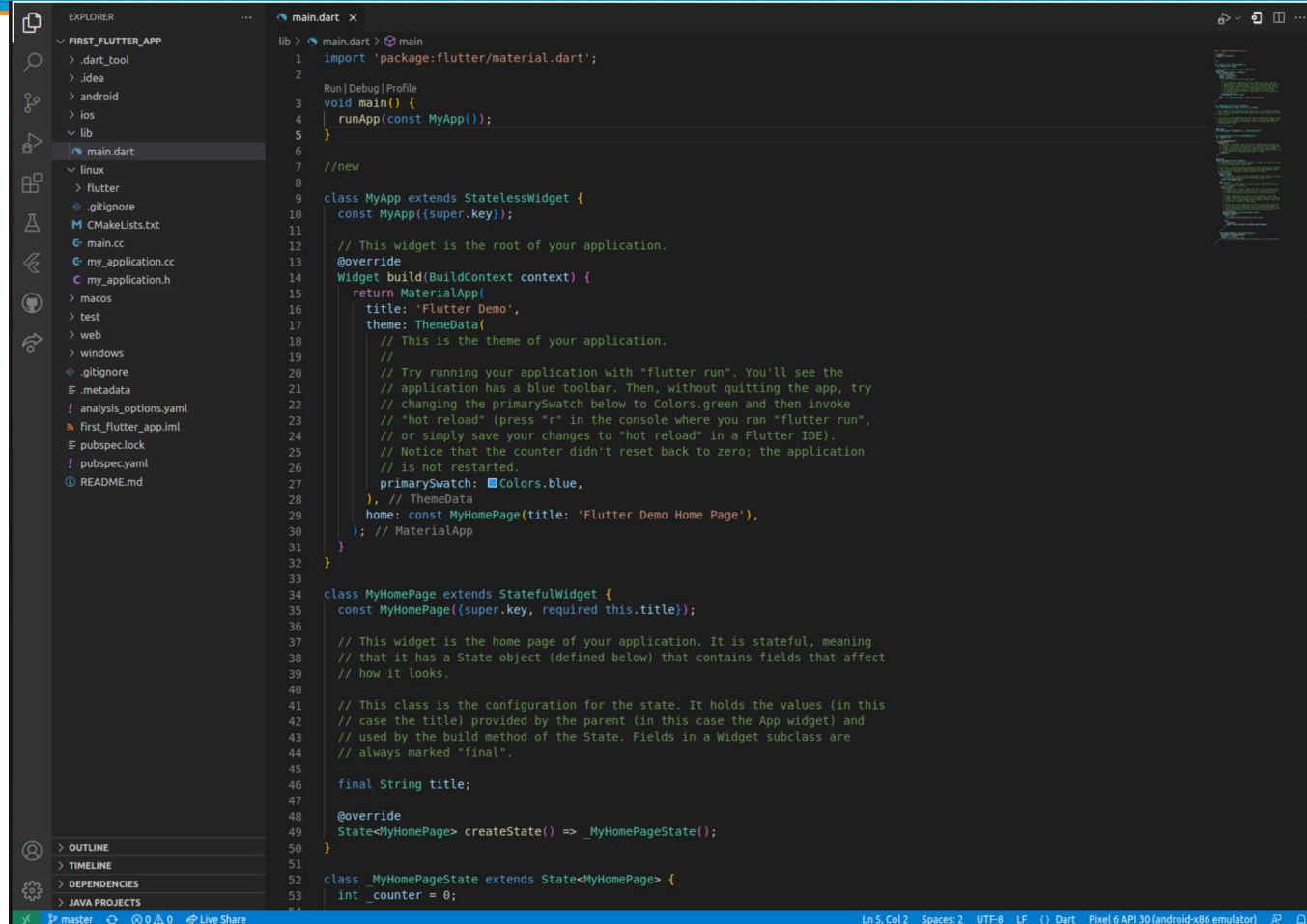
# Programmieren in Flutter





# Programmieren in Flutter

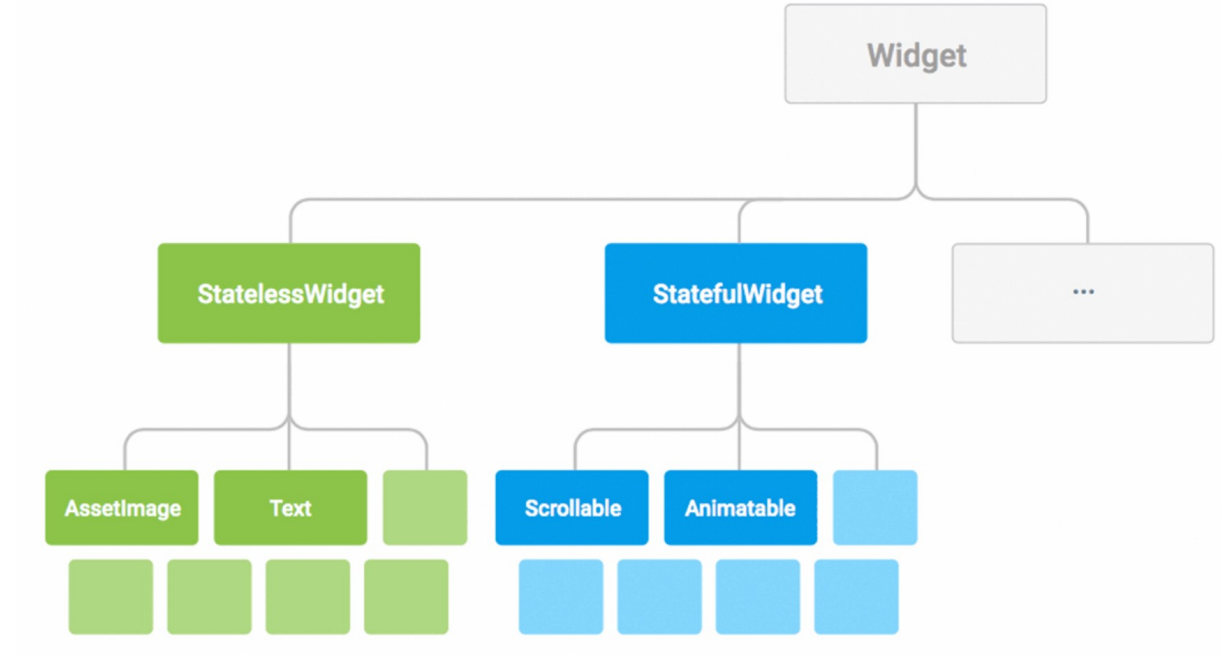
- Visual Studio Code
- main.dart



```
lib > main.dart > main
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   runApp(const MyApp());
6 }
7
8 //new
9 class MyApp extends StatelessWidget {
10   const MyApp({super.key});
11
12   // This widget is the root of your application.
13   @override
14   Widget build(BuildContext context) {
15     return MaterialApp(
16       title: 'Flutter Demo',
17       theme: ThemeData(
18         // This is the theme of your application.
19         //
20         // Try running your application with "flutter run". You'll see the
21         // application has a blue toolbar. Then, without quitting the app, try
22         // changing the primarySwatch below to Colors.green and then invoke
23         // "hot reload" (press "r" in the console where you ran "flutter run",
24         // or simply save your changes to "hot reload" in a Flutter IDE).
25         // Notice that the counter didn't reset back to zero; the application
26         // is not restarted.
27         primarySwatch: Colors.blue,
28       ), // ThemeData
29       home: const MyHomePage(title: 'Flutter Demo Home Page'),
30     ); // MaterialApp
31   }
32 }
33
34 class MyHomePage extends StatefulWidget {
35   const MyHomePage({super.key, required this.title});
36
37   // This widget is the home page of your application. It is stateful, meaning
38   // that it has a State object (defined below) that contains fields that affect
39   // how it looks.
40
41   // This class is the configuration for the state. It holds the values (in this
42   // case the title) provided by the parent (in this case the App widget) and
43   // used by the build method of the State. Fields in a Widget subclass are
44   // always marked "final".
45
46   final String title;
47
48   @override
49   State<MyHomePage> createState() => _MyHomePageState();
50 }
51
52 class _MyHomePageState extends State<MyHomePage> {
53   int _counter = 0;
```

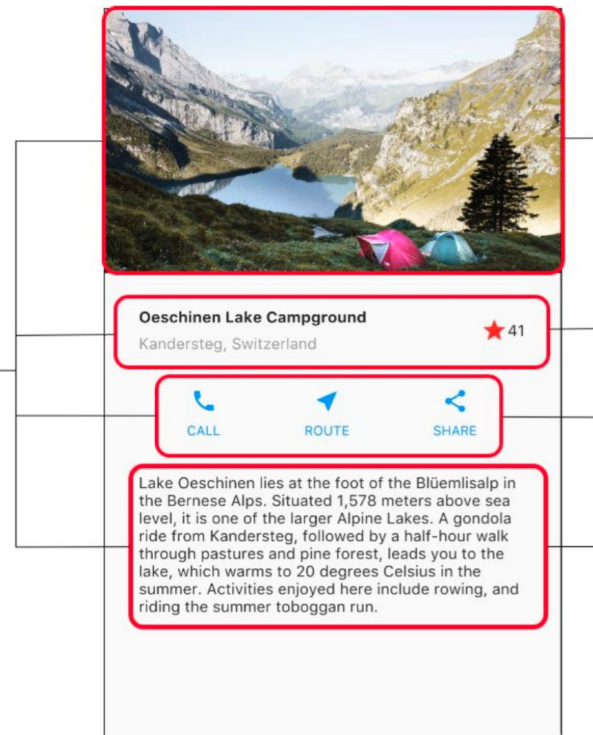
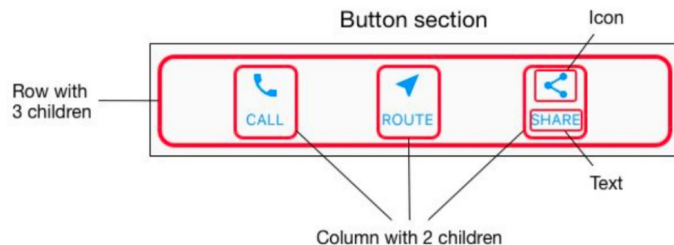
# Programmieren in Flutter

Alles ist ein Widget.

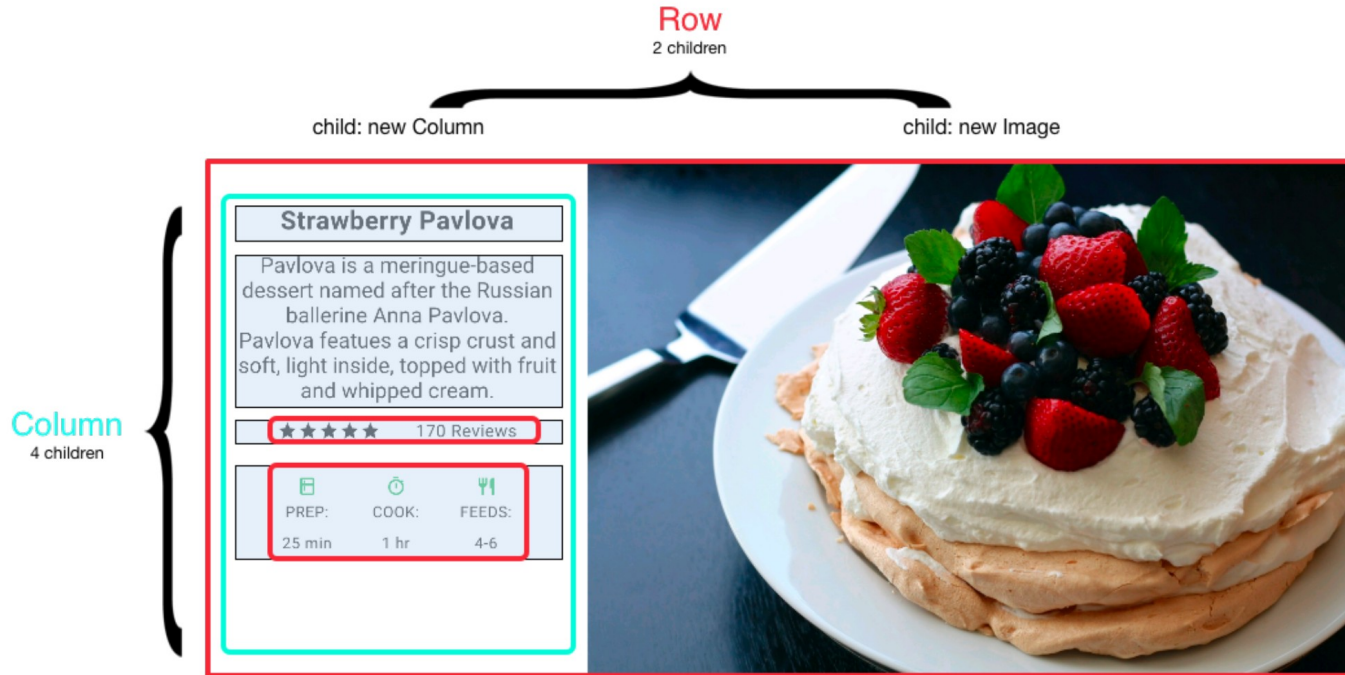


# Programmieren in Flutter

## Rows and columns

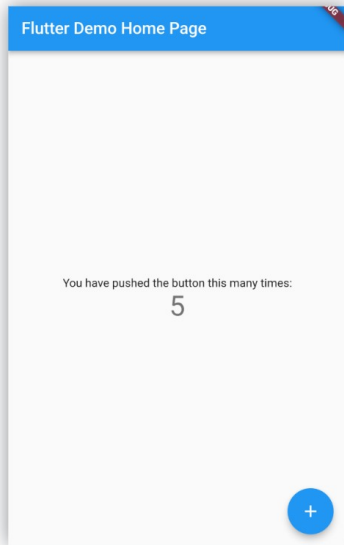


# Programmieren in Flutter

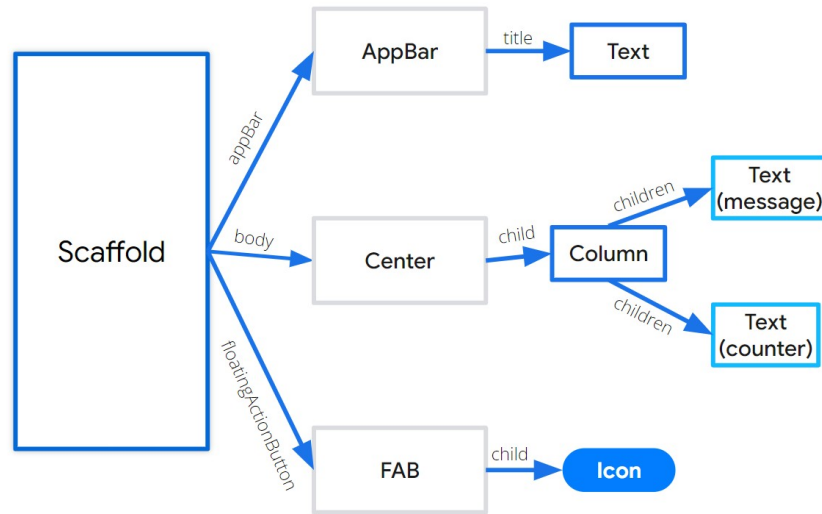


# Widget Tree

UI



Breakdown



# Die erste App

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Welcome to Flutter',
12       home: Scaffold(
13
14         appBar: AppBar(
15           title: const Text('Softwarepraktikum Sommersemester'),
16         ),
17
18         body: const Center(
19           child: Text('Meine erste App'),
20         ),
21       ),
22     );
23   }
24 }
25 }
```



# Die erste App

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Welcome to Flutter',
12       home: Scaffold(
13
14         appBar: AppBar(
15           title: const Text('Softwarepraktikum Sommersemester'),
16         ),
17
18         body: const Center(
19           child: Text('Meine erste App'),
20         ),
21       ),
22     );
23   }
24 }
25 }
```

Importiert das Material Design Widget-Toolkit, das eine umfangreiche Sammlung von Widgets und Tools zur Erstellung von Benutzeroberflächen in Flutter bietet.

# Die erste App

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Welcome to Flutter',
12       home: Scaffold(
13
14         appBar: AppBar(
15           title: const Text('Softwarepraktikum Sommersemester'),
16         ),
17
18         body: const Center(
19           child: Text('Meine erste App'),
20         ),
21       ),
22     );
23   }
24 }
25 }
```

Importiert das Material Design Widget-Toolkit, das eine umfangreiche Sammlung von Widgets und Tools zur Erstellung von Benutzeroberflächen in Flutter bietet.

Diese Funktion definiert den Einstiegspunkt der Anwendung. Hier wird `runApp` aufgerufen, um das Flutter-Widget `MyApp` auszuführen, das die gesamte Anwendung darstellt.



# Die erste App

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Welcome to Flutter',
12       home: Scaffold(
13
14         appBar: AppBar(
15           title: const Text('Softwarepraktikum Sommersemester'),
16         ),
17
18         body: const Center(
19           child: Text('Meine erste App'),
20         ),
21       ),
22     );
23   }
24 }
25 }
```

Importiert das Material Design Widget-Toolkit, das eine umfangreiche Sammlung von Widgets und Tools zur Erstellung von Benutzeroberflächen in Flutter bietet.

Diese Funktion definiert den Einstiegspunkt der Anwendung. Hier wird `runApp` aufgerufen, um das Flutter-Widget `MyApp` auszuführen, das die gesamte Anwendung darstellt.

Definiert eine neue Klasse `MyApp`, die eine erbende Klasse von `StatelessWidget` ist.

# Die erste App

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Welcome to Flutter',
12       home: Scaffold(
13
14         appBar: AppBar(
15           title: const Text('Softwarepraktikum Sommersemester'),
16         ),
17
18         body: const Center(
19           child: Text('Meine erste App'),
20         ),
21
22       ),
23     );
24   }
25 }
```

Importiert das Material Design Widget-Toolkit, das eine umfangreiche Sammlung von Widgets und Tools zur Erstellung von Benutzeroberflächen in Flutter bietet.

Diese Funktion definiert den Einstiegspunkt der Anwendung. Hier wird `runApp` aufgerufen, um das Flutter-Widget `MyApp` auszuführen, das die gesamte Anwendung darstellt.

Definiert eine neue Klasse `MyApp`, die eine erbende Klasse von `StatelessWidget` ist.

Die `build`-Methode ist eine von `StatelessWidget` vererbte Methode, die das eigentliche Widget definiert. Diese Methode gibt eine `MaterialApp`-Instanz zurück, die die gesamte Anwendung enthält.

Die `MaterialApp` verfügt über einen Titel, der im `AppBar` oben auf dem Bildschirm angezeigt wird, und einem `Scaffold`-Widget, das den eigentlichen Inhalt der Anwendung definiert.

Das `Scaffold` enthält eine `AppBar`-Instanz, die den Text "Softwarepraktikum Sommersemester" anzeigt, sowie ein `Center`-Widget, das den Text "Meine erste App" in der Mitte des Bildschirms anzeigt.

Das `child`-Attribut in der `Center`-Instanz nimmt ein `Child`-Widget auf, das hier ein `Text`-Widget ist, das den Text "Meine erste App" anzeigt.

# Die erste App

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Welcome to Flutter',
12       home: Scaffold(
13
14         appBar: AppBar(
15           title: const Text('Softwarepraktikum Sommersemester'),
16         ),
17
18         body: const Center(
19           child: Text('Meine erste App'),
20         ),
21       ),
22     );
23   }
24 }
25 }
```



# Die erste App

```
import 'package:flutter/material.dart';

void main() {
  runApp(
    Center(
      child: Text(
        'Hello, world!',
        textDirection: TextDirection.ltr,
      ),
    ),
  );
}
```



# Die erste App

```
import 'package:flutter/material.dart';

void main() {
  runApp(
    Center(
      child: Text(
        'Hello, world!',
        textDirection: TextDirection.ltr,
      ),
    ),
  );
}
```

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Welcome to Flutter',
12       home: Scaffold(
13
14         appBar: AppBar(
15           title: const Text('Softwarepraktikum Sommersemester'),
16         ),
17
18         body: const Center(
19           child: Text('Meine erste App'),
20         ),
21
22       ),
23     );
24   }
25 }
```

# Die erste App

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Welcome to Flutter',
12       home: Scaffold(
13
14         appBar: AppBar(
15           title: const Text('Softwarepraktikum Sommersemester'),
16         ),
17
18         body: const Center(
19           child: Text('Meine erste App'),
20         ),
21       ),
22     );
23   }
24 }
25 }
```

Aufgabe: Erstellen Sie gemeinsam einen Widget-Tree zu dieser App

# Die erste App

```
1  import 'package:flutter/material.dart';
2
3  void main() {
4    runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Welcome to Flutter',
12       home: Scaffold(
13
14         appBar: AppBar(
15           title: const Text('Softwarepraktikum Sommersemester'),
16         ),
17
18         body: const Center(
19           child: Text('Meine erste App'),
20         ),
21       ),
22     );
23   }
24 }
25 }
```

Aufgabe: Erweitern Sie den Widget Tree um ein weiteres Widget und implementieren Sie es in der App!

# Start!

- 1) Flutter installieren
- 2) VS Code installieren
- 3) GitHub einrichten
- 4) Erste Aufgabe



# Nächste Woche

- 1) Gemeinsame Bearbeitung des Tutorials
- 2) Bearbeitung eines weiteren Tutorials in der Lehrveranstaltung
- 3) Testen auf Geräten