Ziel der App

Die App soll Name und Nachname des Benutzers nehmen und diese Lokal Speichern

Entwicklerumgebung einrichten

---

1. Flutter Dokumentation öffnen
2. Betriebssystem auswählen und Dokumentation folgen

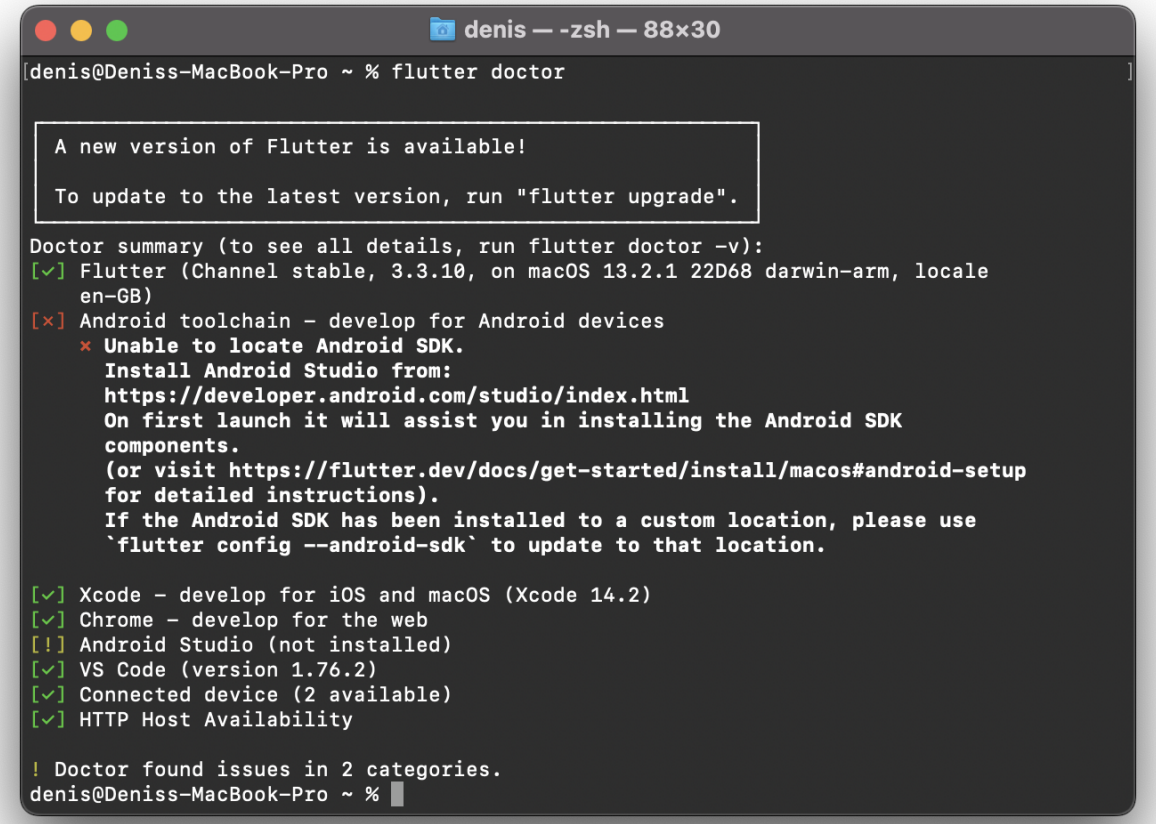
Wichtig zu merken ist:

- iOS app debugging ist nur über MacOS möglich
- Android App Entwicklung benötigt Android Studio (Auf Windows und MacOS)
- iOS App Entwicklung benötigt XCode auf MacOS (ist nicht auf Windows installierbar)
- Command Line Tools müssen installiert sein (Auf Windows sowie MacOS)
- Bei Android Entwicklung müssen USB Drivers heruntergeladen werden
    - Wichtig ist dabei zu achten, dass man die Richtigen Drivers herunterlädt. Jedes Android gerät benötigt eigene Drivers die nur über die Webseite des Herstellers hertunergeladen werden kann

> WICHTIG: Entwickleroptionen auf Andoird bzw. iOS aktivieren (USB Debugging)
> Android: https://developer.android.com/studio/debug/dev-options
> iOS: https://developer.apple.com/documentation/xcode/enabling-developer-mode-on-a-device

> iOS benötigt zudem einen Entwickler-Konto
> https://developer.apple.com/ (Oben rechts auf "Account" drücken und mit Apple ID anmelden)

Wenn alles korrekt installiert ist dann sollte im terminal `flutter doctor` einen ähnlichen Output geben:



Falls fehler aufscheinen sollten (Im screenshot z.B. fehlt Android Studio), anleitungen im terminal folgen

3. Textbearbeitungsprogramm installieren (Visual Studio Code ist empfehlenswert)

Flutter projekt starten

---

1. Terminal in einem beliebigen ordner öffnen (App ordner wird später dort gespeichert)
2. Im Terminal `flutter create <app name hier eingeben>` (<> löschen)
3. Jetzt sollte ein Ordner mit dem ausgewählten Namen zu finden sein, diesen in Visual Studio Code öffnen (oder beliebigen IDE)

App bearbeiten

> Alle Dateien, bis auf `lib/main.dart`, können momentan ignoriert werden

1. Im ordner `lib` die Datei `main.dart` öffnen

> Die `main.dart` Datei ist wichtig und ist die Hauptdatei einer Flutter App
> In `main.dart` findet man schon ein Sourcecode, weil Flutter schon eine Beispiel

> app generiert und diese kann ohne weiteres ausgeführt werden

> Flutter ist ein Framework und basiert sich auf die Programmiersprache "Dart"
> "Dart" ähnelt in vielen bereichen Java und C++, hat aber eine viel einfachere Syntax

```
// Flutter framework wird hier importiert
import 'package:flutter/material.dart';

// Die methode main() wird beim start aufgerufen und führt runApp() aus
// cont MyApp() ist die effektive App die auf dem Bildschirm angezeigt
wird
void main() {
        runApp(const MyApp());
}
```

Flutter Apps werden mit verschiedene Bausteine (Widgets) zusammengestellt

> Sehr empfehlenswert!
> Um Widgets in Flutter besser zu verstehen gibt es einen gratis Online-Kurs auf der Flutter Webseite (https://codelabs.developers.google.com/codelabs/flutter-codelab-first#0)
> Weitere resourcen zum Flutter lernen findet man unter (https://flutter.dev/learn)

App Beispiel (Rainbow-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: 'Flutter Demo',
                        theme: ThemeData(
                                primarySwatch: Colors.blue,
                        ),
                                home: const MyHomePage(),
                        );
        }
}

class MyHomePage extends StatefulWidget {
```

```
        const MyHomePage({super.key});
        @override
        State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
        int _counter = 0;

        // void wird benutzt um Funktionen in Dart zu deklarieren (wie in
Java)
        // Diese Funktion steigert die Variable _counter jedes mal um
einen Wert
        void _incrementCounter() {
                setState(() {
                        // Hier findet die Inkrementation statt (_counter
= _counter + 1)
                        _counter++;

                        // Dieses if-Statement kontrolliert ob _counter
größer ist als die Farbenliste (colors) und setzt _counter wieder auf 0,
damit kein Fehler aufscheint
                        if (_counter > colors.length - 1) {
                                _counter = 0;
                        }
                });
        }

        // List ist ein Array (Liste von variablen)
        // <MaterialColor> gibt an was für ein Datentyp sich in dieser
Liste befindet
        List<MaterialColor> colors = [
                Colors.purple,
                Colors.blue,
                Colors.amber,
                Colors.green,
                Colors.yellow,
                Colors.pink,
                Colors.teal,
                Colors.orange,
                Colors.red,
        ];

        // List ist ein Array (Liste von variablen)
        // <String> In dieser Liste befinden sich, Strings, also Text
        List<String> strings = [
                "Click me UwU",
                "SUS",
                "ok pls stop",
                "Man. stop. pls.",
```

```
                "I SAID STOP",
                "Ok fine I warned you",
                "We're no strangers to loooove...",
                "You know the rules and so do IIIII",
                "Error 69420, sins_johnny.exe stopped working",
        ];


        // Diese Funktion baut die Benutzeroberfläche (muss in jeder
Flutter-App mindestens ein Mal vorkommen)
        @override
        Widget build(BuildContext context) {
                // Alles was nach return steht, beinhaltet alle Widgets,
die auf dem Bildschirm aufscheinen sollen
                return Material(
                        color: colors[_counter],
                        child: Center(
                                child: TextButton(
                                        child: Text(
                                                strings[_counter],
                                                        style:
TextStyle(color: Colors.white, fontSize: 32)),
                                        onPressed: () => {
                                                _incrementCounter(),
                                        }),
                                ),
                        );
        }
}
```

Dies ist ein Beispielcode für eine Flutter-App, die eine Schaltfläche mit einem Text anzeigt, der sich jedes Mal ändert, wenn die Schaltfläche gedrückt wird. Die Hintergrundfarbe ändert sich ebenfalls jedes Mal.

MyHomePage ist ein StatefulWidget (unterschied zwischen StatefulWidget und StatlessWidget), was bedeutet, dass es sich während der Laufzeit ändern kann. Es enthält eine Schaltfläche, die bei jedem Drücken den Text ändert und die Hintergrundfarbe ändert. Der Text und die Hintergrundfarbe werden aus den Listen colors und strings abgefragt, die jeweils eine Reihe von vordefinierten Werten enthalten.

Die App wird mit dem Befehl "flutter run" (oder `Ctrl` + `F5` in VSCode) gestartet. Der Emulator oder das Gerät zeigt die Schaltfläche und den aktuellen Text mit der entsprechenden Hintergrundfarbe an. Wenn Sie auf die Schaltfläche klicken, wird der Text geändert und die Hintergrundfarbe wechselt entsprechend.

Dies ist ein sehr einfaches Beispiel für eine Flutter-App, die zeigt, wie man ein StatefulWidget verwendet, wie man Text und Hintergrundfarbe ändert und wie man

eine Schaltfläche hinzufügt. Es ist ein guter Ausgangspunkt für Anfänger, die lernen möchten, wie man Flutter-Apps entwickelt.

App Besipiel #2 (Text input)

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



void main() {

runApp(const MyApp());

}



class MyApp extends StatelessWidget {

const MyApp({super.key});



// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Input Beispiel',

theme: ThemeData(

// This is the theme of your application.

//

// Try running your application with "flutter run". You'll see the

// application has a blue toolbar. Then, without quitting the app, try

// changing the primarySwatch below to Colors.green and then invoke

// "hot reload" (press "r" in the console where you ran "flutter run",

// or simply save your changes to "hot reload" in a Flutter IDE).
```

```
// Notice that the counter didn't reset back to zero; the application

// is not restarted.

primarySwatch: Colors.blue,

),

home: const MyHomePage(title: 'Input Beispiel'),

);

}

}



class MyHomePage extends StatefulWidget {

const MyHomePage({super.key, required this.title});



// This widget is the home page of your application. It is stateful,
meaning

// that it has a State object (defined below) that contains fields that
affect

// how it looks.



// This class is the configuration for the state. It holds the values (in
this

// case the title) provided by the parent (in this case the App widget)
and

// used by the build method of the State. Fields in a Widget subclass are

// always marked "final".



final String title;
```

```
@override

State<MyHomePage> createState() => _MyHomePageState();

}



class _MyHomePageState extends State<MyHomePage> {

TextEditingController text = TextEditingController();



@override

Widget build(BuildContext context) {

// This method is rerun every time setState is called, for instance as
done

// by the _incrementCounter method above.

//

// The Flutter framework has been optimized to make rerunning build
methods

// fast, so that you can just rebuild anything that needs updating rather

// than having to individually change instances of widgets.

return Scaffold(

appBar: AppBar(

// Here we take the value from the MyHomePage object that was created by

// the App.build method, and use it to set our appbar title.

title: Text(widget.title),

),

body: Column(

children: [
```

```
Spacer(),

Center(

child: Text(

'${text.text}',

style: TextStyle(fontSize: 72),

),

),

Spacer(),

Padding(

padding: const EdgeInsets.symmetric(horizontal: 20.0),

child: TextField(

onChanged: (value) {

setState(() {});

},

decoration: InputDecoration(

border: OutlineInputBorder(),

hintText: 'Geben Sie hier text ein',

),

controller: text,

),

),

Spacer()

],

),

);
```

```
}

}
```

Diese Beispiel-App nimmt Input durch ein Textfeld ein und zeigt dies an, wenn sich dieser ändert (onChange auf Zeile 80)