

Sensoren in Flutter

Zugriff auf das Handy

- Das Handy verfügt über eine Vielzahl an Sensoren und interne Information
- Wie kann man auf diese zugreifen?
 - In der Regel mithilfe von Bibliotheken
 - Diese binden manchmal (system)spezifische Bibliotheken mit ein

Accelerometer und Gyro

- Im Handy sind Sensoren zur Identifikation von Bewegung verbaut
 - Accelerometer und Gyro
- Verwendete Bibliothek
 - sensors

```
19 dependencies:  
20   flutter:  
21     sdk: flutter  
22   sensors: ^0.4.1+3
```

Accelerometer und Gyro

- Der Zugriff erfolgt über die Registrierung von Listenern

```
31      @override
32      void initState() {
33          accelerometerEvents.listen((AccelerometerEvent event) {
34              setState(() {
35                  _acc_x = event.x.toStringAsFixed(2);
36                  _acc_y = event.y.toStringAsFixed(2);
37                  _acc_z = event.z.toStringAsFixed(2);
38              });
39          });
40          // [AccelerometerEvent (x: 0.0, y: 9.8, z: 0.0)]
41
42          userAccelerometerEvents.listen((UserAccelerometerEvent event) {
43              print(event);
44          });
45          // [UserAccelerometerEvent (x: 0.0, y: 0.0, z: 0.0)]
46
47          gyroscopeEvents.listen((GyroscopeEvent event) {
48              print(event);
49          });
50      }
```

UDID: Universal Device ID

- Manchmal ist es notwendig eine eindeutige Identifikation für das Handy zu erzeugen
 - UDID
- Verwendete Bibliotheken
 - flutter_udid 1.0.1

Zugriff auf Kontakte und Permissions

- Handies speichern intern Kontakte.
 - Will man auf diese zugreifen, so muss zunächst die Erlaubnis erteilt werden
 - Im zweiten Schritt fragt das System dann an
- Verwendete Bibliothek
 - `contacts_service` 0.3.9

Zugriff auf die Bildergalerie

- Das Handy ist eine leistungsstarke Kamera, deren Fotos in der Bildergalerie gespeichert werden
- Verwendete Bibliotheken
 - image_picker 0.6.2+1

Zugriff auf die Kamera

- Aus Flutter heraus kann man auch die Kamera ansteuern und Bilder aufnehmen (und diese dann in der App anzeigen)
- Verwendete Bibliotheken
 - <https://flutter.dev/docs/cookbook/plugins/picture-taking-camera>

Da war noch was: beliebige Farben

Das Farbschema einer App

- Flutter verfügt über sogenannte Themes
 - Mit diesen werden die globalen (visuellen) Einstellungen einer App definiert
- Man kann ein eigenes Theme definieren
 - entweder komplett neu
 - Oder als modifizierte Kopie des Originals

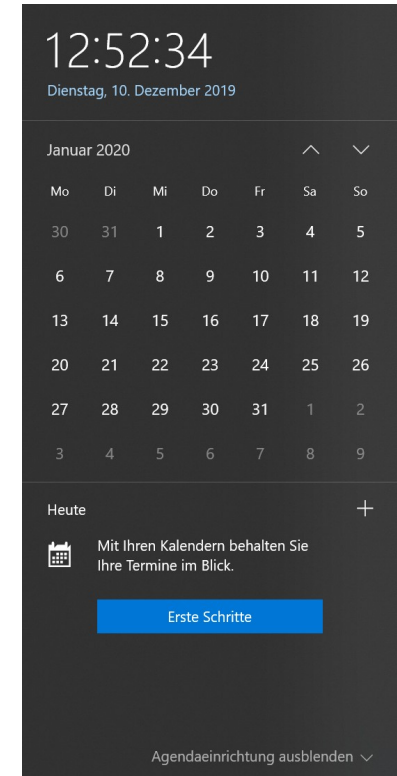
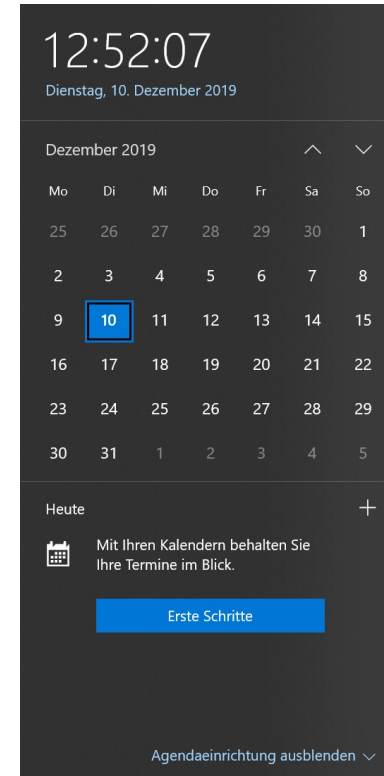
Beispiel: Farbschema

```
10 class MyApp extends StatelessWidget {  
11     @override  
12     Widget build(BuildContext context) {  
13         return MaterialApp(  
14             debugShowCheckedModeBanner: false,  
15             title: 'Leihladen App',  
16             theme: new ThemeData(  
17                 primaryColor: const Color(0xFF004F8B),  
18                 primaryColorDark: const Color(0xFF003F6F),  
19                 accentColor: const Color(0xFF018EFB),  
20                 scaffoldBackgroundColor: const Color(0xFFFFFFFF),  
21                 bottomAppBarColor: const Color(0xFFEEEEEE),  
22             ), // ThemeData  
23             home: StartScreen(),  
24         ); // MaterialApp  
25     }  
26 }
```

Finales Projekt

Ursprüngliche Vorgaben

- Teamorientiertes Arbeiten
- Grundlage sind agile Vorgehensmodelle
 - Pairprogramming
 - Projekte werden hochgeladen,
- Arbeitsfortschritte sollen sichtbar sein
- Abschließendes Projekt
 - Selbst gewähltes Thema
 - Insgesamt 5 Wochen von der Idee
 - zum Ergebnis



Sprint 0: Thema wählen

- Zwei Arbeitsschritte:
 - Gruppen bilden
 - Thema erarbeiten
- Kurzpräsentation des gewählten Themas
 - Kurzbeschreibung Thema
 - Vergleichbare Anwendungen
 - Funktionsbeschreibung
 - Soll und Kann
- Design
 - Farbkonzept
 - Screendesign mit Photoshop
 - Interaktionsdesign mit <https://www.invisionapp.com/>
- Sprintplanung
 - Teilziele für die Sprints festlegen