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

```dart
@override
void initState() {
  accelerometerEvents.listen((AccelerometerEvent event) {
    setState(() {
      _acc_x = event.x.toStringAsFixed(2);
      _acc_y = event.y.toStringAsFixed(2);
      _acc_z = event.z.toStringAsFixed(2);
    });
  });
// [AccelerometerEvent (x: 0.0, y: 9.8, z: 0.0)]

  userAccelerometerEvents.listen((UserAccelerometerEvent event) {
    print(event);
  });
// [UserAccelerometerEvent (x: 0.0, y: 0.0, z: 0.0)]

  gyroscopeEvents.listen((GyroscopeEvent event) {
    print(event);
  });
}
```

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 Kontake.
  - Will man auf diese zugreifen, so muss zunächste 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-using-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

```dart
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Leihladen App',
      theme: new ThemeData(
        primaryColor: const Color(0xFF004F8B),
        primaryColorDark: const Color(0xFF003F6F),
        accentColor: const Color(0xFF018EFB),
        scaffoldBackgroundColor: const Color(0xFFFFFFFF),
        bottomAppBarColor: const Color(0xFFEEEEEE),
      ), // ThemeData
      home: StartScreen(),
    ); // MaterialApp
  }
}
```