Automotive OS

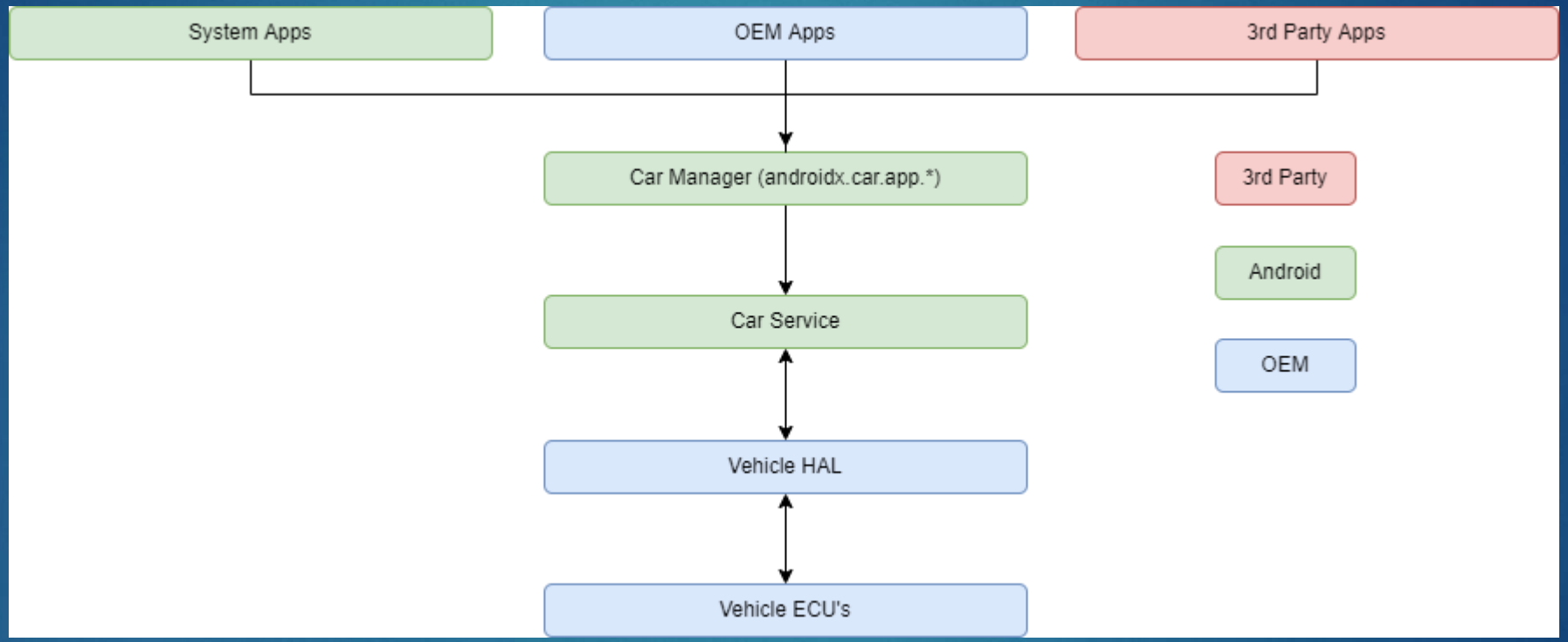
Herausforderungen

- Umgebung
- Sicherheit
- Aktualisierungen
- Stromverbrauch
- Rückfahrkamera

Automotive OS

- Android basiertes OS für das Auto
- Nicht mit Android Auto verwechseln
- Teil von Android Open Source Project (AOSP)
- Funktionserweiterungen durch Automobilhersteller notwendig
- Angekündigt im März 2017
- Google Automotive Services (GAS)

Architektur



Systemeigenschaften

- Über 150
- Werden anhand des VehiclePropertyGroup:SYSTEM Attributes identifiziert

```
PERF_VEHICLE_SPEED = (
  0x0207
  | VehiclePropertyGroup:SYSTEM
  | VehiclePropertyType:Float
  | VehicleArea:GLOBAL
)
```

- (hardware/interfaces/automotive/vehicle/2.0/types.hal)

- Hersteller können mit VehiclePropertyGroup:VENDOR Attribute hinzufügen

Eigenschaften

change_mode:

STATIC

ON_CHANGE

CONTINUOUS

Permissions

- Benötigt um die Car Services zu verwenden `android.car.permission.*`
- 3rd Party Apps sind in der Auswahl eingeschränkt

CAR_INFO

READ_CAR_DISPLAY_UNITS

CONTROL_CAR_DISPLAY_UNITS

CAR_ENERGY_PORTS

CAR_EXTERIOR_ENVIRONMENT

CAR_POWERTRAIN

CAR_SPEED

CAR_ENERGY

- Sonstige sind als **signature | privileged** versehen

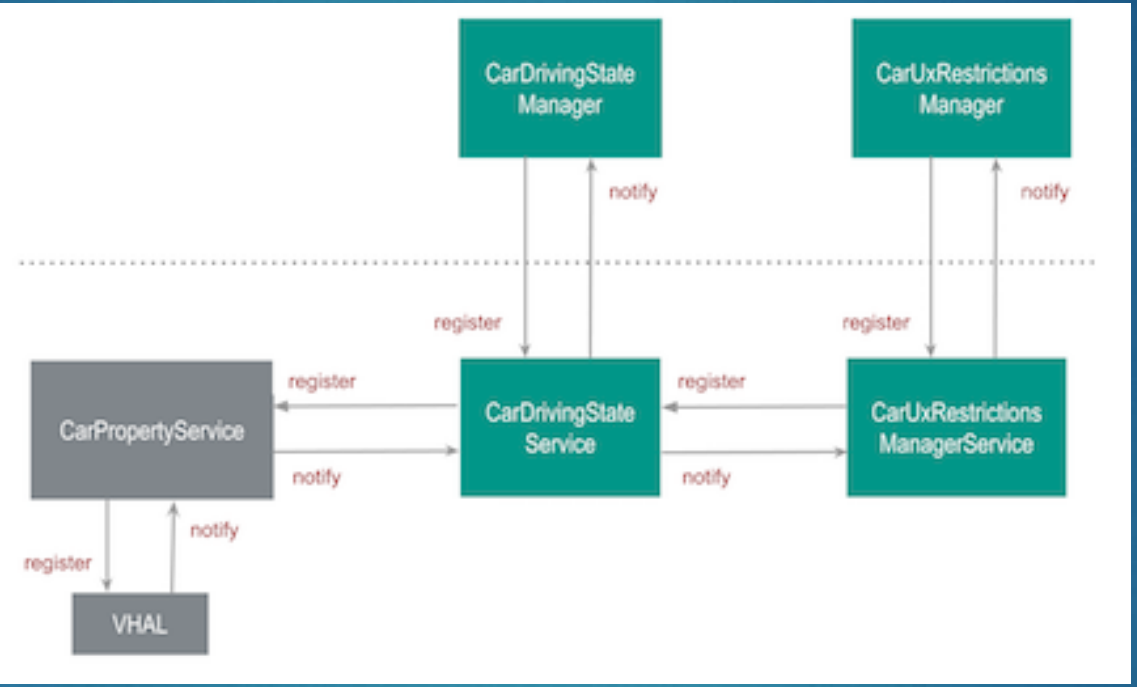
```
val listener: OnClickListener = ParkedOnlyOnClickListener.create {
carContext.requestPermissions(permissions) {
 approved: List<String?>?, rejected: List<String?>? ->
        CarToast.makeText(
            carContext,
            String.format("Approved: %s Rejected: %s", approved,
rejected),
            CarToast.LENGTH_LONG
        ).show()
    }
}
```

Fahrerablenkung

- Ablenkungsoptimierte Anwendungen
- UX-Einschränkungen während der Fahrt

```
<activity …>
….
<meta-data android:name="distractionOptimized"
android:value="true"/>
...
</activity>
```

- Drei Zustände: Geparkt, Leerlauf, Bewegen


CarDrivingState Manager
CarUxRestrictions Manager
notify
notify
register
register
CarPropertyService
register
CarDrivingState Service
register
CarUxRestrictions ManagerService
notify
notify
notify
register
VHAL

3rd Party Apps

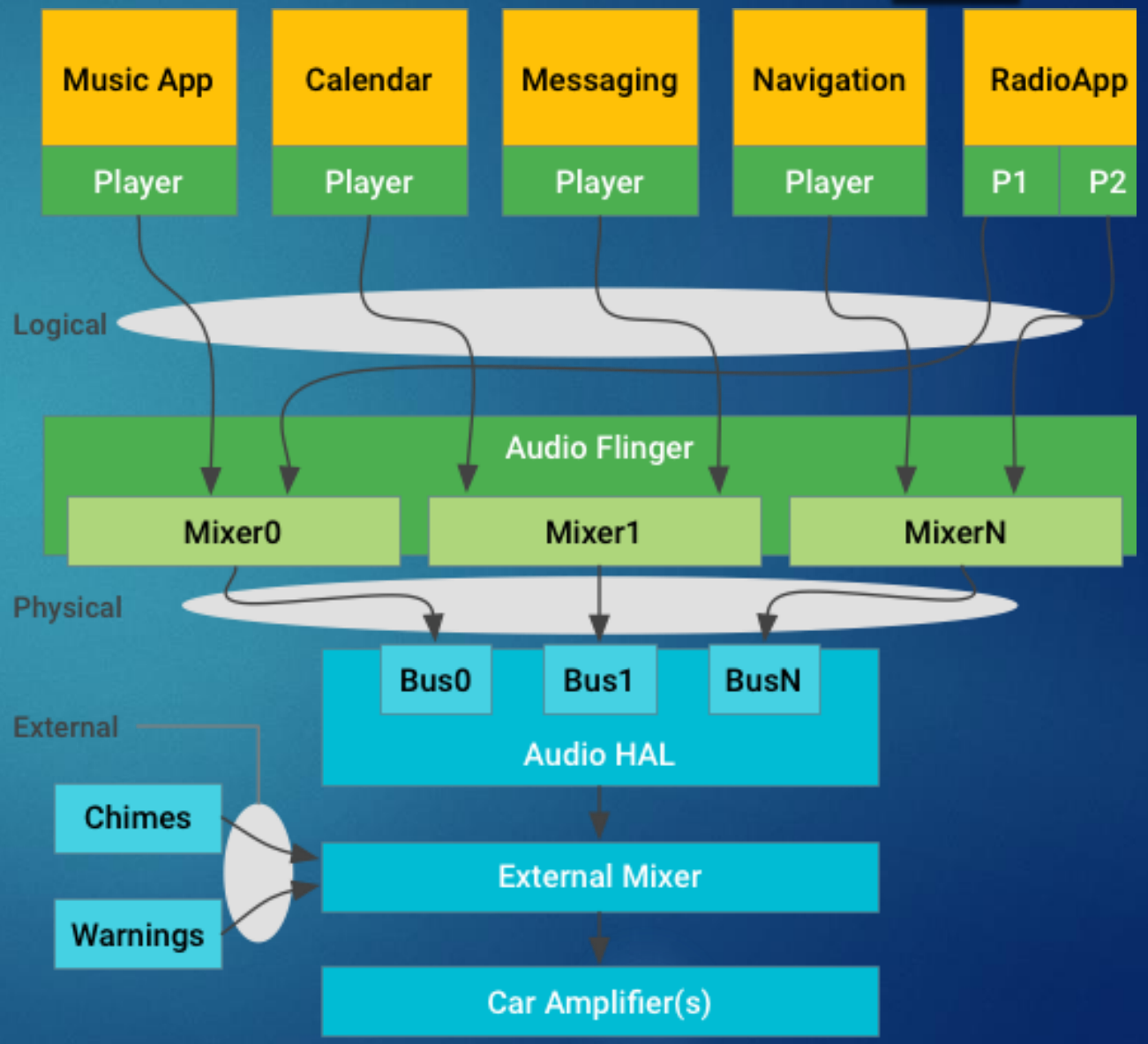
- Messengers
- Media
- Navigation

Extended-View-System

- Zu lange Bootzeiten von Android
- EVS braucht ca. zwei Sekunden zum booten
- Notwendig auf Grund von Androids Stromverbrauch

Audio

- Android ist zuständig für Infotainment Sounds
- AAOS stellt ein Interface zur Audioverwaltung bereit
- OEM muss einen External Mixer bereitstellen
- Audio HAL kann Medien pausieren



AndroidManifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"...>
...
    <uses-feature
      android:name="android.hardware.type.automotive"
      android:required="true" />
     <application ...>
      <activity
       android:name="androidx.car.app.activity.CarAppActivity"
       android:exported="true"
       ...>
       <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
         <meta-data android:name="distractionOptimized" android:value="true"/>
      </activity>
```

```
<service
        android:name=".services.MyCarAppService"
        android:exported="true"
        ...>
        <intent-filter>
            <action android:name="androidx.car.app.CarAppService" />
            <category android:name="androidx.car.app.category.POI" />
        </intent-filter>
    </service>
    <meta-data
        android:name="androidx.car.app.minCarApiLevel"
        android:value="1" />
  </application>
</manifest>
```

build.gradle

- Android Studio version > 4.2
- Mindestens API Level 29 erforderlich

```
dependencies {
  ...
  implementation "androidx.car.app:app-automotive:x.x.x"
  ...
}
```

CarAppService

androidx.car.app.CarAppService

```
class MyCarAppService : CarAppService() {
    override fun createHostValidator(): HostValidator {
        return if (applicationInfo.flags and ApplicationInfo.FLAG_DEBUGGABLE != 0) {
            HostValidator.ALLOW_ALL_HOSTS_VALIDATOR
        } else {
            HostValidator.Builder(this)
                .addAllowedHosts(R.array.hosts_allowlist_sample)
                .build()
        }
    }

    override fun onCreateSession(): Session {
        return HelloSession()
    }
}
```

Text hinzufügen

Session

Androidx.car.app.Session

```
class HelloSession : Session(), DefaultLifecycleObserver {
    override fun onCreateScreen(intent: Intent): Screen {
        return HelloWorldScreen(carContext)
    }

    override fun onDestroy(owner: LifecycleOwner) {
        super.onDestroy(owner)
    }

    override fun onCarConfigurationChanged(newConfiguration: Configuration) {
        super.onCarConfigurationChanged(newConfiguration)
    }

   override fun onNewIntent(intent: Intent){

   }
}
```

Screen

```
Androidx.car.app.Screen
class HelloWorldScreen(carContext: CarContext) : Screen(carContext) {
   init {
      val lifecycle = lifecycle
       lifecycle.addObserver(object : DefaultLifecycleObserver {
          override fun onCreate(owner: LifecycleOwner) {
             ...
          }
          ...
       }
   }
   override fun onGetTemplate(): Template {
      ...
   }
}
```

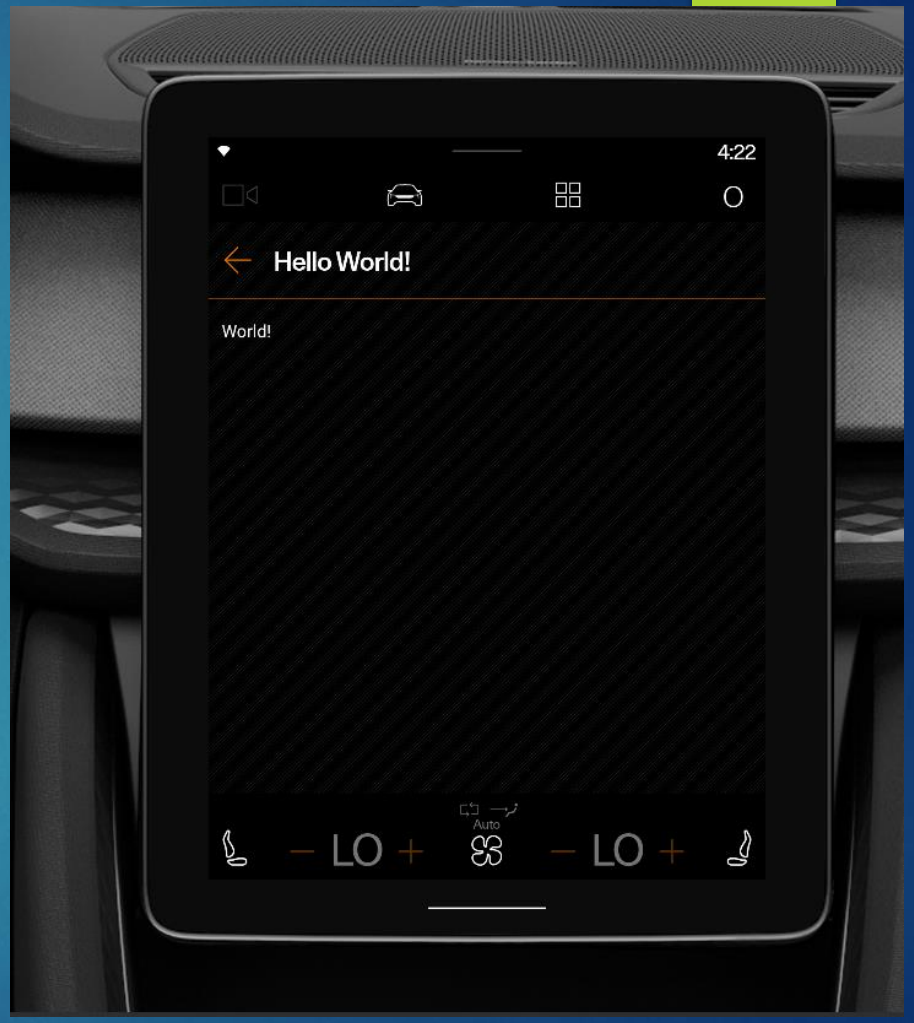
- `invalidate()` um die UI zu updaten

Templates

Androidx.car.app.model.Template

```
override fun onGetTemplate(): Template {
        val row = Row.Builder()
        row.setTitle("World!")
        val pane = Pane.Builder()
        pane.addRow(row.build())
         return PaneTemplate.Builder(pane.build())
            .setTitle("Hello World!")
            .setHeaderAction(Action.BACK)
            .build()
 }
```

androidx.car.app.model.Action



```
override fun onGetTemplate(): Template {
    val listBuilder = ItemList.Builder()
    listBuilder.addItem(
        Row.Builder()
            .setTitle("Grant Permissions")
            .setOnClickListener {
                screenManager.push(
                        RequestPermissionScreen(carContext)
                    )
            }
            .build())
    listBuilder.addItem(
        Row.Builder()
            .setTitle("Show Current Speed")
            .setOnClickListener {
                screenManager.push(
                        SpeedScreen(carContext)
                    )
            }
            .setBrowsable(true)
            .build())
    return ListTemplate.Builder()
        .setSingleList(listBuilder.build())
        .setTitle("Labor MobEng Car Info App")
        .setHeaderAction(Action.APP_ICON)
        .build()
}
```

Fahrzeugeigenschaften

```
val carHardwareManager = carContext.getCarService(
    CarHardwareManager::class.java
)
val carInfo = carHardwareManager.carInfo

carInfo.addSpeedListener(mCarHardwareExecutor!!,
speedListener)


var speedListener = OnCarDataAvailableListener<Speed> {
    synchronized(this) {
        s = it
        invalidate()
    }
}
```

Zukunft von Automotive OS

- Volvo, Polestar, General Motors und Fiat Chrysler sind Kunden
- Geringe Entwicklungskosten
- Bekannte Services
- Abhängigkeit von Google

Laboraufgabe 2

Bitte die bereitgestellte VM auf der Masterfestplatte verwenden

https://github.com/Leantar/Android-Derivates-Aufgaben

Zeit: 30 Minuten