Öffline

Offline Public Transport Application

App Idea

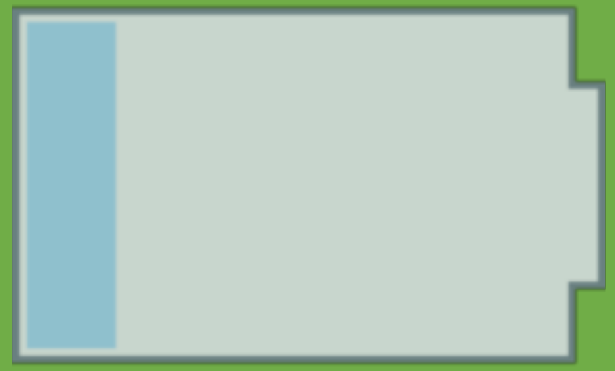
- App for public transport information
- Also usable in offline situations like…
  - … if you are in a underground disco
  - … if you are out of traffic
  - … if you are in rural environment

- Solution:
  - Download of selected schedules for offline usage
    → Automatic update whenever Wi-Fi is available
  - Download of schedules for closed by stops

Challenges

Offline Challenge

Energy Challenge



Challenges

Offline Challenge

- Solution:
  - Download and save in DB (SQLite)
    → Offline Availability
  - Automatic update of saved connections

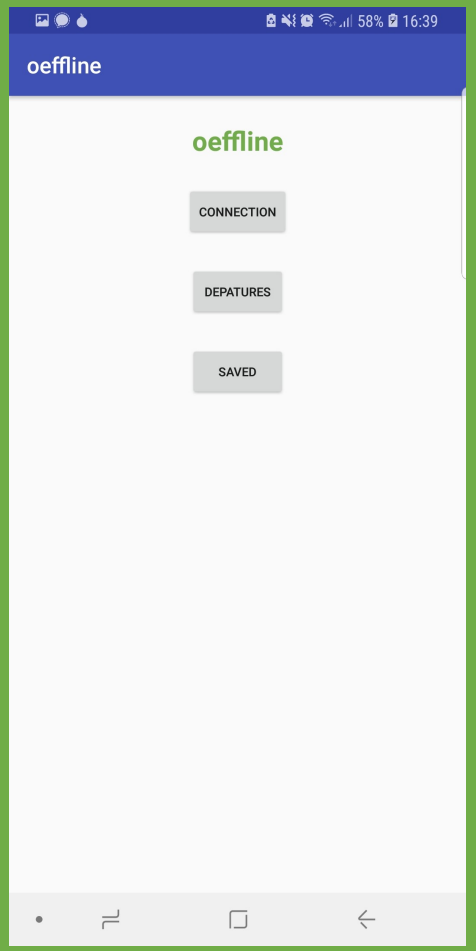
Energy Challenge

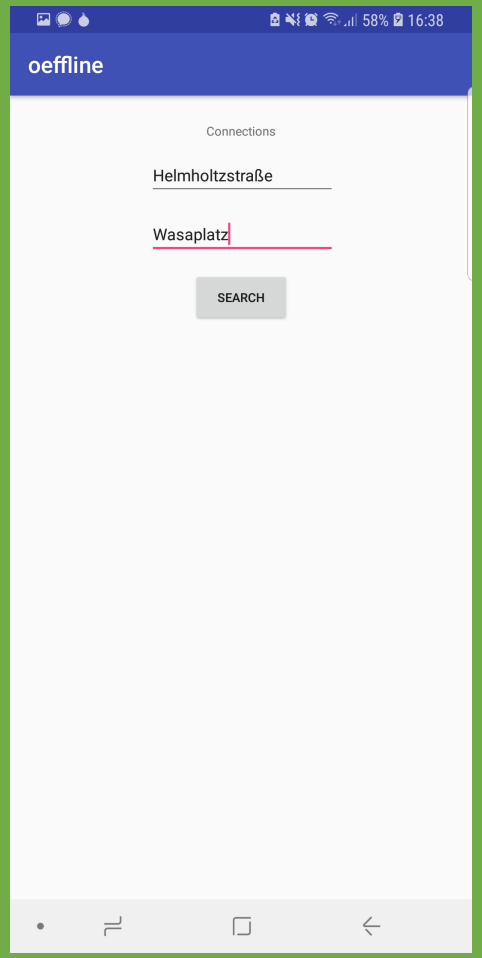
- Solution:
  - Updates only with WiFi connection
  - → No non-stop usage of energy for traffic

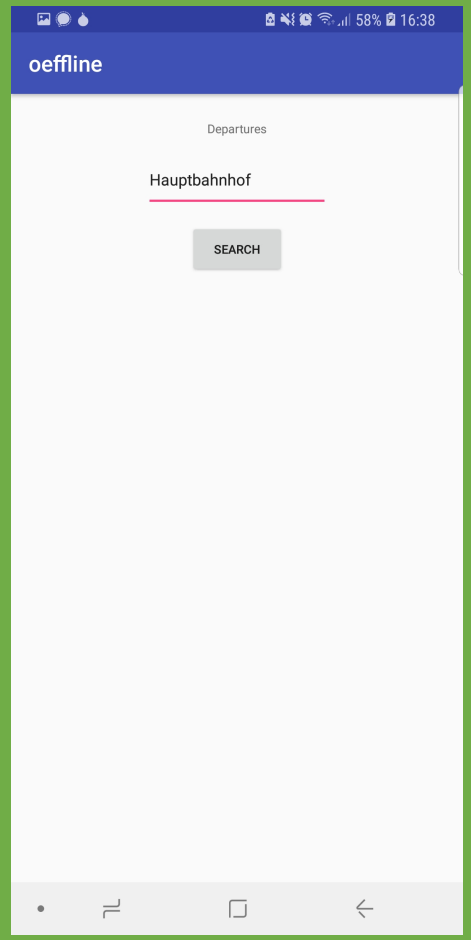
Functions

- Search and store schedules
- Search departures
- Automatic update of saved schedules
- Pre-saved departures from nearest stop

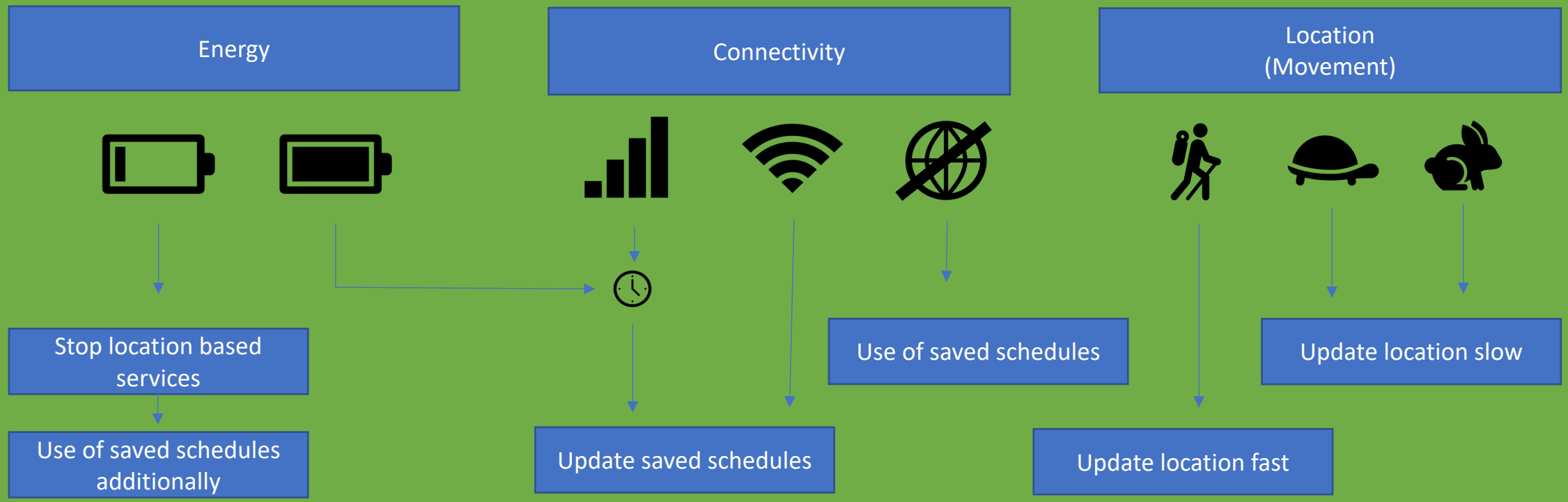
The App


58% 16:39
oeffline
oeffline
CONNECTION
DEPATURES
SAVED


58% 16:38
oeffline
Connections
Helmholtzstraße
Wasaplatz
SEARCH


58% 16:38
oeffline
Departures
Hauptbahnhof
SEARCH

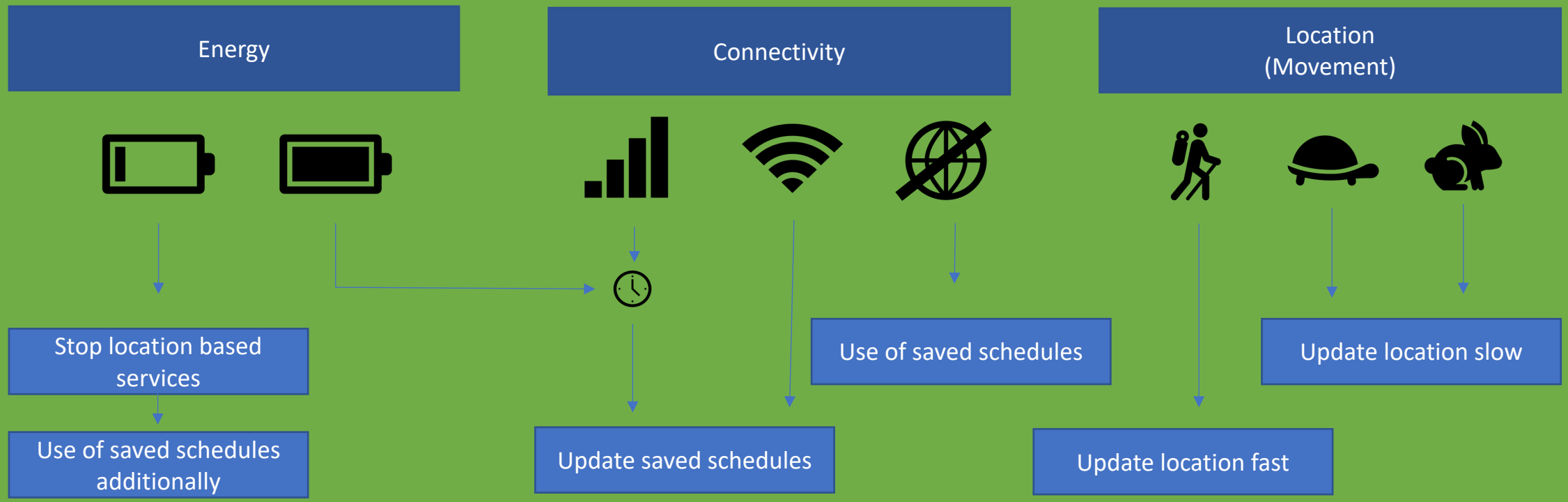
Context Adaption



1 day

30%

Context Adaption



Adaption: Change of the cycle speed

1 day

30%

```
fun setAdaptionTime(connectionState:ConnectionState, energyState: EnergyState){

    when {
        // If pedestrian is normal going (between 5 m/s and 1/ms // 18 km/h and 3,6 km/h
        getSpeed() < MAX_PEDESTRIAN_SPEED && getSpeed() > MIN_PEDESTRIAN_SPEED ->{
            ADAPTION_TIMER = 5
            Log.i(TAG,  msg: "Moving pedestrian, ADAPTION_TIMER = 5")

        }
        // If pedestrian is very slow (stay on one place) or too fast (travels tram/train/car...)
        getSpeed() > MAX_PEDESTRIAN_SPEED || getSpeed() < MIN_PEDESTRIAN_SPEED -> {
            ADAPTION_TIMER = 120
            Log.i(TAG,  msg: "Staying or fast traveling pedestrian, ADAPTION_TIMER = 120")
        }
        else ->{
            ADAPTION_TIMER = 5
        }
    }

    when {
        // An der Steckdose // Vollgeladen
        energyState.isCharging -> {
            Log.i(TAG,  msg: "Charging, no adaption from ADAPTION_TIMER by energy handler useful")
        }
        // If energylevel is lower then 30%
        (energyState.batteryPct?.compareTo(LOW_ENERGY))!! < 0 -> {
            ADAPTION_TIMER = 120
            Log.i(TAG,  msg: "Battery lower then 30%, ADAPTION_TIMER = 120")
        }
        // If energylevel is lower then 15%
        (energyState.batteryPct?.compareTo(LOW_ENERGY / 2))!! < 0 -> {
            ADAPTION_TIMER = 1800
            Log.i(TAG,  msg: "Battery lower then 15%, ADAPTION_TIMER = 1800")
        }
    }
}
```

Technologies

GitHub

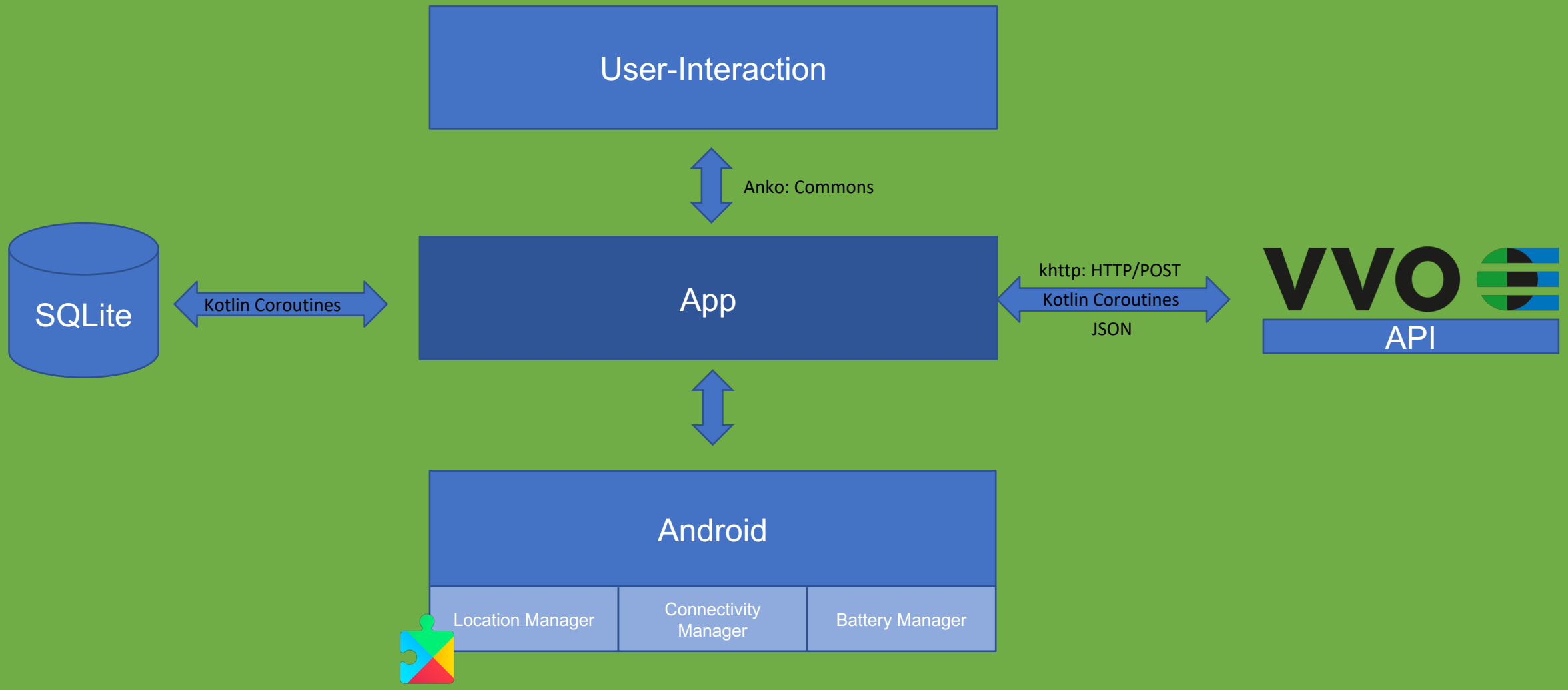
Anko

Klaxon

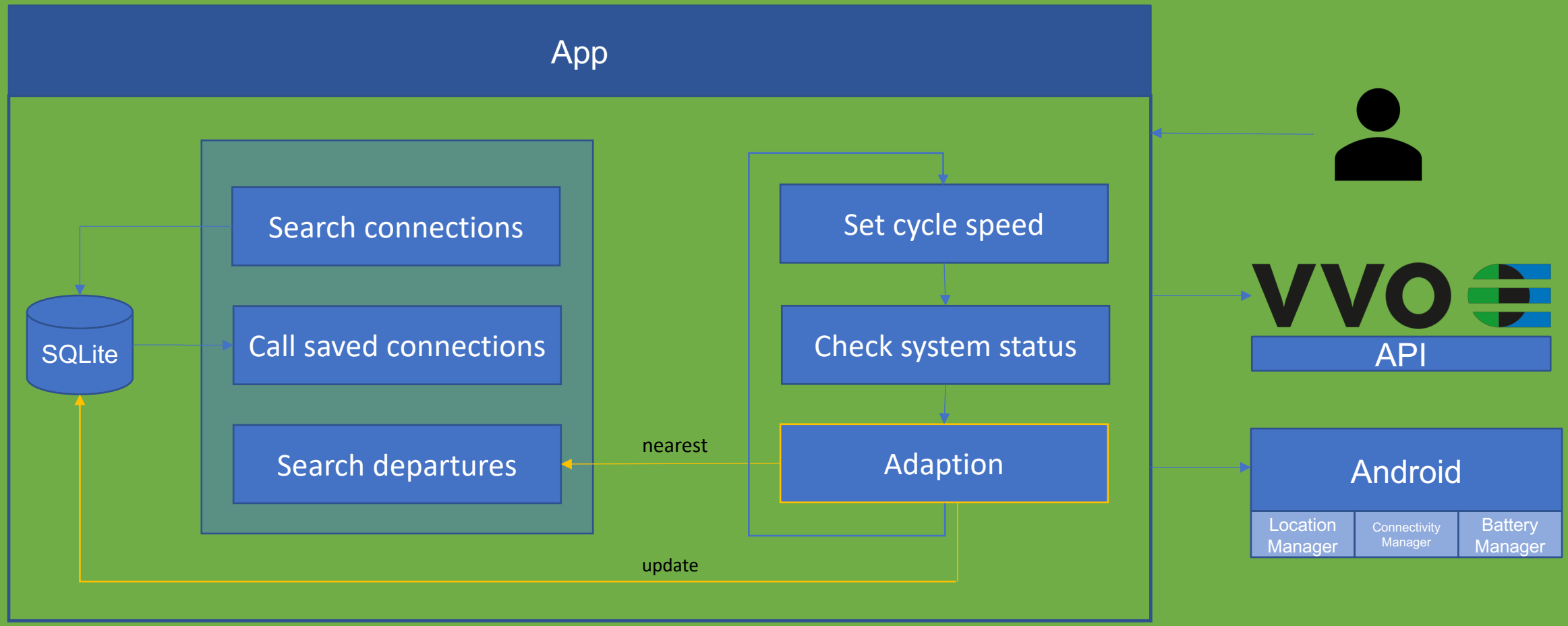
VVO

Web-API

Architecture



Architecture II



Open Issues

- Instead of Mock-Up Data use API Result
- better calculation of speed
- convert location data from GPS to GK4
- Minor bugs

Problems

- Asynchronicity
- MockUp-Data to SQLite
- Ressource-hungry Android Studio
- Get states of system (location, energy, connection)
- Time

Lessons learned

- Android Studio
- Kotlin ♡
- Asynchronicity
- Pair Programming & Meetings
- How to get system information and ask polite for it