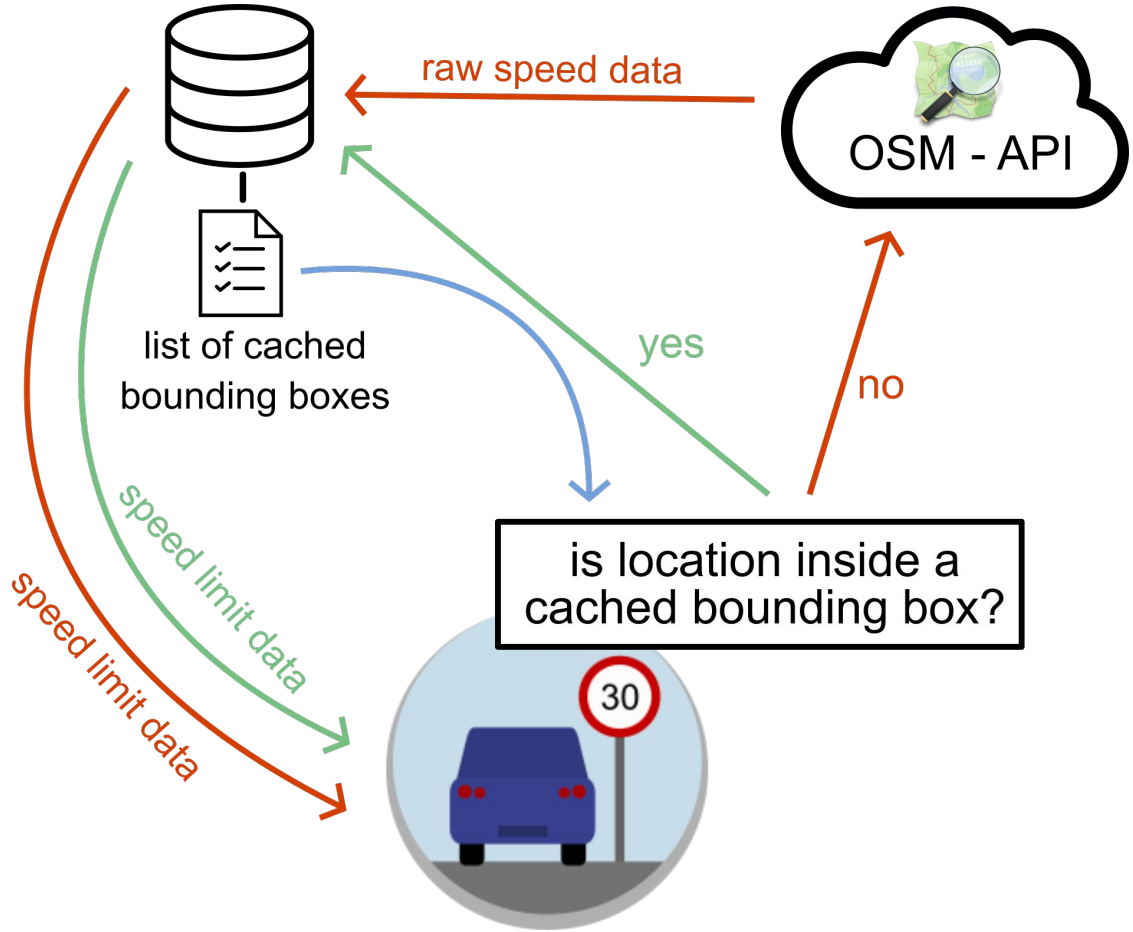
Spover

Speedoverlay for navigation

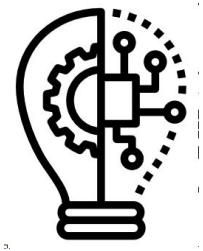
Use cases

1) automatic application start and stop
2) while driving
3) adapts UI based on current speed
4) light change causing a theme change
5) caching speed limit data

Leaving the bounding box

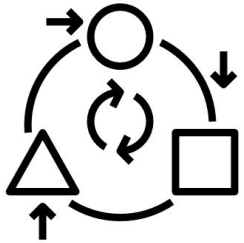


Technologies



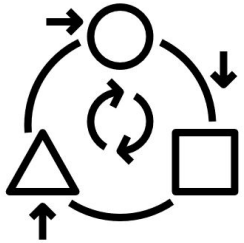
- Android SDK
  - Service to display an UI over other apps
  - HttpsUrlConnection to fetch data from OpenStreetMaps API
  - Room database library to save speed limit data locally
- Jackson library to parse XML response
- JUnit for testing

Context Information



- Current location of device (and thus the users location)
  - obtained via the GPS unit of the device
  - allows calculation of current speed
- Brightness of the environment
  - obtained via the brightness sensor of the device
  - evaluated to higher level context (light modes: light and dark)
- Start of navigation
  - obtained by listening for the Google Maps notification
- Speed limit data
  - fetched from OpenStreetMaps

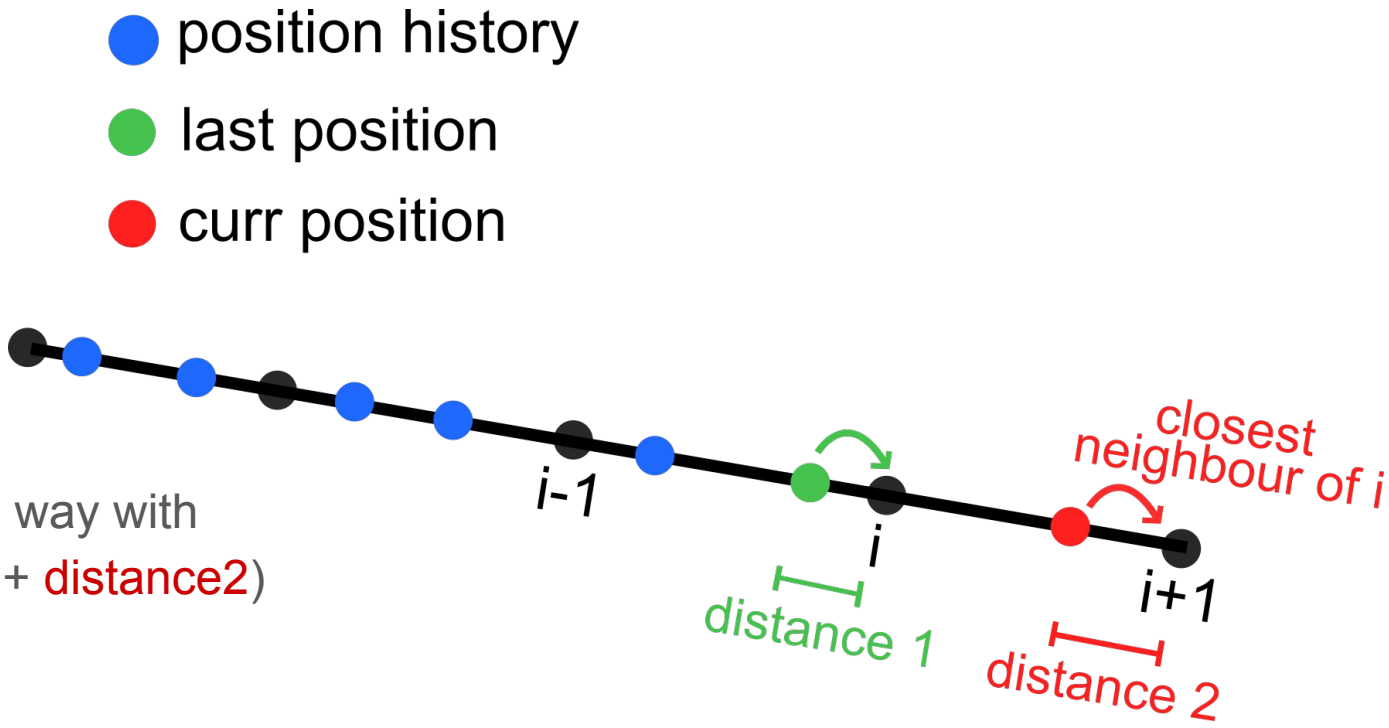
Adaptation Mechanisms



- display current speed
  - current measured speed = traveled distance between last two points divided by needed time
  - current speed is weighted speed history
    - recently added values are higher weighted
    - compensates sudden jumps / location inaccuracies

- display current speed limit
  - mapping to closest way (next slide)
  - evaluating potential speed limit condition (e.g. time, weekdays, ...)
  - setting speed limit as number or emoji

Way mapping



- **current way** = way with min(distance1 + distance2)

Adaptation of application data

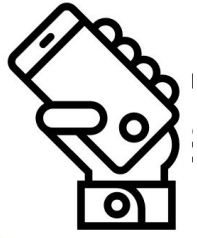
Reduction **on server side** by:

- requesting only data related to streets

Reduction and transformation **on client side** by:

- filtering received data and keeping only minimum to determine the current way and speed limit
- parsing received xml to database entries

Adaptation of UI/UX



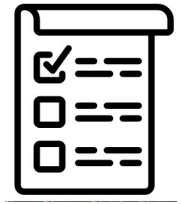
- with help of **current speed** and **speed limit**:
  - adapt the UI to driving style
  - emit warnings when first time exceeding speed limit + self set threshold (can be disabled)

- adapt overlay theme to current light mode
- starting/stopping Spover automatically when a navigation start/end ist detected
  - realized through scanning for sticky notifications from Google Maps

Lesson Learned

- Mapping out features by importance
  - implement important features first
  - don’t implement less important features, when time is running out
- Having one consistent data source
  - automatic and manual download work seamlessly together
- Kotlin is beautiful
  - clean syntax
  - more null safety than with Java

Open Issues



- improve UI
  - settings screen
  - map screen
- improve UX
  - interactions with overlay
  - interactions with map screen
- reduce the amount of data that is being automatically fetched
- add automatic deletion of old map data, that was not fetched manually