

Department of Computer Science

Institute for System Architecture, Chair for Computer Networks

**Application Development for Mobile and Ubiquitous Computing**

# **TRACKSETTER**

Final Presentation

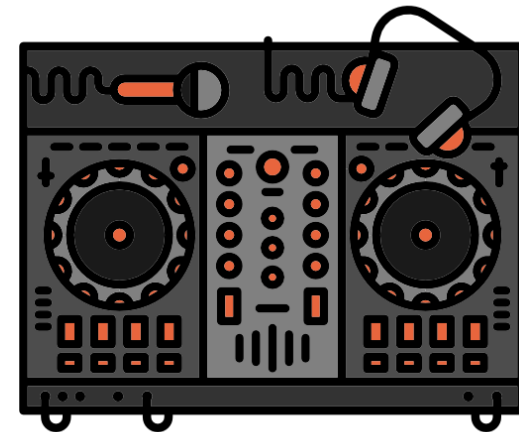
Group 5: Felix Schmiedt, Jakub Letos  
Dresden, 26. Januar 2017


## Overview

If you like spinning real vinyl as a DJ, handling your whole music library can be quite tedious:

- Did you forget a track you wanted to play?
- What next track fits BPM- and/or genre-wise?

For that we propose the app „TrackSetter“ – an electronic catalogue of your music, with which you can also create a playlist.





Artist	Title	BPM
Moderat	A New Error	110
Trent Reznor and Atticus Ross	In Motion	124
Gorillaz	Stylo (feat. Mos Def and Bobby Womack)	100
Pantha Du Prince	Bohemian Forest	126
Ben Klock	Subzero - Original Mix	124
Recondite	Levo	123
Burial	Archangel	135
Jon Hopkins	Open Eye Signal	122
Stephan Bodzin	Singularity	120
Clark	Winter Linn	90
Solumun	Friends	0
Howling	Howling - Âme Remix	12

## Screen 1

- You have a sortable List-View of your current library.
- In which you can also add a track.
- The track can be added via an API (Spotify Web API)
- Or manually (when offline / no track was found).
- All the info & parameters are then gonna be stored locally in a database.

TrackSetter

## Add Track

Clark

Winter Linn

Album

Add a genre

SAVE

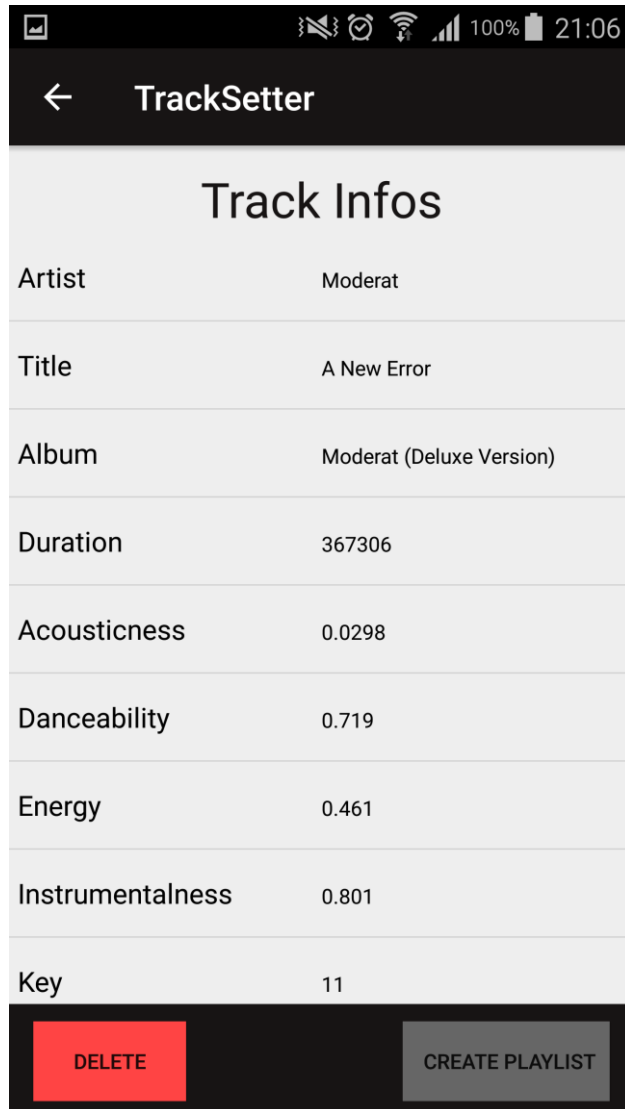
## Screen 2

- Add track manually
- Additional information is added later – if found – online

Burial	Archangel	135
Jon Hopkins	Open Eye Signal	122
Stephan Bodzin	Singularity	120
Clark	Winter Linn	90
Solumun	Friends	0

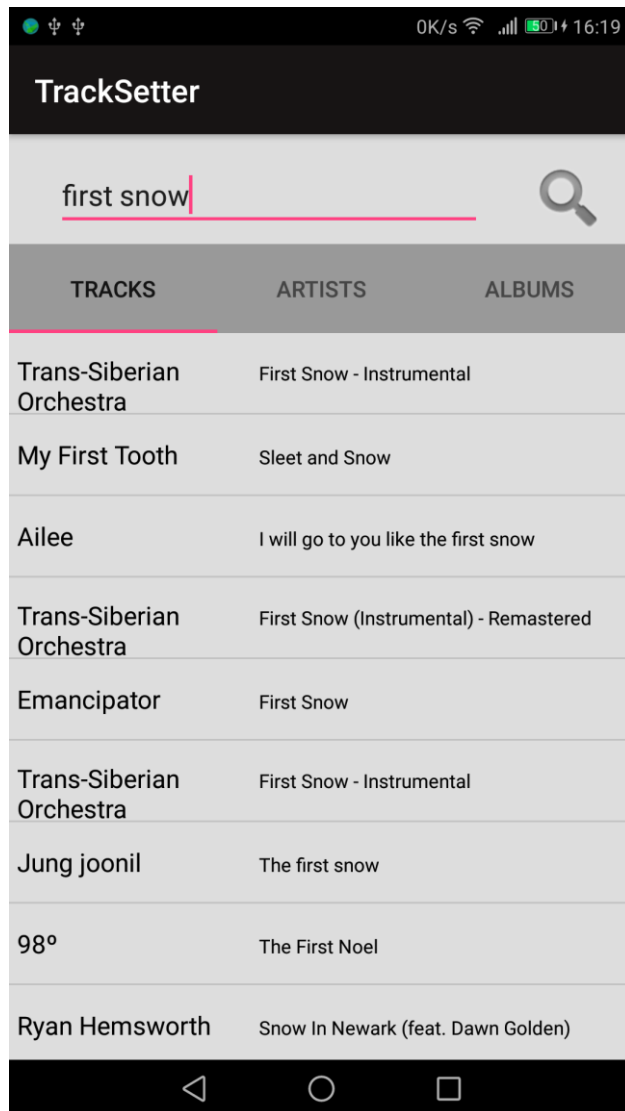
Track was saved.

+



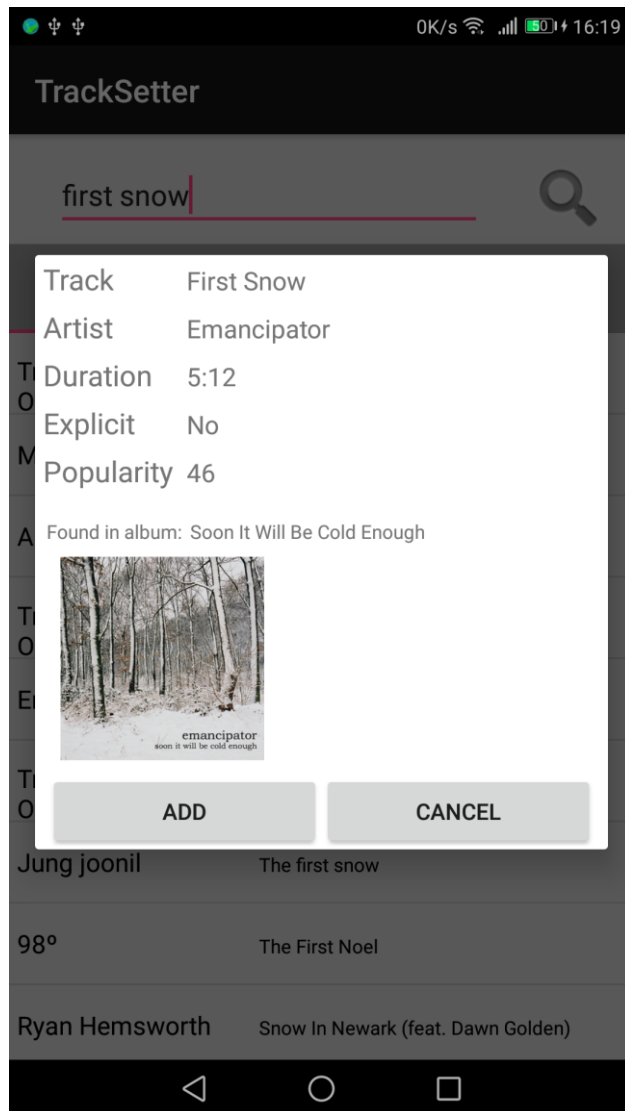
## Screen 2

- Track detail view
- Accessible via onclick
- Only displays currently available data
- Search for missing data online



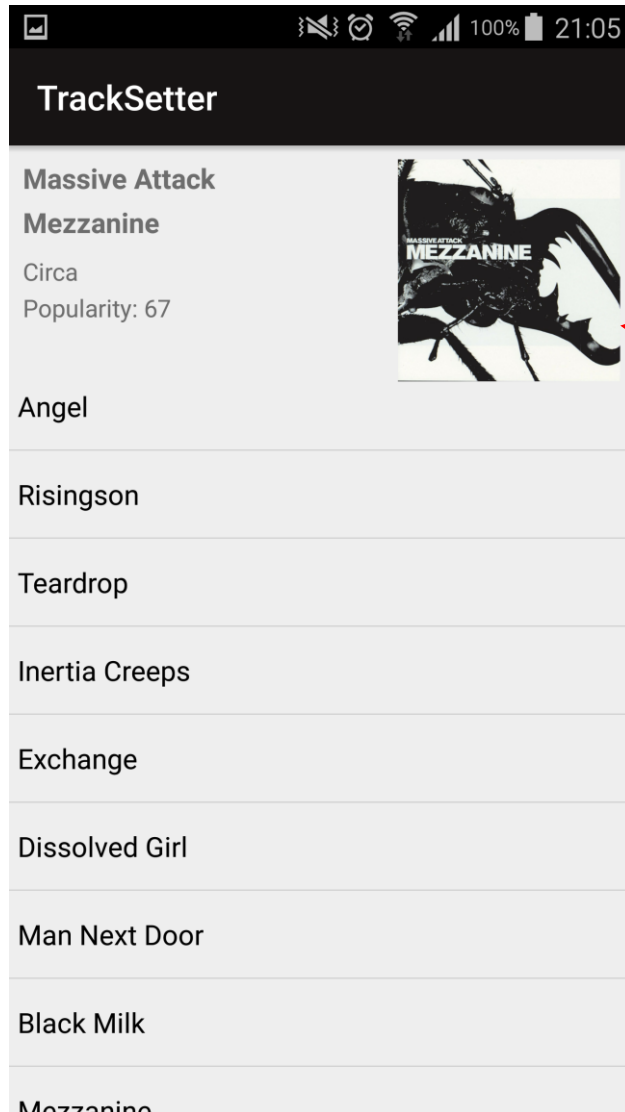
## Screen 3

- Search for tracks, artists, albums via Spotify API
- Click on a result to show its details



## Screen 4

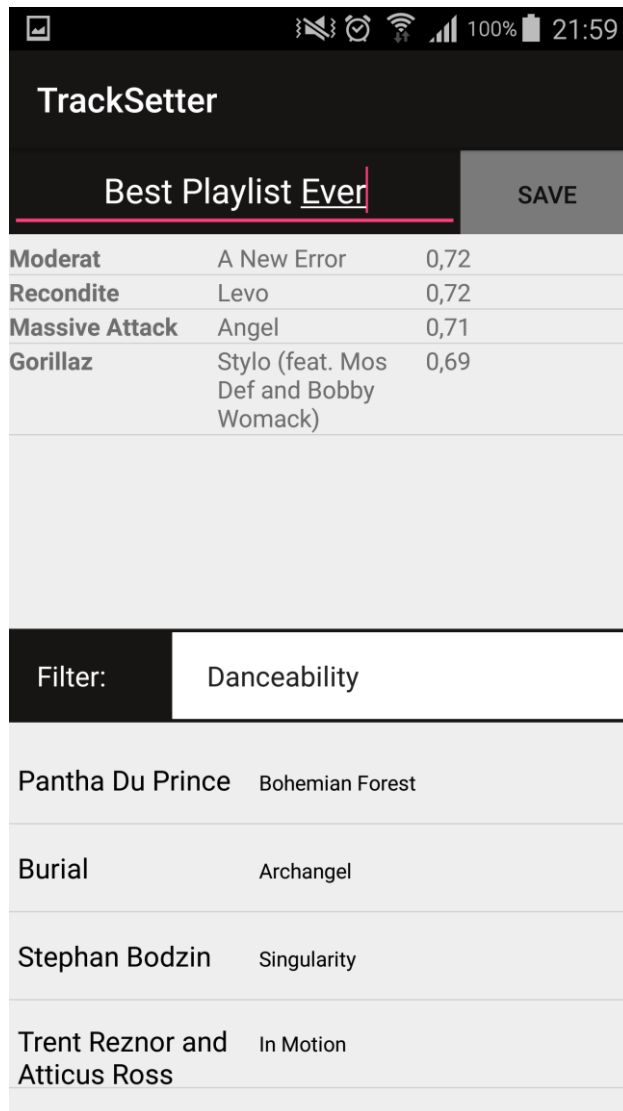
- Search for tracks, artists, albums via Spotify API
- Click on a result to show its details



## Screen 5

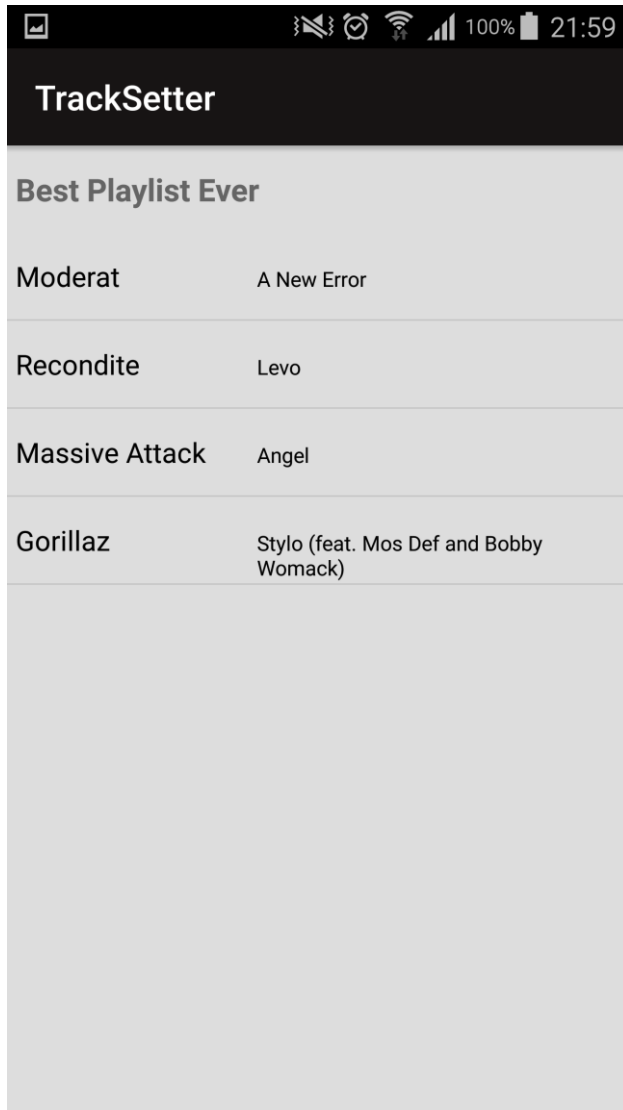
- Ability to add all tracks from an album by clicking the album image (temporary solution)





## Screen 6

- Create a playlist
- Shows all the filter values of the already added songs
- Choose a filter
  - values come from the Spotify API but you can also choose all filter (they are added up internally)
  - first tracks are the best fitting tracks



## Screen 7

- Playlist detail

## Challenges

- **Usability Challenge:**
  - providing color differentiation (lighter to stronger colors) for intuitive selection of more suitable tracks
  - offering a filter to easily narrow down the proposed tracks to the desired main selection criteria
- **Offline Challenge**
  - usable in offline mode with manual insertion of track information

## Adaptation Concepts

- **Offline Usage**

- allow to add tracks manually if not online
- search for missing data when online again -> if artist and track title are the same, add the missing data
- if result is not found or the manually added data was incorrect, the track will be shown with a different color
- clicking on the highlighted track, you are redirected to the normal Track-Search via Spotify (as long as you are online, otherwise the button is not shown)

# Adaptation Concepts

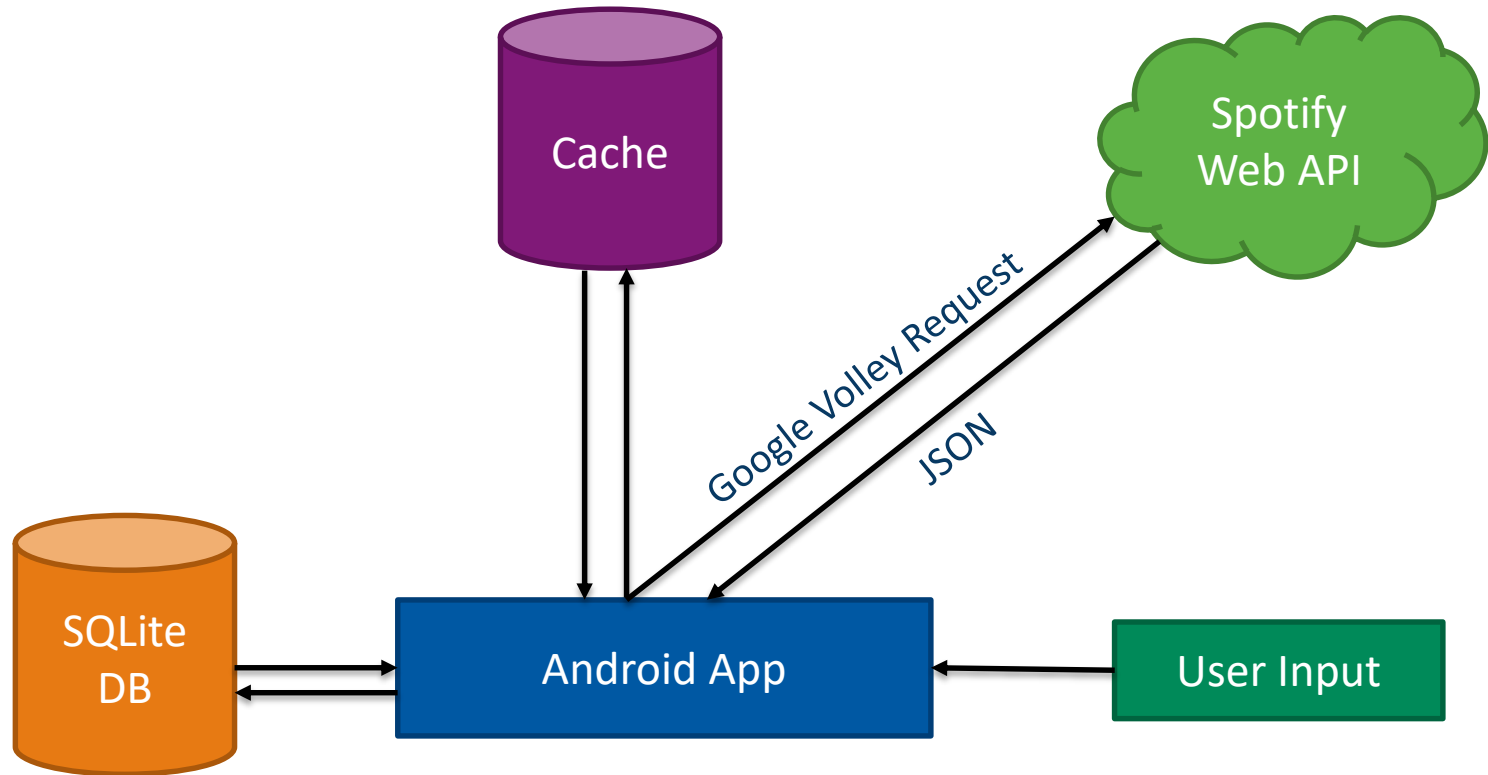
- **Network Awareness**
  - request access token from Spotify on start -> check if online
  - use ConnectivityManager, NetworkInfo, TelephonyManager to determine network type
    - > adapt data fetching
    - (only text when low bandwidth; image prefetching only over WiFi or fast mobile network)

```

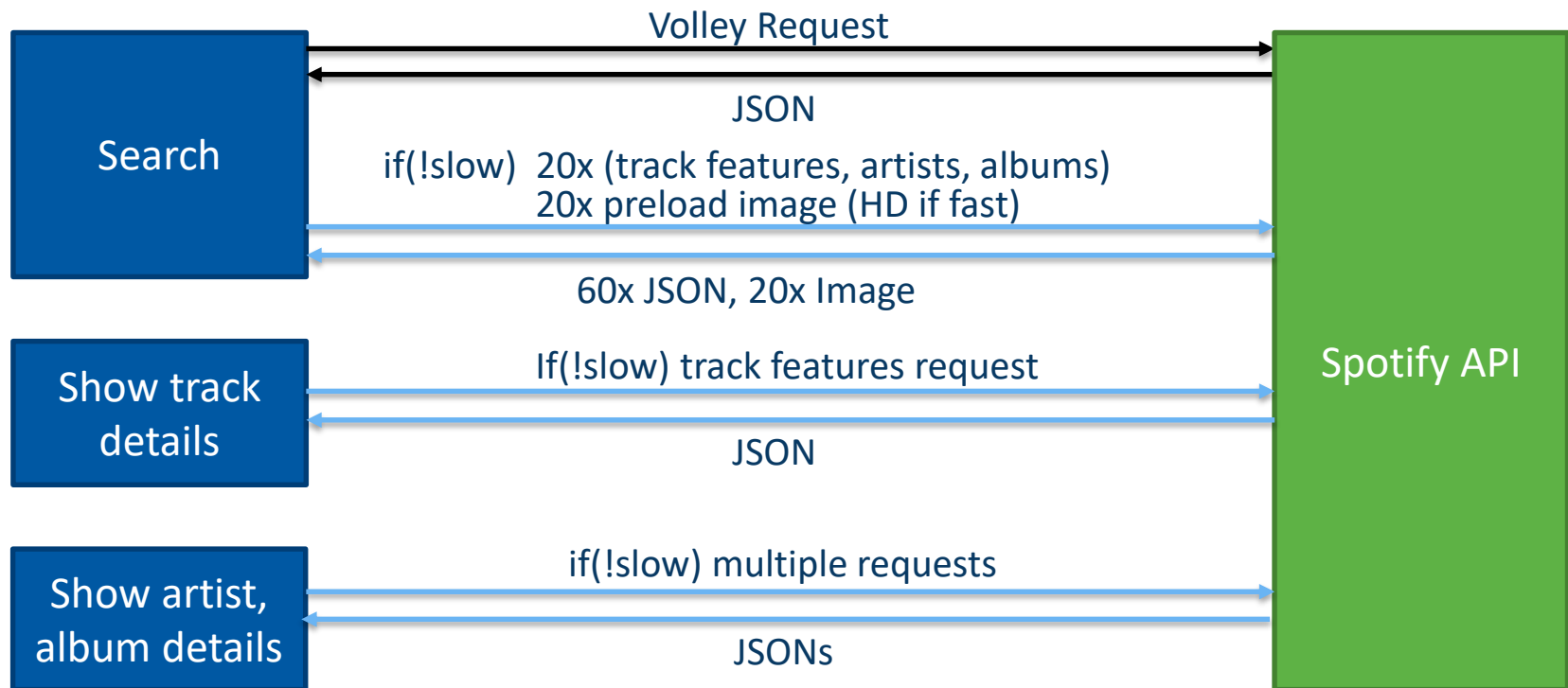
network_subtypes
1  NetworkInfo info = Connectivity.getNetworkInfo(context);
2  if(info.getType() == ConnectivityManager.TYPE_WIFI){
3      // do something
4  } else if(info.getType() == ConnectivityManager.TYPE_MOBILE){
5      // check NetworkInfo subtype
6      if(info.getSubtype() == TelephonyManager.NETWORK_TYPE_GPRS){
7          // Bandwidth between 100 kbps and below
8      } else if(info.getSubtype() == TelephonyManager.NETWORK_TYPE_EDGE){
9          // Bandwidth between 50-100 kbps
10     } else if(info.getSubtype() == TelephonyManager.NETWORK_TYPE_EVDO_0){
11         // Bandwidth between 400-1000 kbps
12     } else if(info.getSubtype() == TelephonyManager.NETWORK_TYPE_EVDO_A){
13         // Bandwidth between 600-1400 kbps
14     }
15
16     // Other list of various subtypes you can check for and their bandwidth
17     // TelephonyManager.NETWORK_TYPE_1xRTT      ~ 50-100 kbps
18     // TelephonyManager.NETWORK_TYPE_CDMA       ~ 14-64 kbps
19     // TelephonyManager.NETWORK_TYPE_HSDPA      ~ 2-14 Mbps
20     // TelephonyManager.NETWORK_TYPE_HSPA       ~ 700-1700 kbps
21     // TelephonyManager.NETWORK_TYPE_HSUPA      ~ 1-23 Mbps
22     // TelephonyManager.NETWORK_TYPE_UMTS      ~ 400-7000 kbps
23     // TelephonyManager.NETWORK_TYPE_UNKNOWN    ~ Unknown
24
25 }

```

# Architecture

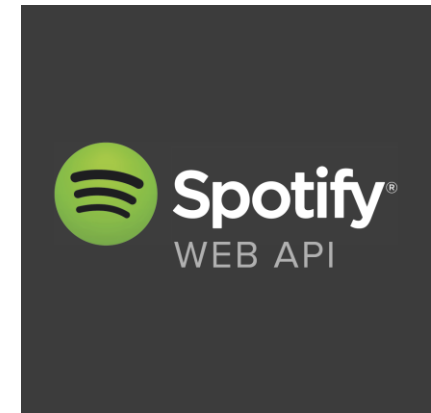
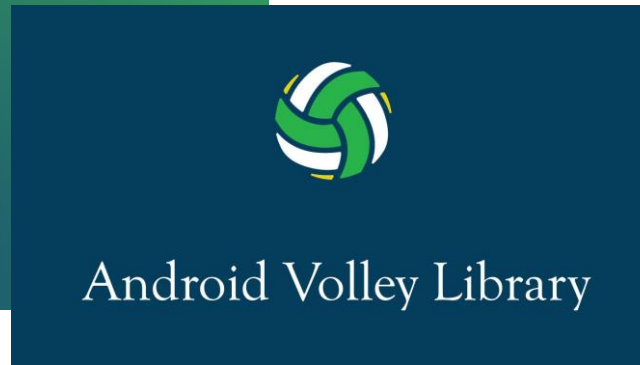


## Detailed Communication Model



slow: < 0.384 (GPRS, EDGE, 1xRTT, GSM, all others)  
 medium: 0.384 – 5 (UMTS, CDMA, EVDO\_0, EVDO\_A, HSDPA)  
 fast: > 5 (WiFi, LTE, HSPA+, EVDO\_B)

# Technologies





## Open Issues

- No listener for network change  
-> recognizable only on action
- Determined network speed to some extent arbitrary
- Color-Differentiation for Filter
- Some unclean code, though some refactoring has already been done

## Lessons Learned

- Difficult to implement app as originally planned
  - unforeseen necessary features get higher priority than originally planned features (e.g. color differentiation for filter)
- Working closely together as a team
- Working with Android
- Not all ideas can be implemented, e.g.:
  - Add EAN-Scan (ZXing Library)
  - Improved search for misspelled added tracks

**Thank you for your attention.**

