The app allows the user to track their movement. The user can start or stop tracking at any time using a switch in the app. Data is stored using Room Ape; queries are executed on the main thread since the amount of data is quite small. The database contains two tables: path and point, with a 1 to many relation. A sticky background service is started the first time the app is used or at boot time. The service and the only activity in the app communicate using broadcast messages. Broadcast messages are of three different types:

* START\_TRACKING\_ACTION: sent from activity to service when the user activates the switch, tells the service to start track user’s movements
* STOP\_TRACKING\_ACTION: sent from activity to service when the user de-activates the switch, tells the service to stop track user’s movements
* UPDATE\_UI: sent from service to activity, tells the activity that something in the data has changed and to update the UI. This message is sent following these events: a new path has been created (=user started tracking), a new point has been added to the path, a path has been closed (=user stopped tracking), the service has been restarted after being killed and tracking has resumed.

When tracking is turned on the service creates a new path and register for location updates that comes every 5 seconds and are at least 10 meters far away from the last one. When an update comes in, a new point is added to the path. When the user turns off tracking, the service unregisters for location updates and set the path as closed. If the service is terminated by the system during tracking, when is started again it resumes tracking using the last available open path (there shouldn’t be more than one since a path is closed only when the user turns off tracking). The service consumes battery only when in tracking mode.

The app architecture follows the MVC pattern and makes no use of databinding.