5.2 A closer look at the technological development process

The roots of StationRadar can be traced back to the preparatory work for the research presented in Caset et al. (forthcoming) (see Chapter 2). As part of this research, two network centrality measures (travel time centrality and transfer centrality) were calculated in R and were afterwards plotted on a web interface using the R packages called ‘Shiny’ and ‘Leaflet’. Later, in the run-up to the research presented in Chapter 3 (Caset et al. 2019), the radar diagrams were added to the web interface, along with the vector and raster layers[[1]](#footnote-1), the data tables and the other informative tab pages that were present in the beta version of the tool (see Figure 26 for an illustration).

The radar diagrams were created using the R package ‘ggplot2’ which is easy to use and allows for flexibility in terms of code modifications. For example, in order to create the radar diagrams, we built on the ‘polar graph’ template and hardcoded the appropriate widths of the diagram ‘slices’[[2]](#footnote-2). However, one of the limitations behind this approach, lies on the fact that the Shiny Server Open Source version, doesn’t allow multiple processes to run in parallel. For example, imagine there are three users querying the website. Instead of simultaneously processing the queries, they are processed one after the other, which may in some cases result in sizeable loading time. Besides that, there are limited possibilities in terms of web design as there are certain website templates that you need to stick to. Although it is still possible to change some templates by tweaking the css and html code, it is a cumbersome process which still does not allow a full customization of the website.

Following the outcomes of the workshops, one of the main usability limitations centered around the lack of tool interactivity. More specifically, the vast majority of participants expressed the need of plotting radar diagrams as a function of their own desired station selections. In order to live up to these expectations, we had to rethink the way in which the radar diagrams were created. Whereas ggplot2 offers many interesting features, it does not allow for the kind of flexibility required. We therefore opted to go for an open-source javascript framework by using ‘Vue.js’, ‘Vuetify’, and the javascript libraries ‘D3.js’ and ‘Highcharts’. We designed the radar diagrams using D3.js and Highcharts by drawing on the ccs-styled ‘mode-pie’[[3]](#footnote-4) (Highcharts and D3.js). The indicator graphs detailed in the previous section were generated drawing on the ‘spline with inverted axes’ template (also using Highcharts and D3.js). We are currently still figuring out how to include the reactive element into the calculations that allows dimension and indicator performance to be calculated on the fly relative to the selected group of stations.

The move to a javascript framework at the same time addressed the reactivity and design limitations mentioned above. In terms of reactivity, a major advantage of using Vue.js is that all the calculations are performed on the side of the user, meaning that the server requirements are not too high. Another advantage is that it is possible to have a fully reactive website, meaning that the content adapts automatically to the user’s device, as well as it opens the door for a better interactivity between the user and the website.

By combining all of these features, we were able to create a more mature and scalable version of the tool: StationRadar 2.0.

1. In order to speed up the loading process, these layers were tiled using the QGIS plug-in QTiles. [↑](#footnote-ref-1)
2. The code for these tailored radar diagrams is provided open-access at https://github.com/FilipeamTeixeira/radardiagram [↑](#footnote-ref-2)
3. In chosing this type of visualization, we were greatly inspired by the Urban Mobility Index website by ‘Here’: https://urbanmobilityindex.here.com/. [↑](#footnote-ref-4)