OPEN DATA PROJECT - SS 14

SHOULD I RUN? also known as SIR

IDEA

Create a mobile application, that tells the user when the next public transport connection comes and if it is possible for him to get to the public transport stop in time. This is done by calculating the route to the stop from his current position. The idea of the design was to stick to the KISS-principle (keep it simple, stupid) with the goal to require the least possible input from the user, to make the app usable even if he is running.

RESOURCES

Public Transport Stops data:
https://offenedaten.de/dataset/vbb-haltestellen/resource/4075a186-463b-400b-b46f-85ebcc87f5f1

VBB-API-Testsystem:
http://demo.hafas.de/bin/pub/vbb-fahrinfo/relaunch2011/extxml.exe/

FRAMEWORKS

jQuery & jQueryUI (JavaScript Handling and simple UI)
Phonegap (for deployment as app)
d3.js (for parsing of data)
Visual Studio (as development environment)

PREPARATION

The Public Transport stops data had to be parsed with the help of d3.js. This enabled us to get the dataset with only the names of the stops, which was important for the automatic completion. During a second round of parsing, we focussed on the station details, assigning according public transport stop number and coordinates to the stop name.
Duration: 4 hours

The next phase was the incorporation to the VBB interface. This first meant to find some kind of documentation, to find a possible query that could be send to the test system (see above). After successful connection to the test system, the data needed to be parsed and answers filtered. Here we needed to analyze the structure of the answer in different situations, to be able to handle the data reliable.
Duration: 10 hours

Developed by: Kaj-Sören Mossdorf, Sona Pecenakova, Viet Trinh, René Vos

IMPLEMENTATION

1. Implementing the automatic completion, so that the user can only enter a part of the stop name and the application will display a list of stop names that include / start with the user's input string.
Duration: 2 hours

2. Write code to handle and display all the stations connections and finding a way to save the data, so it could be retrieved in later calculations
Duration: 4 hours

3. Implement the recording of the users time over a distance of 100 meters. Here we first choose a 10 meter distance, but due to the accuracy of the GPS-Signal, we had to choose a higher distance to get a reliable time. To prevent uneccesary request to the yournavigation.com-Service (see below), we are using the Haversine formula (http://www.movable-type.co.uk/scripts/latlong.html) here, which allows the calculation of distances based on latitude and longitude.
Duration: 3 hours

4. Recording of the time the user needs to overcome one floor, which we planned to do via the altitude-data of the GPS-Sensor of the Smartphone / Tablet. Unfortunately as it turned out, the data provided (if at all - Android-devices don't provide any), was way too inaccurate. So instead we settled on asking the user to run one story and simply stop the time. As for the height of a story, we used a mean value of the ceiling height ending up with an average of 2.75 meters. This will not prove 100 percent accurate every time, but should result in a stable and usable value for the calculations in step 6.
Duration: 4 hours (including misfired GPS-trials)

5. Calculating the distance between the current position of the user and the public transport stop; This is done using the navigation service under http://yournavigation.org/ . This service is based on the free and open data of the OpenStreetMap-Project and offers a web api to calculate complete routes (we only needed the distance though).
Duration: 2 hours

6. Next step was calculating the time needed to get to the stop from the user's current position. This calculation was based on the previously collected and computed data, such as the time needed for the distance (see point 3), the time needed to overcome a story (see point 4), the distance to the stop, the departure time of the tram (which why the storage in step 2 was crucial) and the current time. Displaying this in an according visualization was also part of this stage. Here we are using seven different colors to provide feedback for the user, as seen in the table below.

COLOR CODES - feedback on connection times for the user

black	The connection left up to one minute ago (still in the list, as the user might beam himself to the station and reach the connection after all - also used, to make up for inaccuracies)
purple	The user has a 30 seconds long span between his arrival and the departure
red	The user has a 60 seconds long span between his arrival and the departure
orange	The user has a 120 seconds long span between his arrival and the departure
yellow	The user has a 150 seconds long span between his arrival and the departure
darkgreen	The user has a 180 seconds long span between his arrival and the departure
lime	The user will reach the connection without any trouble at all

Duration: 4 hours

7. The last step was to build the application for Windows Phone. Here the Phonegap-Framework came into play, allowing us, to turn the HTML- & JavaScript-Code into a app, that would run on a smartphone.
Duration: 10 hours (see “problems” below for the reasons)

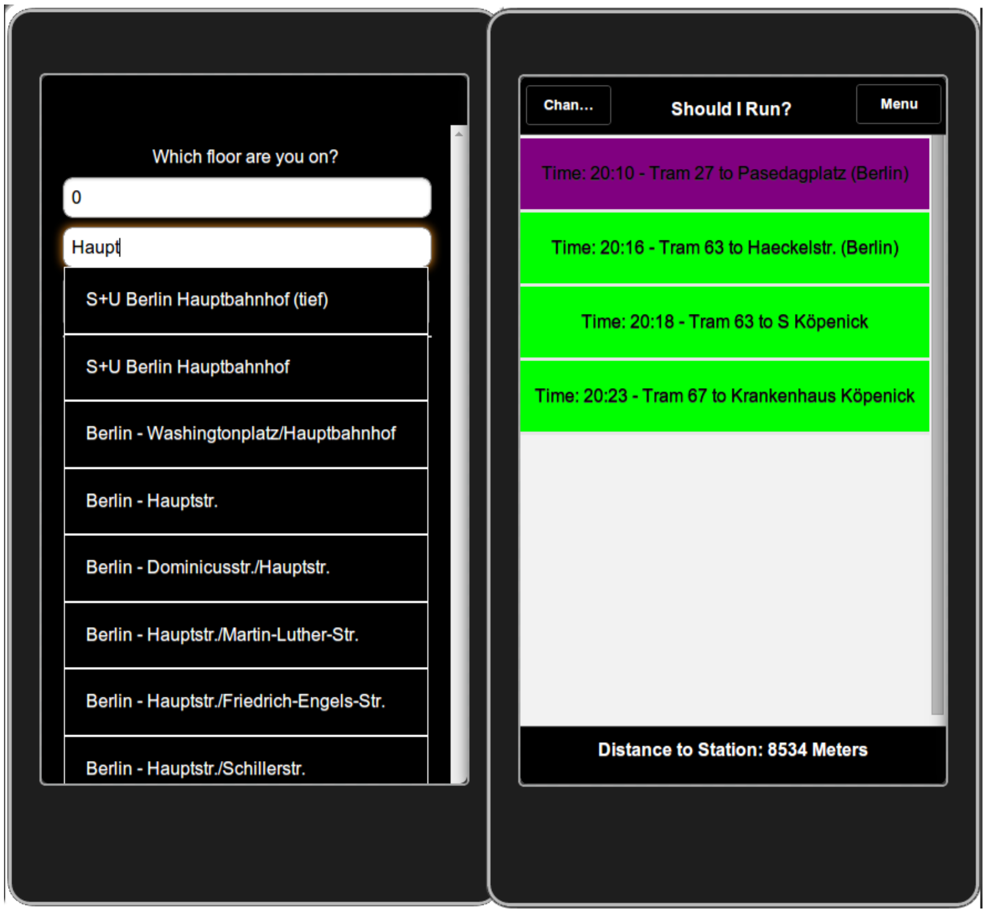
PROBLEMS

- Almost non existing documentation of the VBB-API and the test systems it is running on. This seriously would be a nice project for a next group (or given its complexity - generations to come). What made it worse, was that the connection data was only provided in xml format and not in JSON format as in other cities, this could apparently be easily changed through the VBB, but would probably mean a new configuration for their systems as well.
- Once we had the app running on a smartphone we needed to change the UI and the layout to be easily usable via touch input.
- It also turned out, that the accuracy of the GPS-Chip of smartphones wasn’t nearly as precise as we had hoped. That meant, that we first needed to implement an accuracy filter for the GPS-signal (if the signal was first found, the accuracy was as worse as 90 meters).
- Also GPS-related was our problem with the height of the user - as described in point 4. under implementation.

TESTING THE APP

Due to the used technologies, this part unfortunately is not that easy. The app (which basically is a website) could be accessed via the browser of a computer or smartphone, but a security guideline in the W3C-Standard prevents cross domain queries (which we use, to collect data from the different services). You can however install Chrome (with the Ripple-Emulator-Addon) and start the browser with the --disable-web-security Option.

This problem with the regulation is circumvented if the app is used as an app, since it is running in a different environment. If you have access to a Windows Phone, we will happily provide you a copy of this app. We also build an Android-Version, which can be installed on any Android device, please remember though, that the experience can vary from device to device. Especially since the app was developed for Windows Phone. The source code of the project as well as the two Apps can be found on GitHub: https://github.com/stiller-leser/ShouldIRun .


Which floor are you on?
0
Haupt
S+U Berlin Hauptbahnhof (tief)
S+U Berlin Hauptbahnhof
Berlin - Washingtonplatz/Hauptbahnhof
Berlin - Hauptstr.
Berlin - Dominicusstr./Hauptstr.
Berlin - Hauptstr./Martin-Luther-Str.
Berlin - Hauptstr./Friedrich-Engels-Str.
Berlin - Hauptstr./Schillerstr.
Chan...
Should I Run?
Menu
Time: 20:10 - Tram 27 to Pasedagplatz (Berlin)
Time: 20:16 - Tram 63 to Haeckelstr. (Berlin)
Time: 20:18 - Tram 63 to S Köpenick
Time: 20:23 - Tram 67 to Krankenhaus Köpenick
Distance to Station: 8534 Meters