Waiting for the Bus – Pebble Watchapp

DOCUMENTATION:

The source code of the app is inside the file app.js. The app is structured as follows:

Macintosh:Users:davidemerzi:Dropbox:University:Multimedia Networking:progetto:app_schema.pdf

The menu of the app is divided into three sections: *Stops Nearby*, *SmartStops* and *Settings*.

STOPS NEARBY

The Stops Nearby section is defined by the function *createStopsNearbySections*. It provides with all the stops and the first two arriving bus per stop in a given range with respect to the user’s position. The algorithm behind this section is the following:

1. with an ajax request the app finds the stop\_id of all the stops around you in a given range
2. for each one of these stops the app gets the times of the buses with another ajax request, but filters only the buses that pass after the current time and saves only the first two
3. the app shows the results on the Pebble screen.

The screen is organized as follows:

Macintosh HD:Users:DaveS:Desktop:layout_pebble.pdf

Each section represents a bus stop; in the title the user can see the distance between your coordinates and the ones of the bus stop plus the name of the stop. Stops are ordered by proximity. Each section has a maximum of two items that represent the first two buses that are arriving, with their estimated time of arrival (ETA) and the trip destination, which is shown one line below. In this menu if the select button is pressed bus’s details are shown, whereas if the long-select button is pressed a check-in is performed.

SMART STOPS

The *SmartStops* section is defined inside the function *createSmartMenuSection*s and it is the learning part of the application. This section shows results only if the storage is not empty. In fact the application saves in the storage not only the checked-in bus and related bus stop, but also the previous and the following bus stop. The algorithm behind this learning part is the following:

1. with an ajax request the app finds the stop\_id of all the stops around the user in a given range
2. it filters only the ones that are saved in the storage (current, previous and following bus stop)
3. with a second ajax call the app finds the times of the next buses for all the filtered bus stops
4. it filters only the next buses whose route\_id (univocal number for identifying a bus line) is present in the storage
5. it prints the results on the screen like the Stops Nearby section, ordered by proximity
6. it also catches the main exceptions

The learning process is based on the user’s decision, in fact the user can decide with the initial check-in what the application can learn and what not. Furthermore SmartStops menu shows the buses you take more often based on your GPS coordinates, because it checks the stops in your range. Moreover the fact that it saves also the stop before and the stop after the one checked allows iteration after iteration to expand the range of visible stops, creating a path of habitual stops.

Given your GPS position the app will be able to find a stop and inform the user if this stop is related to the main checked-in one. If more than one stop is shown, the app will be able to “weigh” them according to their distance with respect to the user’s GPS coordinates and send the user to the nearest one.

After checking-in more and more bus stops of the same bus line, a path will be progressively built.

SETTINGS

The Settings section consists of three options:

1. *Clear storage*: it cleans the app storage where data about check-in are saved.
2. *Set range*, i.e. the radius in which the app will search for bus stops (from 100 meters to 1 Km)
3. *List check-in* that shows info on the checked buses and permits to delete them singularly.

Other used functions are:

* *updateMenuSections* for loading sections on the screen
* *initApp* that retrieves the coordinates from the phone GPS and saves them
* *distance* that calculates the distance between two coordinates