TraceMySteps - Visualizing And Analyzing Personal Geolocation Data

Pedro João Cordeiro Francisco

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Daniel Jorge Viegas Gonçalves

Examination Committee

Chairperson: Nuno João Neves Mamede

Supervisor: Daniel Jorge Viegas Gonçalves

Member of the Committee: João Manuel Brisson Lopes

October 2016

For my family

Acknowledgments

This work would not have been achievable without the support of my family, professors and friends.
To my family, thank you all for giving me the determination, grit and inspiration to achieve my objectives.
I am principally indebted to my parents and brother, who supported me emotionally and financially.
Also, I would like to take this opportunity to express my deep gratitude and admiration towards my supervisor, Daniel Jorge Viegas Gonçalves for his commendable teaching, monitoring, guidance and incentive throughout the progress of this thesis.
To all my friends, and especially my closest ones, thank you for the advice, help, support, insights and understanding through all these days, since my thesis started.

Resumo

Com o aumento da utilização de dispositivos de localização GPS, como os smartphones, são produzidas grandes quantidades de dados. As pessoas começam a utilizar estes volumes de informação de forma a compreender melhor a sua mobilidade - lugares que visitaram, distâncias percorridas, caminhos tomados, etc. No entanto, é difícil analisar este tipo de informação e observar padrões que sejam pessoalmente relevantes. Nesta dissertação apresentamos o nosso trabalho, que permite aos utilizadores entender a sua informação espácio-temporal, com especial destaque para a vertente da semântica pessoal dessa informação. Para tal acontecer, a nossa interface apresenta um conjunto de visualizações que os utilizadores podem personalizar e filtrar, de modo a poderem meticulosamente analisar os seus dados (que abrangem intervalos temporais desde dias a anos). Por fim, a realizámos uma avaliação com utilizadores, que mostrou que as pessoas conseguiam usar e compreender o sistema no seu todo, provando assim que os objectivos iniciais foram alcançados.

Palavras-Chave: semântica pessoal, informação espácio-temporal, visualização escalável, dados de movimento, visualização de informação.

Abstract

As the use of GPS tracking devices such as smartphones keeps growing, large amounts of data are produced. People are starting to use these volumes of data as means to gain insights about their mobility - places they have been to, distances travelled, routes taken, etc. It is hard to analyze this type of data and observe patterns with personal relevance. In this dissertation we present our crafted solution that allows people to understand their spatio-temporal information, with focus on the personal side of that information. In order for that to happen, our interface presents a number of visualizations that users can customize and filter, so that they can analyze their data (spanning from days to years) thoroughly. Finally, an evaluation with users showed that people could use and understand the system in its whole, thus proving the initial objectives were achieved.

Keywords: personal semantics, spatiotemporal information, scalable visualization, movement data, data visualization.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, with the exponential growth in the usage of GPS tracking devices such as smartphones, smartwatches and tablets, massive amounts of data derived from that usage are generated. That data is usually associated with mobility, behaviors and patterns of animals and people. This situation leads to an increasing number of people carefully exploring the information they have available in order to plan their life and activities. For instance, one can consult weather reports to see if the day is good for a jogging session, or consult if the buses will arrive to the stop on time.

Lifelogging can be described as a practice in which people track their personal data. This data is generated by the person's physiological and social activities such as running, eating, traveling and working. As we will explore, this is an area that has started to gain interest from many people in the last couple of years, with an increase in blogs, websites and applications that offer various solutions and insights about this subject. Most of the solutions implemented by users are very specific and individual, becoming valid only for a certain context.

Concerning this GPS tracking data, there are tools that allow users to manipulate, analyze and manage it - Geographic Information Systems (GIS). Features included in these tools range from interactive queries to map visualizations of data and various modeling types.

Most approaches to mobility tracking do not cope with the personal aspect of data. Exceptions to this rule include lifelogging applications and a few social networks that consider spatio-temporal locations. Besides this one, other problems arise when discussing this matter of tracking mobility data: scalability, ethics and personal privacy, less focus on time and users, and the need to develop better evaluation standards (Andrienko 2011). Despite these problems, there has been an evolution in the topic of preprocessing the collected tracking data: many approaches use clustering algorithms, association and track simplification in order to erase redundant and irrelevant information. Nonetheless, this demonstrates the need for a universal

solution that all types of users can use and further enhance their lives. The outline here is that collecting spatio-temporal data has become a frequent and simple process, but analyzing it and deriving information with personal significance is still a blurred area. The need for expert knowledge in order to manipulate GPS tracking data also presents a challenge. All of this is because the study and design of visualizations related to lifelogging is not very evolved, with very simple approaches. So our aim is to design and create an application for users to thoroughly understand and analyze their spatio-temporal information, with focus on the personal aspects of the data regarding travels, stays and places the user visited. This way we can add a solution to this lacking field of lifelogging apps and further push its growth.

To achieve this goal, we created a system capable of processing and storing a user's mobility data - it handles previously recorded GPS tracks - together with visualizations that include and present the personal semantic notes associated with those routes or locations. This way users can be aware of the places they have been, time spent there, frequency of visits, routes taken, etc.

First we needed to know what factors, to be observed in our solution, were important for potential users. With this survey we uncovered the main requirements. Next, we had to understand which type of user mobility data we needed, and how we could process and simplify it. Then we planned the architecture of our system and designed back-end, containing the database, its tables and attributes, and the scripts that processed the data and populated the database. Then, with the database running, we studied how to create a web server and a REST interface so as to connect to the database and have dedicated methods that query and format the information for each visualization. We also analyze the back-end scalability.

On the front-end, each visualization is connected to the REST interface and its specific methods through the use of a URL provided by it. A service manages requests, acts as a broker between the front-end and the web server, and also acts as a cache. The visualizations are based on maps and all sorts of graphs, and are connected between them to observe patterns and relate information. A timeline slider allows users to select the date interval in which they want to analyze the information. In the same way we did for the back-end, we also analyze the front-end scalability.

Finally, we proceeded to execute user tests. Besides the validation of our objective, the focus was to get the global feeling of the solution, to understand user satisfaction and to evaluate the way users deal with the personal semantics of the data, at the same time as they customize and interact with the various visualizations, because it is the critical part of our solution.

All the development and testing procedures/results are documented in the following pages, with more detail and deeper investigation about our architecture and technology choices. We also approach how the back-end, front-end and overall solution morphed and evolved throughout the design and development lifespan.

1.1 Objectives

In order to tackle the above mentioned problems, we need to define a goal that better summarizes our work. Thus, our prime objective is to:

Design a visualization that allows users to understand and analyze their spatio-temporal information, with focus on personal semantics.

The envisaged solution needs to primarily address the problems of scalability and personal semantics. We will also need to take into consideration that we will be using the dataset already preprocessed. From this main goal, three sub goals derive, which we will describe below:

One will be to **understand which factors are relevant for the users**. In other words, what will our solution show. For instance, we must know if the user gives more attention to routes or places, etc. It is essential to know what do the visualizations need to present in order for them to be successful and produce an evolution in the field of personal semantics.

We will also need to **determine the proper information visualization techniques**. Be it a timeline, map, graph, heatmap and so on. If the chosen techniques have flaws in terms of scalability, color picking, text, etc., the user will not be able to analyze the information correctly and will tend to not use the visualization.

Lastly, we will **develop the system and its visualizations**, integrating with an already existent one that deals with visual queries of the information.

1.2 Contributions

Here we specify the main contributions of our work:

- Web-application for analyzing the data: The developed front-end that contains a number of customizable visualizations for users to analyze their geolocation data. It is available on GitHub [1]
- Back-end for supplying the web-app: The developed back-end parses user movement data and personal semantic information, and populates a database. Then, a routing interface connects both ends. It is also available on GitHub [2]

[1] `https://github.com/PedroF20/angular-traceMySteps`

[2] `https://github.com/PedroF20/traceMySteps-backend`

1.3 Structure

In the next chapter we will discuss various published works concerning the topic of visualization of geospatial data (personal or other type). This topic is therefore divided in four subsections: the **Representation using tracks and timeline**; the **Representation using heatmaps**; the **Representation using space-time cube** and the **Representations that use customization panels and menus**. On Chapter 3 we will scrutinize the issue of data collection and discuss some ways to bypass or solve them. On Chapter 4 we describe the design and implementation of our back-end. Chapter 5 presents all the development stages of our front-end, as well as the integration with the said back-end. On Chapter 6 we display the evaluations made to our solution - namely user tests. Finally, Chapter 7 wraps up our work by summarizing our methodology, the attained results, and by proposing some guidelines for future work/research.

Chapter 2

Related Work

In this section we will explore existing works in the field of visual analytics, more specifically regarding the visualization of geospatial data (personal or other type). This topic is divided in four subsections: the **Representation using tracks and timeline** where we will scrutiny works that present data through the use of GPS tracks on map and through timeline to inform the user of its activities on a given temporal space; the **Representation using heatmaps** to show the frequency that a user passes by a certain location, finding patterns relevant to the user; and the **Representation using space-time cube** that enables the spatio-temporal data to be presented in a 3D style. Many representations use customization panels and menus, where data can be filtered and customized in a more classical approach. And finally we will have a section for summarizing and discussing the works, with focus on the visualization of personal mobility data.

2.1 Representations using tracks and timeline

These works provide visualizations where the GPS tracks are represented on a map. They also use timelines, although the design and features of the timelines vary from work to work.

2.1.1 Smart Reality Testbed: visualization of smartphone-based lifelogging data

This work (Jeon 2014) aims to collect data about smartphone usage patterns associated with apps such as Facebook, Twitter or Youtube. With the help of the chronological ordering, users can make some insights about details of their activities and with the help of the visualization tool they can recall those daily activities. The representation is based on the use of circles, where the size of the radius of the circle in the map is directly related to application usage duration.

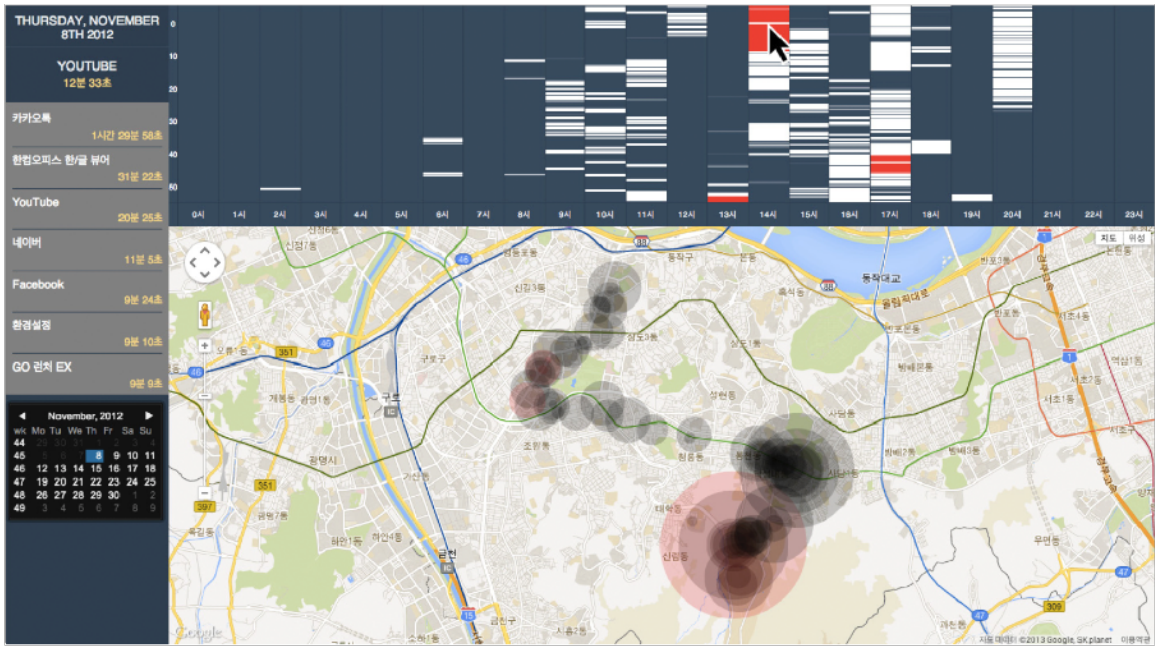


Fig.1: Visual interface of Smart Reality Testbed

After the logs (GPS, SMS, etc.) are extracted and aggregated, a clustering algorithm is applied in order to erase meaningless points and overlapping circles, so that the scattered circles are transformed into a few big circles.

The visualization tool consists of two more parts: an information panel and a timeline view. The panel shows the date, information about the focused app, a list of apps used on that day, the total use duration of the applications and a calendar control where the user can see a detailed log information. As seen on figure 1, the timeline view on top shows stacked white blocks, where each block encodes a use event of the phone. The corresponding circles on the map turn their color from black to red if the mouse hovers on a white block in the timeline.

One of the advantages of this tool is the high detail that is presented to the user. Also, the timeline is very useful. The drawbacks were the heavy CPU-work delegated to the client side and a scalability problem caused by massive app usage, showing too much circles in the map which creates difficulties in observing it.

2.1.2 Patterns of Life

This project (Attfield 2014) is an answer to a challenge proposed by VAST (2014). The challenge is about identifying suspects and possible victims and detecting underlying circumstances of an abduction. The scenario is set up on the fictitious island of Kronos (Abila is the capital) where the affected company GasTech is located. The available data sources are GPS tracks of company cars, a table

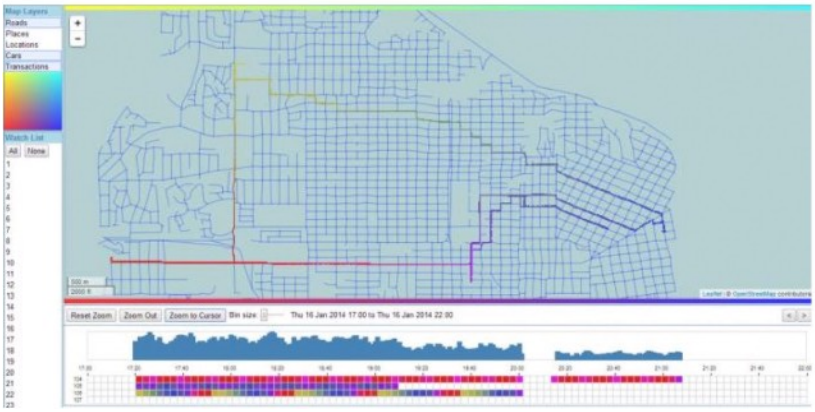


Fig.2: PoL Atlas showing vehicle movement

containing the assignments of cars to certain people, credit-card data and loyalty-card data of employees and other persons, and a simplified map of Abila. This challenge led to other solutions described in this survey.

The first component, PoL Atlas, is shown on figure 2. It provides the visualization of vehicle tracking GPS data by means of an interactive map on top and a timeline on the bottom. Users can select the vehicle they see. The timeline consists of a grid in which rows correspond to vehicles and columns to time interval. Users can zoom in to change the time gap and the resolution of the view. Finally, color patterns on the timeline allow to see repeating journeys.

The second component, PoL Classifier, helps users to infer and label locations by segmenting GPS data into trips and visually plotting successive individual trips.

The third component, PoL Location Timeline, lets the user see time spent at named locations with vehicles represented as horizontal bars and time read from left to right. A different color is used to represent different locations (coffee shop, home, etc.).

The advantage of this tool is the great level of detail of both timelines. The drawbacks are the uncertainty in data, a scalability problem regarding the labeling (a big number of labels that can clutter the map) and the color scheme in the timeline becomes confusing with too many elements.

2.1.3 Dodeca-Rings Map

Dodeca-Rings (Guo 2014) is one of the answers for the VAST challenge. This system was created in order to analyze geo-temporal traffic problems, using data sets that included two weeks of vehicle GPS tracking data, card transaction data associated with the drivers their assignments. It uses dodecagons as an approach to integrate and visualize space, time and objects.

One dodecagon on the map visualizes one car’s stays and activities in different locations in one day and several dodecagons nested together correspond to that

amount of cars. The user can then zoom in on the dodecagon and observe more carefully the data.

Each dodecagon consists of the parking location in its center, numeric ID's and color logos (for an office, restaurant, etc.). For instance, an orange thick segment indicates periods of stay. Finally, as it is a twelve-sided polygon, each side corresponds to an interval of two hours.

Fig.3: Dodeca-Rings map and timeline

The system has also two modes: one shows multiple cars' activities in one day and the other shows one cars' activities on multiple days. It also includes a Social Relationship Matrix that shows all the social connections between vehicle drivers and an activity temporal chart shown on the right side of figure 3 that shows car activity during each day.

The advantages of this system lie on the fact that the dodecagons nested together create concentric rings, helping the user to see patterns and the two visualization modes are very detailed. The drawbacks are related to scalability (too many vehicles clutter the map with dodecagons - we can see an example on figure 4) and time taken to understand the visualization method.

2.1.4 Safeguarding Abila

This work (Saraf 2014) also relates to the VAST challenge. It collects, registers and analyzes geo-spatial data, temporal and financial data (card transactions).

The system is based on the concept of Point of Interest (POI): a specific point location that someone may find useful or interesting (restaurant, shopping center, etc.). These POIs are then represented on map, associated with user locations and card transactions in order to group customers.

A second interface provides two more views: POI distribution and frequency over time. POI distribution over time, represented on figure 4, is a scatter plot that acts as

a simple timeline and displays all the POIs where a particular user was present over the 2 week duration. This helps to identify home and work POIs for each user.

POI frequency over time displays the total number of users present at a particular POI during a 24 hour window, helping the characterization of POIs as home, recreational or work locations.

A third interface allows queries and visualization of spatio-temporal GPS data. The analyst can observe a sequence of locations visited by employees. To observe this on map, the analyst has to choose the date, employee, time and POIs by means of a query.

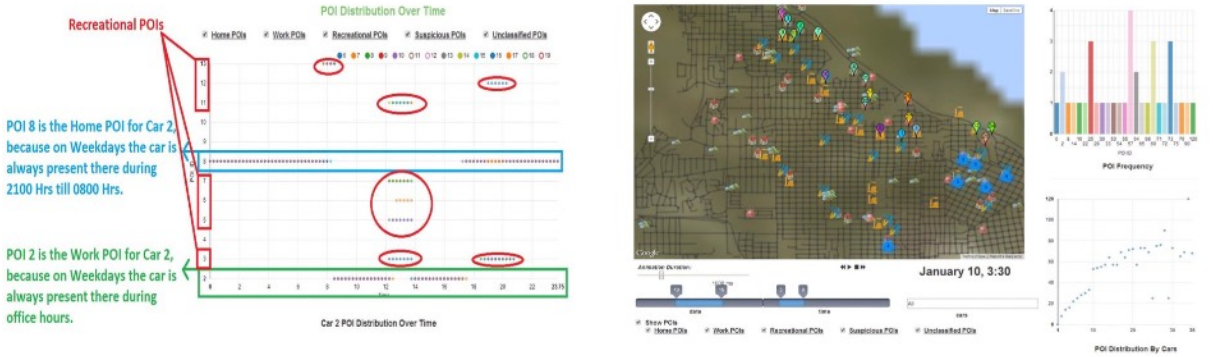


Fig.4: POI Distribution over Time Plots and map interface

The advantage relies on the easiness of finding employee patterns while using the time plots (both distribution and frequency). The drawbacks are the lack of detail and information on the map interface, as well as scalability problems regarding the high number of vehicles or employees on a POI, cluttering the map view.

2.1.5 MovementFinder

MovementFinder (Chen 2014) is a multi-filter visual analytics system that supports data investigation from several aspects: location, time, people and events. Together with maps, transaction records and GPS tracks, it provides many views and filters for that information. It is also one of the answers for the VAST challenge and is based on the existence of Points of Interest (POIs). They are used to identify places like home, coffee shops,work, etc. The application has five linked views (figure 5) that act as a visualization and an interactive filter:

A map view that shows the information by using lines, polygons and colors that represent tracks, POIs and POI categories, respectively. Users can apply spatial filters on the map; A timeline view that shows the temporal distribution of GPS records. Users can apply temporal filters to select tracks; An entity view that shows a name list of the employees. Users can select people on the list; An event timeline

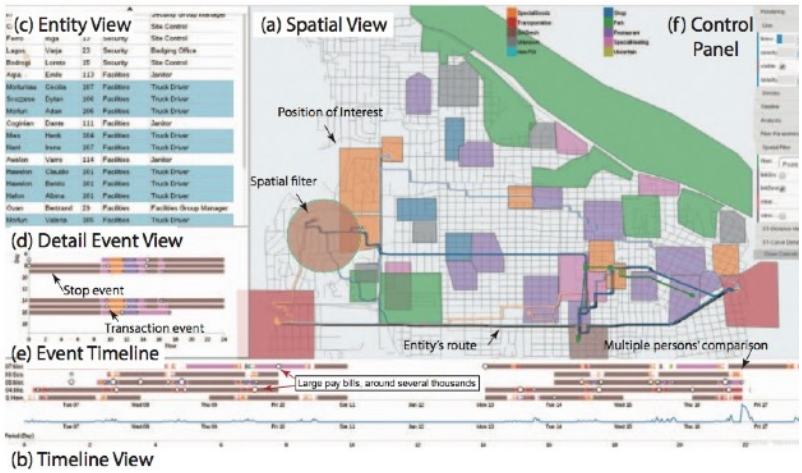


Fig.5: MovementFinder interface

that shows the event subsequence in a selected time range for multiple employees; A detail event view that shows the whole event sequence of one employee, where each row represents one day and colored bars represent time spans of events, with the colors associated to POIs. A circle represents transactions. There is also a control panel, used for parameter adjusting.
The advantage is the great level of detail that the five types of views provide, making it very easy to find spatio-temporal patterns. The drawback is once again related to scalability, because a larger scale of data (more tracks, more employees, etc.) clutters the map and difficults visualization.

2.1.6 Visual Analytics For Detecting Behavior Patterns In Geo-Temporal Data

This project (Hundt 2014) is an answer to the VAST challenge. The authors found several limitations in the challenge such as high interaction costs, low reproducibility, etc. They overcame these difficulties by designing a tool to facilitate interactive exploration of the data.

The tool represented on figure 6 consists of a map view in the center of the window that shows the city and parks. Users can filter by person in order for stops and GPS data to be presented accordingly. It is possible to look at a certain point in time, select the time span that is shown and jump through data in predefined time steps.

Furthermore, there is a line-chart that shows the aggregation of movements of all cars, pre-defined sorting of the person table to help exploration and a heat-map to show the most frequent stops of vehicles and their drivers. The state of the system is always visible by means of highlighting persons, position of the time-slider, and labels showing information (e.g date and timespan).

The advantages here are the existence of the heat-map that complements the map and timeline, and the low complexity of filtering. On the other hand it is difficult to access detailed information.

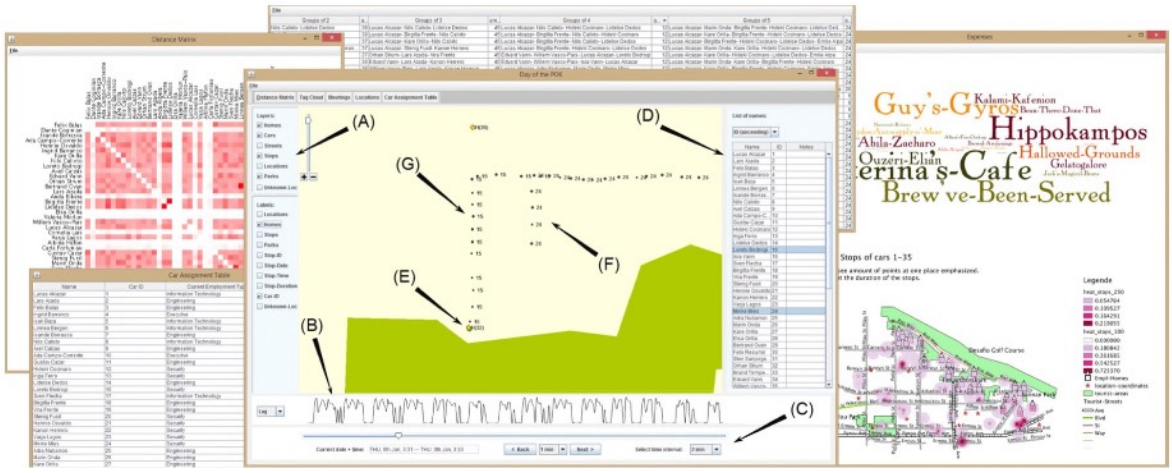


Fig.6: Overview of the tool

2.1.7 Visual analytics of GPS tracks: From location to place to behavior

Another project (Wood 2014) that is a solution proposed for the VAST challenge, it consists of different visualizations of data. This data is based on three types: geospatial, temporal and attributes. These types can be represented by variables such as day of the year, hour of the day, easting, etc. The challenge is to map these data variables to the corresponding visual variables.

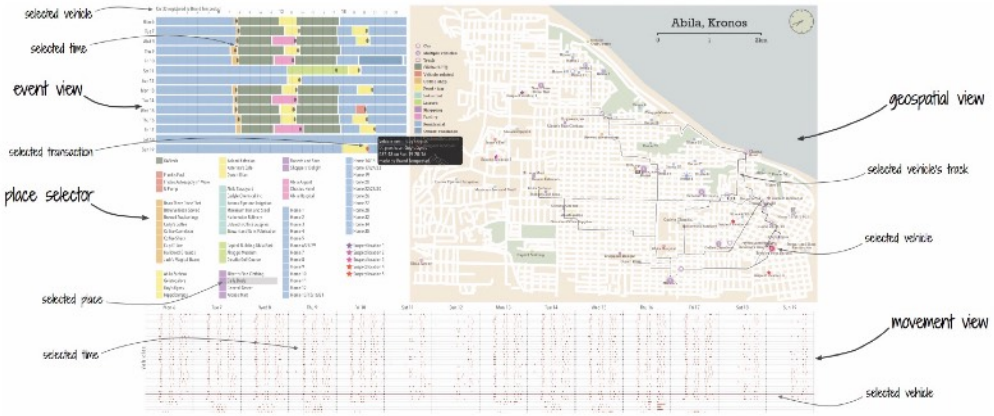


Fig.7: Design of the system

The geospatial view maps GPS coordinates to the graphical space in a mapping style. This particular mapping style allows for tracks to be shown as lines or animated movement of vehicle icons. It also supports spatial comparisons. The moment view in

the bottom of figure 7 shows periods of movement for all vehicles and acts as a selector. Coordinated with the geospatial view, it allows spatio-temporal comparison.

The event view uses a calendar-type layout, colored by place type. This is possible after place types and locations have been identified and associated with proximity detection. It allows common routines of individuals to be easily identified.

The co-location view uses a layout similar to the temporal view but colors accordingly to stationary location at known places, showing if, where and when multiple vehicles met at the same location.

The pro of the system is the detail and design of all the views available. The con is related to scalability, because a high quantity of data can obstruct the movement and geospatial view.

2.1.8 Visits

The purpose of Visits (Thudt 2013) is to use data as means of reflecting how people recall their trips. This is based on the fact that our mind captures those trips as a narrative sequence of events.

That being said, Visits collects spatio-temporal data and creates a map-timeline (figure 8) to display location histories, similarly to how people remember their trips: a sequence of places, in which there are more and less memorable places.

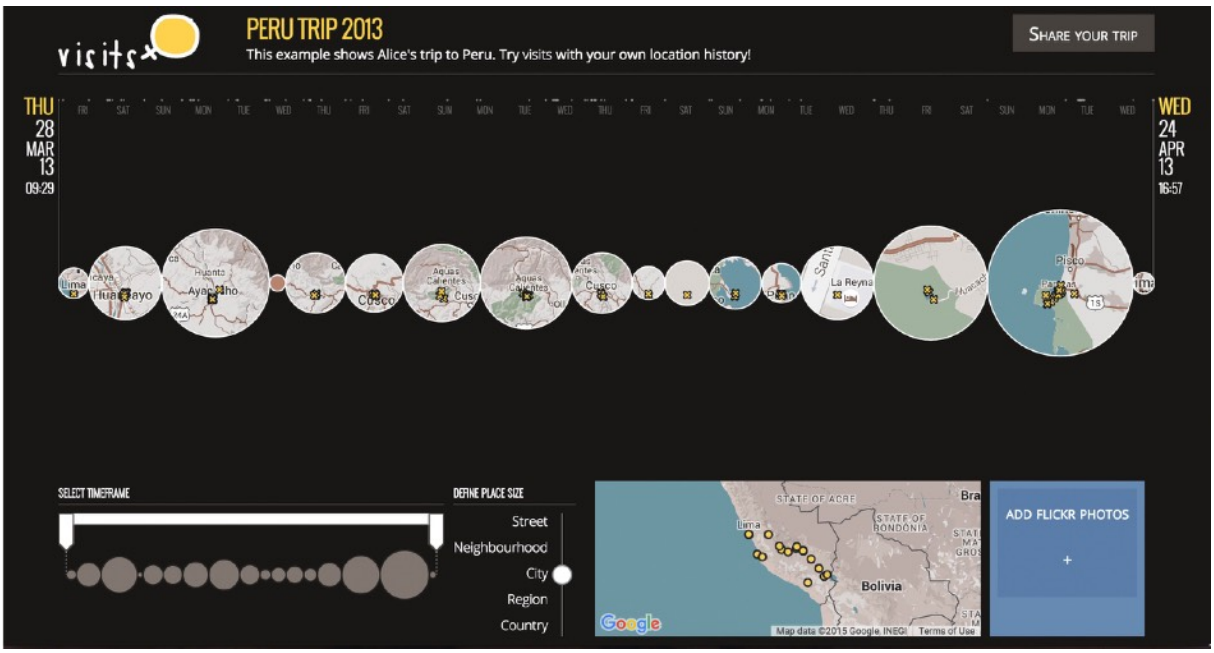


Fig.8: Visits' map-timeline interface

This concept of map-timeline is a hybrid between timelines and maps where each map segment has its size determined by the duration on the timeline, while showing a certain section of the map. The chronological order is from left to right. The user can zoom on specific sections of the timeline in order to highlight the stays and places visited during that time span. It is possible to determine the distance threshold and control the frequency of location measurements.

The biggest advantage of Visits is the way the information is represented and chronologically ordered, making it very easy for the user to locate an event in time. The biggest disadvantage is its scalability: if there is a large number of events shown in the map-timeline, it will become very big, rendering some circles too small to be identified.

2.1.9 The Variable Tree: I Know Where You Were Last Summer

Fig.9: Visualization of journeys

This is a website created by James Siddle [3]. The author in this specific blog entry (Siddle 2014) uses a publicly available London Transport [4] dataset that contains records of bike journeys for London's bicycle hire scheme. This dataset was used to track the movements of individual cyclists across London, for a six month period between 2012 and 2013.

Cyclists' journeys are cataloged on map, where purple lines indicate journeys made in both directions and orange lines with arrows, such as the ones on figure 9, indicate one-way journeys. The size of the line shows the number of times a certain journey was made.

There is also a circle where its size represents the total of different destinations that the cyclist has travelled to and from that station. A bigger line implies a frequent journey. By activating the filters, a slider is shown, where the user can see journeys on a certain time range.

The biggest advantages are the design of the arrows that show the direction of movement, the size of the circle and lines that show the importance of a certain station and the filters used to get more detail. The drawback is associated with

[3] http://vartree.blogspot.co.uk/2014/04/i-know-where-you-were-last-summer.html

4 https://tfl.gov.uk/info-for/open-data-users/our-feeds

scalability: if there are too many circles represented, the map becomes cluttered and confusing to analyze.

2.1.10 Visual analysis of movement behavior using web data

This tool (Krueger 2014) was created for a study about trajectories and behavior patterns of a few hundred electric scooters and their drivers, over the course of one year using on-board GPS devices.

The data is integrated in a system with two views: a geographic map view and a temporal view. The geographic map view shows routes and destinations for identified

Fig.10: Overview of the map, lens and timeline

POIs, where the venues are divided into categories. For more detail the user can select the POI and change the category level. To avoid clutter, icons they are aggregated with other nearby icons, showing an icon with that number of POIs. A lens magnifier tool is represented on top of figure 10 and (following the guidelines of (R. Kruger, 2013)) can be used to filter a certain area for individual scooters and observe their movements in that area.

The temporal component is used by selecting one or more scooters. The user can select a timeframe and then see the most frequently visited POIs displayed on top with a stronger color representing a bigger chance of that location, while the less frequent ones are stacked below in descending order with a faded color. This is shown for each slice of the timeframe.

The advantages are the use of the lens as a filter, enabling the user to focus on a specific area and the design of the timeline that uses the icons, coloring and sorting. The drawbacks are related to the grouping of POIs because it can assign some POIs to a wrong category and related to scalability because even with the grouping, it can sometimes clutter the map.

2.1.11 Geovisual Analytics and Storytelling Using HTML5

This work (Lundblad 2013) introduces a geovisual analytics software based on HTML5/Javascript with integrated storytelling, applied to large spatio-temporal data. This storytelling can be related to news or statistical articles, where the tool provides support, making easier for the authors to express their data and stories through visual means.In this paper we will investigate the existent types of visualizations and how they can be integrated with storytelling, with less detail about technicalities and implementation of the system. The framework is based on HTML5 and JavaScript.

These techniques include choropleth maps (figure 11), scatter plots, bar charts, time graphs, table lenses, parallel coordinates plots, scatter matrixes, distribution plots and tree-maps. It also supports data analysis algorithms, connects the components to each other, and supports other data providers.

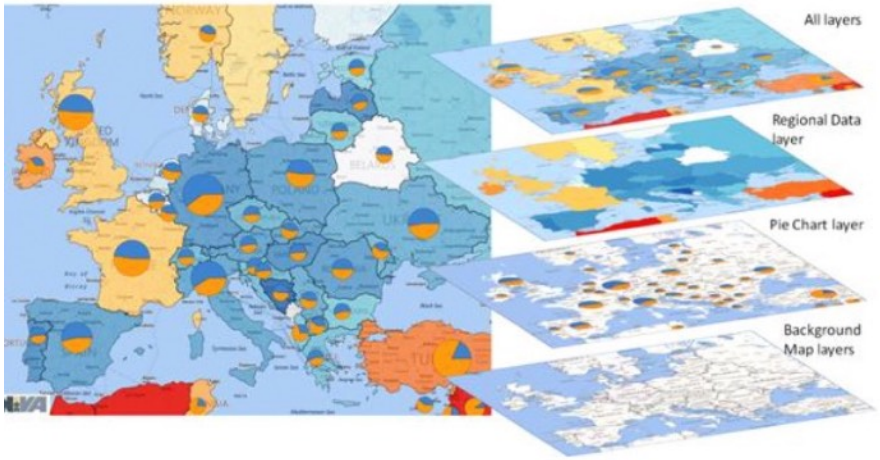


Fig.11: Layers that compose the choropleth map - visuals are created in layers independent from the map to allow many combinations between them

The storytelling is done by using the Snapshot Manager that stores events and inserts them in the story via a hyperlink, showing the state of visualization when the reader clicks it in the page.

The advantage of this framework is the amount of different visualizations available and their detail. The drawback is its scalability - this framework is more suitable to enormous amounts of data.

2.1.12 Visual Analysis of Route Diversity

This work (Liu 2011) proposes a movement visualization method that examines route diversity, to help taxi drivers. It is based on the trajectory data of over ten thousand taxis in Shanghai, China.

The system has three major components, as seen in figure 12: the global view, the trip view and the road view. Users first start with the global view. It consists of three layers: heatmap, location and custom. After the trajectories are analyzed, an overview of traffic flow and diversity is displayed. Users can then select a source and destination to show the trip view. Users can also open the road view to analyze all trajectories passing through a road segment. All views are linked together.

The trip view shows the major routes for a given source/destination pair with the trajectories shown in the central area, the numbers showing times of a day, bars encoding the total numbers of trajectories recorded at different time periods and a

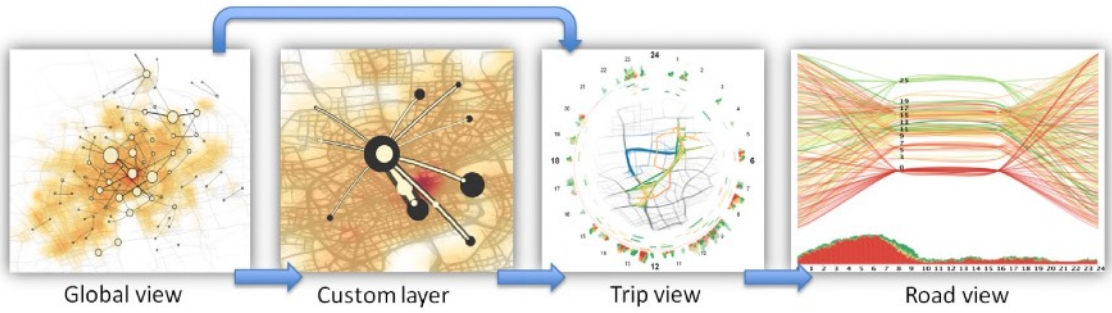


Fig.12: Overview of the system's available visualizations

circular trace with two end points to represent the start/finish times of the trajectory. A color map is used to show the high or low speed of the taxi.

The road view shows two aspects of it: importance and traffic conditions. The view is split into two parts, with the upper part showing the importance of the road and statistical information associated with each route; the lower part showing the speed distribution which reflects traffic conditions.

The advantage is the great detail in all the layers, especially how the information is shown on the trip view. The drawback is the scalability because with too large datasets and trajectories, visual clutter becomes a serious problem.

2.1.13 An Event-Based Conceptual Model for Movement Analysis

The approach of this work (Andrienko 2011) is based on extraction of interesting events from trajectories. It was tested on data about movement of wild animals. The basic visualization tools available in the system include cartographic map display and space-time cube. A set of interactive tools for filtering allows the user to select portions of movement or context data to be visualized.

Also, temporal view of trajectories is a visual display used in one of two modes: time graph and time bars. In the time graph mode shown in figure 13, the vertical dimension

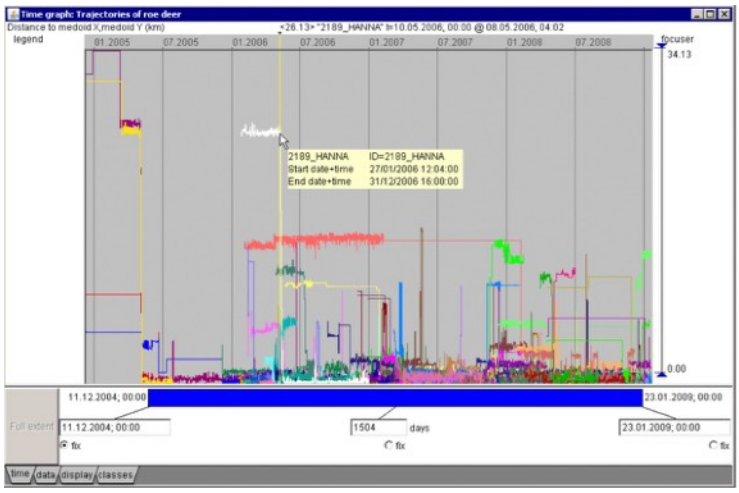


Fig.13: Time graph mode

represents the value range of some time-dependent numeric variable with each trajectory represented by a polygonal line. In the time bars mode trajectories are represented by horizontal bars positioned one below another. The horizontal positions and the lengths of the bars correspond to the temporal positions and durations of the respective trajectories. Non-absent values of the selected variable are represented by colors of bar segments.

The tool's advantage is the number and detail of all filters. The drawbacks are the superimposition of lines in the time graph, visual clutter due to scalability, and less detail of the time bars mode.

2.1.14 Analyzing Temporal Patterns of Visits

This work (Andrienko 2013) presents a tool to visualize data from GPS and GSM tracks, and georeferenced tweets. The data was collected in France and USA for a period of almost five hundred days. After the data was extracted and processed, individual POIs for each person were identified.

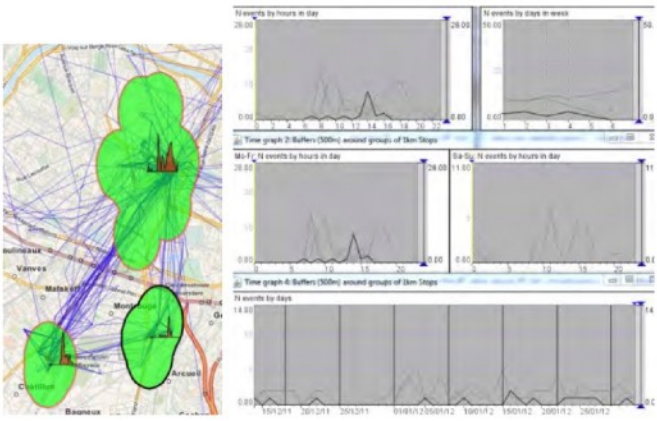


Fig.14: POIs and temporal profiles for person A

The visual interface for POI exploration, interpretation and semantic annotation includes two main components: a map and a set of time graphs. For the GSM data, figure 14 shows that after selecting a trajectory for one person, the major POIs are outlined and the diagrams show the temporal profiles of the stops in these areas. There are five graphs that show the temporal signatures of the POIs, associated

with hours of a day, days of a week, hours of a day for the working days and weekends.

GPS data is also analyzed using the time graphs, as the map view shows the routes of the person in green and the extracted POIs with the profiles of stop occurrences represented by diagrams.

Limitations of this system are mostly related to the lack of detail in the map and time graphs, and little information about the identified POIs.

2.1.15 QS Spiral

This work (Larsen 2013) aims to capture the continuity and periodic properties of quantified self data by introducing visualization of time-series data on a spiral, enabling the user to interact and interpret the data through touch-based gestures.

By using collected data from GPS and Wi-Fi, the visualization builds on top of a clock-dial metaphor with a circle corresponding to a time span (hour, day, etc.). Data points can be shown using different color coding of the timeline, symbols or animations.

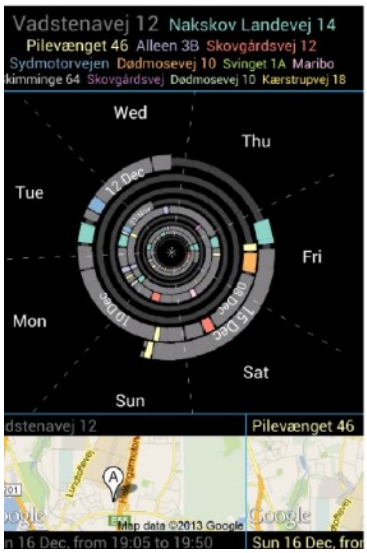


Fig.15: QS Spiral week view

Figure 15 shows the visualization, with a tag cloud of color-coded geolocations shown at the top and a timeline of map segments at the bottom. The user can tap map segments, highlighting those geolocations in the spiral.

A single tap allows highlighting specific elements. A pinch allows to zoom into a particular part of the data and swipe gestures allow to pan. The double-tap gesture switches between different time views. Although the spiral allows for repetitive data deviations to stand out, the system has a limitation related to scalability: a big data set clutters the spiral, with parts too small to tap.

2.1.16 Aprilzero

This is a website created by Anand Sharma [5]. Its main purpose is to act as a health and fitness database for the author [6], showing personal data such as vitamin levels or body fat. It also presents the distance covered by the author while running or walking, for each day of the month and in total. When a run is clicked, the explorer shows the timeline for that day (figure 16), with information such as the time spent running and where the author was and where he went during each hour of that day. The icons represent visited places, with an horizontal bar indicating the start and

[5] http://aprilzero.com/

[6] http://aprilzero.com/sport/

finish of the stay. The website presents a very complete lifelogging experience, with great scalability.

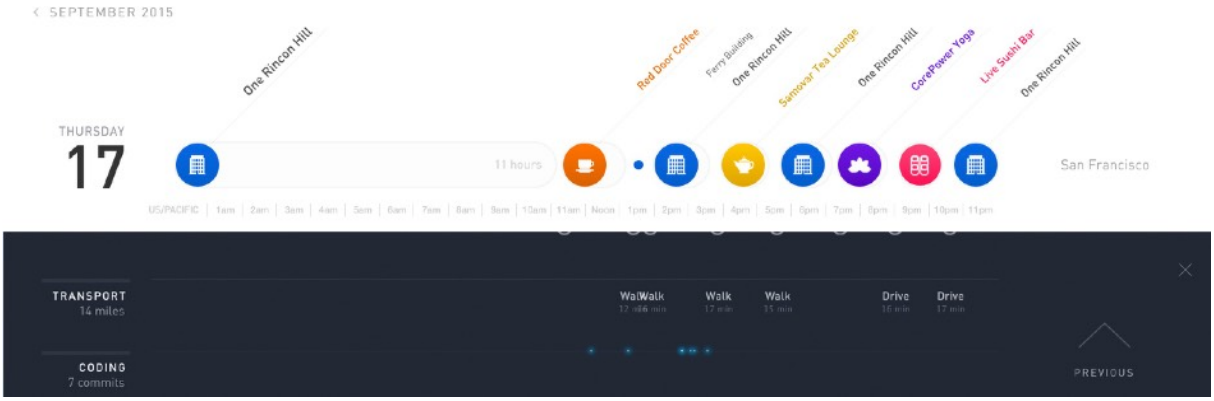


Fig.16: Aprilzero interface

2.1.17 Jehiah 14 - Personal Annual Report

This is a website created by Jehiah Czebotar [7] and in this specific entry he presents his lifelogging report in the year of 2014 [8]. By using different visualizations that we will address, he represents information such as the difference between 2013 and 2014 in terms of spendings on transportation modes (air, auto, train, subway, bike), the distribution of visits to coffee shops, his bike trips in Manhattan, commits and merges made in his work, and the battery capacity of his computer.

The first visualization addresses the difference between the transportation spendings and is represented by two columns, with five colors for each of the five transportation methods. A bar between each column allows to see the matches and while hovering with the mouse we can see the amount spent.

The second one, regarding the visits to coffee shops, is a circular chart that shows the number of visits for each one, and as we hover on each coffee shop, it shows the distribution of visits per morning, afternoon and day of the week.

The third and most important one addresses the mobility of the author, showing his trips on Manhattan (figure 17). It uses a different color for each different road covered in a certain trip. As we hover on a colored square corresponding to a trip, it shows that specific track. This allows for a good level of scalability, although the size of the squares could be a problem in a smartphone screen.

[7] https://jehiah.cz/

[8] https://jehiah.cz/one-four/

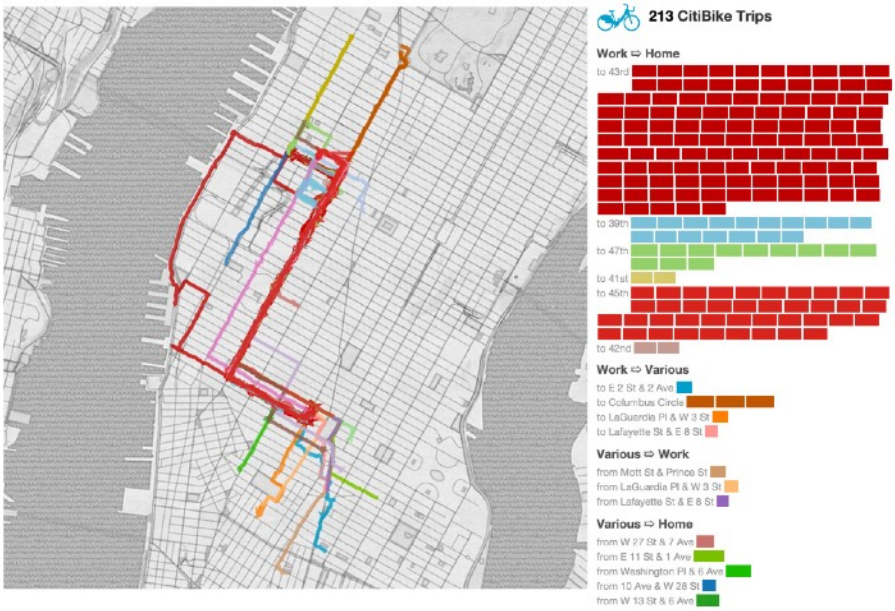


Fig.17: Visualization of the author's trips

2.1.18 Shifted Maps

This work (Otten 2015) proposes a visualization based on a map network that shows visited places and their connections. The places are shown as circular map extracts scaled according to the time spent there and the movements between the places are represented as edges between them, as shown in figure 18. The thickness of the ring

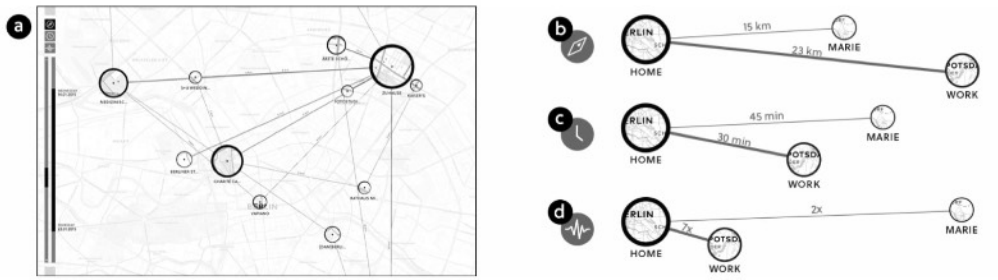


Fig.18: Shifted Maps and its three viewing modes

represents the relative number of visits. The dataset used was based on the movements of three participants for about two weeks.

The first available view, the geographic view, organizes the places based on their geographic positions and distances between them are labeled on the edges. The temporal view organizes the places based on the average time it takes to go from one place to another, with those times labeled on the edge. The frequency view organizes places with frequent connections close to each other. Less frequent connections are more distanced. The frequency of travels is also labeled on the

edge. There is a time-range slider to select and observe the information of a certain temporal period. Each view can be explored using semantic zoom, with bigger levels of detail in bigger zoom levels.

The advantages of this work rely on the simple design of the three views, the focus on personal semantics, as well as some scalability. The drawbacks are related to a lack of detailed information about the places, and low interaction with the circles or edges.

2.1.19 VSÝNTHESIS

This is an entry [9] on a website dedicated to personal data visualization projects [10]. In this work we will focus on only one of the visualizations presented: the Activity River (figure 19), which allows users to visualize daily activities as a stream of colors in a timeline. The different colors represent different activities. Although there is some lack of detail and no places or routes shown, this view is scalable and could be modified to show months or years instead of days, allowing users to observe even more patterns.

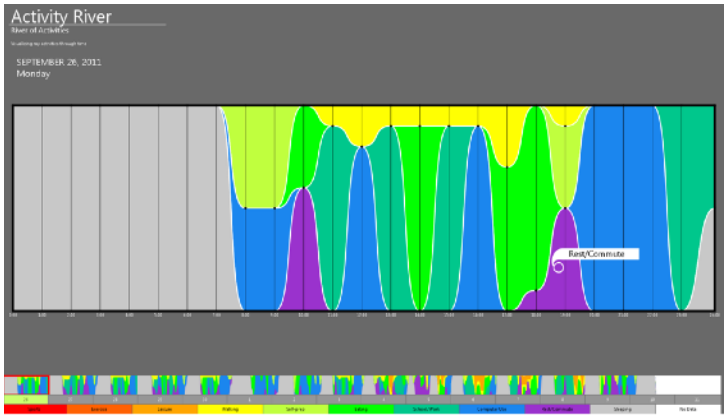


Fig.19: Activity River

2.1.20 Urban Trajectory Timeline Visualization

This paper (Wang 2014) presents a set of timelines to show one trajectory, based on different criteria. The used data comprised the mobility of 492 taxis roaming from a railway station to an airport, with a span of 24 days. Associated with the sampling points are speed, turns, direction and whether there is a passenger inside the taxi (the extraction only selects these ones). By using a created algorithm, they are able to split the trajectory into stops and moves that will be processed in order to correctly display attributes in the timelines.

The interface is represented in figure 20. It shows a timeline as a 2D heatmap to visualize trajectory curvature change over time; a timeline as a 3D terrain, to also visualize trajectory curvature; ThemeRiver to show the temporal distribution of trajectories in a day; a map view to show spatial information of trajectories; a

[9] http://vis4me.com/portfolio/vsynthesis/

[10] http://vis4me.com/

projection view to show proximity of trajectories; a matrix view to show trajectory attributes and a color palette for group tagging.

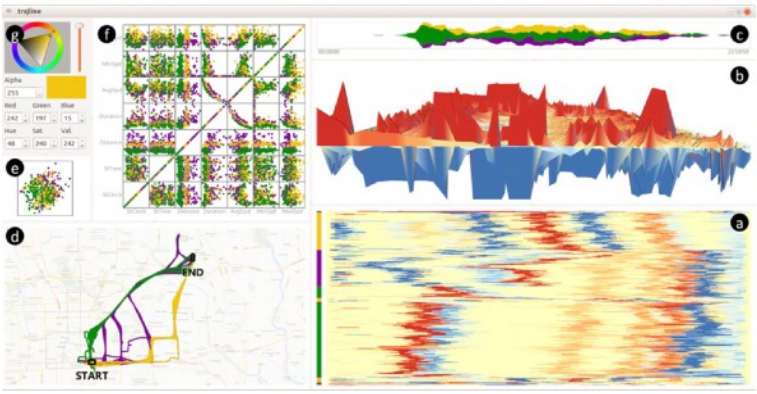


Fig.20: Red/blue represent turning direction and yellow/green/purple tag 3 groups

The timelines here complement each other, offering good detail about the trajectory and how each taxi behaves. The matrix can be especially informative. However, the 3D timeline may not be scalable, cluttering the visualization and providing inaccurate information.

2.1.21 OD-Wheel

This project (Lu 2015) proposes a new visualization and analysis tool, in order to explore origin-destination (OD) patterns. The dataset used is based on GPS data from almost 30 thousand taxis in Beijing, in a span of 24 days. Each sample contains attributes such as time, direction or latitude.

Users can define a region, with all taxi trips starting from or ending on that region are selected and grouped into OD clusters. The hybrid circular-linear visualization allows users to explore patterns of each OD cluster, such as the variation of traffic flow volume and traveling time.

After the data is preprocessed, users do interactive filtering on the trajectories by direct manipulation from spatial and temporal aspects. Then, a clustering algorithm groups trips into OD clusters with three different views available: an overview, a map with the clusters' spatial distribution and line plots with the attributes' statistics. The OD-Wheel also supports comparisons between clusters and has a temporal filter to view data from a selected time range.

Each OD cluster is a stack of bars along a time axis, drawn in the wheel (they can be moved around). The width of the bar encodes the temporal dynamic value during a corresponding time interval. A linear view is connected to the wheel (figure 21) for less distortion and more accuracy. The inward arrows represent O clusters and the outward arrows represent D clusters.

The big advantages are the easy comparison of data and the interactivity that the wheel allows. The drawbacks here are related to the visual cluttering of the wheel when there is a big number of OD clusters and the low scalability of the map overview.

Fig.21: Overview of the wheel and the system's views

2.1.22 TrajRank

This work (Lu 2015) presents a visual analysis method based on trajectory ranking. The dataset used was the same as the one used in the previous work (the authors are the same and this work aims to continue the previous one). The ranking system tries to merge trajectories with a similar calculated ranking on all road segments as bands. Each band corresponds to one kind of travel behavior. This ranking method allows users to compare the behavior of vehicles based on their relative position, instead of the absolute time.

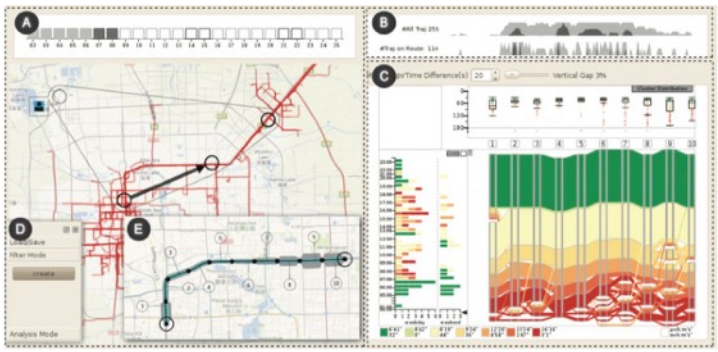


Fig.22: Interface of TrajRank

The interface, as shown in figure 22, consists of 4 views: a spatio-temporal view with trajectory filtering and route segmentation; a horizon graph view that shows the distribution of trajectories over a day; the aforementioned ranking view where a red color corresponds to a low score, indicating low speed or bad traffic condition, and a green color represents the opposite; a temporal distribution view displays the distribution of trajectory groups over a customizable time interval that is also used as a vertical axis for the view; and finally a modified box-plot that shows a statistical description of travel time on each road segment.

Although the scalability problems of the ranking view, this graph can be tweaked to show bigger quantities and different types of information. The temporal distribution view also proves itself very simple and useful.

2.2 Representations using heatmap

These works provide visualizations where movement data is represented through heatmaps.

2.2.1 Activity Patterns in Public Space

Fig.23: Heatmap to represent stops

This is a work (Spek 2010) by the University of Delft that aims to use tracking devices as means to gain insights and define interventions in public spaces, in order to improve walkability of the city centre. We will not explore how data was collected or what were the conclusions of the study, but rather analyze the tracks visualization methods presented, as well as their advantages and disadvantages.

In figure 23 the heatmap is used to show the highest density of people movement. This can be used to identify POIs such as restaurants, grocery stores or parking lots.

In figure 24, a variation of the previous heatmap allows to see which streets/roads have the biggest flow of people. In the context of this work, it is used to understand which roads are more problematic and need intervention in order to alleviate the flow of people. It could be used to find relevant movement patterns such as the roads taken while going home from work.

Limitations with this technique are related to the map having close to no detail besides the patterns. It would be necessary to add some sort of descriptive view of the information.

Fig.24: Heatmap variation

2.2.2 Visual Analysis of Route Diversity

This work, discussed in 2.1.12, contains a heatmap layer in the interactive visualization framework. This layer that is represented on figure 25 is used to identify hot spots in which higher percentages of vehicles travel around. The red areas represent regions of high densities of vehicles while white areas represent regions of relatively low densities. Users can use the heatmap to choose locations for further analysis.

Fig.25: Heatmap layer of the system

2.2.3 Visual Analytics For Detecting Behavior Patterns In Geo-Temporal Data

This system described in 2.1.6 has other incentives in order to further explore data. One of these is a heatmap visualization that can be observed on figure 26. It has a map with POIs and streets that allows to analyze the stops and time spent in them.

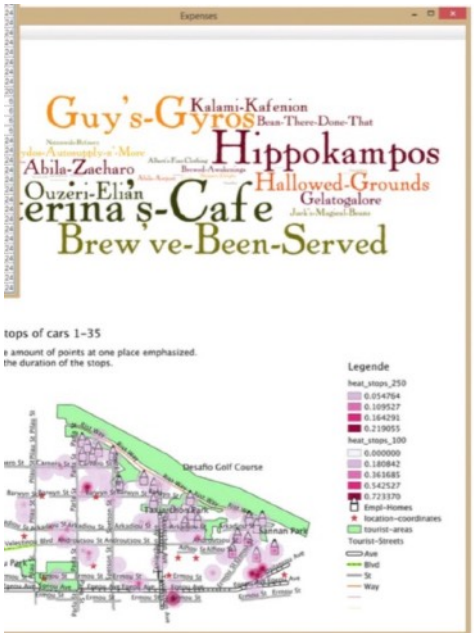


Fig.26: Heatmap visualization of the system

2.2.4 Tracking Bicycle Mobility - 9 Days In Amsterdam

This is a blog about mapping, data visualization and urbanism created and operated by Patrick Stotz and Achim Tack [11]. In this specific blog entry [12], one of the authors (Stotz 2014) creates a simple heatmap of his tracks in Amsterdam [13]. By collecting GPS data from his smartphone and OpenPaths service, and then completing and correcting it, we are able to observe the most used transportation method and the roads with higher density of tracks (figure 27). These roads are the ones in which the orange color is brighter.

Fig.27: Tracks on the heatmap

Again, the biggest advantage here is the simplicity in observing the mobility patterns of one person. The drawback is that although we can see the transportation method, there is not much detail. It would be interesting to see this also applied to places the author visited.

2.3 Representations using space-time cube

These works provide visualizations where movement data is represented through a space-time cube. Following the guidelines of (Kraak 2003), a basic space-time cube consists of a cube with a representation of geography on its base (along the x- and y-axis), while the cube's height represents time (z-axis). Typically it contains space-time paths for individuals or groups of individuals.

2.3.1 Analyzing Temporal Patterns of Visits

The work explored in 2.1.14 also uses a space-time cube in order to explore the data collected from GPS data. On the top of the cube in figure 28, the trajectories and stops are temporally aligned to the weekly cycle. On bottom the trajectories and stops of the work days are temporally aligned to the daily cycle. The trips are represented by lines and stops by spheres.

The limitations of the space time cube are mostly related to scalability: it is hard to identify and explore massive amounts of data.

---

[11] http://mappable.info/

[12] http://mappable.info/blog/2014/9/5/amsterdam

[13] https://www.closr.it/canvas/1306/#/

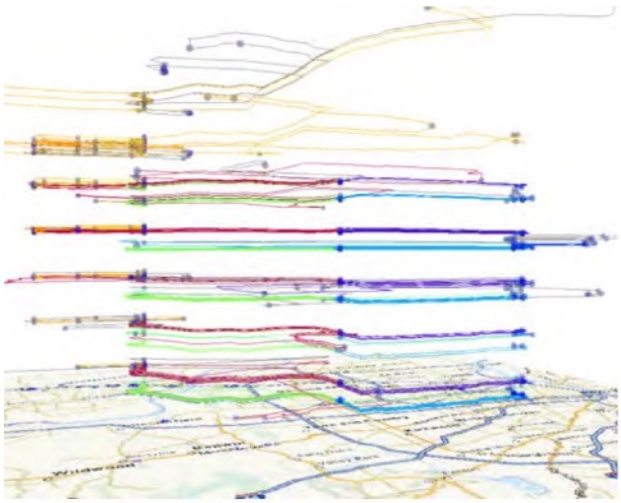


Fig.28: Temporal distribution of a person's trips

2.3.2 An Event-Based Conceptual Model for Movement Analysis

As explored in 2.1.13, this work also uses a space-time cube as means of visualization of the trajectories of the animals, together with the map display, as is displayed on figure 29. It gives an insight about animals' movement in relation to space. Each position of a trajectory is located in a certain cell of the space-time continuum. The yellow circles represent the events of the roe deer and the pink circles the events of the lynxes. The mouse cursor is positioned on one of the segments in the temporal view of the trajectories of the roe deer. This shows the corresponding temporal and spatial positions.

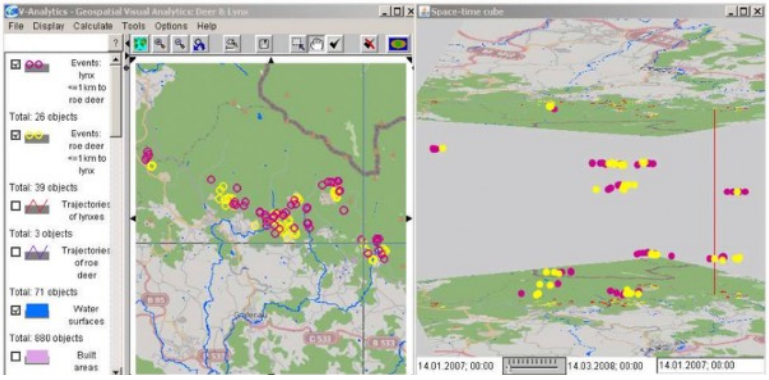


Fig.29: Map view and space-time cube

2.4 Discussion

Table 1 summarizes the most important metrics of the works explored in the previous sub-sections.

The first topic explored is scalability. This is one of the biggest issues in the field of visual analytics (reference to problems of geospatial visual analytics paper) - most of the explored works use existing datasets or data that comprises short time intervals. When data volume increases, map visualizations tend to clutter with too much information for the user to absorb. Works like (Saraf 2014), (Krueger 2014), (Andrienko 2013) and (Otten 2015) still have some scalability, mainly due to the fact that (Saraf 2014) does not rely heavily on map visualization and relies more on timeline and plots; (Krueger 2014) groups spatially close icons into an icon showing the number of aggregated icons; (Andrienko 2013) is scalable because the visualization presented is a heatmap, dealing with no more than density information; (Otten 2015) relies more on the connections between places rather than their geographic positions. Only works like (Stotz 2014), (Aprilzero) (Jehiah 2014) and (VSYNTHESIS) have the capacity to show a lifetime quantity of data without losing detail or cluttering visualizations.

Regarding the data collection methods on the second column, all the works that answered the VAST challenge were given the same predefined dataset based on GPS locations and routes, with the teams processing and using it distinctively. Works like (Visits), (Lundblad 2013), (Aprilzero), (Jehiah 2014), (Otten 2015) and (VSYNTHESIS) used other data sources such as smartphone apps (Foursquare, Moves, etc.), sensors, statistics and news. GSM and Wi Fi data is also collected ((Smart Reality Testbed), (Andrienko 2013), (QS Spiral)) but the main source here is GPS.

About the visualization methods on the third column, almost all of the works present the information in 2D maps and use timelines. The maps in their majority are based on Google Maps, although they differ in the way the tracks and/or locations are shown. The timelines also differ from each other. For instance, (Krueger 2014) shows its timeline with icons and colors and (Wood 2014) shows in a calendar type view. (Saraf 2014) also shows plots as a somewhat successful method that does not clutter visualization and allows for instant identification of patterns. There is a minority, represented by heatmaps that although are very easy to analyze, provide low detail, space-time cubes that seem outdated and clutter the view, and a spiral visualization in (QS Spiral) that has a serious scalability problem.

Regarding the topic of preprocessing, it englobes methods such as data cleaning, track simplification, clustering, removing inconsistencies, repetitions or redundancies. This is important for a good visualization experience, because it allows for data to be previously analyzed and transformed into information that makes sense for that

	Scalability	Data Source	Visualization Method	Preprocessing	Panels and Menus	Personal Semantics	Paths?	Places?
1) Smart Reality Testbed	No	GPS, Wi Fi, cell, others	Timeline + map	Clustering	Yes	No	No	Yes
2) Patterns Of Life	No	Existing dataset	Timeline + map	Derive latitudes and longitudes	Yes	No	Yes	Yes
3) Visits	No	Others	Timeline	No	No	No	No	Yes
4) Dodeca-Rings	No	Existing dataset	Timeline + map	Association, anomaly detection, clustering	Yes	No	No	Yes
5) (Siddle 2014)	No	Existing dataset	Map	Track simplification	Yes	No	Yes	Yes
6) (Saraf 2014)	Some	Existing dataset	Timeline + map + plots	Clustering, association	Yes	No	No	Yes
7) Movement Finder	No	Existing dataset	Timeline + map	Association, manual GPS error removal	Yes	No	Yes	Yes
8) (Hundt 2014)	Yes	Existing dataset	Map + heat map	Using KNIME	Yes	No	No	Yes
9) (Wood 2014)	No	Existing dataset	Timeline + map	Automatic proximity analysis	No	No	Yes	Yes
10) (Krueger 2014)	Some	GPS	Timeline + map	Clustering	Yes	No	Yes	Yes
11) (Lundblad 2013)	No	Others	Plots + maps + charts + matrixes	No	No	No	No	Yes
12) (Liu 2011)	No	GPS	Map + heat map + timeline	Map-matching algorithm	No	No	Yes	Yes
13) (Andrienko 2011)	No	GPS	Timeline + map + space-time cube	No	Yes	No	Yes	Yes
14) (Andrienko 2013)	Some	GPS, GSM	Map + time graphs + space time cube	Temporal aggregation, clustering	No	No	Yes	Yes
15) QS Spiral	No	Wi-Fi, GPS	Spiral	No	No	No	No	Yes
16) (Spek 2010)	Yes	GPS	Heat map	No	No	No	Yes	Yes
17) (Stotz 2014)	Yes	GPS	Heat map	QGIS	Yes	Yes	Yes	No
18) Aprilzero	Yes	Others, sensors	Timeline	No	Yes	Yes	No	Yes
19) (Jehiah 14)	Yes	Others	Map + charts + plots	No	No	Yes	Yes	Yes
20) (Otten 2015)	Some	Collected Dataset	Map	Unknown	Yes	Yes	No	Yes
21) VSÝNTHESIS	Yes	Others	Timeline	Unknown	No	Yes	No	No
22) (Wang 2014)	Some	GPS	Map + timelines + matrixes	Custom algorithm	Yes	No	Yes	No
23) OD-Wheel	Some	GPS	Timeline + map	Clustering, association	No	No	Yes	Yes
24) TrajRank	Some	GPS	Timeline + map + graph + plots	Clustering, association	Yes	No	Yes	No

Table 1: Visualization Summary

context - a GPS track from A to B with minimal deviations in the path can be simplified into a straight line with no intermediate points, or dividing a big track into smaller tracks that correspond to a trip. More than half of the explored works use preprocessing tools, with focus on clustering algorithms.

Another topic addressed is whether or not the works deal with personal semantics. This is related to the information shown having personal meaning to the user (eg. my house, my work, dinner time, etc.). A minority of the explored works support locations and times only important to the user (they only analyze non personal data), being this a problem that we will address in our work, since it is a project for analyzing personal geolocation data. The exceptions are (Stotz 2014), (Aprilzero), (Jehiah 2014), (Otten

2015) and (VSYNTHESIS), works in which the personal lives of the authors or testing users are represented.

The remain three columns refer to the existence of panels and menus, if the works show paths instead of tracks, vice versa, or both. The majority of works rely on the use of menus and/or panels for navigation, selecting and filtering the information to be displayed. Almost all works show places in their visualizations, with more than half also showing the paths that lead to those places.

Recapitulating, the most important aspects of our solution are the focus on scalability (the ability to support massive amounts of data), personal semantics while analyzing the data and preprocessing.

Chapter 3

Data Collection

Based on our already stated objective to design a visualization that allows users to understand and analyze their spatio-temporal information, with focus on personal semantics, we have to collect personal mobility data to investigate if we achieved this objective. This data comes from two sources: the first is our personal mobility data and some selected users' data, collected through the use of GPS-enabled devices (namely a smartphone) and stored in files; the second are notes made by the author, about his tracks. This type of notes allow users to give a personal semantic context to their tracks, to give meaning to their trips and stays.

Even if the goal of our work is not focused on the data collection and processing, to evaluate our solution we will need to test it with datasets that contain personal mobility data. In this chapter we will describe the collected data types, understand how we can use them and their related issues. Collecting this type of data can be a lackadaisical process, as there are issues that can lower the data quality or even discourage users to stop doing it.

The said data types will be described in section 3.2. Then, their related problems and some solutions are described in sections 3.3 and 3.4. Nonetheless, before we started the design of our solution, we had to conduct a survey on potential users in order to know what they would like to observe on it. This study is described in section 3.1.

3.1 User survey - which factors are relevant

First we have to know which factors are relevant for the users. This is one of the sub goals we mentioned, and in order to achieve it and assess the interest of potential users in our solution, we decided to create a survey. The survey was made using the Google Forms platform and was published and distributed via friends and colleagues through social media (LinkedIn, Twitter and Facebook).

After a short introduction to the work that we will unfold, the users considered the following case: during an interval of several months or years, they collected GPS

data of the places they visited, paths they took to get there and temporal data associated to it (date, hour, etc.). Now, they could explore that data in order to better understand their mobility patterns. Then, they answered the 16 questions provided in Appendix D, ticking in the scale of 1 (least interesting) to 5 (most interesting) their degree of interest in the presented functionalities.

After the questions were made, we decided to group them. We used the topics of time, places, routes, distances and patterns. Associated with time are the questions 12, 10, 3, 2, 1; related with places we have questions 15, 13, 16, 12, 10, 4; related with routes we have questions 16, 10, 6, 5, 3; related with distances we have questions 6 and 7; and finally, related with patterns we have questions 15 and 11. The survey was online for about one week and we collected 171 answers. Then we went ahead with the processing and statistical analysis of the results:

We first started by removing invalid answers: incomplete or incorrectly filled questionnaires. We also removed outliers. In order to find them, we calculated the subtraction of the mean with two standard deviations and the sum of the mean with two standard deviations. The results were the values 2.53 and 4.73. Ten answers fell out of this range and thus were not considered.

Before removing the outliers, the mean of the responses to the survey was 3.6. The standard deviation is 0.6. Afterwards, the results did not change much, as the mean just went up from 3.6 to 3.7 and the standard deviation went down from 0.6 to 0.5. We can conclude that outliers did not have a strong effect on the mean and the standard deviation. The average of 3.7 is above the central value of the scale and thus represents interest in the topic and functionalities presented. This shows that the values were not very dispersed from the mean, backing the idea that this topic and presented objectives is appealing to people. Finally, we used a color scale in order to identify the questions that had better results. Questions that had answers with a mean between 3 and 3.5 were colored orange, meaning that people had low interest. If the mean was between 3.5 and 4 they were colored yellow, meaning that people had some interest. If the mean was bigger than 4 they were colored green, meaning that people had large interest. With a meanbelow 3 they would have been colored red, meaning no interest, but that did not happen. Then, using the previously mentioned pairing between the questions and topics, we created the following table:

Time	2	2	1	
Places		3	3	
Routes	1	4		
Distances		2		
Patterns			2	

Table 2: Summary of the survey

Here we can see that time is the topic that users find more critical to understand and plan their life activities, with focus on the time taken on public transportation and time spent on a route to a certain destination. The next one is the routes they take to and from their destinations, with focus on the quickest, the different routes taken or the distances associated to them. The third most critical topic are the places they visit, with potential users showing interest in knowing how many times they visited a certain place, where they spend most of their time, and being able to assign a photo or album to a place. Distances and patterns - with focus on, for instance, being able to know the total distance covered on a given timespan, or observing if the mobility patterns changed - are the less critical topics.

3.2 Types of collected data

Our work will use two different types of data. The first type is GPS data collected by a device (smartphone) and stored in GPX files. The second type is personal semantic data describing the trips made and places visited by the users.

3.2.1 GPS tracks

To achieve our main objective of having a visualization that allows users to analyze their spatio-temporal information with focus on personal semantics, users must first collect that said information by collecting GPS data on a daily basis. The used device for the data collection is not important, although nowadays is far easier to download a mobile application (available for various smartphones and operating systems) for that purpose. What is imperative here is that the GPS data collection has to produce some file output that we can use as input for our application, namely its back-end.

Since the use of smartphone applications for GPS data recording is the most widespread manner of doing it, we will assume the common user of our application does it as well. So, we will center our attention on it. These various applications allowthe exporting of the recorded information to a file or a set of files, in a common format: GPX. GPX stands for GPS Exchange Format, an XML schema that stores GPS data and allows its use in other software applications. This is the file format that we will use and manipulate in our application, as exemplified in figure 30.

Each file that is generated by these recording applications is called a track. It contains location data - latitude, longitude, elevation, a timestamp for each point. Note that a recorded track does not necessarily represent a trip. The concept of trip is, among others, a very important concept in the context our work. A trip is a more subjective term, related to user behavior. It represents a travel that a user made from place X to place Y without taking in consideration the distance, paths or intermediate places covered. All in all, a track can represent a trip, but it is not a strict representation. For instance, a trip can be divided into many different tracks because the GPS recording was stopped several times due to loss of signal or user

interaction. Also, due to user error or forgetfulness, a single generated track can coincide with more than one trip.

To address these problems, one solution would be for the user to turn on the GPS collection when moving from one place to another and turn it off when entering a place. This would simplify the recording and optimize the use of the device's battery, since GPS sensors can consume a lot of it, and nowadays smartphones rarely have the capability to endure a full day of usage on one battery charge. As we will see in section 3.2, this is one of the problems associated with mobility data collection.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- GPSTrack 2.2.1 - http://bafford.com/gpstrack -->
<gpx xmlns="http://www.topografix.com/GPX/1/1">
<trk>
<name><![CDATA[2016-01-16 21-36-19]]></name>
<trkseg>
<trkpt lat="38.704938" lon="-8.970171"><ele>14.135672</ele><time>2016-01-16T21:36:18Z</time>
<!-- hAcc=65.000000 vAcc=11.453109 --></trkpt>
<trkpt lat="38.705009" lon="-8.970241"><ele>14.543830</ele><time>2016-01-16T21:36:24Z</time>
<!-- hAcc=65.000000 vAcc=10.443358 --></trkpt>
<trkpt lat="38.705052" lon="-8.970250"><ele>14.933756</ele><time>2016-01-16T21:36:31Z</time>
<!-- hAcc=65.000000 vAcc=10.478768 --></trkpt>
<trkpt lat="38.705123" lon="-8.970269"><ele>14.933756</ele><time>2016-01-16T21:36:40Z</time>
<!-- hAcc=147.353491 vAcc=12.864718 --></trkpt>
<trkpt lat="38.705168" lon="-8.970295"><ele>14.933756</ele><time>2016-01-16T21:36:41Z</time>
<!-- hAcc=148.618549 vAcc=12.864718 --></trkpt>
```

Fig.30: Sample of a GPX file

3.2.2 Personal semantic notes

As described in the previous section, we now know where the spatio-temporal data that will be used to achieve our main objective comes from. Now we need the personal meaning of that information, to help us fully achieve our main objective. To create these personal semantic notes, we will adopt the LIFE format created by Daniel Gonçalves and available on GitHub [14]. This library also contains helper functions that will later allow us to process the semantic notes on our back-end.

This format allows the association of labels (meaningful names to the user, such as "grandmother's home", for instance) to specific coordinates of the data collected by the GPS devices. This means the user can later search and visualize his personal semantic information on the application front-end, using the associated labels that represent the real world coordinates.

The base goal of this format is to assign a semantic meaning to a certain time interval. In the context of our work, it is mainly used to assign a name to the time a user was indoors. As we will see next, it can also be used to assign a name to a trip (underground or bus, for instance).

[14] https://github.com/domiriel/LIFE

This format is centered on the days of a given year and month. For each day we have a number of spans that describe where the user was on a certain time interval. We know the user was traveling when a time interval does not have a span. So the spans represent the time the user was indoors - a stay. These stays represent real world places the user has been and that have personal meaning to him. Trips are an exception in this format. A trip is a special type of span, in which two places are mentioned, instead of one. These two places point out the origin and destination of the trip. Spans can be annotated with tags, comments and specific periods of time. This format also supports information about the categories of places, their physical location (latitude/longitude) and places inside other places (e.g. a shop inside a mall).

```
--2016_01_15
@UTC
0000-0915: home
0924-0930: seixalinho station
1000-1008: cais do sodre station
1009-1024: cais do sodre station->saldanha station
1025-1027: saldanha station
1030-1743: INESC
1746-1750: saldanha station
1751-1815: saldanha station->cais do sodre station
1815-1834: cais do sodre station {Waiting for the boat trip back to Montijo. The boat was late}
1920-2359: home [dinner]
```

Fig.31: Semantic notes for a single day

A day, as exemplified on figure 31, is represented with a date in the format `yyyy_mm_dd`, preceded by two hyphens `--` and then followed by spans that cover the entire day. Each span is represented by `<start>-<end>: <place>`, in which `<start>` and `<end>` are the times of start and end of the spam, following the `hhmm` military standard. Times begin at `0000` and end at `2359`. The name of the personally meaningful place is enclosed in the `<place>` tag. A timezone change can also be indicated as an offset from UTC, in the format `@UTC(+/-)<offset>`. This command must appear inside a day and just after the date of that day. Its effect lasts until another timezone tag is found. As stated before, trips require a special kind of span that represents the start and end point, separated by a `->` sign. The span represents the actual travel time. To represent the time spent at one or both places, a new span must be created for that purpose. These are the stand-out syntax details we need to understand in order to operate this format accurately.

We can consider the time intervals present in these semantic notes to be the complement of the time intervals in which we have GPS tracks recorded. There are two ways of creating these annotations: the first, more traditional way, is writing them by hand on a text editor, normally in the end of each day when we finish the GPS data collection. This is the way we created our notes. The second way is related to another on-going project, developed by a colleague, that complements our work. Shortly, it takes as input a set of GPS tracks and automatically parses them, suggests the associated locations and creates the LIFE file. That on-going project

allows users to simply concentrate on collecting their GPS data, without the need to be constantly writing down the semantic notes.

3.3 Real world data problems

Collecting and managing mobility data in the real world is not always a straightforward task. It is important to enumerate and analyze the challenges posed to users as they collect their real world mobility data. These challenges, addressed below, are related to the GPS tracks and personal semantic notes.

3.3.1 Related to GPS tracks

The following problems are related to erroneous behavior of the GPS devices or created by the user during operation of them:

- Dataset size: the collection of data from many days/weeks/years can create copious amounts of files;
- GPS accuracy: errors created by the devices, such as spikes or inconsistencies;
- Cold start/end: When the user forgets to turn the recording on and off, storing many trips in one single track file. This can lead to some data inconsistencies;
- Loss of signal: Some routes may have missing points due to loss of signal when entering a tunnel or an underground station, for instance;
- Battery capacity: Even nowadays GPS sensors consume a lot of battery capacity, making the user quit the recording process and/or leading to low data quality.

3.3.2 Related to personal semantic notes

These problems arise during the creation of a user's personal semantic notes:

- Multiple meanings: Different labels for the same location, stay or trip;
- User forgetfulness: Whilst creating their personal semantic notes, users may not remember the exact place or hours of a certain event took place;
- Adapting to real world changes: Through the course of time, a place can change names and a name may refer to different places, like when the user moves to a new home, although the label is the same, its real world coordinates change.

3.3.3 Related to both

These problems are common to both collected data types:

- Privacy: Users may not want their personal mobility data to be of public knowledge.

- User burden: The data collection process involves more than recording tracks and creating personal semantic notes, such as cleaning and correcting that information, which can put off some users and lead to low data quality.

3.4 Solving real world data problems

While data collection and cleaning is not the focus of this work, we still needed to address the most serious problems, so that collected data can have enough quality. We achieve this through the use of an algorithm. As we have seen in this chapter, data collection - what data do we need to collect, and how do we collect it - is important as it allows users to have real world data with personal meaning to them, which in turn will be used in our solution. This allows them to analyze their personal mobility data, as previously stated on our objectives. Yet this does not mean we will not be doing any kind of data treatment at all. In this section we will analyze a way to reduce the size of our dataset and mitigate some of the other problems we found. Also, on Chapter 4 we will discuss some ways we can clean and tweak the raw data that comes from both the data collection and the database, so as to optimize it and facilitate its display on the front-end.

3.4.1 Dataset size

As a user gathers more and more mobility data throughout the years, the size of his dataset becomes enormous. There is the need to lower the dataset size as much as possible, while not losing the overall spatial and temporal information about the places and routes taken intact. Also, as we will see on Chapter 5, some of our visualizations deal directly with information such as spatial coordinates, and to have them as quick and responsive as possible, we want a simplified dataset. For this to happen, we need a data simplification algorithm instead of a visual simplification, meaning the paths will still be the same, without losing significant data.

To reduce the number of points we chose to use the Ramer-Douglas-Peucker algorithm [15] (RDP), a well-known line simplification algorithm. It keeps a sufficient quantity of points so that little to no data is lost. Usually, this algorithm ignores the temporal information (which is not efficient), but for our visualization purposes, the spatial information is more vital. To apply the algorithm to the tracks and create the simplified versions of them, we will use the implementation provided by Tkrajina [16].

To check the performance of the algorithm, we compared two versions of a track: the first is an original, non-simplified version, while the other is simplified by the RDP algorithm. We used a short track, of about 400 meters of movement. The original track contained 153 points, while the simplified version contained 13 points. It is a 12

[15] https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm

[16] https://github.com/tkrajina/gpxpy

times reduction in the number of points. This reduction value is not constant, and can average from five to 13 times the original size of the track, depending on its size and on the maximum distance between points - a parameter we pass to the algorithm. Figure 32 also compares the information shown by both tracks. We can see that despite the number of points is reduced, the path is still essentially the same.

Fig.32: Comparing an original track (green) with a RDP-simplified track (orange)

3.4.2 Battery capacity

Collecting GPS data through the use of a device like a smartphone drains a good portion of its battery. Even so, the remaining battery capacity is certainly enough to perform the other usual tasks. As the technology developments in this area are still not enough, one solution can be the use of a spare battery or a power bank.

3.4.3 Adapting to real world changes

Some places, during the course of time, may change their names. Every time a location changes name, a mechanism is needed to inform all the stored locations referring to that place that they should also change names. This is also valid when a place changes location but the name stays the same.

3.4.4 Privacy

Our application is not made with the intent of sending data to the internet. It does not use any external service and all the needed files are stored locally. However, as future work, it may need some privacy towards the existence of multiple users (login).

Chapter 4

TraceMySteps - Back-end

Here we will present our approach regarding the back-end of our solution. In order to do that, we must first explain the scenario in which our system will be used, and also the architecture we have planned for it. Section 4.1 will describe the database creation process, while section 4.2 will describe how our web server operates.

The scenario is clear-cut: a user will utilize his smartphone (with GPS capabilities) on a daily basis, recording his paths when moving from one place to another, stopping the recording each time he enters a place. There are several applications available for the different mobile operating systems, to do this recording task, but we decided to use GPSTrack [17]. Therefore, a different track is created each time a user moves: for instance, if the user makes a trip from place A to place B and then from place B to place C, then this will result in two collected GPS tracks. As described on Chapter 3, the collected tracks will be complemented with personal semantic notes that give meaning to the places and trips the user made. By processing these two types of data on our application, the user will be able to analyze his personal mobility data, focusing on the personal aspect of it.

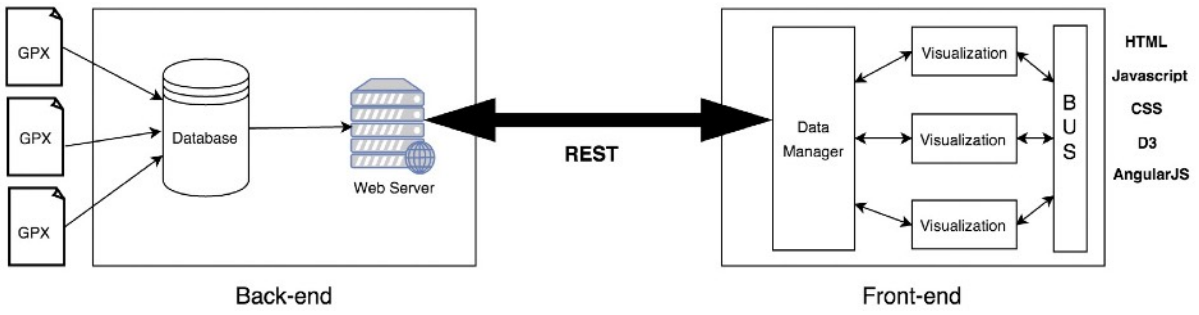


Fig.33: Solution schema

The designed architecture for our solution, as depicted in figure 33, adopted a client-server model. This was attained through the use of a REST between the front-end

[17] https://itunes.apple.com/us/app/gpstrack/id378708797?mt=8

and the back-end. The RESTful state was achieved by using a Python based web server on the back-end, that handles the requests from the client side. Our choice here relied on Python in order to continue previous work done in this area also made in Python, and because of the support and libraries publicly available. We provide all our work through GitHub [18], for others who wish to continue and refine our work.

The concept here is that the client-side does not have to do the heavy work, only sending requests and get the specific information to be visualized, with little to no need to treat the data or do the queries itself. That heavy work is made by the server, as it communicates with the database. We firstly planned to implement this Python web server using the Flask framework [19] with extensions such as Flask-Restless [20] and Flask-SQLAlchemy [21] (SQLAlchemy is a Python SQL toolkit and ORM), but in the end we opted to use other extensions. This change occurred because of the need to handle geographic information not only on the database population, but also on the queries and response construction. Thus, the chosen extensions consist of Flask-RESTful [22], a Flask extension that allows the creation of REST API's based on a lightweight abstraction, Psycopg [23], a PostgreSQL adapter for Python, and PPyGIS [24], an extension for Psycopg that adds support for geometry objects.

The back-end also comprises the said database. This database will be fed with the GPX files and personal semantic notes described on Chapter 3. It is a common relational database. But because we will use geographic information, PostgreSQL alone is not enough, since it does not support geographic types. We need our database to handle points, edges, rectangles, etc. The object-relational database system we will be using is, as said, PostgreSQL. And to have this spatial support we have chosen a PostgreSQL extension: PostGIS [25]. It is a well known service with support to points, line string and polygons. It also follows the SQL specification from the Open Geospatial Consortium. For a much more in-depth manipulation and simplification of the tracks, there is an ongoing project by a colleague that does this work. We will assume the data is already considerably cleaned and consistent, in order to put it on the database.

---

[18] https://github.com/PedroF20/traceMySteps-backend

[19] http://flask.pocoo.org/docs/0.10/

[20] https://flask-restless.readthedocs.org/en/latest/

[21] http://flask-sqlalchemy.pocoo.org/2.1/

[22] http://flask-restful.readthedocs.io/en/latest/index.html

[23] http://initd.org/psycopg/

[24] http://www.fabianowski.eu/projects/ppygis/

[25] http://postgis.net/

The front-end consists of a manager, the visualizations and a bus. After some research, related on the next chapter, we decided to use the AngularJS [26] framework to build it. The data manager acts as a broker: sends the requests made by the user to the server and then receives the information, sending it to the connected visualizations to be displayed. It further manages the requests from the multiple visualizations, prioritizing and sending the correct information to each one. Also, the manager has a cache that stores previously requested information or processes in the background, in order to increase speed and response time. Together with HTML5, we will use Javascript and CSS for the development, as they are the standards in the industry. We will also use D3 [27], a Javascript library which helps creating dynamic and interactive data visualizations in web browsers. For some of the visualizations we will also use Leaflet [28], a library for interactive maps, and some of its plugins. Finally, there is a bus that links all the active visualizations. This bus was planned to be implemented using the Observer Design Pattern in Javascript. But as we will see on the next chapter, we will make use of a tool that AngularJS provides us in order to achieve the same result. This bus will allow all visualizations to show the same information in a given moment, maintaining coherence. When a visualization is updated, it will notify the bus the changes that occurred. Then, the bus communicates that update and its information to the visualizations that can use it. The updates can be actions like requesting data or interactions like filtering or highlighting parts of a visualization.

4.1 Database

It is one half of our back-end, and to have it fully functional we have to address three components: the data model, that consists on the scheme of the database and is used to create the tables in which we will store the data from both the tracks and LIFE files; the track processing, in which we simplify the GPX files using the RDP algorithm described on Chapter 3; and the database population, in which both the user's LIFE file and his simplified tracks are parsed and inserted in it using methods provided by the Psycopg and PPyGIS extensions.

4.1.1 Data model

The data model is a SQL schema that contains the instructions for the creation of the tables we will be using to store our data. Note that its base version was not created by us - it was created by Jorge Saldanha [29] and adapted by Rui Gil for his

[26] https://angularjs.org/

[27] http://d3js.org/

[28] http://leafletjs.com/

[29] https://github.com/jmsfilipe/where-have-i-been-backend

dissertation [30] (GatherMySteps) - we just adapted some tables by adding new columns, in order to store specific data related to our solution. Also, in our case, one of tables is not being used. The base data model is being used in all projects related to the data visualization line of investigation, and may be adapted accordingly to each project, as we did.

Since we are working on a spatial database, some instructions contain special operators in order to store, represent and access correctly the objects defined in a geometric space. These instructions add to the typical SQL queries such as SELECT, and common types such as INTEGER or TEXT. The following and other operations are specified by the Open Geospatial Consortium standard:

- Spatial measurements: computes line length, polygon area, distance between geometries, etc.;
- Spatial functions: modify existing features to create new ones, like intersecting features;
- Spatial predicates: allows for true/false queries about spatial relations between geometries, like “which points are located within a kilometer of this point?”;
- Geometry constructors: creates new geometries, usually by specifying the vertices (points or nodes) that define the shape. The types include Point, Linestring, Polygon, MultiPoint, MultiLinestring, MultiPolygon and GeometryCollection;
- Observer function: queries which return specific information about a feature such as the location of the center of a circle.

With this in mind, we need to choose a spatial reference system (datum) to use. Here, a datum is a model of the earth used in mapping. It consists of a set of numbers that define the shape and size of the ellipsoid and its orientation in space. It gives the best available fit to the shape of the Earth. There are many datums, with each of them optimized to use in a certain part of the world. For our work, we decided to use the WGS-84 (4326) [31], the global standard for use with GPS coordinates.

Our adapted data model file starts by dropping any databases with the same name. Then we create the new instance of the database, and connect to it. After creating the extension to allow the use of PostGIS, we create the tables. There are four tables, as shown in figure 34: *locations*, *trips*, *stays* and *trips_transportation_modes.* Although the data model was not created by us, for the *locations, trips* and *stays* tables, we added some columns. As we will see, these columns are important to store dates and frequencies, so as to allow actions as data filter (using the timeline slider) in the front-end and observation of most common places.

---

[30] `https://github.com/ruipgil/ProcessMySteps`

[31] `https://nsidc.org/data/atlas/epsg_4326.html`

The *locations* table aims to store information about each unique location, and contains four columns: the first one contains the label of the location; in the third column we store the point cluster, a set of points that derived the location. Here we are using the Linestring type, because it requires at least two positions and defines a line between these points in the given order. The second column consists of the centroid of the location. It is a single point in which we calculate the mean of the points in the cluster. This way we have a single point identifying a location.

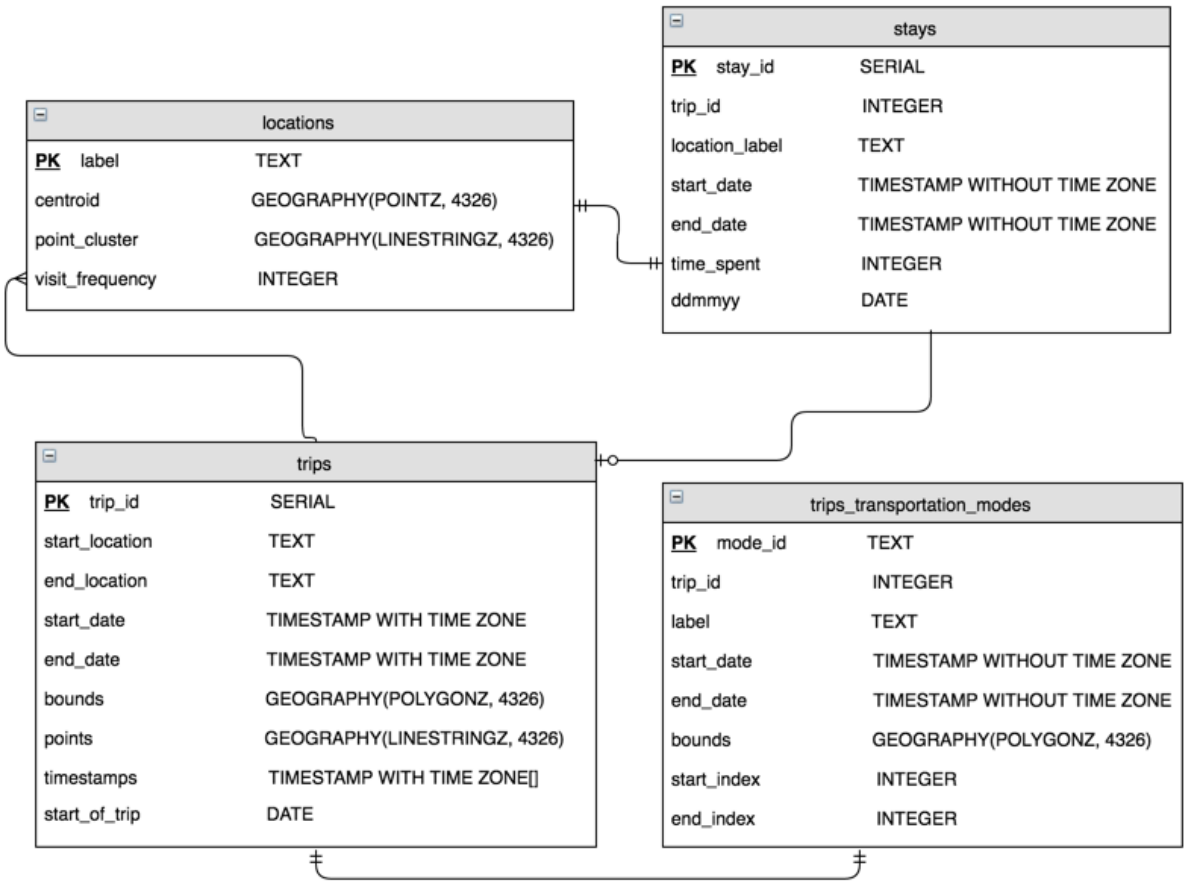


Fig.34: Database Entity-Relationship diagram

The number of times that location has been visited is stored in the last column - ***visit_frequency***. This column is not present in the base data model. We added it to help feed data to some of our visualizations. Each time a location is found while parsing the LIFE file, this column is updated, meaning we can then easily show the user the places he visited the most. On the first versions of the database this column was not present, mostly because we were making a decision and architecture mistake. We were processing the data and feeding the database as if the GPS tracks were the most important part of them, and then we would complement it with the personal semantic notes. But as we have seen on Chapter 3, collecting tracks is not always a simple task, and has some problems. Besides, the focal point of our work is on the personal aspect of the data. So we decided to change the approach and consider the LIFE file as the critical aspect in the back-end data processing and

database population, while using the GPS tracks to complement that information, instead of the opposite. This decision led to the adding of new columns such as the forenamed and to changes in the way we populate the database, as we will see next.

The *trips* table aims to store trips made by a person, and contains nine columns: the first one is an identifier for each trip; the next column contains the label for the starting location of the trip. The next column contains the end location of the trip. The fourth and fifth columns represent the start and end dates of the trip, stored as timestamps. In this table we account for different timezones in the timestamps. The sixth column stores the bounds of the trip as a polygon and the seventh column stores the points that compose the trip as a Linestring. The penultimate column stores timestamps of the trip. In the scope of our work, each trip corresponds to a GPS track. The last column, ***start_of_trip***, was added by us and stores a single date marking the start of each trip. It will be useful to filter the data in the front-end.

The *stays* table contains seven columns: the first one is an identifier for each stay; the second column contains a *trip_id*. This column is optional, meaning a stay may or may not refer to a trip. The third column contains the label of the location in which the stay occurred. The next two columns store the start and end dates of the stay. In this case we do not need to account for timezones. The penultimate column, ***time_spent*** is another one we decided to add to our model, and stores the length of each stay, in minutes. For the purposes of our work, each span of a LIFE file is considered to be a stay. Finally, we have added another column, ***ddmmyy****,* to store the date in which each stay started. Again, this will be helpful to filter the data.

Lastly, the *trips_transportation_modes* table is the only one from this model we will not be using in our work, since it is not imperative in our context of personal semantics and we do not have support for transportation modes (bus, train, etc.) in our visualizations - they require other type of processing.

4.1.2 Track processing

So as to start populating the database and its tables, we first need to apply the Ramer-Douglas-Peucker (RDP) algorithm to the collected tracks. As said on Chapter 3, this is a line simplification algorithm that, in our case, reduces the number of points existent on each track. Our script is based on the work of Jorge Saldanha [32], a colleague in the same line of investigation as us, and while he uses his version of the algorithm, we will stand by the version of Tkrajina, as expressed earlier. We decided to do this because Jorge's version takes the temporal aspect of the track points as another variable on the processing, and for our context (track visualization) that is nonessential - we only need the date in which the trip happened, not the time for all points. The focus is to. Our script, for each track on a folder, divides it in various

[32] https://github.com/jmsfilipe/where-have-i-been-backend

segments and then applies the RDP algorithm on every segment. The RDP algorithm takes as argument the expected maximum space between track points after the simplification, in kilometers. Using a set of three tracks - a small one, one of about two kilometers and the other comprising several kilometers - we did a short test with some values (ranging from 0.001 to 100) for the function argument. This way we could find the value that produced more balanced results (table 3). Note that tracks can average a few meters to hundreds of kilometers, and for that reason we needed to find a value that would optimize the simplification for different distances.

Track	Distance (Km)	Number of points before	Number of points after
1	0.38	153	13
2	1.71	489	28
3	36.31	1818	250
Total	**38.4**	**2460**	**291**

Table 3: Track simplification with an argument value of 5

We can notice a reduction of almost nine times in the number of points of this track set. After each segment is simplified, it is written again on a new file that will contain the same path as the initial track but with those segments simplified by the algorithm. The new file will be named with the date of the track and a number to differentiate tracks within the same day. Finally, each new file is saved in a directory also created in the script, for the purposes of storing the simplified tracks.

4.1.3 Populating the database

So far we have created the database, its tables, and simplified the tracks. We can finally populate the database. This is done using another script created by us. As we can see in the snippet in figure 35, the script has access to the directory of the simplified tracks. Then, it opens and parses the tracks using the Tkrajina library (the same that contains the RDP algorithm) and loads each one. Notice that although it is not used/supported, the script also provides functions to load transportation modes inside the correspondent table in the database.

This loading operation is the centerpiece of the database population process and its structure was basically the same throughout the design of the script, with only some conditions added in order to further clean the data to be inserted. This way we make sure that we do not insert points in which labels are empty, non existent, or are not strings. We always had in mind that to correctly interpret the tracks, we had to break them down in segments, and then in points. That way we could always access the time and date of a specific position and match it with the semantic notes, so as to

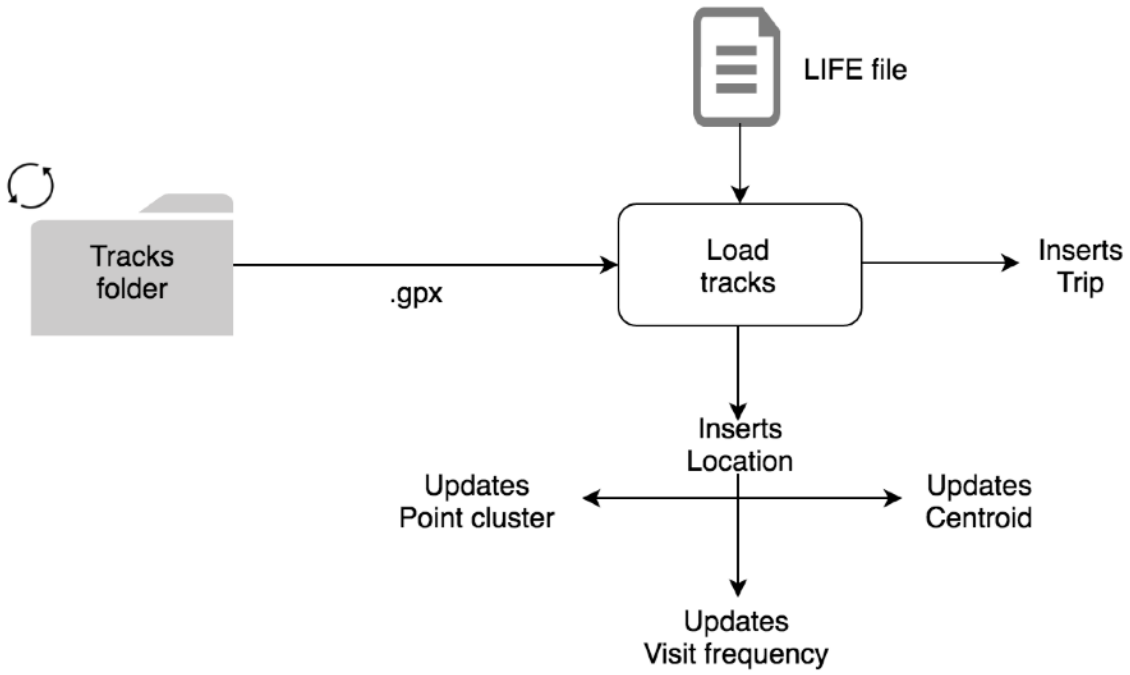


Fig.35: Processing of a folder containing simplified tracks

give a geographic position to the many existent places. Considering this thought process, we obviously start by decomposing each track in segments, and each segment in points. Then, we get the timestamp of each point and access the LIFE file in order to know where the user has been at that timestamp. If he had indeed been at a place at that time and a label for start locations is empty, we consider that point a starting location. A similar process is applied for end locations. This means that in the end, after processing the segments and finding the proper points and labels, we can insert a trip in the database. To insert the trips, we were formerly using a different abstraction because of our different approach. That abstraction considers a trip as a segment and then inserts it. Since in the final version each segment corresponds to the full trip, we decided to maintain this abstraction. It inserts the start and end locations in the database, while simultaneously creates a timestamp for each point. Next, using the geographic features of our libraries, it calculates the bounds and points of the trip. Finally the trip is inserted, together with an identifier for the trip.

Also, in our loading process for each track, we observe that if a location is not a starting or end point of a trip, we consider it as a single location and in order to have more information on the database, we also add it to the *locations* table. The other instance in which we add a location is in the trip insertion, as aforementioned. When a location already exists we update the point cluster of the location. Then we determine the centroid by calculating the mean of the latitudes and longitudes of the point cluster. After these calculations we update the centroid and the point cluster of the location in the table. If the location does not yet exist, we insert the new location, where the cluster and the centroid are the first collected point. The frequency in

which the user visits the location is also calculated and inserted at once, using the LIFE file, so that this column shows the correct data, consistent with the semantic annotations. There is a worst-case condition in the insertion of a new location, in which if there is not a geographic point for it, we only insert its label. Additionally, our script has the ability to manipulate the tracks and the geometry data in order to calculate information such as start and ending times, centroid or bounds. They also cast the geographic information provided by the track points to the PostGIS geometry formats (in our case Point, Geography and Linestring), using the PPyGIS library.

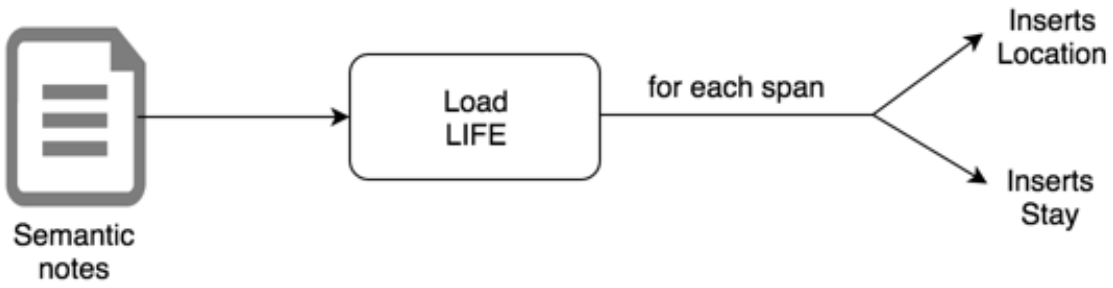


Fig.36: Processing of the semantic notes

With the tracks loaded, we open a new connection to the database and call a function that parses the LIFE file and populates the database with all the stays and labels of locations (figure 36) that have no tracks available - thus with no geographic points. Their spatial attributes will be updated in the database the next time a parsing of new tracks, containing those locations, is called. This procedure allows us to process the semantic notes, independently of the geographic information. This way we make sure that all the stays and locations are correct, and also that the database population is consistent with our architecture choice of shifting the importance towards the semantic notes (LIFE file), as said on section 4.1.1. In the first versions of our script we were doing the LIFE file handling while loading tracks, by inserting the stays each time we parsed a track and inserted the corresponding trip in the database. This meant the stays and the number of times a location was visited were depending on the trips, when they should be depending on the personal semantic notes the user wrote. Also, a number of data inconsistencies would arise due to the problems associated with collecting GPS tracks. Ultimately, when we were comparing the stays we loaded with the real annotated stays, they were fairly distinct in terms of completeness. To help us parse the semantic notations, we use helper functions from the LIFE library. We go through every span inside each day present in the file, calculate the time spent on each stay and convert the dates to a timestamp. Then, for each of those spans, it passes that information together with the label of the place as arguments to the another function we have which in turn will use the Psycopg module to execute only the insertions in the database.

After having the database fully functional, we also performed load testing in our solution, in order to observe if it could handle with a lifetime of semantic annotations and personal geolocation data. To achieve that, we replicated our data to match the quantity collected by the author's supervisor - circa 5 years, 9337 GPX files containing 774477 points, 637 MB of information and 1385 distinct places. We used 2812 GPX files, where each year averages 563 files and 123 MB of data. We also assume the collection of about two to three tracks a day. These tracks totalled 608.3 MB of information and contained 4392048 points. We also had to replicate our LIFE file, leading to the creation of 1073 different locations.

We started by simplifying the tracks using the RDP algorithm, which resulted in the same number of GPX files, although with less total size. The total file size was reduced to 59.3 MB, with the processing completed in 13 minutes and 38 seconds. This is a size reduction of 90.25%. Albeit it took almost 15 minutes, we think it is an adequate processing time, since in a real world situation a user will not process five years of data at once. Ideally they will process new data daily, weekly or monthly, worst case. The insertion of all data in the database together with some web server preprocessing took 10 minutes and 24 seconds. These testes were ran using an Intel Core i7 Dual-Core 3 GHz, 8 GB 1600 MHz DDR3 and a APPLE SSD SM0512F disk. The back-end processed and inserted all the data correctly, thus showing that our database handles big amounts of spatio-temporal data and personal semantic notes.

4.2 Web server

As the database is fully functional, we now need an interface that queries the data and feeds it to our front-end. As we explained before, this is attained through the use of a REST between the front-end and the back-end. The RESTful state is achieved with the creation of a Python based web server on the back-end. The Flask micro-framework provides the tools to quickly develop a RESTful API:

- We start by adding the framework-based instructions that create the server (with the *debug* flag activated, to help the testing and debugging);
- Then we run the Python script in the console;
- We can observe in the console that the server is running, in our case in the port 5000 of the *localhost*, and is waiting for requests.

Now what is missing is the ability to connect to the database and feed each visualization with the data they need. This framework, like others available, also allows methods to manage data in the database, such as POST, PUT or DELETE. But since the database is already populated, we only need to read data from it, hence we will exclusively use GET methods. Multiple visualizations connected to this web server, each with their specific type and format of data, need multiple functions in order to be supplied. Flask offers resourceful routing, allowing us to write in a file a

single interface that supplies all visualizations. This way we have as many URL's as there are visualizations. In the front-end, as we will see on Chapter 5, each one will connect to a URL (with the help of a service) and that specific URL provides the data that is needed. The responses from each resource are always encoded in the JSON format, meaning we need to be careful with some object types that are not supported. A set composed by a URL and a Python class that contains a method that produces the particular data for a visualization, is called an endpoint.

4.2.1 Endpoints

An endpoint is created by calling a method to add a new resource on the Api object. Within this resource and using the aforementioned method, we create the operations that handle the data for each visualization. This means each URL will be routed to the Flask Resource class. This class represents an abstract RESTful resource. Each new resource added to the api extends from this class and exposes methods for each supported HTTP method. We have to pass the name of the class (resource) that will handle the data as the first argument of the method and as the second argument a path (as a string), identifying the respective URL. Although we do not use it, one relevant feature is that we can match parts of the path as variables - such as strings or integers - to the resource methods, meaning we could send, for instance, variables such as names or dates through the visualizations to the endpoints. This can provide multiple options to query and filter the data before it gets to the front-end, though in our context we do not need it.

The process is similar for all endpoints: we start by defining the name of the class that will process the data for one of the visualizations. Then we assign to the resource its specific URL, so that the front-end can make the request to the correct address (ex: "*localhost:5000/endpoint_name")* and get the information it needs. Finally, inside the class, we build the method that queries and formats the data. For our application we created 13 endpoints, where each one is responsible for handling the data for each visualization. Their names and behavior are the following:

- *"hexbinPlaces"* - This endpoint is associated with our Hexbin Places map visualization, so for this case we need to send via the JSON response an array containing the centroids of all the locations visited. This is achieved by querying the coordinates from the centroid of each label. We also add that label to our response. Since this visualization aims to show on map the most frequently visited places, we also manipulate the coordinates of the centroids by replicating them in the array the number of times they were visited;
- *"hexbinTracks"* - This endpoint is also associated with a hexbin map visualization, but this time it is the Hexbin Tracks. The difference to the previous is that this one considers all the points in each track. This way we can use the hexagonal binning

to observe the most traveled paths. The endpoint processes the simplified tracks, returning a set containing the longitude, latitude and date of each point;

- *“calendar”* - Here we process the data regarding our Calendar visualization. We have to first construct an array of JSON objects containing the stays (label, date, time and duration) for each day. Then we have to append each element of that array inside another array of JSON objects. This last array contains the date for the whole day and the total time spent on stays that same day. Shortly, we break each day in their stays. This is an example of nested JSON objects;
- *“areagradient”* - For this endpoint, the JSON response is quite simple: for each day present in the LIFE file, it calculates how long the user spent on the move (walking, bus, etc.), i.e., the user was not staying somewhere;
- *“gpstracklist”* - This endpoint returns a list of the available GPX tracks;
- *“barchartFrequency”* - This endpoint returns the data regarding one half of the Places visualization. It queries the label and visit frequency of each location and creates a JSON object containing every datum of that set;
- *“barchartTime”* - It is the other half of the Places visualization. This time it queries the label of the location, the date in which the stay took place and the time spent on that stay. It also returns a created JSON object containing every datum of that set;
- *“chord”* - This endpoint returns a JSON object where each datum is composed by the start an end location of a trip, and the date in which the trip occurred. This is not the only endpoint in which we return the date of an event (stay of trip) and it was one of the last changes we made in our back-end. By accessing the date of an event, we can filter the data using the timeline slider;
- *“arcedges”* - Here we process half of the data needed for the Trips Network visualization. We get the start and end locations of a trip, assign it a frequency of 1 and build the JSON object. Then we remove datums in which both start and end locations are the same. Finally, we go through the resulting list and for equal travels we remove the remaining copies and increment their frequency based on the number of times each trip happened;
- *“arcnodes”* - This endpoint processes the second half of the data needed for the Trips Network. It queries the database and builds a JSON object containing all the locations that are part of a trip, being them either start or end locations. We do not return more than one copy of each location;
- *“staysgraph”* - It is one of the endpoints that does the most processing, because of the way the Stays visualization is built (as we will see in the next chapter). Although this happens, the number of datums returned is always the same: It processes a set containing an hour from 1 to 24 (to represent each hour period of a day) and a

number representing the day of the week (0 - Monday to 6 - Sunday). Then, for each combination of day & hour we store the label(s) of the stay(s) associated with that tuple. Then it gets their duration, in that exact day & hour, through some calculations and verifications. The JSON object always returns 168 datums that correspond to the number of combinations between the day & hour values. Besides the day & hour, the datum obviously also contains the label(s) of the correspondent stay(s) and the duration;

- *"slidermin"* - This endpoint accesses the days present in the LIFE file and returns the first day for which there are semantic notes;
- *"slidermax"* - The same as the previous, but returns the last day.

As you may note, some of our endpoints do not handle only database information such as centroids. With our architecture decision to change the focus to the LIFE file, a few resource methods also deal directly with it. For example, for the endpoints that supply our timeline slider - we can consider as a visualization, since it allows the user to check the amount of time in which he gathered his mobility and personal semantic notes and select which time interval he wishes to analyze - we simply store the days in the LIFE file and return the first and last values for each endpoint, respectively. If that design change was not made, we had to get the dates for our timeline slider using the GPX tracks. This would create data inconsistencies such as not showing events that did not have tracks associated.

In the end, even if the endpoints deal with the geographic data, information about stays or deal directly with the LIFE file, the procedures are similar. For endpoints that need database information, we open a connection to it using the Psycopg adapter. For endpoints that need the LIFE file, we just call it and use its methods, since we create a variable for the file in the beginning of the script. One of the problems with the assembling of the JSON responses was the fact that JSON does not support Date objects (in this case the Python ones). If there is some kind of manipulation on the dates, we always need to cast the Date object to a string, which does not change the information itself, but only the type. The exception is when we pass the dates directly from the LIFE file, meaning the conversions will be made on the front-end.

By using some JSON-oriented PostgreSQL functions on our queries we guarantee that all the results are correctly encoded. In the first versions of our interface we were not paying attention to this encoding issue, meaning we would get a lot of serialization errors, despite the fact the data was correct. This routing interface and its methods was one of the last things we developed, since we would only know which data was needed when the visualizations were fully designed and built. It would not make sense to create the methods without knowing what data the visualizations needed and how they would behave.

With the load testing of our solution, previously explained on section 4.1.3, we also used that data to test the scalability of both the back-end and front-end. This would show if our solution is scalable enough to handle a lifetime of semantic annotations and personal geolocation data. For the back-end, all of our endpoints correctly handled and processed the requests, returning the responses with the complete and correct data in an appropriate time interval (averaging less than half a second). Only one endpoint took more than the average time (“hexbinTracks” - about 50 MB of data in just under 10 seconds of loading time). Although considering the large amounts of data used, it is still a somewhat slow time to process. To make our back-end fully interactive it would need to get the results in under one second. Nonetheless our solution continues to be responsive, but more preprocessing and/or other algorithms (to help the hexagonal binning for instance) can be implemented. The perfection of our back-end data processing is flagged as future work. Yet, having these tests in mind and except this particular endpoint, overall we believe our back-end is still robust and scalable enough to deal with a lifetime of semantic notes and personal geolocation data. Our appraisal for the front-end is shown in the next chapter.

Chapter 5

TraceMySteps - Front-end

In this chapter we will explain our approach regarding the front-end of our solution. But first, in order to get some context on how the front-end was designed and implemented, we will discuss the technology we have chosen to do it. Then, in section 5.1 we describe the interface of our solution. In section 5.2 we describe the data manager and how it deals with requests from the visualizations and responses from the back-end. Next, in section 5.3, we describe the design of the visualizations, their features and what information they show. In section 5.4 we describe the process of connecting the visualizations between them and with the timeline slider, in order to filter the data. In section 5.5 we will analyze how scalable our front-end is, in a manner similar to that done for the back-end, which is described in the end of the previous chapter. Finally, section 5.6 describes some use cases for our application.

Due to the context of our line of investigation and the features (RESTful, etc.) we needed for our back-end, the choice of the framework to create it was rather simple. However, for the front-end the decision was not as straightforward. As we have seen in the previous chapter, we decided to use the AngularJS framework to build the front-end, yet in the beginning we were torn between AngularJS, Ember [33] and React [34]. But because it is still the industry standard framework, has bigger and better community support, and has a massive amount of add-ons (in the form of directives and modules) that can be used to enhance an application, the choice was AngularJS.

The AngularJS framework works by first reading the root HTML page, which has embedded into it additional custom tag attributes. In our case we have a single view defined in a tag. This way the framework renders our single page application inside that view. An AngularJS web-app can have multiple views (associated to different pages for example). Then, that view has its respective HTML page(s) (that in our case we use to design the interface) and its respective controller. Shortly, a controller is a Javascript object that manages the scope of the view and controls the data of the

[33] http://emberjs.com/

[34] https://facebook.github.io/react/

app. AngularJS also uses modules as containers for different parts of the application such as controllers, filters or services. Finally, AngularJS provides markers for the DOM (Document Object Model) element that tell AngularJS's HTML compiler to attach a specified behavior to that DOM element. These markers are called directives and are what allow us to create and bind the visualizations to the view's HTML, extending its functionality. We created a directive for each visualization and as we will see next, every time we click its respective button on the menu, the visualization is created and added to the interface. It is a common practice to integrate D3 into AngularJS using directives. Since we also have map-based (using Leaflet) visualizations, we analogously use directives to integrate them into the framework.

5.1 User interface

Our interface is divided in three different areas: **visualizations area**, **timeline slider area** and **menu area** (figure 37). Each of these areas has a different function and below we will analyze them. However, the interface and consequently the visualizations were not always looking like this. We first started by designing them through many low fidelity prototypes and sketches (available on Appendix C). So as to start the implementation of the interface, we first wanted to have a more concise idea about what we needed on it - the timeline slider, how the visualizations would be added, what type of data they would need, how they would behave, their tooltips, etc. Then, as explained before, we chose the framework. From there we created our interface with a design very similar to the prototypes, using the tools provided by the main libraries (AngularJS, D3 and Leaflet) and then integrating it with the back-end.

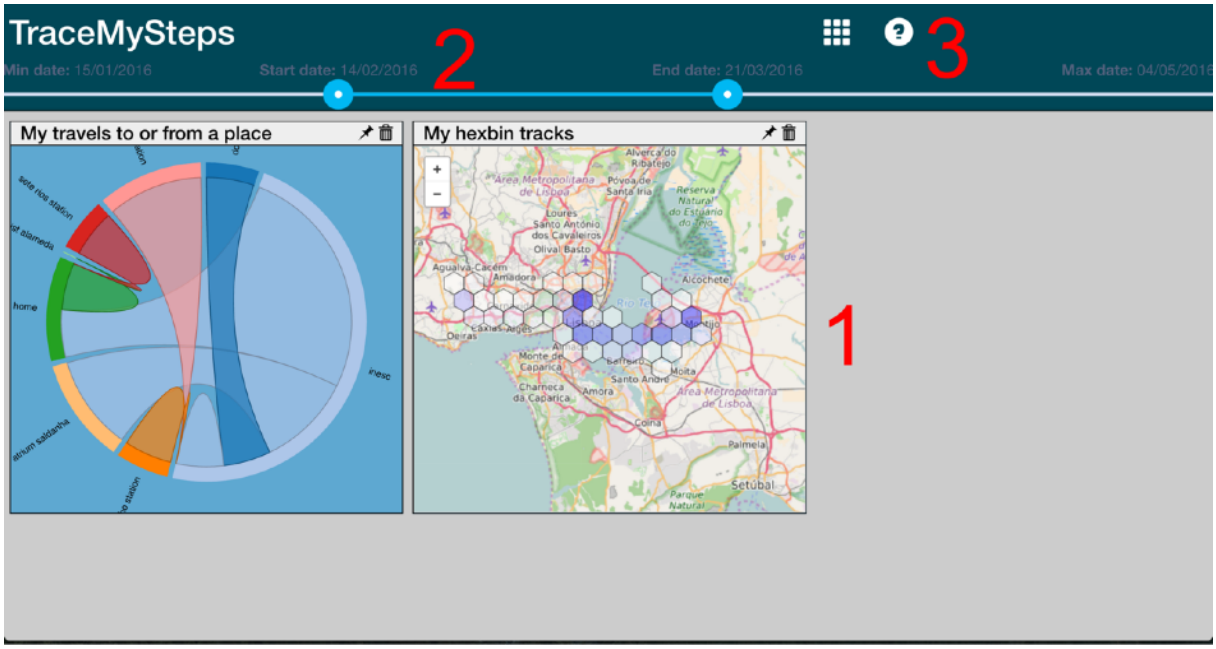


Fig.37: TraceMySteps user interface

1. **Visualizations area** - Here is where the added visualizations are drawn. Each visualization is wrapped in a widget (stylized using CSS) that contains a header with its description and other buttons to pin and delete it. Users can drag the visualization to any position on the grid, using the header of the widget. This way they can customize the grid, disposing the visualizations as best suited to them, easing the process of analyzing their data and finding patterns. This grid layout was created using a directive for AngularJS called “angular-gridster” [35]. Widgets can also be resized to a maximum or minimum size defined for each of them. Users simply need to click and drag any border of the chosen widget to the position they wish to resize. The size of the visualization inside the widget changes accordingly, maintaining the consistency of the information (resizing does not erase or change data). This grid layout obviously allows for multiple widgets to be active at the same time, with them never overlapping each other. By default we ensure that they always stack and push each other, in order to occupy empty space. This happens either when they are added using the menu or when they are re-organized by the user.

2. **Timeline slider** - The slider is used to adjust the time interval in which the user wants to observe the data. There are two handles that the user can click and drag to adjust the minimum and maximum dates of the interval. The step for the handles can vary from a day to weeks or months. Visualizations react accordingly to the changes in the slider and redraw to show the corresponding information. Some visualizations provide a more general view, showing only all-time data or data from the last year. They are more suited to this role since they are more focused on showing frequencies (of stays/visits and trips). This way, when comparing with the other visualizations affected by the timeline slider, users can for instance see if and when they stopped going to a certain place, or stopped doing a certain trip or path. Therefore visualizations like the Hexbin Places, Calendar, Stays, Trips Network and one half of the Bar Chart (that shows the more frequent places, as we will see), do not react to changes in the slider. It is useful to have these two types, because this way users can compare certain periods of their life with the general picture, thus making it easier to find relevant patterns and happenings in their personal geolocation data.

3. **Menu area** - As we can see in figure 38, our interface features two different menus, with each one associated to one button. By clicking the question mark button, the “Help” menu slides from the left. This menu explains the concept of the interface and how to perform all of its operations. By clicking the menu button, the “Add visualizations” menu slides from the right. This menu allows users to add the widgets containing the visualizations they wish to analyze. They can add as many copies of each one as they want. When many copies of one visualization are

[35] https://github.com/ManifestWebDesign/angular-gridster

present, they all react the same, consistently and with no information loss. There is also a button to clear all the widgets from the grid. Note that these menus do not push the content of the page. They overlap it, not interfering with the size of the grid and widgets. Users can still do all the operations, albeit the space is reduced when both menus are active. Ideally, a user adds the desired widget(s) and then immediately closes the respective menu. The “Help” menu acts as quick reference.

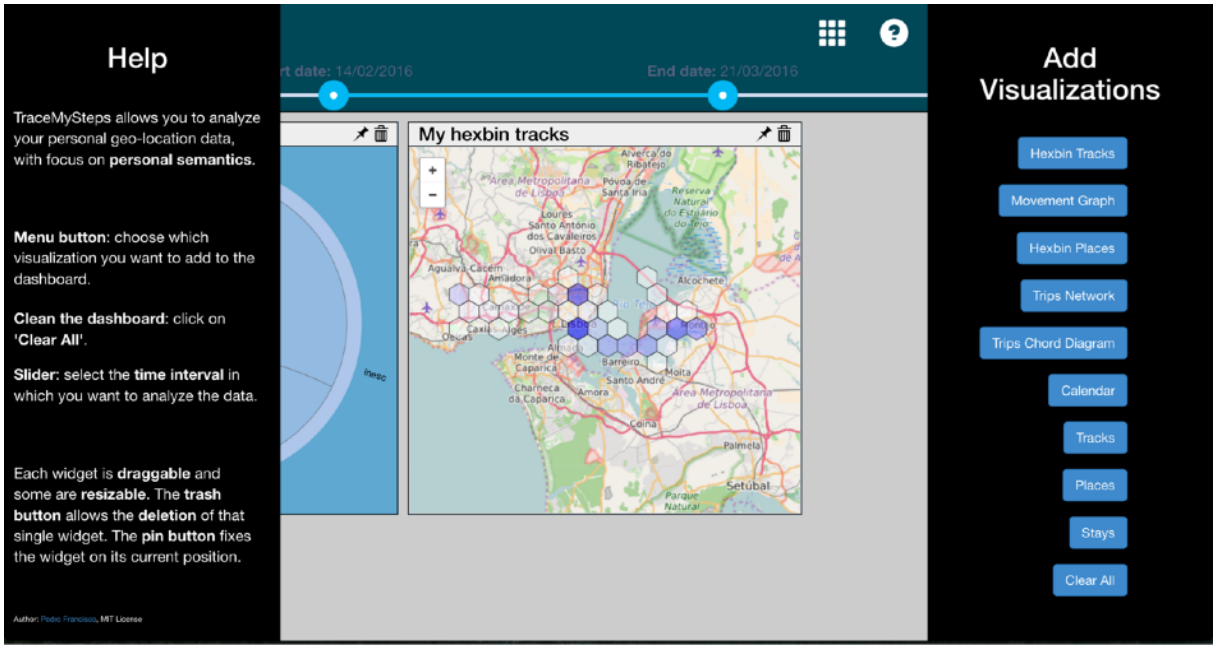


Fig.38: TraceMySteps interface menus

5.2 Data manager

The intermediate between the visualizations in our interface and the endpoints of our back-end is the data manager. It acts as a broker, receiving requests from the visualizations, gathering the data they need and sending it back to them. To achieve this, we made use of another AngularJS tool: services. In our case we use the HTTP service, one of the most common used services in AngularJS apps. The service makes a request to the server, and lets the application handle the response. So we can consider our data manager to be a service that contains extra verifications so as to correctly cache and send the data. We always thought of using a service as a manager, to get the data from the back-end and pass it to the visualizations, so the design and implementation of this component did not change significantly over time. When the visualizations receive the data, it is already prepared by the back-end (as we have seen on the previous chapter), and as a result the processing made by the front-end is reduced to the strictly necessary - the timeline slider requires data to be filtered; the connection between visualizations does not. Later, we will study that they are only visual operations made to highlight elements such places or stays.

The model explaining how the data manager operates is represented in figure 39. Each visualization (associated to a directive) contains a method that calls the data manager and sends its request. The request contains the name of the respective endpoint as a parameter. This method is called one time per visualization - only when the user decides to add it to the grid for the first time.

The data manager receives that first request, and using the name of the endpoint that came with it, connects to the respective web server endpoint URL (ex: "*localhost:*

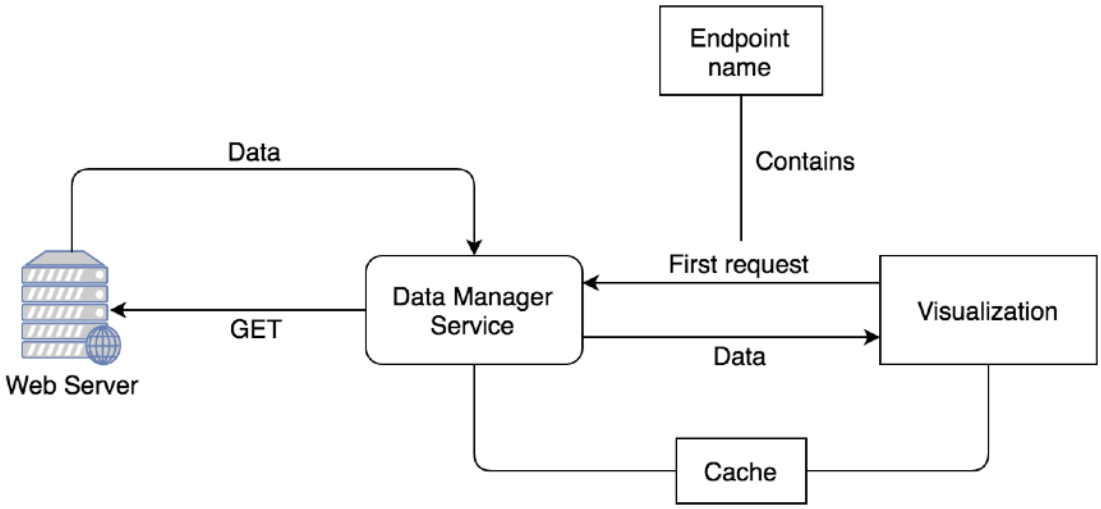


Fig.39: Data manager operation

5000/endpoint_name" - the service knows the address of *localhost*). As analyzed before, the web server will respond to the manager with the correct data, already prepared for the visualization. Lastly, that data is sent from the manager to the visualization. From there, every time a visualization is deleted from the grid and then later re-added, the cache mechanism returns the data without having to do any new requests. This is specially useful when the data needs to be filtered via the timeline slider, because we avoid doing an excessive number of requests to the server. Our front-end and caching mechanisms are scalable enough for this to happen, as we will see later when we re-approach the issue of scalability. Now the visualizations have the data they need, and thus can be drawn.

5.3 Visualizations

Implementation-wise, each visualization is contained in a directive. Together with caching and control variables/mechanisms (such as loading times: when a user adds a visualization for the first time, it has a loading period, and after that the loading is minimal, since the data is cached) all the directives have similar methods: the method to communicate with the data manager, as observed before; the methods that make a visualization connect and relate with others, where applicable; the methods that deal with the timeline slider, where applicable; and finally the essential

methods that draw the visualizations, using D3 or Leaflet. However, we will see that there are some slight changes between how the map visualizations and the D3 ones are drawn. There are nine different directives:

- Hexbin Tracks - Using a Leaflet plugin [36], this map visualization (figure 40) shows the points of a user's GPS tracks through an hexagonal binning algorithm. Shortly, this algorithm is used for grouping a dataset of N values into less than N discrete groups. There are some reasons for using hexagons instead of squares for binning a 2D surface. The most noticeable is that hexagons are more similar to circles than squares. This translates in more efficient data aggregation around the binning center. The bluer the hexagons in the visualization, the more movement that area has. From that, users can derive information such as their most frequent paths. The visualization can be panned, and also has two control buttons to zoom in and out. The hexagonal binning algorithm updates the hexagons each time the zoom level is changed, so the information is always consistent. It is one case in which the drawing of the visualization is slightly different from the ones created with D3. There are two more map visualizations, and in each one we have to use an AngularJS function to create a new element on the DOM. Then we associate a Leaflet map to that new element. Next we create a layer for that map. Finally we draw the visualization on the layer. Remember that all of this happens inside each map directive. Each of these directives also has an array for maps with different identifiers, so as to allow multiple copies of the visualization to coexist on the grid;

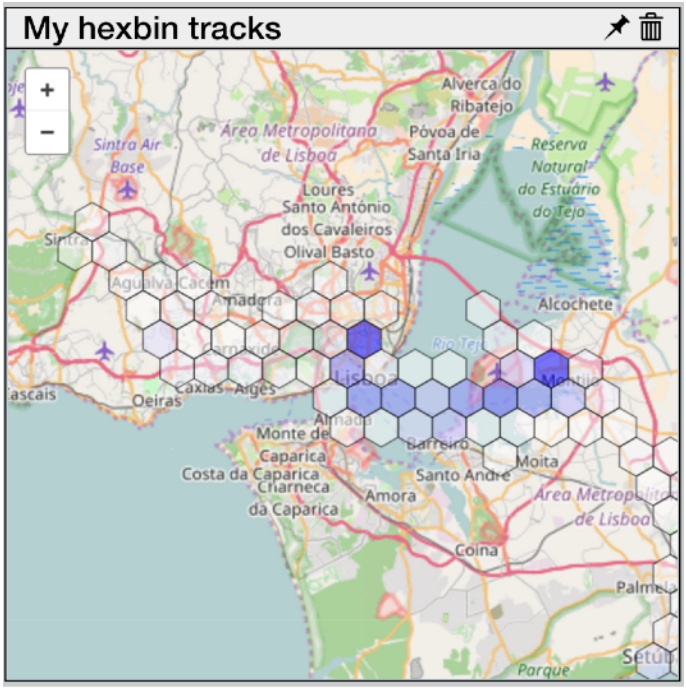


Fig.40: Hexagonal binning to show frequent paths

[36] https://github.com/Asymmetrik/leaflet-d3

- Movement Graph - This visualization shows, for a certain time period, the number of minutes a user spent moving (on trips, walking, etc.). This one was created with D3. It uses interpolation to create an area graph, instead of a plot. This way a user can clearly see the variations of his movement time. The visualization also contains a brush tool. It is useful to select a more specific time period and analyze it more thoroughly, since the X-axis of the graph changes accordingly to the brush, showing hours, days, months or years. It is a good complement to other visualizations that show data related to stays;
- Hexbin Places - It is the second map visualization. It has a behavior very similar to “Hexbin Tracks”. The difference is the type of data it shows and the presence of a tooltip. Instead of showing all the paths, it shows the centroids (as stored in the database) of the locations repeated a number of times equal to the frequency that they were visited. Then, the binning algorithm processes that information and darkens the blue color for the most frequent places. As said, this way users can observe what places they visited the most, thus complementing the “Hexbin Tracks” visualization. When a user hovers on an hexbin, and again depending on the zoom level, a tooltip pops up, showing the name of the place in question;
- Trips Network - It consists of an arc diagram that shows the all-time most frequent trips. This visualization needs to make two requests to the data manager: one to get the locations that are going to be the nodes of the diagram, and the other request is to get the actual trips, with each trip containing the frequency in which they happened. By observing figure 41, we can see the locations represented as circles. If the user hovers on them, the tooltip shows the name of that location and highlights the arcs related to it. Then, trips that go from a node on the left to one on the right are represented above the nodes. Trips from a right node to a left one are represented below. This allows to show both ways of two trips that share the same locations. The thicker the arc of a trip, the more frequent that trip is. If the user hovers on an arc, the tooltip shows the start and end locations of the trip, while also highlighting those said locations (like in figure 41).

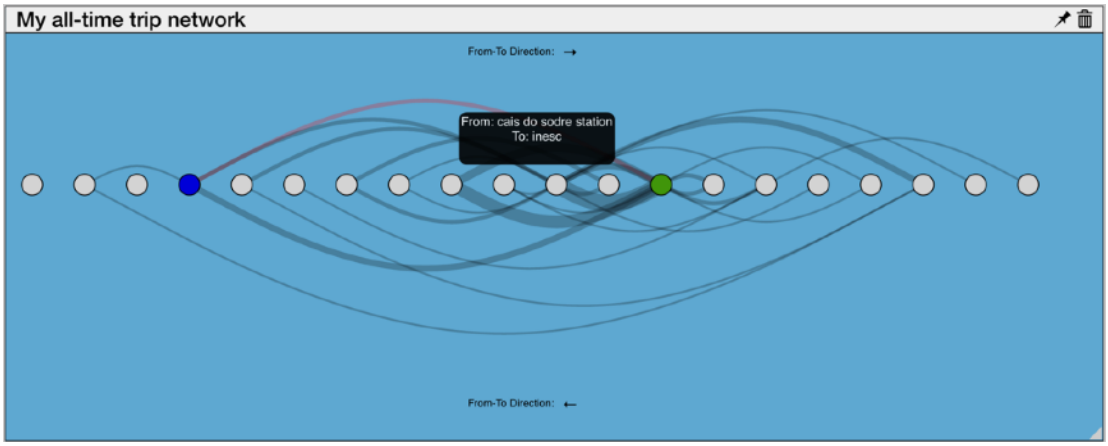


Fig.41: Trips Network

- Trips Chord Diagram - Represents all the trips made in a certain time period using a chord diagram. It is a graphical method of displaying the inter-relationships between data in a matrix. The data is arranged radially around a circle with the relationships between the points typically drawn as arcs. This means we use a D3 function to process the data and create a matrix. We pass all the trips made, with each one containing the start, end location and the date of the trip. Each arc contains the label of the location, and when a user hovers on the edge of an arc, all trips related to that arc are highlighted while the others are faded. This eases the process of analyzing trips in the diagram. Likewise, when a user hovers on a section of the arc itself (a chord), a tooltip shows the number of trips made between the two locations of that chord, in both ways.
- Calendar - Based on a D3 calendar directive [37], it represents time series in order to visualize tracked time over the last year. We adapted it so it could work with personal semantics data, be used as a grid widget and make requests to the data manager to get its data. The calendar contains four different overviews: the year overview, that shows the days of the week and months. Here, a circle represents each day of the year, and the bigger and darker the circle, the more stays that day has. On hover, a tooltip shows the places and the time spent. The second overview is monthly, when a user selects a month. It shows the weeks of the month and each day of the week. Each day contains the stays divided in rectangles. The tooltip also shows the name of the place and time spent. The third overview is weekly, when a user selects a week. For each day of that week, it shows the stays as rectangles and the same tooltip, similarly to the previous overview. The final overview is daily, and shows the places where the user spent his time, for each hour of a selected day. The tooltip shows the total time spent for the respective stay.
- Tracks - It is the last map visualization. The process of creating the map and layer is the same as the two previous map visualizations. The difference here is that we do not use the hexagonal binning algorithm. As observed in figure 42, we present each GPS track for a certain time period as a path, with the aid of a Leaflet plugin [38]. Then, if desired, we can see the trips made in certain days (using the timeline slider as we will see next). In the case of the figure, the map shows the tracks from five months. There is an additional control button, allowing the user to change from a Leaflet map to a Google Earth one.

[37] https://github.com/g1eb/angular-calendar-heatmap

[38] https://github.com/mapbox/leaflet-omnivore

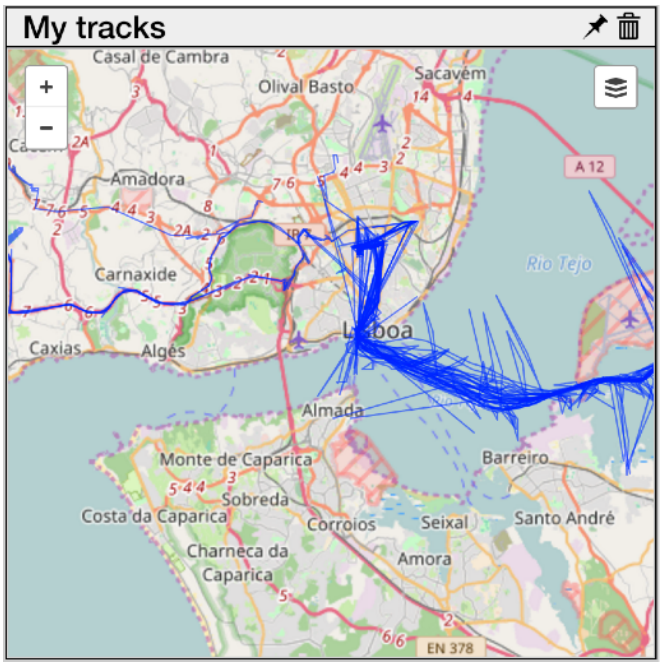


Fig. 42: GPS tracks on map

- Places - This visualization contains two bar charts. These bar charts show the information in descending order, from the most common place to the less common place. The first bar chart contains the all-time most visited places, while the second bar chart contains the places where the user spent most time, during a certain time period. As we can see in figure 43, two buttons allow the user to choose what chart they wish to analyze. We can also observe the tooltip, that for each case either shows the number of visits made to a place, or the time spent in a place, during that time period. We fit the values for the places inside a certain range, depending on the maximum and minimum ones. This happens so as to not have too many places in the chart, that could clutter the visualization and show irrelevant information. The data used for the chart in figure 43 had many more places than the ones shown, but by having a threshold the chart looks cleaner and we can clearly see the differences in the time spent.

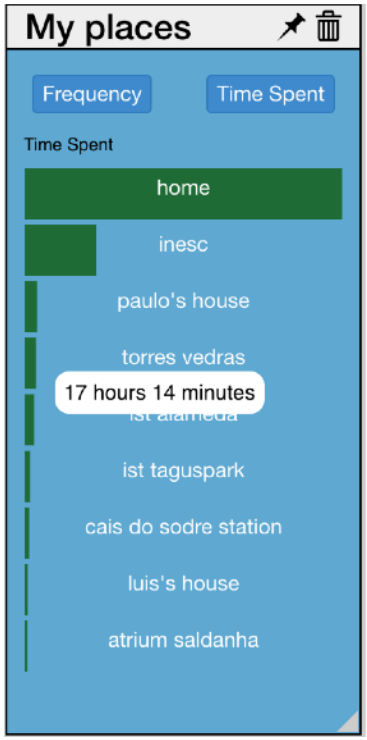


Fig. 43: Places bar chart

- Stays - It is the final visualization, and acts as an all-time calendar heatmap. It is composed by squares, and each square corresponds to a day of the week and a hour of a day. The objective is to associate stays to each square. This way, the darker the square, the more and/or longer stays that day and hour had, through the course of time. On hover, each square shows the place or places where the stays

happened. As an example, by looking at figure 44, on Monday from midnight to 8 AM the squares are colored with a dark blue. Probably because the user was always at home sleeping at that specific time. And although it is not in figure 44, the tooltip for those squares confirms the user was mainly at home. Other example, Wednesday from 9 to 10 AM, the squares are light colored, meaning the stays were very short. Probably the user was moving from home to his workplace. And again, the tooltips attest it. So, this heatmap helps the quick detection of patterns of stays.

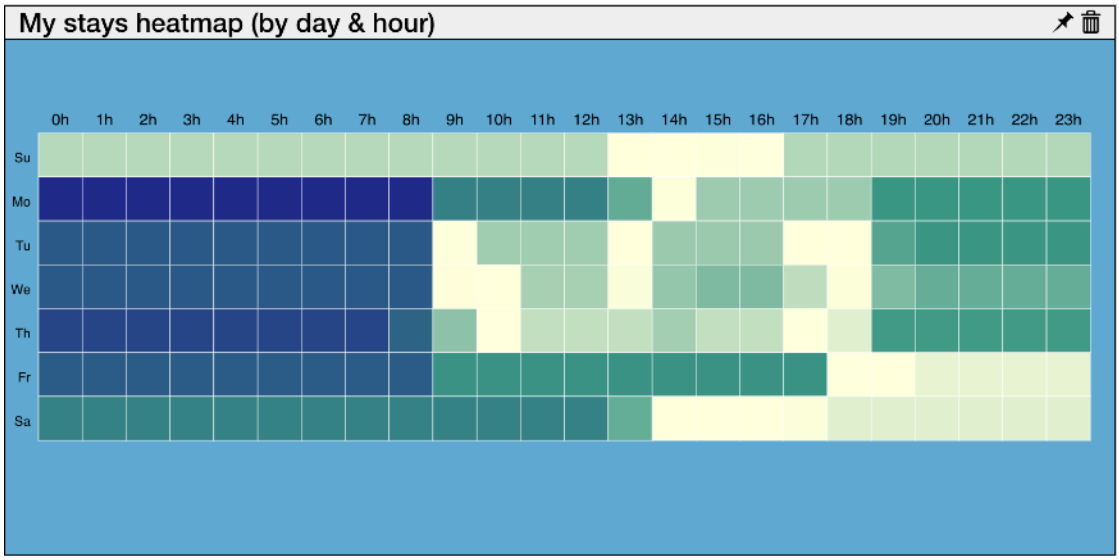


Fig.44: Stays heatmap

5.4 Connecting the visualizations

As the user places the desired widgets on the grid, and analyzes each one by dragging, panning, using the tooltips, etc., the information they show needs to be consistent between each other. For instance, if we have a Trips Chord Diagram and a Hexbin Places active on the grid, and hover on a location in the Chord Diagram, besides highlighting in the Chord Diagram as we seen in the previous section, it would make sense to highlight the corresponding place in the Hexbin Places map. By linking the visualizations when they are on the grid and making them exchange information, users can understand the various aspects surrounding their data such as the location associated to a place, the time spent there or the number of visits made. This also makes our interface more dynamic and slick, since the actions a user make on one visualization can have repercussions on other visualizations.

Figure 45 illustrates the previous example, together with the Places visualization. We can see in the figure that the user selected “*inesc*” in the Chord Diagram. When this happened, besides the normal behavior of the diagram, the map automatically zoomed and centered on the location of “*inesc*”. The Places visualization also highlighted the corresponding label. From here, the user can observe how that specific location positions itself in the other visualizations, allowing for quick

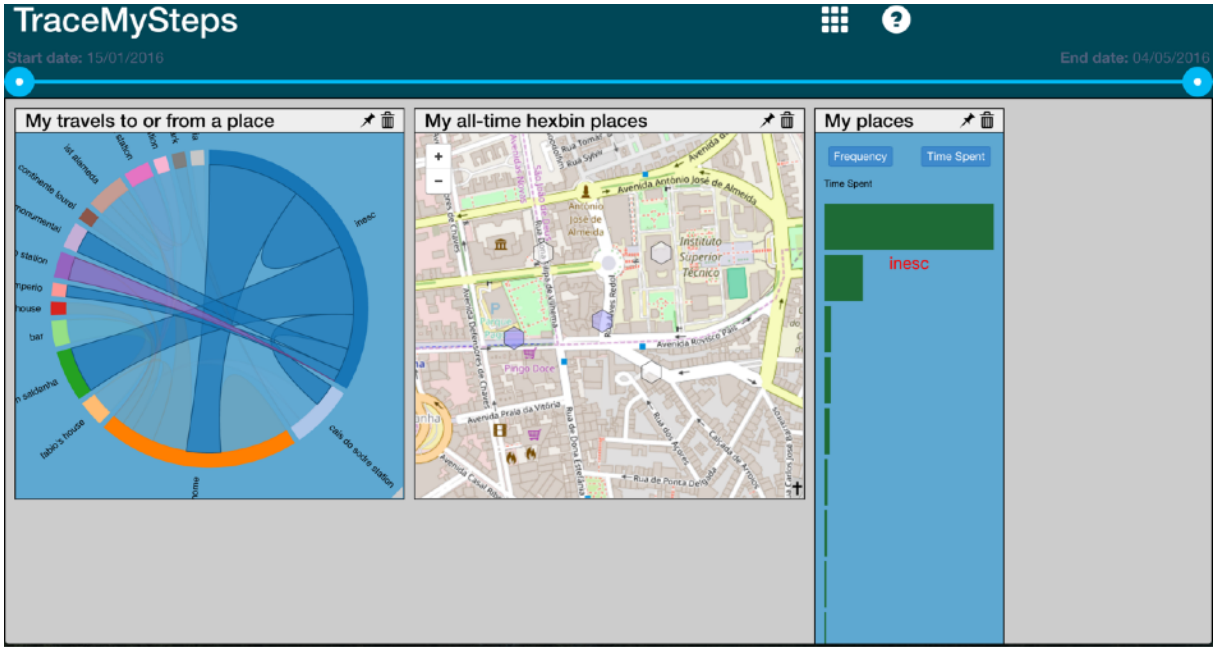


Fig.45: Three linked visualizations

comparison and pattern finding. These links exist between almost all possible combinations of visualizations. We left some out, like the Hexbin Tracks, the Stays, the Tracks and the Movement Graph. We decided to do that because the tracks visualizations only react to the timeline slider and the Stays is a more general one, showing all-time data. This way we keep the customizable aspect of our interface together with the ability to always compare different time periods.

The communication between the visualizations is possible thanks to another AngularJS tool: broadcasts. This broadcast mechanism tinkers with the scopes present in the application. In the case of the broadcast operation, it is the root scope of the application that deals with it. Every AngularJS application has a single root scope. All other scopes are descendant scopes of the root scope. Scopes provide separation between the model and the view, via a mechanism for watching the model for changes. They also provide event emission/broadcast and subscription facility. So, this tool provides a solution to create the bus between the visualizations, as we stated before. And it is rather easier than creating a new observer design pattern, since this tool acts as an event listener and dispatcher. The design decision to use this tool happened in the first stages of the development, when we were looking for solutions to have the said bus. Root scope broadcasting is especially useful to send events when we want to reach a scope that is not a direct parent - it is exactly our case, as it is between the child scopes of each directive. We found this tool to be much more easy to implement for all the visualizations and more suited to the type of data and communication we wanted to do, than to implement an Observer Pattern. That implementation would be from scratch, with a lot of testing. Broadcast offers a service similar to the Observer, but more adequate since it is a framework tool.

This mechanism is straightforward: when we call each directive (that contains each visualization), each one is created with its own scope. These scopes inherit the root scope of our application. So, each directive scope is a child scope. Hence, changes made on the root scope will be reflected in its children (the directives). Therefore, to pass the data between visualizations, we need to call the root scope in the directive. Using figure 45 again as an example, when a user hovers on the location in the Chord Diagram, we call the root scope and its broadcast method on the diagram. That method dispatches two parameters: a string identifying the specific directive broadcast and the data we wish to pass, wrapped as a JSON object. In that case we send the label of the place. Then, the two other visualizations contain a listener for the Chord Diagram, and when they get notified that the Chord Diagram sent information to them via the root scope, they gather the information and do a series of visual operations in order to highlight the corresponding data in the different visualizations. When a widget is removed, it means the directive is also removed and its scope is deleted. We use a special listener for that event, in order to erase the broadcast listeners and avoid memory leaks. It would not make sense to have a removed visualization listening for broadcasts. However, as we will see next, the timeline slider requires some of them to keep listening for dataset changes in the background, so as to save state. This process is the same for all linked visualizations, with several broadcasts and listeners between them.

5.4.1 Timeline slider

It can be considered one visualization, although of small proportions, because it allows users to check the time period in which they are analyzing their data. It was one of the last components we implemented, since it is based on the broadcast system that links the visualizations. We first wanted to have the visualizations properly linked so that for the timeline slider the implementation could be simpler. The slider is based on an AngularJS plugin [39]. Its creation and operation is not done in a directive, but in the application controller. In the controller, the root scope can still operate and send information to child scopes. We adapted it to process and show dates instead of numbers. When the user ends the slider dragging, the root scope broadcasts the minimum and maximum date of the time period, together with its identifier. Figure 46 shows the timeline slider in action. The user selected a time period of a month, and then the Chord Diagram, Hexbin Tracks and Tracks visualizations redrew, in order to show the data for that month. The redrawing process for the D3 visualizations is simple, with the typical call for the draw function, with a different dataset. For the map visualizations we need to make sure the layers are removed. The map and DOM elements remain. With the layers removed, new ones are added with the new set of tracks and hexbins for that time period.

[39] https://github.com/angular-slider/angularjs-slider

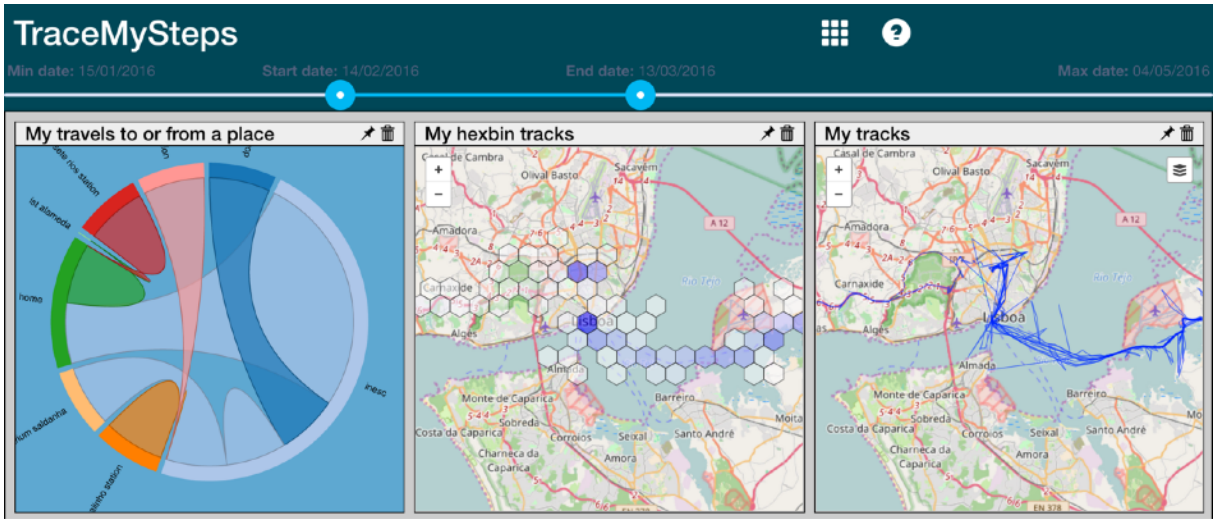


Fig.46: Observing data from the 14th of February to the 13th of March

The first time visualizations are added to the grid, they start listening to the changes in the slider and react accordingly. As said, this reaction is to redraw and show the information of a certain time period. When a visualization is deleted and is not on the grid we do not delete its timeline slider listener. This way each one saves state and updates its dataset in the background, always keeping up with the slider. When a user re-adds the visualization, it already shows the information consistent with the time period. Before this solution was implemented, in some final versions of the prototype we were always deleting the timeline listener. This led to disparities between the time period and the information shown, when we deleted a visualization, changed the slider and then re-added it later.

5.5 Front-end scalability

As we witnessed on Chapter 4, when we load tested our solution with a big dataset, we used that dataset to test the scalability of our back-end. To test the scalability of our front-end, we used two datasets: the same dataset from the load testing, and a smaller one. Recapitulating, before the simplification algorithm, that big dataset contains 2812 GPX files, where each year averages 563 files and 123 MB of data. We also assume the collection of about two to three tracks a day. These tracks totalled 608.3 MB of information and contained 4392048 points. We also had to replicate our LIFE file, leading to the creation of 1073 different locations. This amount of data simulates circa 5 years of mobility and personal annotations, as disclosed before. But first we used the smaller dataset, the same that will be used in the evaluation - five months of mobility tracking and semantic notes, collected by the author.

For this smaller dataset, a response containing the data for a visualization averaged a collection time of 90 milliseconds, which allowed for very small loading and drawing

times. For our front-end, this means full interactivity and responsiveness. There is no visual clutter or missing information in the majority of the visualizations. Minor clutter is only found in the Trips Network, with some arcs overlapping. The other components, such as the timeline or the link between the visualizations, operate without issues. For small datasets we can conclude that our system is very scalable.

Regarding the bigger dataset, the loading times for the visualizations averaged a collection time of 800 milliseconds - just under one second. The drawing time does not rise much, but data loading times are bigger, especially for the Hexbin Tracks visualization. This affects the interactivity a bit. Still, full responsiveness in the drawing, and in other actions such as resizing, is ensured. The timeline slider works without any problem. The visualizations are still correctly linked, as they pass the correct information between them and highlight everything accurately and quickly. Regarding the drawing of the visualizations, eight out of nine adequately scale and show information without cluttering the widget. The Tracks visualization is somewhat slower in the display the GPS tracks, but the speed is acceptable given the vast number of tracks. The Trips Network is again the only one that does not scale very well. Although it draws and uses the correct information, the space it uses is bigger than the space its widget provides. This led to several hidden nodes and arcs. Nonetheless, the results are satisfying given the big amount of data. Albeit more processing can be made on the back-end, as shown in Chapter 4, for big datasets we still consider our front-end to be as robust and scalable as our back-end.

5.6 Use cases

To understand how our system works and to show its customisation potential, we present three use cases and for each one we show how would a user normally proceed to find the desired information. We will use the small dataset from the previous section. The first use case is: "*Find the place I have stayed the most during the month of April*". The second is: "*Find my usual Wednesday mobility pattern*". The third use case is: "*Observe the tracks for the 26th of January 2016, finding what trips were made inside Lisbon that day*".

For the first use case, we quickly observe that it asks for a place. We also observe that we need to compare the time spent between several places. Thus, the most suited visualization for this is Places. Inside it we click the "Time Spent" button, in order to see information about stays. Then, we observe that we need to analyze data from a specific month. So we need to filter the data using the timeline slider. We adjust the slider to show information from the first day to the last day of April. After adjusting the slider, the visualization shows the data we wanted: during the month of April I have stayed the most at "*home*". By hovering on the bar, we see that during the month of April, I spent 47 hours and 35 minutes at home. If we add a Hexbin Places visualization and then click on the bar, we can see the location of "*home*" in

map, as additional information. Figures 47 and 48 show some steps a user would take and the final result, with the timeline selected for the month of April, the tooltip (on hover) showing the time spent there, and the selected bar (when clicked it turns black) requesting the Hexbin Map visualization to show the location of "*home*" - a case of linked visualizations.

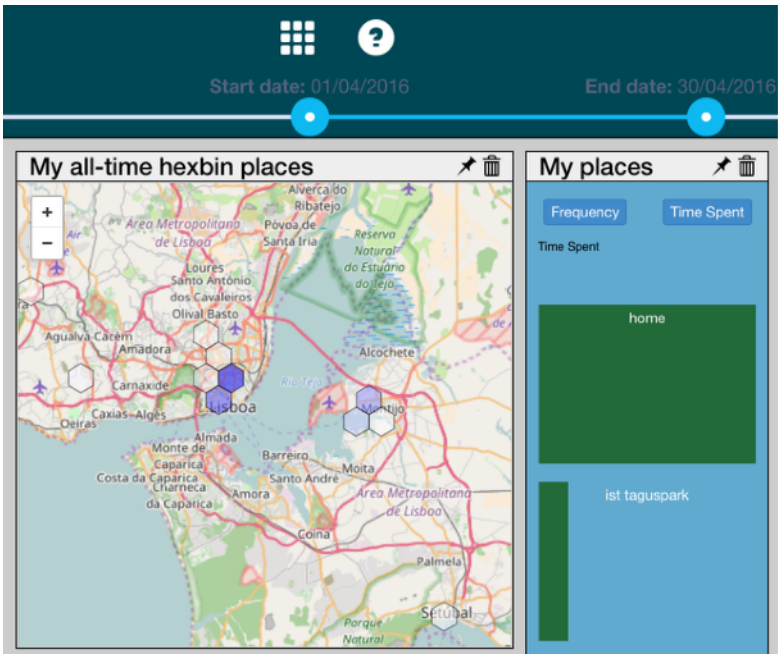


Fig.47: Adding a Hexbin Places in the first use case

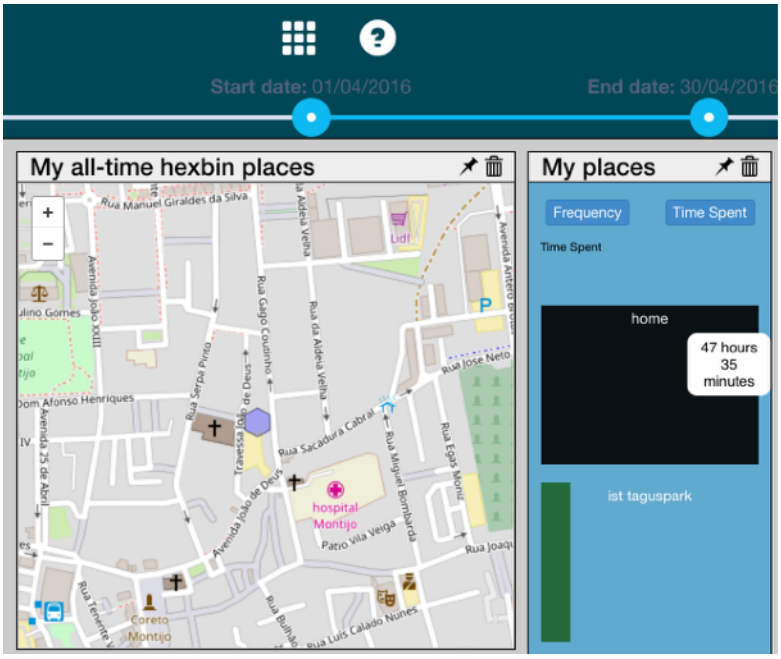


Fig.48: Linked visualizations after clicking the bar in the first use case

The second use case asks for a pattern, in a specific weekday. The first thought would be to use the Calendar visualization, but that is not very suited to observe patterns, it is more adequate to analyze specific stays during a certain day. To observe patterns regarding weekdays we have the Stays visualization. Figure 48 helps describe the correct pattern: as we can see, on hover, the location I mainly stay on Wednesdays at 7am is "*home*". At 9 and 10am the squares are brighter - this shows a period of movement. If we hover we see that at that time I briefly stay on a station, waiting for transportation. Then the squares darken until lunch time - I stayed in "*inesc*". After lunch time on "*atrium saldanha*", I stayed again in "*inesc*". Finally, around 6pm I use transportation again (brighter squares), to return to "*home*".

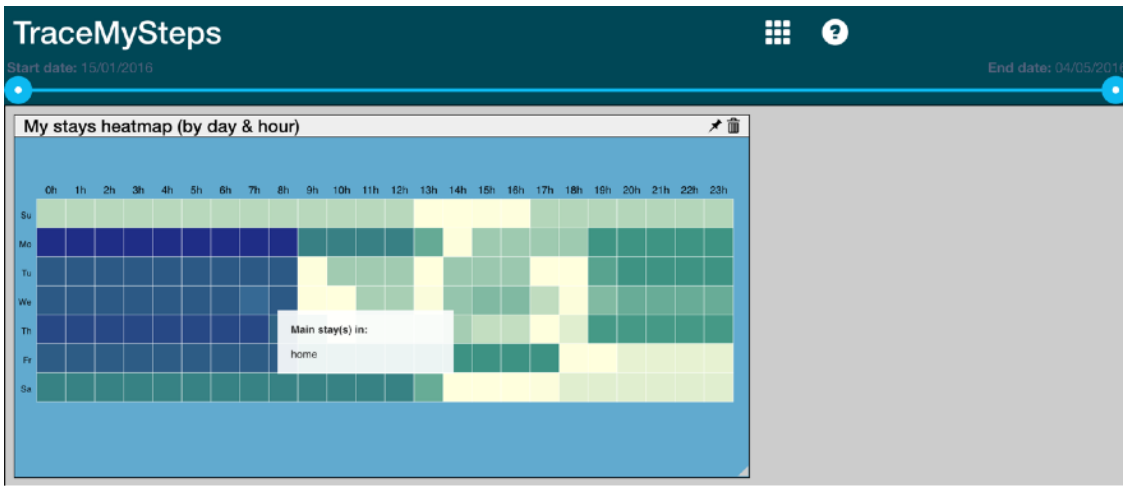


Fig.49: Depiction of the second use case

For the third use case we observe that we need both tracks and trips for a certain period of time. For that, we add the Tracks visualization and the Trips Chord Diagram, as they react to the timeline slider. Then we use the timeline slider to filter the data to the one desired. So we need to observe the trips made inside Lisbon on that day. By zooming in we can see the respective tracks (figure 50). Next we need to analyze which places are associated with those trips: we start by adding a Hexbin Places visualization; then, with the linked visualizations feature, we use the Chord Diagram to observe where the locations of the trips are placed on map; we see that three locations are inside Lisbon. These locations are "*inesc*", "*cafe imperio*" and "*cais do sodre station*". Finally, as we can see in figure 51, we use the Chord Diagram to show the trips made between those three places, inside Lisbon. These are complemented with the respective tracks, showing the paths in the Tracks' map.

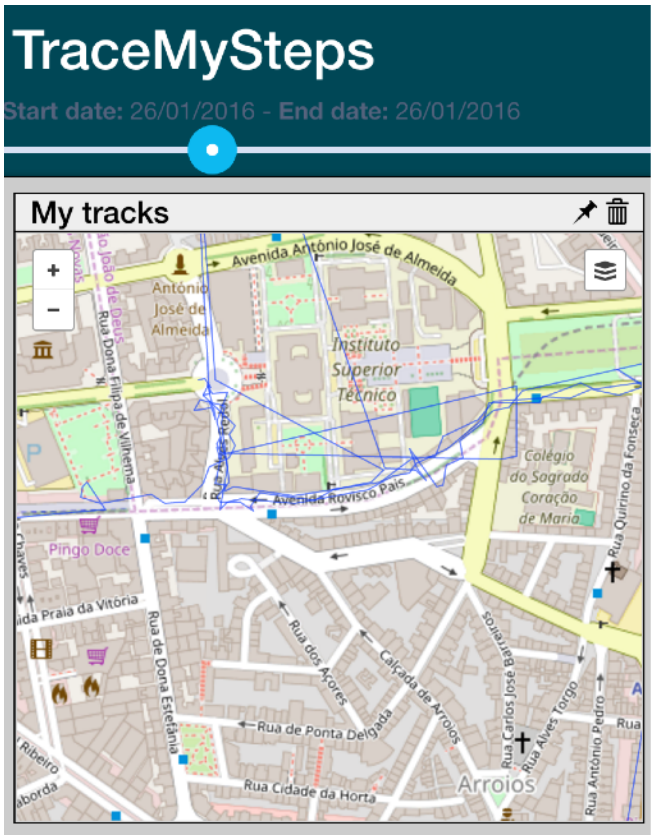


Fig.50: Zooming in the tracks for the third use case

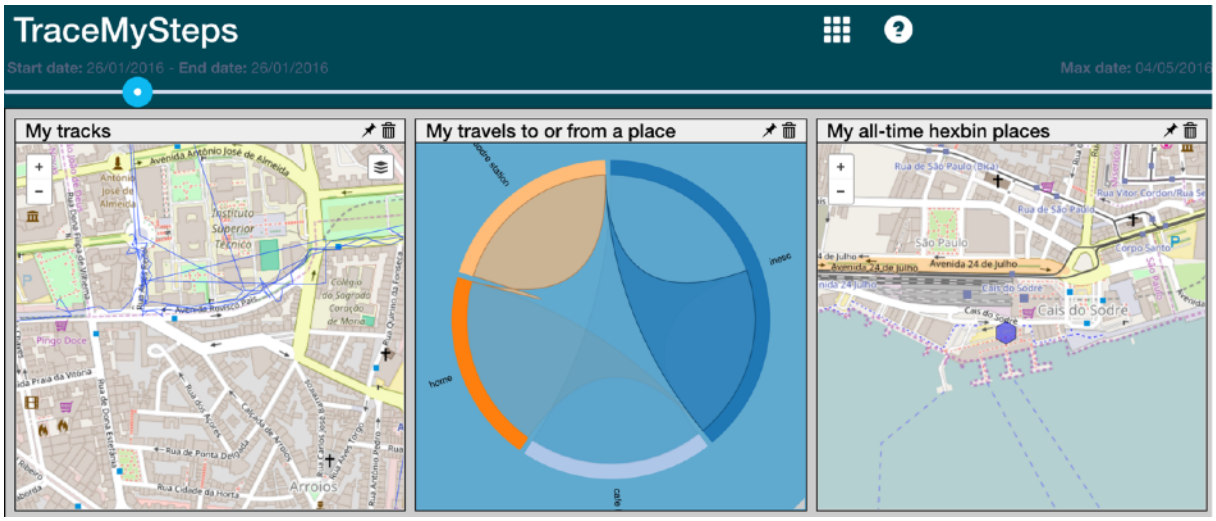


Fig.51: Depiction of the third use case

Chapter 6

Evaluation

In this chapter we will evaluate our system. This way we can assess if our solution reaches the objective we have set. This objective is stated in the beginning of this document. The interest here lays in evaluating how users deal with the solution as a whole, while having in mind components like the menus and the widgets that contain the visualizations, how these visualizations relate between them, and the timeline slider. We also intended to do a heuristic evaluation, but due to time constraints we were not able to do it. In the upcoming sections of this chapter all the data about users, tasks, questionnaires, protocols and results will be scrutinized.

6.1 Experimental protocol

With the user evaluation, we wanted to observe if users could understand and correctly use our application for its intended purpose: analyze spatio-temporal information, with focus on personal semantics. Twenty users were part of this evaluation. We held a session for each single user, and each session was composed by four different stages:

1. Initial questionnaire to trace the basic profile of the testers - age, gender, studies, etc. (Appendix B.1). The results are shown in section 6.2.1;
2. We explain and demonstrate the system for about five to ten minutes. Then the user has an additional five minutes to freely explore the interface, get comfortable, and lower the number of errors and execution times for the first tasks;
3. A set of tasks, presented to the user in a given order. That order varies for each user. These tasks consist on actions to be performed on our solution, covering its main components;
4. Final questionnaire to assess usability and global feeling of our solution (Appendix B.2). The first part of this questionnaire is based on the System Usability Scale

(SUS) [40]. The second part is aimed at user satisfaction and has additional domain specific questions to know if users are willing to collect their spatiotemporal data.

We used the same dataset for all users: five months of mobility tracking and semantic notes, collected by the author. The tasks are divided in four different groups, related to components of the solution:

- Evaluating the menus and widgets - how users deal with actions such as adding and removing widgets (that obviously contain visualizations). Additionally, we will focus on how they deal with searching information in individual visualizations;
- Evaluating the connected visualizations - with these tasks we will assess if users can find certain information in certain visualizations, with the help of other visualizations as means to highlight and filter to them what they need. This way we can observe if visualizations are correctly connected and complement each other;
- Evaluating the timeline slider - tasks that ask users to filter the data using the timeline slider of the interface, in order to point information about a specific day;
- Evaluate overall use of the solution - a final, more broad task, in which users have no clear indication of what component of the interface to use, but are still asked to find certain information. It is a task aimed to check if users can find specific information in the interface as a whole, after being somewhat familiarized to it.

For these tests we decided to use the following metrics to collect data: task duration time, completion within an expected time interval (each task has a maximum convenient time for its completion; we have to compare if the task duration time is within this set interval), number of errors, number of clicks, correct answers and other relevant annotations.

6.1.1 Tasks

The task set (Appendix A) consists of 16 different tasks, divided in the aforementioned four groups. An example of a task related to the menus and widgets is "*Remove the Places and Calendar widgets*". Related to the connected visualizations, an example can be "*Using the Chord Diagram, find and point us inesc in the Trips Network*". Another example, this time related to the timeline slider is "*Set the slider date to the 8th of March and explore the tracks for that day*". Besides collecting data regarding the metrics for the tasks, we asked users to express their doubts, concerns and obstacles (if there were any) at the end of each task. This allowed us to understand if they were making any progress and kept the users focused on performing the tasks. Firstly we thought about asking users to think out loud during the execution of the tasks, but we came to the conclusion that this constant feedback could mess with metrics such as the duration time of each task. It

[40] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

could also distract users from the process of executing the task. We also let the users take a short break in the middle of the tasks.

6.1.2 Final questionnaire

After each user has completed his task set, a second questionnaire was made. The objective was to assess usability, global feeling of our solution, user satisfaction and also ask additional domain specific questions, together with a request for some comments and suggestions as feedback for future work. As said before, the questionnaire was divided in two parts. In the first part the system usability is evaluated using the SUS methodology and in the second part there are some additional domain specific questions.

The benefits of SUS is that it is a very easy scale to administer to participants, it can be used on small sample sizes with reliable results, and it can effectively differentiate between usable and unusable systems.

Scores for SUS range from 0 to 100, and so we have to understand which range can be considered a good result. Based on research, a SUS score above a 68 would be considered above average and anything below 68 is below average. With a score of 90 or more, a system is considered to be exceptional.

For the second part we wanted to understand if users were willing to collect their spatio-temporal data, and in the future use it in our application. That is the reason why we asked questions related to smartphone and GPS familiarity/usage such as if they used the GPS capabilities of their phones. If users were not willing to track their daily spatio-temporal data, we asked what reasons prevent them from doing it (privacy, battery, etc.). Using a linear scale, we also asked users about the ease of use and understanding of the solution and its components.

6.2 Results

Here we present the results from the task execution and from both questionnaires.

6.2.1 User profiling results

In the set of 20 people, 75% were male and 25% were female. 65% were aged between 18 and 25 years old, 15% between 26 and 35 years old, 10% between 36 and 50 years old and 10% over 51 years old. About their academic studies, 60% had a background in Science/Engineering, 25% answered “Other” and the rest (15%) had a Humanities and Social Sciences background. About their academic degree, 45% had a Bachelor’s Degree, 20% had a Master’s Degree and the rest (35%) finished High School.

6.2.2 Results by task

Table 4 shows the results for each task, based on the metrics we collected. The first task stated "*Add a Places widget, a Calendar widget, a Stays widget and a Tracks widget*". On average this task took 13 seconds. And in this case, for a confidence interval with a 95% level, the range for the true mean is 11.55 to 15.05. These values are inside the convenient target time we considered for the task. Along with a suitable number of clicks and low number of errors, we can say that this task had a high success rate. This range includes the opening and closing of the menu, together with the loading times for those visualizations.

The second task, "*In the Places widget, find and tell us the name of the place with the second highest time spent and its value*" still had an acceptable percentage of users completing it within the expected target time. Delays were caused by mis-clicks, and users going back and forth from the "Frequency" area to the "Time Spent" area, in order to answer. The percentage of correct answers was high, with the wrong answers coming from the fact some users did not select "Time Spent" and instead answered with the frequency of visit for the place. All the other metrics have values inside the ranges we expected, showing this task was carried out successfully.

The third task, "*In the Calendar, search and tell us the name of the places I have been on Tuesday, in the second week of February*" had a high success rate in both the correct answer and completion within the target time, since it is a more guided task - we say where to go and what to observe (calendar, February, second week). A lower than expected average of clicks shows the user did not have the need to go further into the weekly view (two clicks were needed to enter the weekly view) to observe the correct places.

The fourth task, "*Remove the Places and Calendar widgets*" had a lower completion rate within the target time and an average of clicks (3.5) bigger than the expected two clicks (one to remove each widget). This is because the icon for the widget removal is not very well placed in the widget header, and when users try to click it, sometimes it instead resizes the widget or drags it.

The fifth task, "*Add a Trips Chord Diagram widget*" was completed by all users within the expected limit, since it is a rather easy task. Its errors were based on the fact that some users did not close the menu after adding the widget.

The sixth task, "*In the Chord Diagram, find and tell us the number of trips made between inesc and cais do sodre station*" had a correct answer rate of 100%, showing that users could find the correct answer. However, the completion rate within the expected target time was just above average, with a rate of 55%. This is explained by the fact that the tooltip of the chord diagram takes some time to draw and needs the cursor to be completely still, leading to some errors and an unexpected amount of time needed to see that specific number of trips made.

The seventh task, "*Add a Hexbin Places widget and then use the Chord Diagram to find and tell us the exact location of cais do sodre station*" had a complete success rate regarding the correctness of the answer. The majority of the users also completed it within the expected interval. This is mainly due to the fact that linked visualizations respond quickly to user actions, and so when a user hovers on a location in the Chord Diagram, the Hexbin Places shows the location immediately.

The eight task, "*Use a new Places widget to find and point us where is my home*" also had a high success rate for both answer correctness and completion within the target time. Although in this visualization users have to click to select the desired location, it did not affect the task metrics. The success here is explained using the same reason as the previous task.

The ninth task, "*Add a Trips Network widget*" had no problems to be completed within the target time, although a common error delayed its completion for some users: when they added the widget, the grid was already filled with other widgets, so the Trips Network would be drawn in the bottom part of the grid. So they had to scroll to confirm it was added. Some added it twice as some sort of confirmation.

The tenth task, "*Using the Chord Diagram, find and point us inesc in the Trips Network*" had a high success rate for answer correctness, again due to the quickness in which the linked visualizations show the desired data. Some errors were because users mis-hovered on other locations. However, the rate for completion within the target time was lower because the widgets were far apart on the grid, leading to some seconds spent on scrolling or clicking, in order to find the correct location.

The eleventh task, "*In the Trips Network, find and tell us the most frequent trip containing inesc*" had a lower than average answer correctness rate, with the biggest average error of these tests. This is explained by its low scalability: some of the arcs are too close together, and users could not discern which arc showed the most frequent trip. They mostly answered the second or third most frequent trips. Other errors were due to users observing trips regarding other locations than the one asked. However, the majority answered within the expected target time.

The twelfth task, "*Drag the Tracks, Chord Diagram and Places widgets so as to put the three together*" was mostly completed within its interval, with some errors and delays due to users resizing the widgets instead of dragging, or when they were dragging one widget close to the other, it would send the other down in the grid.

The thirteenth task, "*Set the slider date to the 8th of March and explore the tracks for that day*" was always completed within the target time, given the simplicity of the timeline slider. The standard deviation in the time taken is big here, since some users explored the tracks with few clicks and during a small amount of time, while other did a more thorough search, taking more time and clicks.

The fourteenth task, "*Find and point us how much time I spent at home on that day*" makes use of the timeline slider and all users answered right. The fact that the Places widget was already on the "Time Spent" mode helped, although some users took more time and clicks because they did not notice that. Also, the visualizations respond quickly to the timeline slider changes, which definitely helped.

The fifteenth task, "*Remove all remaining widgets using the "Clear All" option*" was completed without any problems, since it is just a click in a menu button. Very few users did not complete it on time, but it was for a matter of one to two seconds. More familiarization with the interface would mean a full completion rate within the interval.

The sixteenth task, "*Find and tell us the day in which I spent the most time in a single place, and what was that place*" was a more broad task, not guiding the users to specific visualizations. The answer would be to go to the Calendar, since we are asking for a specific day, and not a weekday for instance. The completion rate within the interval was satisfactory, since this task involved further exploration: the majority of the users started to search on the Places visualization, then went to the Stays visualization. Some answered wrongly here because it was on the Calendar that we wanted an answer. Others understood they needed to point out a specific day and then went to the Calendar. Here, some of them answered what we were expecting while others answered with another possible day - comments were that the color scale in the Calendar was a bit deceiving. There are circles with very similar colors. And because of that we accepted a second answer, in which the stay differs from the stay in the first answer by a matter of minutes. This happens probably because the dataset is not too big to have a distinct scale.

Task	avg clicks	stdev click	avg time (s)	stdev time (s)	avg error	stdev error	Correct answer (%)	Within interval (%)
1	6.7	1	13.3	4	0.3	0.4	NA	90
2	1.5	1.3	8.8	3.7	0.5	0.7	85	80
3	1.8	0.9	15.1	6.6	0.4	0.6	95	90
4	3.5	1.1	6.7	2.4	0.4	0.5	NA	65
5	2.8	0.4	4.3	1.3	0.3	0.5	NA	100
6	1.5	1.4	14.9	6.4	0.6	0.6	100	55
7	3.4	0.6	15.8	4.2	0.4	0.6	100	95
8	4.2	0.6	16	4.3	0.3	0.5	100	90
9	3.3	0.4	5.6	1.4	0.2	0.4	NA	85
10	0.9	1	8.9	2.6	0.3	0.6	90	80
11	1	0.9	14.4	6.9	1.5	1.3	45	70
12	3.7	1.1	13.6	7.6	0.6	0.9	NA	80
13	9.4	3.8	24.4	9.6	0.2	0.4	NA	100
14	0.3	0.5	9.6	3.4	0.4	0.8	100	80
15	2.7	0.5	5.9	1.4	0	0	NA	90
16	7.7	2.8	98.3	31.1	1	1	50	65

Table 4: Results by task

6.2.3 Results by component

Here we analyze the results for each component of our solution.

- Menus and widgets - Some users had problems with the removal of the widgets - task 4 had an average of clicks higher than expected, with some users not completing the task within the expected time. As stated before, the remove button when clicked can be confused with the dragging or resizing, leading to some errors. Also, there were some comments that the text in the header of each widget should match the name of the respective button, so that users could understand more quickly which widgets were on the grid. The closing of the visualizations menu also had some comments, with suggestions to add another button inside the menu, so as to close it. Apart from these problems, users could correctly customize the grid, operate the menus and smoothly add the correct widgets.
- Connected visualizations - Users did not have any serious problems observing the data between the different linked visualizations. We can conclude this by observing that for the tasks that deal with connected visualizations (6, 7, 8, 10, 11), the number of correct answers is high. Also, most users completed the tasks within the expected target time, with few errors. They all responded quickly to user actions, highlighting the correct data between them. The non-existent loading times allow for immediate identification of the data. As we have seen, the problems came when the user was observing the Trips Network visualization. This visualization is the least scalable and as future work we point to the improvement of its scalability.
- Timeline slider - Regarding this component there were not any problems, since users found the timeline easy to use and the visualizations also responded quickly to the temporal filter - tasks 13 and 14 had a low amount of errors, a high percentage of correct answers and a high percentage of completion within expected target time. They could correctly explore the timeline-filtered data.
- Overall use - Users had some difficulties with the color scale in the Calendar view, as the lower percentage of correct answers and completion within target time shows for task 16. Also, sometimes when they were scrolling it would accidentally zoom in or out on a map, if the mouse was hovering it. Besides that, users found the interface to be working efficiently, with no excessive loading times of freezes.

6.2.4 Final questionnaire results

After the completion of the task set by the users and after we have taken our metrics and annotations, we asked the users to answer a final questionnaire. As seen before, the questionnaire contains two parts: the first one regarding the System Usability Scale, and the second part with domain specific questions.

Table 5, in Appendix E, details the SUS scores for each user. Figure 52 shows a box plot with the classification for each question of the System Usability Scale. For the first question, the plot shows users are open to use the system frequently. In question 2, users find the system to be of low complexity, although an outlier thinks it is a bit more complex. Question 3 shows users think our system is easy to use. Regarding question 4, a low number of users think they would need the help of a technical person, in order to use our system. Question 5 has more disparate results, but the majority of people find the system to be well integrated. For question 6, few people found inconsistencies in the system. For question 7, the majority of people find the system to be quick to learn. Almost no people found the system cumbersome to use, according to the results of question 8. Results for question 9 show that people are rather confident using our system. Finally, results for question 10 show that the majority of users did not have to learn a lot of things before using our system. In the end, our overall score is 75.5 - above average. This means our system is good and we are on the right track. With minor improvements (detailed in the next chapter), the score of our system will be even higher. This score does not reflect a percentage.

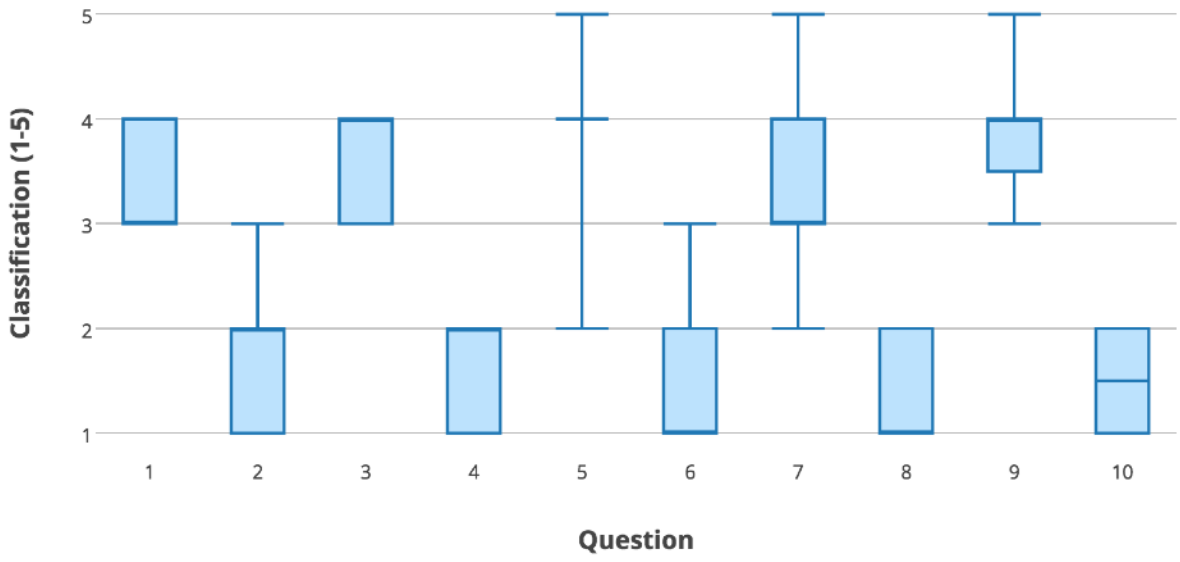


Fig.52: Classification by question

Concerning the second part of the questionnaire (results in Appendix E.3), 60% of the users gave an appreciation of 4 (on a scale from 1- terrible to 5 - very good) to the ease of use of the different visualizations and 5% gave the maximum classification. 45% gave a classification of 4 for the ease of understanding and 35% gave the maximum classification. Regarding the menus, 45% gave a score of 4 to the ease of use of the menus and 10% gave maximum score. 45% were neutral. 45% gave a classification of 5 for the ease of understanding and 35% gave a

classification of 4. About the timeline slider, 60% gave the maximum classification for the ease of use and 30% a classification of 4. 55% gave the maximum classification for the ease of understanding and the rest gave a classification of 4. For the ease of use of the overall application, 65% gave a classification of 4, 20% were neutral and 15% gave the maximum classification. 55% gave a classification of 4 for the ease of understanding, 15% gave the maximum classification and 30% were neutral.

6.3 Discussion

We recognize the results of our experiments positive, and people considered our system quick and well designed. Tasks were also adequate, despite a few problems with some of them. They confirmed the existence of some identified issues and presented us some other suggestions. Even with a good score of 75.5 on SUS, we know there are areas that need improvement.

Overall, tasks were well executed and confirmed that our interface contains the visualizations functional, and well linked between them and with the timeline slider. Tests proved the usability of our system, although we will have to deal with some minor interface improvements, regarding the menus and widgets of our system. These problems were identified during the tests, with some being user suggestions. Apart from this, user comments and feedback were fairly positive. Many users did not know about lifelogging or analysis of personal semantics and geolocation data. They found the activity to be intriguing, although showing some privacy concerns.

Regarding our objective, specified in section 1.1, we can say it was achieved. The objective was to d*esign a visualization that allows users to understand and analyze their spatio-temporal information, with focus on personal semantics*. User tests shown us that we have a good quality system, with users being able to customize, filter and find information and patterns, regarding personal semantics and geolocation data. Respecting our many visualizations, we only have one scalability issue to correct and improve.

Chapter 7

Conclusions

With the exponential growth in the usage of GPS tracking devices such as smartphones, smartwatches and tablets, huge amounts of personal mobility data derived from that usage are generated. Most approaches to the exploration of mobility tracking still do not cope with the personal aspect of data. Hence, there is no adequate way to analyze and derive information with personal significance from collected spatio-temporal data.

For that reason, we found the need to design and create an application for users to thoroughly understand and analyze their spatio-temporal information, with focus on the personal aspects of the data regarding travels, stays and places the user visited. To know about the components it needed, we first had to analyze several different works that provided different interfaces and visualizations (maps, timelines, heatmaps, etc.) for users to analyze mobility data. We observed the advantages and disadvantages of those different approaches. Few works included approaches to the personal aspect of the data, confirming the lack of solutions for this issue. Then, we made a survey with potential users, in order to understand the requirements - which factors are relevant for the users.

All things considered, we established the architecture of our solution. It is described in Chapter 4. Our back-end comprises a database and a web server and it is detailed in the same chapter. We also approach data collection (problems and processing) in Chapter 3. For the front-end, we identified the three components we needed: the visualization area, the menu area and the timeline slider area. The details about our interface and these components are described in Chapter 5. We also study the scalability of both the front-end and back-end by load testing a large dataset (it simulates circa five years of mobility tracking and personal semantic annotations). The results were very satisfactory and so we can consider our solution to be quite scalable. Still, we found one problem regarding the scalability, and we propose it as future work, together with some other features.

After designing our solution, we had to analyze if users could use and understand it. For that to happen, we conducted a series of tests. Our objective of *designing a visualization that allows users to understand and analyze their spatio-temporal information, with focus on personal semantics* was attained, as our evaluation describes in Chapter 6. There, we detail the protocol, tasks and test results.

7.1 Future Work

After disclosing the proposed implementation, there are still some issues that can be addressed in the future:

- Small interface enhancements: improve the scalability of the Trips Network visualization; also, the evaluation we conducted showed the need to perfect the widget removal and menu closing mechanisms;
- Map network and lens visualization: based on (Otten 2015), this is a complex visualization that uses a lens to zoom in and out on a map, so as to observe various types of networks. In this case the lens would be tweaked to also show several semantic information regarding the network that is being analyzed. Ideally, this visualization would work on a fullscreen instead of a single widget on the grid;
- Security measures: the next step to make our system more secure, regarding data privacy, is to add features such as a login system;
- Perfect the back-end data processing: Chapter 3 shows how scalable our back-end is. Nonetheless, the idea here is to implement more algorithms and processing methods to further clean and optimize the data, namely data related to GPS tracks.

Bibliography

Andrienko, Gennady, et al. 2013. Extracting Semantics of Individual Places from Movement Data by Analyzing Temporal Patterns of Visits. In *Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place* (COMP '13). ACM, New York, NY, USA, , Pages 9 , 8 pages. DOI=http://dx.doi.org/10.1145/2534848.2534851

Andrienko, Gennady, et al. 2011. Challenging problems of geospatial visual analytics. In *Journal of Visual Languages & Computing. 22.4.* p. 251-256. https://www.researchgate.net/profile/Gennady_Andrienko/publication/220578976_Challenging_problems_of_geospatial_visual_analytics/links/546c6d7c0cf257ec78ffeb82.pdf

Andrienko, Gennady, et al. 2011. An event-based conceptual model for context-aware movement analysis. Int. J. Geogr. Inf. Sci. 25, 9 (September 2011), 1347-1370. DOI=http://dx.doi.org/10.1080/13658816.2011.556120

Andrienko, Natalia; Andrienko, Gennady. 2013. Visual analytics of movement: an overview of methods, tools and procedures. In *Information Visualization 12*, 1 (January 2013), 3-24. DOI=http://dx.doi.org/10.1177/1473871612457601

Attfield, Simon, et al. 2014. Patterns of life visual analytics suite for analyzing GPS movement patterns. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 343-344. DOI=http://dx.doi.org/10.1109/VAST.2014.7042557

Chen, Siming, et al. 2014. MovementFinder: A Multi-filter Visual Analytics Design for Movement Data Investigation. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 345-346. http://vis.pku.edu.cn/people/simingchen/docs/vastchallenge14-mc2.pdf

Fonseca, Manuel J., et al. 2012. Introdução ao design de interfaces, FCA-Editora de Informática, Lisboa. ISBN: 978-972-722-738-9

Guo, Chen, et al. 2014. Dodeca-rings map: Interactively finding patterns and events in large geo-temporal data. In *Visual Analytics Science and Technology (VAST), 2014*

IEEE Conference on. IEEE. p. 353-354. DOI=http://dx.doi.org/10.1109/VAST.2014.7042562

Hundt, Michael, et al. 2014. Visual analytics for detecting behaviour patterns in geo-temporal data. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 355-356. DOI=http://dx.doi.org/10.1109/VAST.2014.7042563

Jeon, Jae Ho, et al. 2014. Exploratory visualization of smarphone-based life-logging data using Smart Reality Testbed. In *Big Data and Smart Computing (BIGCOMP).* 2014 International Conference on. IEEE, p. 29-33. DOI=10.1109/BIGCOMP.2014.6741400

Kraak, Menno-Jan. 2003. The space-time cube revisited from a geovisualization perspective. In *Proc. 21st International Cartographic Conference*. p. 1988-1996. http://www.itc.eu/library/Papers_2003/art_proc/kraak.pdf

Krueger, Robert, et al. 2013. TrajectoryLenses - a set-based filtering and exploration technique for long-term trajectory data. In *Proceedings of the 15th Eurographics Conference on Visualization (EuroVis '13)*. The Eurographs Association & John Wiley & Sons, Ltd., Chichester, UK, 451-460. DOI=http://dx.doi.org/10.1111/cgf.12132

Krueger, Robert, et al. 2014. Visual analysis of movement behavior using web data for context enrichment. In *Pacific Visualization Symposium (PacificVis), 2014 IEEE*. p. 193-200. DOI=http://dx.doi.org/10.1109/PacificVis.2014.57

Larsen, Jakob Eg, et al. 2013. QS Spiral: Visualizing periodic quantified self data. In *CHI 2013 Workshop on Personal Informatics in the Wild: Hacking Habits for Health & Happiness*. http://orbit.dtu.dk/files/73859557/chi2013_pi.pdf

Liu, He, et al. 2011. Visual analysis of route diversity. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on. IEEE*. p. 171-180. DOI=http://dx.doi.org/10.1109/VAST.2011.6102455

Lu, Min, et al. 2015. OD-Wheel: Visual Design to Explore OD Patterns of a Central Region. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, Hangzhou, China. http://vis.pku.edu.cn/research/publication/PacificVis15_ODWheel.pdf

Lu, Min, et al. 2015. Trajrank: Exploring travel behaviour on a route by trajectory ranking. In *Visualization Symposium (PacificVis), 2015 IEEE Pacific. IEEE*. p. 311-318. DOI=http://dx.doi.org/10.1109/PACIFICVIS.2015.7156392

Lundblad, Patrik; Jern, Mikael. 2013. Geovisual Analytics and Storytelling Using HTML5. In *Information Visualisation (IV), 2013 17th International Conference. IEEE*. p. 263-271. DOI=http://dx.doi.org/10.1109/IV.2013.35

Otten, Heike, et al. 2015 Are there networks in maps? An experimental visualization of personal movement data. In *IEEE VIS 2015*. http://mariandoerk.de/papers/personalvis2015.pdf

Saraf, Parang, et al. 2014. Safeguarding Abila: Spatio-temporal activity modeling VAST 2014 Mini Challenge 2 award: Honorable mention for effective presentation. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 359-360. DOI=http://dx.doi.org/10.1109/VAST.2014.7042565

Thudt, Alice, et al. 2013. Visits: A spatiotemporal visualization of location histories. In *Proceedings of the eurographics conference on visualization*. p. 79-83. http://innovis.cpsc.ucalgary.ca/innovis/uploads/Publications/Publications/visits.pdf

Van Der Spek, S. C. Activity patterns in public space; a tool for assessing city centres. 2010. In *Walk 21, 11th conference, The Hague.* Nov. 16-19, 2010. https://www.researchgate.net/profile/Stefan_Van_der_SPEK/publication/254822298_Activity_patterns_in_public_space_a_tool_for_assessing_city_centres/links/0deec525f96d73d4e5000000.pdf

Wang, Zuchao; Yuan, Xiaoru. 2014. Urban trajectory timeline visualization. In *Big Data and Smart Computing (BIGCOMP), 2014 International Conference on. IEEE*. p. 13-18. DOI=http://dx.doi.org/10.1109/BIGCOMP.2014.6741397

Wood, Jo. 2014. Visual analytics of GPS tracks: From location to place to behaviour. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 367-368. DOI=http://dx.doi.org/10.1109/VAST.2014.7042569

Appendix A

User Test Protocol

TraceMySteps

Test Protocol

Before starting, we would like to thank you for taking your time in testing this application.

TraceMySteps allows you to understand and analyze your spatio-temporal information, with focus on personal semantics. In the specific case of this test, it will be my personal information that will be used and scrutinized.

Firstly, you will be asked to freely navigate around the interface, in order to become familiar with it. Then you will perform some tasks.

We will record the time taken, but do not feel any pressure or need to hurry. It is only for statistical purposes.

We will not be taking not of any identity information. Your responses are completely anonymous. We assure you complete confidentiality. The information you provide will be stored only to track survey completion. All data will be recorded in the aggregate, with no individual identified.

You will have around 5 minutes to freely explore TraceMySteps' interface. You can do as many and different actions as you want. It is just to get you comfortable.

Then we will ask you to perform the tasks described in the next page, in a given order. We will record the screen and time for each task, so that we can later analyze your task and count the number of clicks. We will also take written notes about your execution of the tasks.

There will be a break in the middle of the tasks.

At the end of each task, we will ask you to express your doubts, concerns and obstacles (if there are any). This allows us to understand if you are making progress. You can also leave comments or suggestions in the last question of the next questionnaire.

The final questionnaire contains two parts: the first part in which you have to classify each sentence on a scale from 1 - Strongly disagree to 5 - Strongly agree; the second part in which you just have to answer using a linear scale or choose one of the answers.

Tasks

First, you will have around five minutes to explore the interface. But if you think you are good to go, tell us. Each task contains the expected target time for its completion (still, the duration time of each task can exceed it - they are two separate metrics).

Ex: Task 1 has a expected 25 seconds interval, but User 1 took 30 seconds to complete it. Metrics: Duration time = 30 seconds; Completed within interval = No.

A) Dealing with widgets

1. Add a Places widget, a Calendar widget, a Stays widget and a Tracks widget - 20 secs
2. In the Places widget, find and tell us the name of the place with the second highest time spent and its value - 10 secs
3. In the Calendar, search and tell us the name of the places I have been on Tuesday, in the second week of February - 25 secs
4. Remove the Places and Calendar widgets - 7 secs

B) Connected visualizations

5. Add a Trips Chord Diagram widget - 7 secs
6. In the Chord Diagram, find and tell us the number of trips made between *inesc* and *cais do sodre station* - 15 secs
7. Add a Hexbin Places widget and then use the Chord Diagram to find and tell us the exact location of *cais do sodre station* - 22 secs
8. Use a new Places widget to find and point us where is my *home* - 22 secs
9. Add a Trips Network widget - 7 secs
10. Using the Chord Diagram, find and point us *inesc* in the Trips Network - 10 secs
11. In the Trips Network, find and tell us the most frequent trip containing *inesc* - 20 secs

C) Adjusting the slider

12. Drag the Tracks, Chord Diagram and Places widgets so as to put the three together - 20 secs
13. Set the slider date to the 8th of March and explore the tracks for that day - 50 secs
14. Find and point us how much time I spent at *home* on that day - 12 secs

15. Remove all remaining widgets using the “Clear All” option - 7 secs

D) Overall

16. Find and tell us the day in which I spent the most time in a single place, and what was that place - 120 secs

Appendix B

Questionnaires

B.1 User profile questionnaire

1. Gender
   A. Male
   B. Female
2. Age
   A. 18-25
   B. 26-35
   C. 36-50
   D. 51+
3. Academic studies
   A. Science/Engineering
   B. Humanities and Social Sciences
   C. Health Sciences
   D. Other
4. Academic degree
   A. Did not complete High School
   B. High School
   C. Bachelor's Degree
   D. Master's Degree
   E. Advanced Graduate work or Ph.D.

B.2 Overview questionnaire

Part One

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Part Two

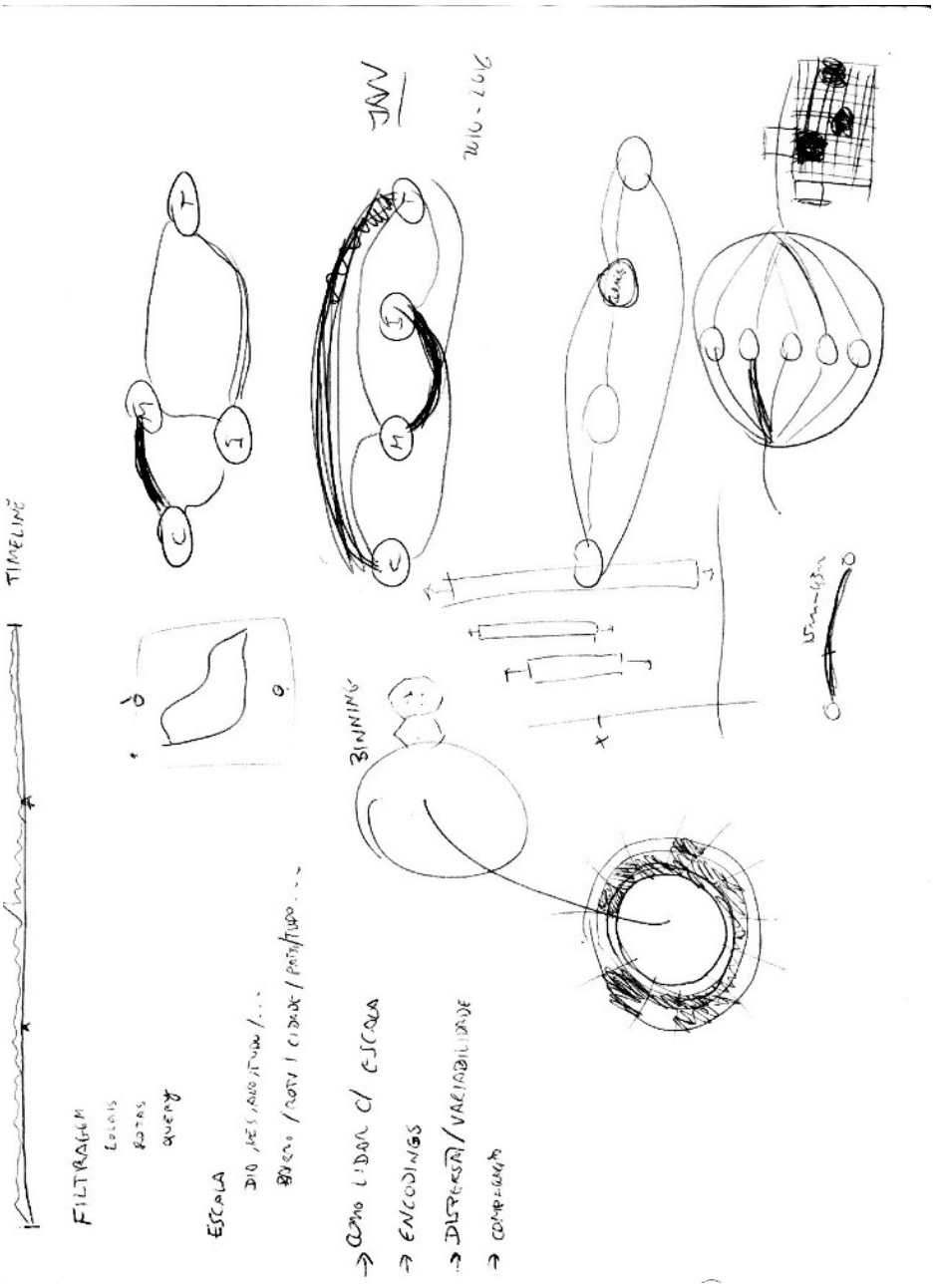
1. Regarding the different visualizations, they were: (1 - Terrible to 5 - Very good)
   A. Easy to use
   B. Easy to understand
2. Regarding the menus of the application, they were: (1 - Terrible to 5 - Very good)
   A. Easy to use
   B. Easy to understand
3. Regarding the timeline slider of the application, it was: (1 - Terrible to 5 - Very good)
   A. Easy to use
   B. Easy to understand
4. About the application, overall, it was: (1 - Terrible to 5 - Very good)
   A. Easy to use

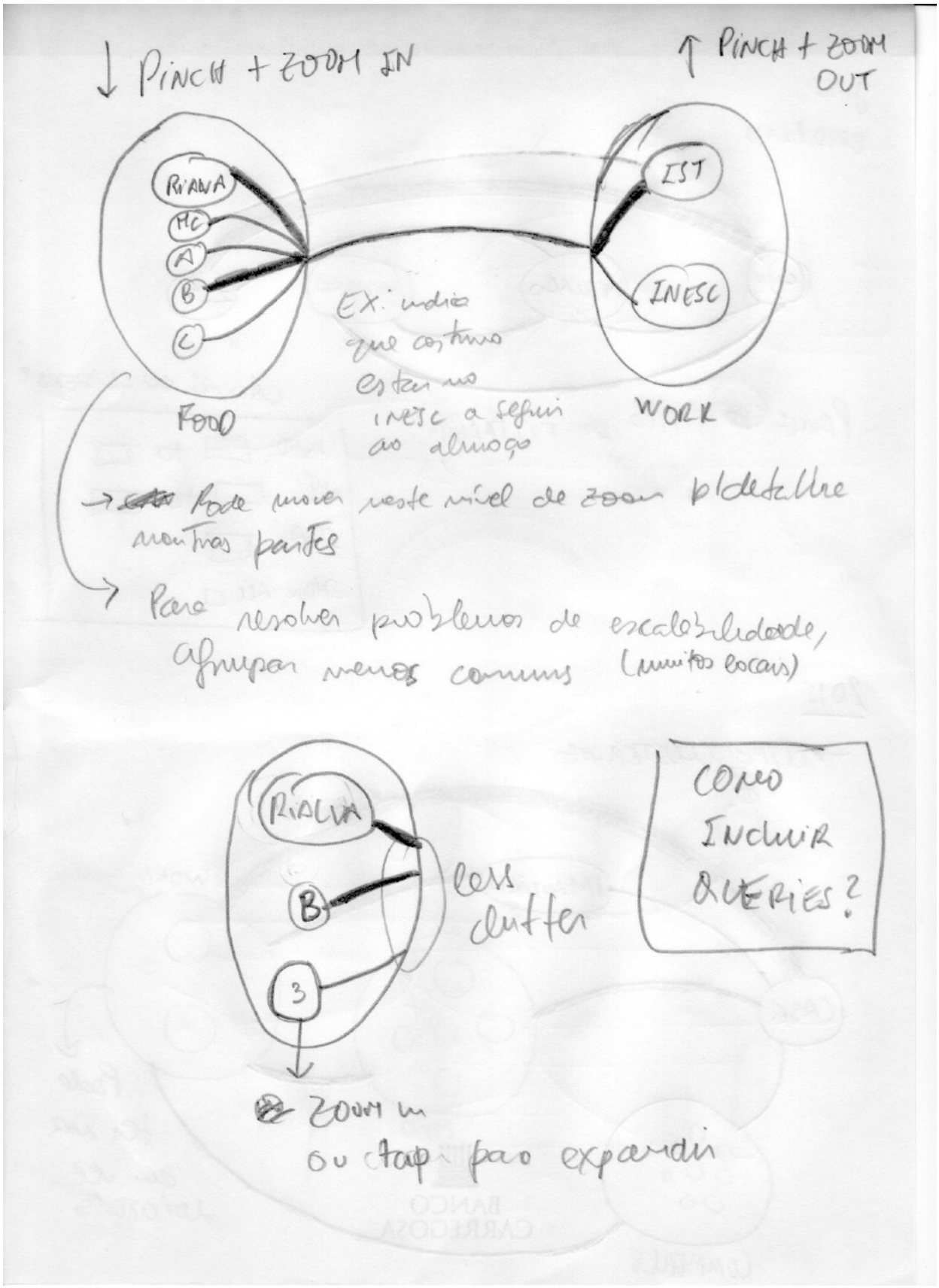
B. Easy to understand

5. Do you have a smartphone with GPS?
   A. Yes
   B. No
6. Do you make use of the GPS?
   A. Yes
   B. No
7. If yes, in which situations?
   A. Map navigation
   B. Track recording
   C. Photo geotagging
   D. Other
8. Would you track your daily spatio-temporal data?
   A. Yes
   B. No
9. If not, what reasons prevent you from tracking your spatio-temporal data?
   A. Privacy issues
   B. Battery issues
   C. Other
10. Leave any comments or suggestions

Appendix C

Prototype & visualizations sketches

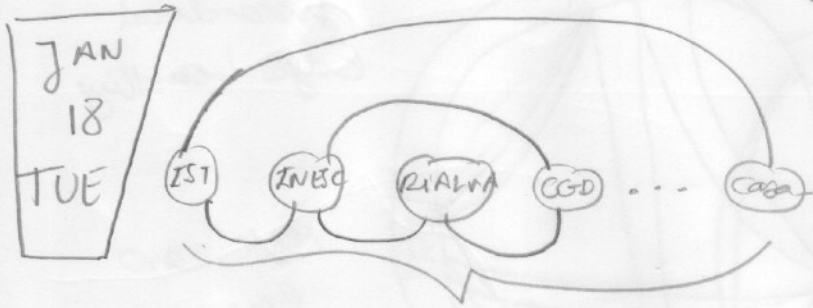



↓ PINCH + ZOOM IN
↑ PINCH + ZOOM OUT
RIANA
MC
A
B
C
FOOD
IST
INESC
WORK
EX: indica que costuma estar no INESC a seguir ao almoço
Pode mover neste nível de zoom p/ detalhe
Para resolver problemas de escalabilidade, agrupar menos comuns (muitos locais)
RIALVA
B
3
less clutter
COMO INCLUIR QUERIES?
Zoom in
ou tap para expandir

edge bundling


BE
FE
/loc/car.
Python
RESTful
ANGULAR.JS
BUS
AH
User tests
Usability
20 users
Temp, errors
Utilidade

Routes density – Bundles / thread arc



Highlight de um arco permite ver detalhes (nº vezes, tempo, etc.)

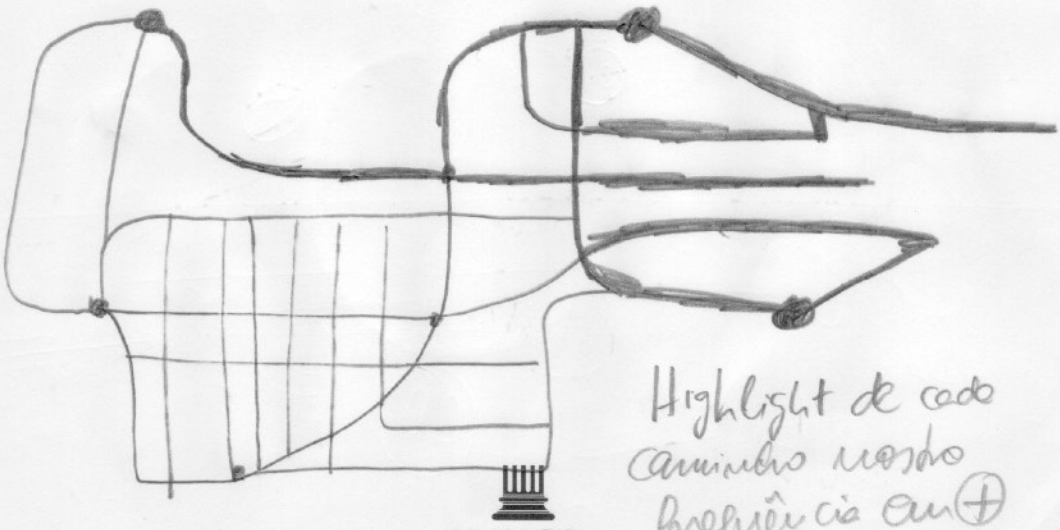
ou

pequena parte do mapa

ordenados por importância / proximidade, etc.

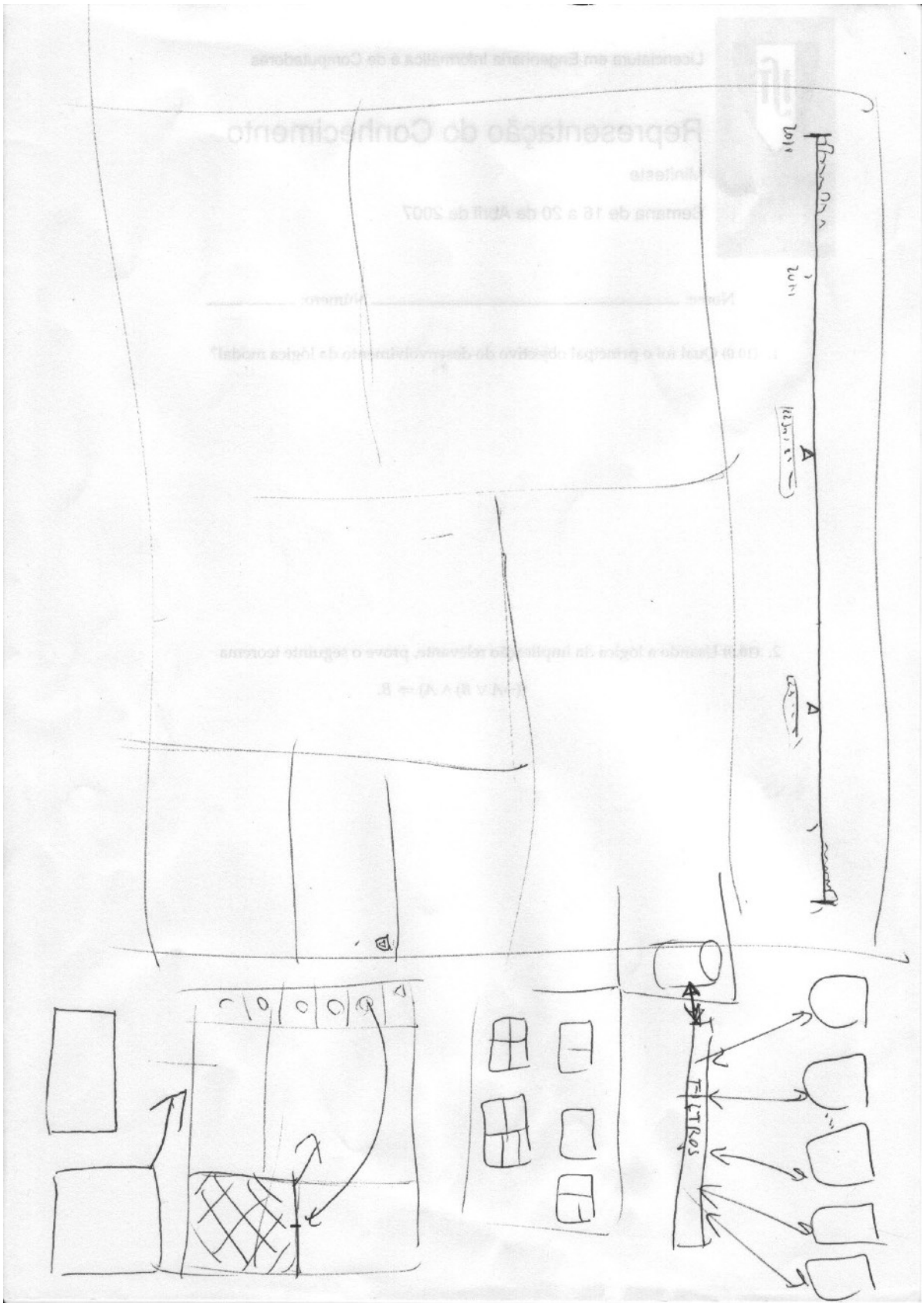
→ melhor para vistas diárias (evita problemas de clutter)

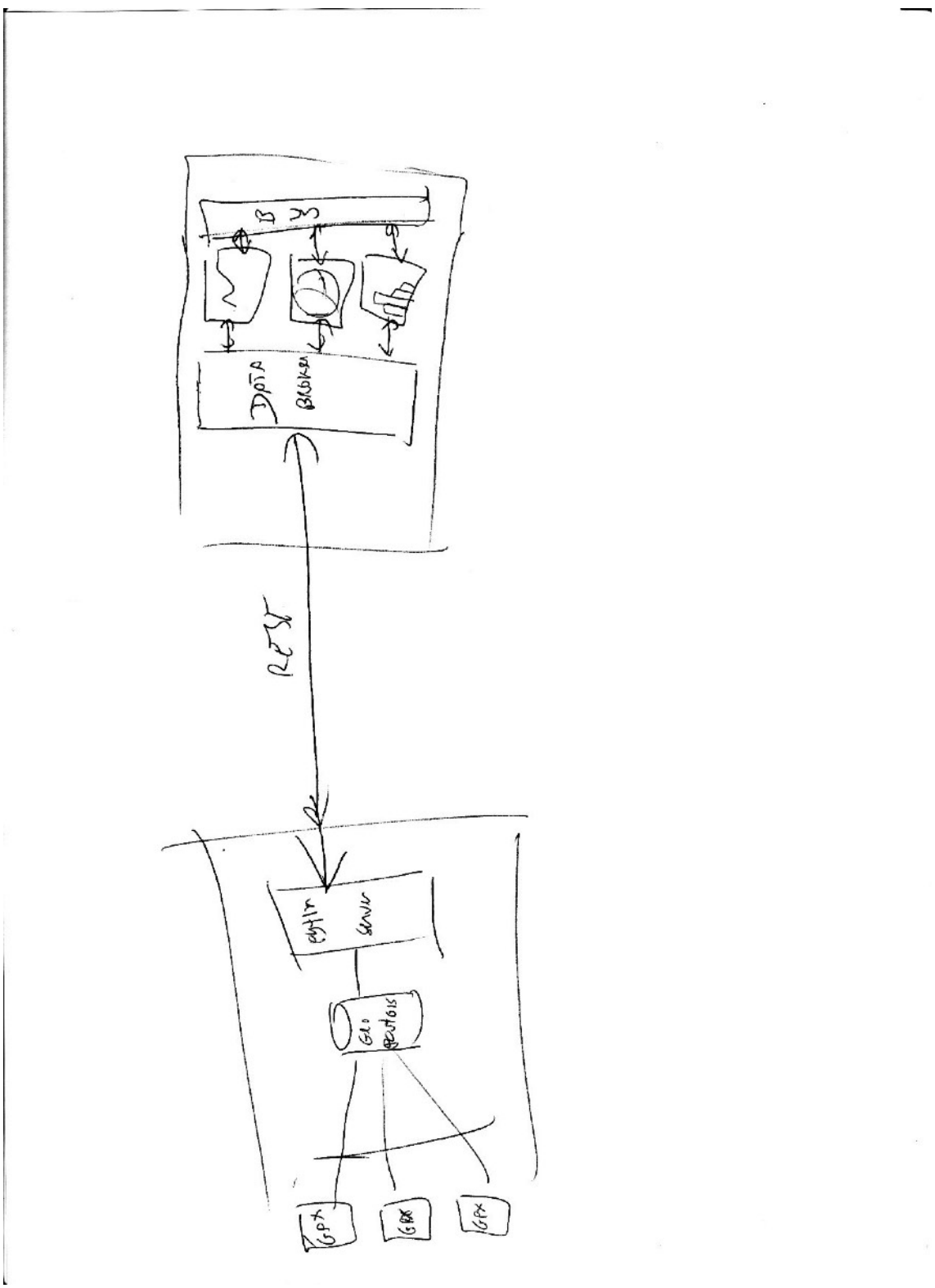
→ Para vistas anuais / mensais
→ hierarchical edge bundling?



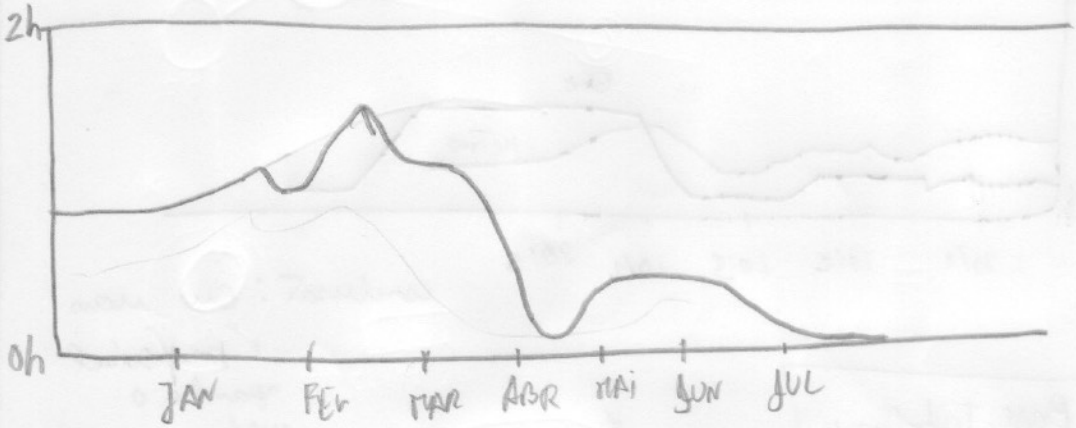
BANCO CARREGOSA

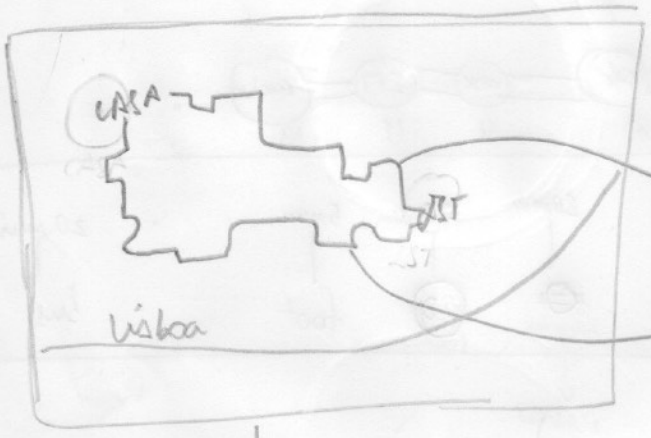
Highlight de cada caminho mostra frequência ou detalhe (nº vezes, tempo, ...)





TIME SPENT ON ROUTE 1 HOME - IST





EX usando Trabalho 17

com highlight mostra que demora 20 mins (médios)

Fazendo

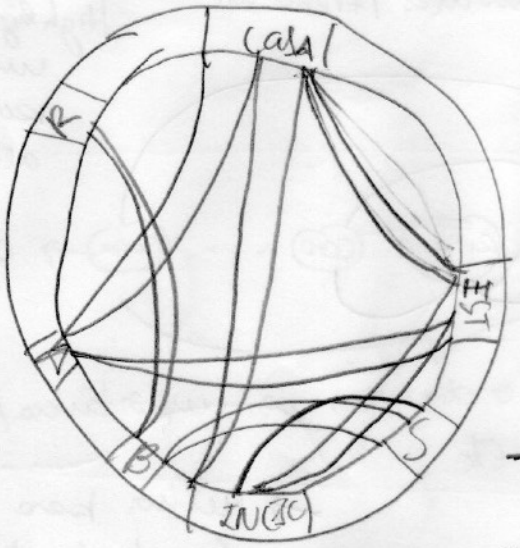
highlight mostra que demora 15 mins

OPTIMAL

Pode-se usar com o percurso dos transportes públicos, para ver qual demora ⊖

BANCO CARREGOSA

CHORD DIAGRAM WITH RELATIONS



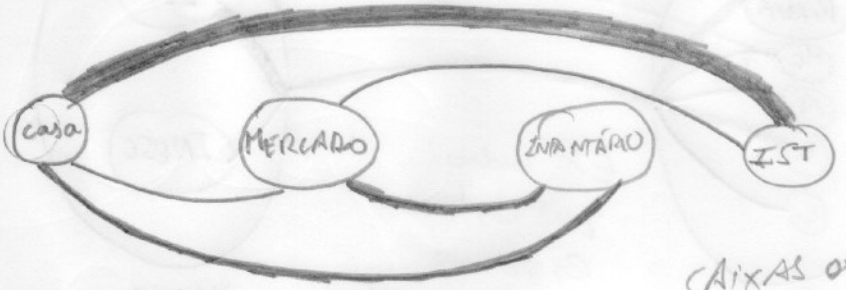
Hierarchical Edge bundling

Neste caso locais

→ Podem ser comunidades

→ Pessoas

JAN
2010/2013



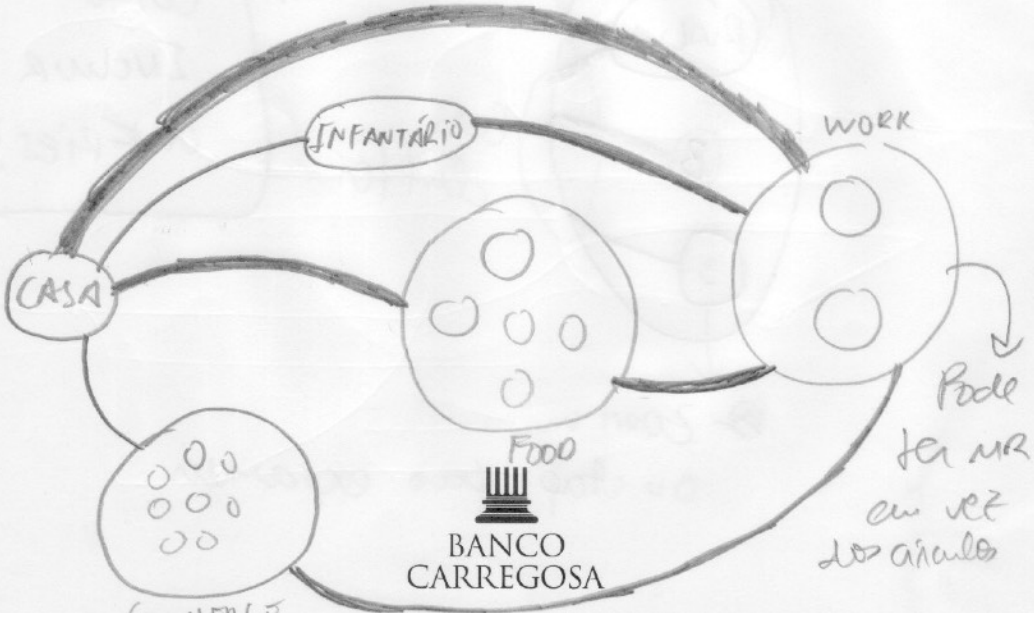
PAINEL TÍPICO DE FILTRAGEM?

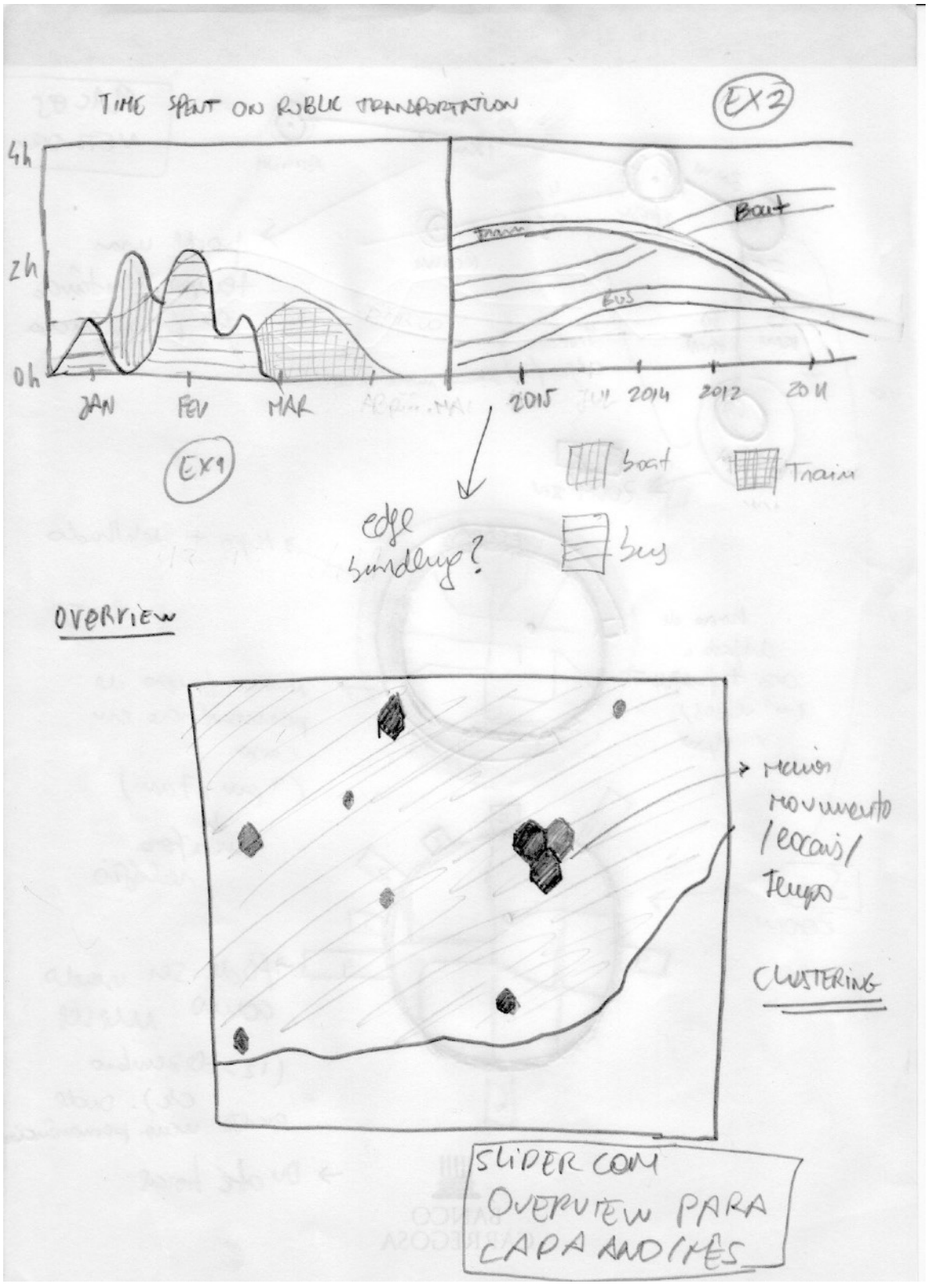
CAIXAS OU SLIDERS?

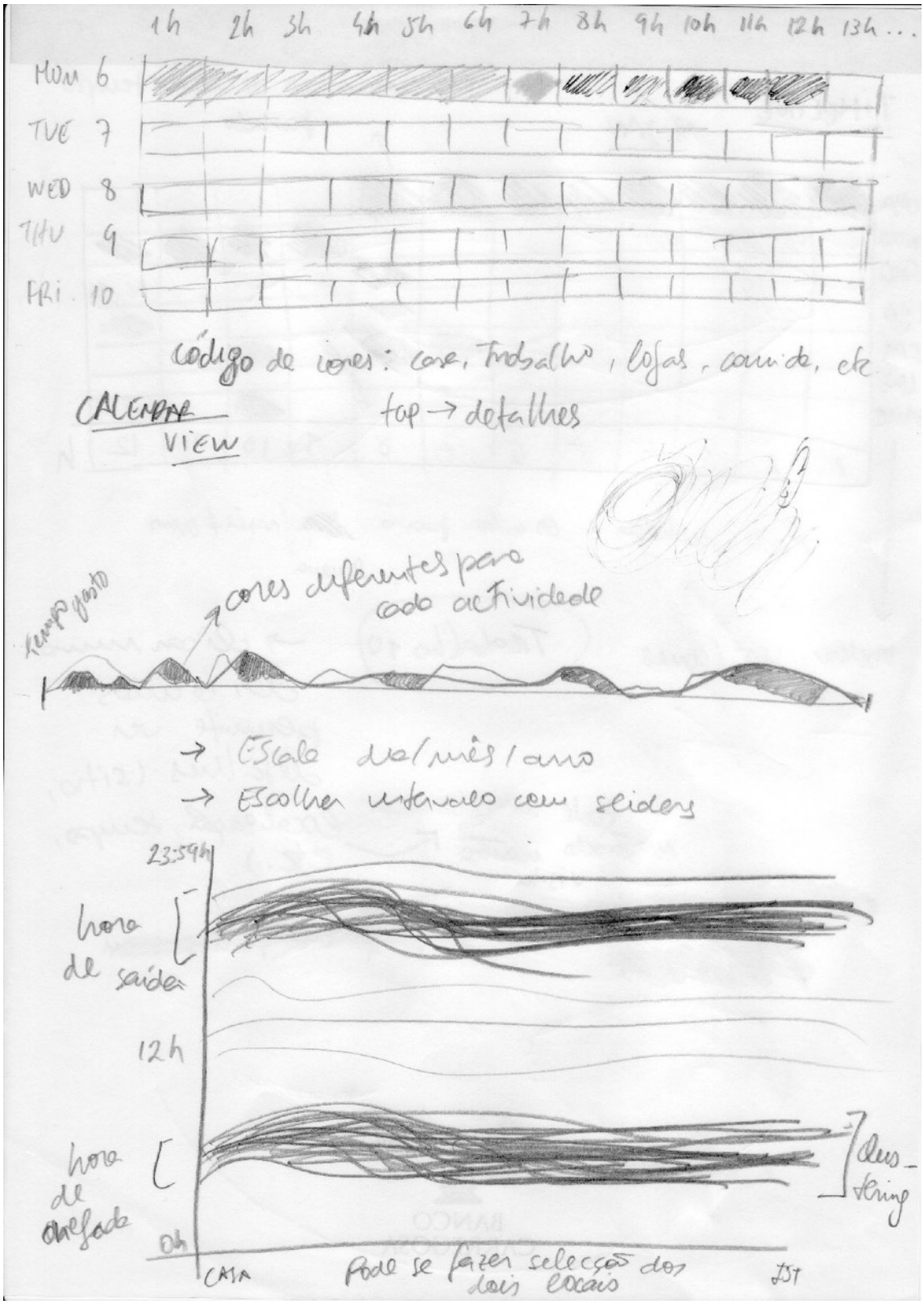
ANO [ ] to [ ]
MÊS [ ] to [ ]
DIA [ ]
SHOW ALL [ ]

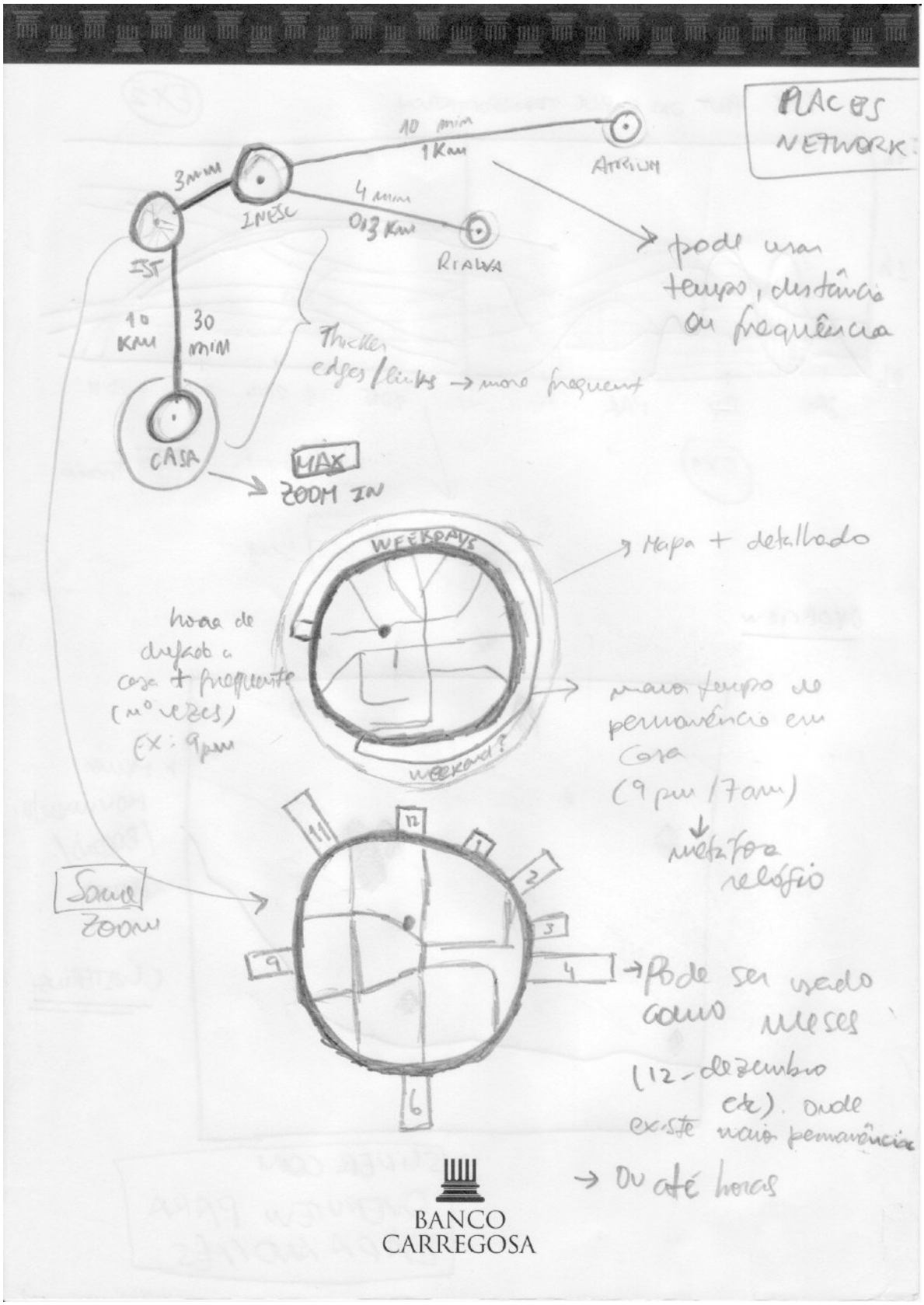
2015

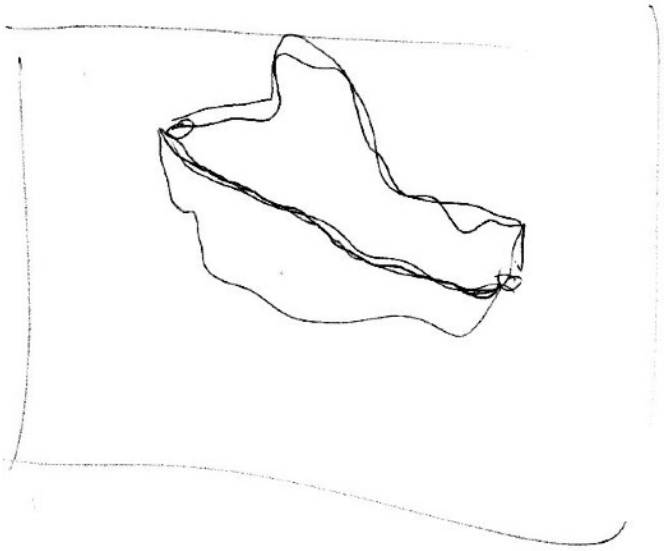
→ TYPE CLUSTERING

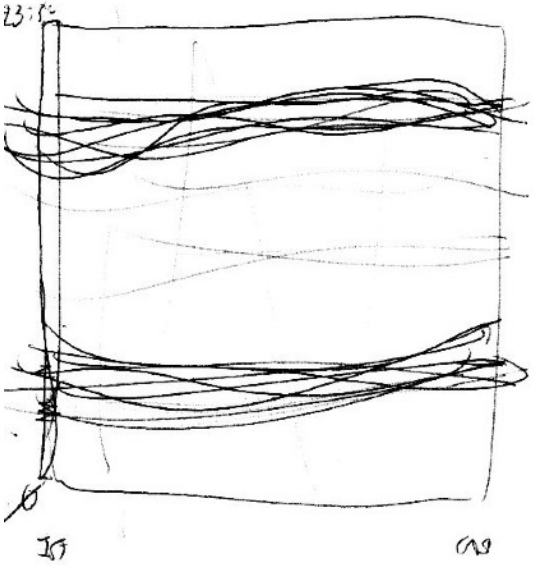


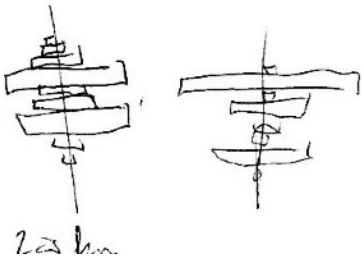

TIME SPENT ON PUBLIC TRANSPORTATION
EX2
4h
2h
0h
JAN
FEV
MAR
2015
2014
2013
2011
Boat
Train
Bus
EX1
edge bundling?
Boat
Train
Bus
OVERVIEW
menor movimento / ocaso / tempo
CLUSTERING
SLIDER COM OVERVIEW PARA CAPA ANOTAÇÕES

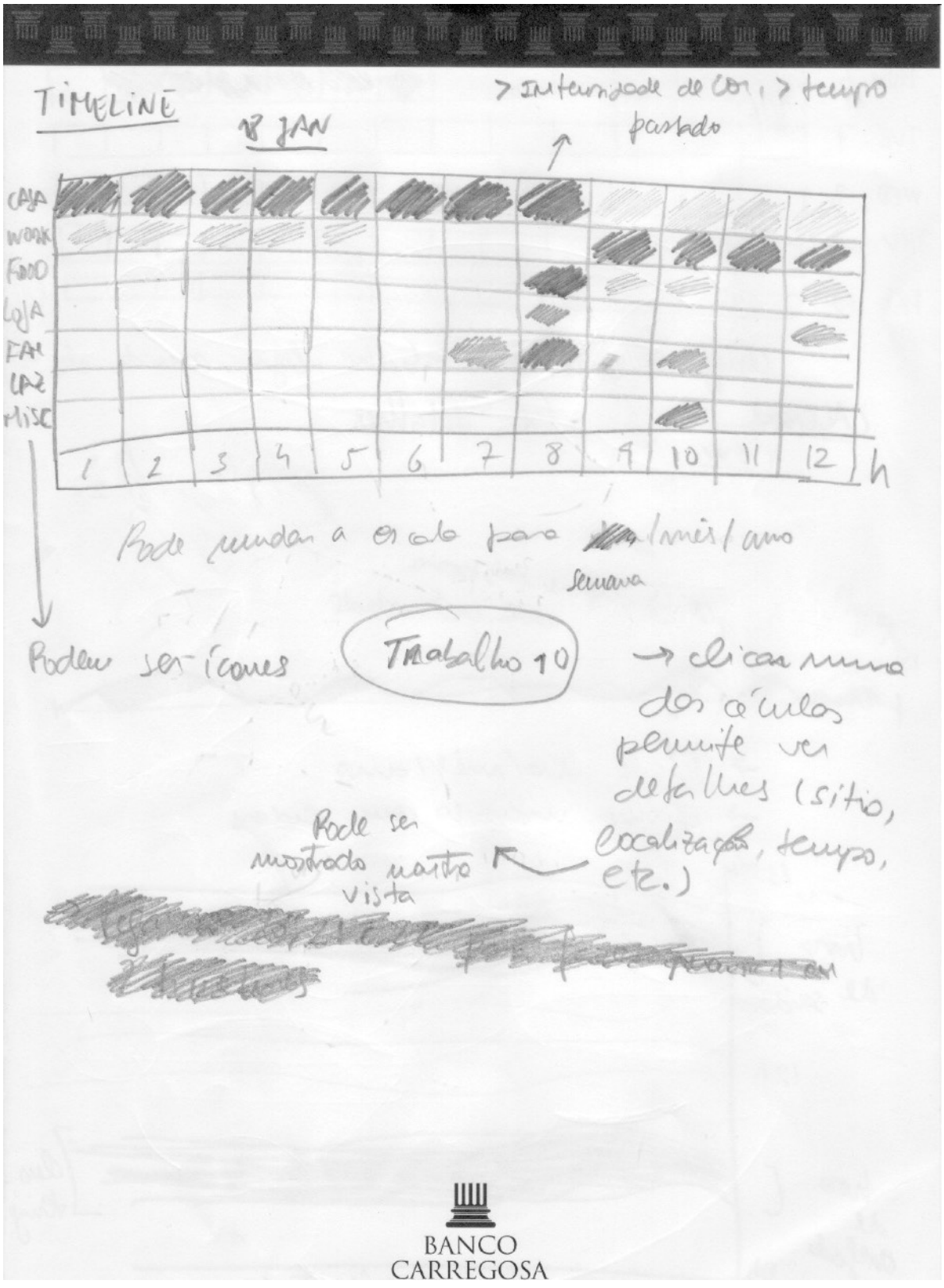

1h 2h 3h 4h 5h 6h 7h 8h 9h 10h 11h 12h 13h...
MON 6
TUE 7
WED 8
THU 9
FRI 10
código de cores: casa, trabalho, lojas, comida, etc
CALENDAR VIEW
top → detalhes
tempo gasto
cores diferentes para cada actividade
→ Escala dia/mês/ano
→ Escolher intervalo com sliders
23:59h
hora de saída
12h
hora de chegada
0h
CASA
Pode se fazer selecção dos dois locais
IST
dias


PLACES NETWORK:
10 min
1 Km
ATRIUM
3 min
INESC
4 min
0,3 Km
RIALVA
IST
10 KM
30 min
CASA
pode usar tempo, distância ou frequência
Thicker edges/lines → more frequent
MAX ZOOM IN
WEEKDAYS
weekend?
Mapa + detalhado
horas de dormir a casa + frequente (nº vezes) (ex: 9pm
mais tempo de permanência em casa (9pm / 7am)
mesma coisa relógio
Semi Zoom
Pode ser usado como meses (12 - dezembro etc). onde existe mais permanência
→ ou até horas
BANCO CARREGOSA









BANCO CARREGOSA

Appendix D

User survey - relevant factors

1 - O total de tempo passado num determinado local? (Ex: No café X despendi o total de 5 horas; Nos últimos três anos despendi 1000 horas na faculdade)

2 - De entre os meios de transporte que usa quais os que demoram mais tempo a chegar a um certo destino? (Ex: De metro demoro 30 minutos a chegar à faculdade, enquanto que de autocarro demoro 20 minutos). Isto pode ser especialmente útil num dia em que tem um teste e não pode chegar atrasado.

3 - Qual o caminho que demora mais tempo até chegar a um certo destino? (Ex: Para chegar à faculdade pelas ruas A, B e C demoro 20 minutos. Pelas ruas D, E e F demoro 15 minutos). Pode ser útil na perspectiva referida na pergunta anterior, mas desta vez sem contar com transportes.

4 - Quantas vezes visitou um determinado local? (Ex: Visitei o restaurante Y 15 vezes)

5 - Os diferentes percursos (caminhos) que fez?

6 - As distâncias percorridas nos diferentes percursos feitos?

7 - A distância total percorrida num certo intervalo de tempo?

8 - Ao observar os caminhos percorridos, quais os mais e os menos frequentes?

9 - Que locais nunca ou raramente explorou? (Ex: Saber que nunca visitou a Rua de Sta. Catarina no Porto)

10 - Quando foi a um certo local? (Ex: Fui ao café X no dia 7, 9 e 11). Isto pode ser útil para associar eventos pessoais a essas visitas e datas.

11 - E ter acesso a uma rede de deslocações, que mostre as relações entre elas? Isto poderia permitir a visualização de destinos e percursos mais comuns a partir de um certo ponto (Ex: Começando no local de trabalho, o percurso mais vezes feito é para casa, o segundo mais vezes feito é para o ginásio, etc.)

12 - Em que locais despende mais tempo? (Ex: Maior parte do tempo passado em casa, seguidamente na faculdade, em terceiro lugar no local de trabalho, etc.)

13 - O número de vezes que vai a locais diferentes, dentro do mesmo tipo? (Ex: Quando vou ao cinema, costumo ir mais vezes ao cinema Z e menos vezes ao cinema W)

14 - Ao entrar num local, se no passado já esteve nesse local? (Ex: Entrei no cinema A e fiquei informado que a última vez que lá estive foi há 5 meses para ver o filme ABC)

15 - A qualquer altura, se mudou os seus padrões de mobilidade? Poderia ser interessante para analisar por exemplo a diferença de caminhos utilizados entre uma origem e destino, ao longo de um intervalo de tempo ou no total do tempo de utilização. Também poderia ser útil para analisar como vai sendo distribuído o tempo de visita e permanência em certos locais, e como estes variam. (Ex: Uso o mesmo caminho casa-trabalho que usava no ano passado? Continuo a gastar as minhas horas de almoço no Restaurante ABCD? Depois de a Maria casar, isso afectou ou não as minhas visitas à casa dela?)

16 - Gostaria de associar a um percurso ou local, uma foto ou álbum? Isto poderia permitir a fácil identificação de eventos ou pessoas associadas. (Ex: uma foto com amigos num bar permitiria relembrar um evento importante como um aniversário)

Appendix E

Evaluation Results

E.1 User profile results

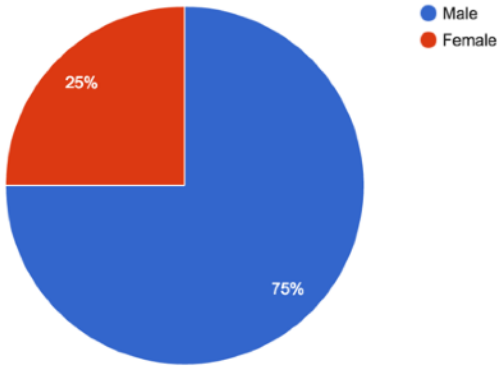


Fig.E.1: Users' gender

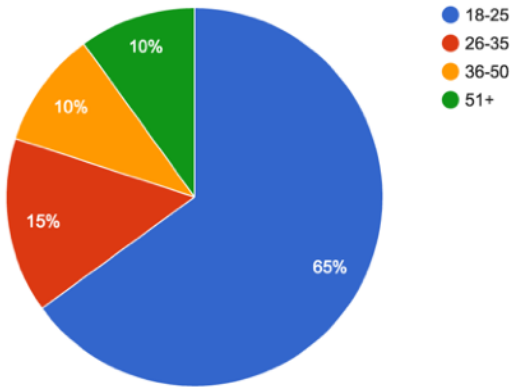


Fig.E.2: Users' age

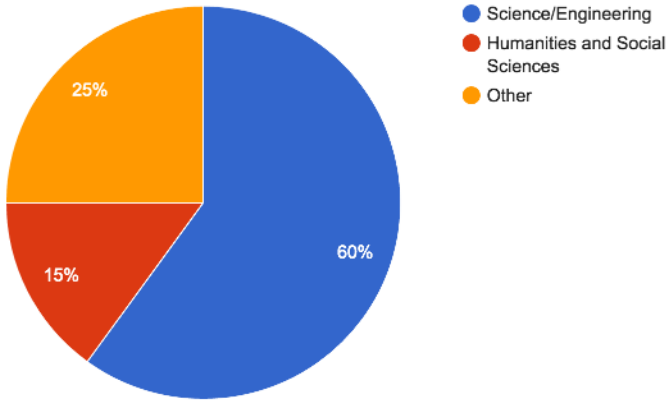


Fig.E.3: Users' studies

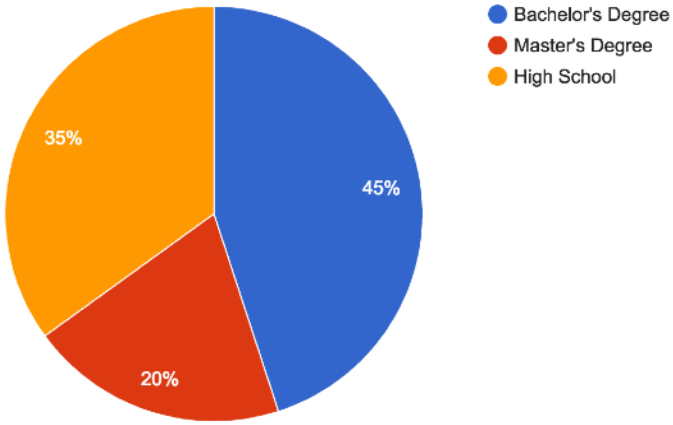


Fig.E.4: Users' degree

E.2 Task results

Task	User	Time (s)	Within interval?	Errors	Clicks	Correct answer?
1	1	15	Y	0	6	NA
	2	12	Y	1	6	
	3	11	Y	1	6	
	4	10	Y	0	6	
	5	11	Y	0	6	
	6	10	Y	0	6	
	7	13	Y	0	7	
	8	12	Y	0	7	
	9	21	N	1	7	
	10	15	Y	0	6	
	11	10	Y	0	7	
	12	10	Y	0	6	
	13	15	Y	0	7	
	14	14	Y	0	6	
	15	17	Y	1	8	
	16	12	Y	0	6	
	17	14	Y	0	6	
	18	10	Y	0	6	
	19	25	N	1	10	
	20	8	Y	0	8	
2	1	13	N	0	1	Y
	2	6	Y	0	1	Y
	3	15	N	2	6	Y
	4	7	Y	0	2	Y
	5	6	Y	0	1	Y
	6	9	Y	0	1	Y
	7	6	Y	0	1	Y
	8	8	Y	1	1	N
	9	10	Y	2	2	Y
	10	16	N	1	3	Y
	11	6	Y	0	0	N
	12	7	Y	0	1	Y
	13	7	Y	1	0	N
	14	18	N	1	3	Y
	15	9	Y	0	1	Y
	16	5	Y	1	2	Y
	17	6	Y	0	1	Y
	18	7	Y	0	1	Y
	19	7	Y	0	1	Y
	20	8	Y	0	1	Y

Task	User	Time (s)	Within interval?	Errors	Clicks	Correct answer?
3	1	26	N	2	4	Y
	2	10	Y	1	1	Y
	3	8	Y	0	2	Y
	4	14	Y	0	1	Y
	5	16	Y	1	1	Y
	6	35	N	0	1	Y
	7	20	Y	1	3	Y
	8	8	Y	0	1	Y
	9	11	Y	0	1	Y
	10	17	Y	0	1	Y
	11	18	Y	1	3	N
	12	11	Y	0	3	Y
	13	9	Y	0	2	Y
	14	15	Y	0	2	Y
	15	19	Y	0	1	Y
	16	12	Y	0	2	Y
	17	18	Y	1	1	Y
	18	13	Y	0	2	Y
	19	10	Y	0	1	Y
	20	12	Y	0	2	Y
4	1	5	Y	0	2	NA
	2	10	N	1	4	
	3	4	Y	0	3	
	4	6	Y	0	3	
	5	8	N	1	5	
	6	4	Y	0	4	
	7	5	Y	0	3	
	8	10	N	1	4	
	9	7	Y	0	3	
	10	6	Y	0	2	
	11	8	N	1	4	
	12	4	Y	0	2	
	13	10	N	1	6	
	14	9	N	1	4	
	15	5	Y	0	3	
	16	6	Y	0	4	
	17	4	Y	0	4	
	18	7	Y	0	2	
	19	4	Y	0	3	
	20	11	N	1	5	

Task	User	Time (s)	Within interval?	Errors	Clicks	Correct answer?
5	1	5	Y	0	3	NA
	2	3	Y	0	3	
	3	4	Y	0	3	
	4	4	Y	0	3	
	5	5	Y	1	2	
	6	3	Y	0	3	
	7	5	Y	0	3	
	8	3	Y	1	3	
	9	4	Y	0	3	
	10	4	Y	0	3	
	11	4	Y	1	2	
	12	7	Y	0	3	
	13	3	Y	0	3	
	14	3	Y	0	3	
	15	3	Y	1	2	
	16	4	Y	0	3	
	17	7	Y	0	3	
	18	6	Y	1	2	
	19	5	Y	0	3	
	20	3	Y	1	2	
6	1	15	Y	1	1	Y
	2	18	N	1	2	Y
	3	7	Y	0	1	Y
	4	9	Y	0	0	Y
	5	8	Y	0	1	Y
	6	26	N	1	3	Y
	7	10	Y	0	1	Y
	8	19	N	1	4	Y
	9	16	N	2	2	Y
	10	10	Y	0	0	Y
	11	21	N	1	3	Y
	12	9	Y	0	0	Y
	13	20	N	1	2	Y
	14	11	Y	0	0	Y
	15	28	N	1	4	Y
	16	16	N	1	0	Y
	17	8	Y	1	0	Y
	18	24	N	1	3	Y
	19	12	Y	0	1	Y
	20	11	Y	0	1	Y

Task	User	Time (s)	Within interval?	Errors	Clicks	Correct answer?
7	1	15	Y	0	3	Y
	2	9	Y	0	3	Y
	3	15	Y	0	3	Y
	4	10	Y	0	3	Y
	5	13	Y	0	3	Y
	6	20	Y	1	4	Y
	7	10	Y	0	3	Y
	8	15	Y	0	5	Y
	9	16	Y	1	3	Y
	10	18	Y	0	4	Y
	11	20	Y	1	3	Y
	12	21	Y	0	4	Y
	13	17	Y	0	3	Y
	14	16	Y	0	4	Y
	15	18	Y	1	4	Y
	16	19	Y	0	3	Y
	17	12	Y	0	3	Y
	18	25	N	2	3	Y
	19	15	Y	0	4	Y
	20	11	Y	1	3	Y
8	1	20	Y	1	4	Y
	2	13	Y	0	4	Y
	3	11	Y	0	4	Y
	4	19	Y	0	4	Y
	5	12	Y	0	3	Y
	6	14	Y	1	4	Y
	7	15	Y	0	4	Y
	8	23	N	1	4	Y
	9	13	Y	0	4	Y
	10	18	Y	0	4	Y
	11	20	Y	1	5	Y
	12	22	Y	1	6	Y
	13	13	Y	0	4	Y
	14	14	Y	0	4	Y
	15	16	Y	0	4	Y
	16	25	N	1	5	Y
	17	12	Y	0	4	Y
	18	17	Y	0	5	Y
	19	11	Y	0	4	Y
	20	11	Y	0	4	Y

Task	User	Time (s)	Within interval?	Errors	Clicks	Correct answer?
9	1	4	Y	0	3	NA
	2	5	Y	0	3	
	3	8	N	0	3	
	4	5	Y	0	3	
	5	5	Y	0	3	
	6	5	Y	0	4	
	7	4	Y	0	3	
	8	6	Y	0	3	
	9	4	Y	0	3	
	10	5	Y	0	4	
	11	9	N	0	3	
	12	5	Y	0	3	
	13	5	Y	1	4	
	14	7	Y	0	3	
	15	6	Y	0	3	
	16	8	N	1	4	
	17	5	Y	0	3	
	18	6	Y	0	3	
	19	5	Y	0	3	
	20	4	Y	1	4	
10	1	6	Y	0	0	Y
	2	8	Y	0	3	Y
	3	5	Y	0	1	Y
	4	9	Y	0	0	Y
	5	7	Y	0	0	Y
	6	6	Y	0	3	Y
	7	11	N	1	0	Y
	8	7	Y	0	0	Y
	9	15	N	1	2	Y
	10	8	Y	0	0	Y
	11	10	Y	0	2	Y
	12	7	Y	0	0	Y
	13	8	Y	0	0	N
	14	9	Y	0	1	Y
	15	8	Y	1	0	Y
	16	14	N	2	1	N
	17	9	Y	0	1	Y
	18	10	Y	0	2	Y
	19	9	Y	0	1	Y
	20	12	N	1	0	Y

Task	User	Time (s)	Within interval?	Errors	Clicks	Correct answer?
11	1	21	N	1	2	Y
	2	4	Y	0	0	Y
	3	23	N	1	1	Y
	4	6	Y	0	0	Y
	5	10	Y	1	0	N
	6	5	Y	0	1	Y
	7	7	Y	2	0	N
	8	25	N	3	2	N
	9	18	Y	3	2	N
	10	9	Y	1	0	Y
	11	17	Y	3	1	N
	12	8	Y	0	0	Y
	13	15	Y	2	1	N
	14	23	N	1	1	N
	15	10	Y	0	0	Y
	16	11	Y	2	1	N
	17	14	Y	0	0	Y
	18	23	N	3	3	N
	19	18	Y	4	2	N
	20	21	N	2	2	N
12	1	12	Y	0	4	NA
	2	5	Y	0	3	
	3	30	N	2	6	
	4	9	Y	0	3	
	5	14	Y	0	3	
	6	12	Y	0	3	
	7	11	Y	0	3	
	8	9	Y	0	3	
	9	25	N	2	4	
	10	12	Y	0	3	
	11	8	Y	0	3	
	12	11	Y	0	3	
	13	8	Y	0	3	
	14	7	Y	2	5	
	15	8	Y	0	3	
	16	28	N	1	3	
	17	9	Y	0	3	
	18	11	Y	0	4	
	19	15	Y	1	4	
	20	27	N	3	7	

Task	User	Time (s)	Within interval?	Errors	Clicks	Correct answer?
13	1	30	Y	0	10	NA
	2	22	Y	0	8	
	3	18	Y	0	5	
	4	40	Y	0	10	
	5	40	Y	0	4	
	6	30	Y	0	9	
	7	12	Y	1	15	
	8	15	Y	0	20	
	9	17	Y	0	8	
	10	21	Y	0	8	
	11	12	Y	1	14	
	12	18	Y	0	7	
	13	23	Y	0	12	
	14	30	Y	0	6	
	15	29	Y	0	10	
	16	16	Y	0	4	
	17	31	Y	1	10	
	18	20	Y	0	8	
	19	45	Y	0	11	
	20	19	Y	0	9	
14	1	9	Y	0	0	Y
	2	4	Y	0	1	Y
	3	5	Y	0	1	Y
	4	8	Y	0	0	Y
	5	7	Y	0	1	Y
	6	9	Y	0	0	Y
	7	6	Y	0	0	Y
	8	7	Y	0	0	Y
	9	15	N	2	1	Y
	10	9	Y	0	0	Y
	11	14	N	1	0	Y
	12	7	Y	0	0	Y
	13	10	Y	0	0	Y
	14	17	N	2	1	Y
	15	11	Y	0	0	Y
	16	8	Y	0	0	Y
	17	9	Y	0	1	Y
	18	10	Y	0	0	Y
	19	14	N	2	0	Y
	20	12	Y	1	0	Y

Task	User	Time (s)	Within interval?	Errors	Clicks	Correct answer?
15	1	5	Y	0	3	NA
	2	7	Y	0	3	
	3	5	Y	0	3	
	4	4	Y	0	3	
	5	6	Y	0	2	
	6	7	Y	0	3	
	7	5	Y	0	3	
	8	4	Y	0	3	
	9	7	Y	0	3	
	10	6	Y	0	2	
	11	9	N	0	3	
	12	7	Y	0	3	
	13	6	Y	0	2	
	14	5	Y	0	3	
	15	8	N	0	3	
	16	5	Y	0	2	
	17	5	Y	0	3	
	18	4	Y	0	2	
	19	7	Y	0	3	
	20	6	Y	0	2	
16	1	68	Y	1	3	Y
	2	52	Y	1	4	Y
	3	106	Y	2	10	Y
	4	90	Y	0	8	N
	5	110	Y	0	5	N
	6	150	N	0	10	Y
	7	124	N	2	5	N
	8	55	Y	0	6	N
	9	67	Y	1	9	N
	10	135	N	1	11	N
	11	129	N	2	7	N
	12	70	Y	0	4	N
	13	75	Y	0	6	Y
	14	131	N	3	9	N
	15	62	Y	3	7	N
	16	85	Y	1	10	Y
	17	125	N	0	13	Y
	18	90	Y	0	8	Y
	19	141	N	1	12	Y
	20	100	Y	1	7	Y

E.3 Overall questionnaire results

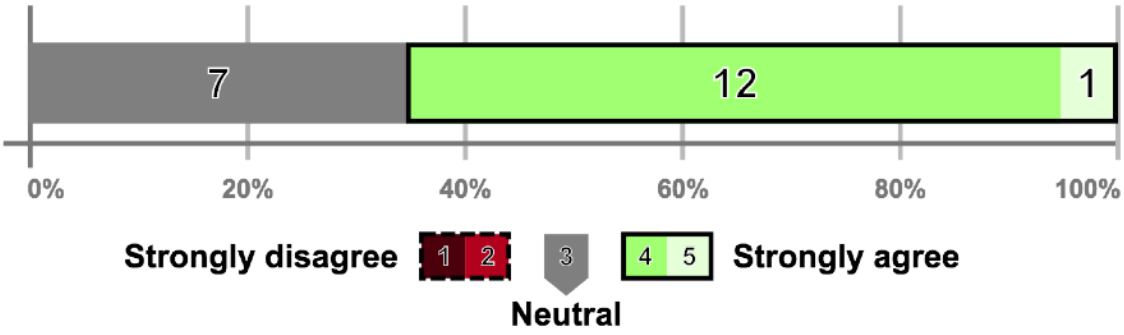


Fig.E.1: Ease of use for different visualizations

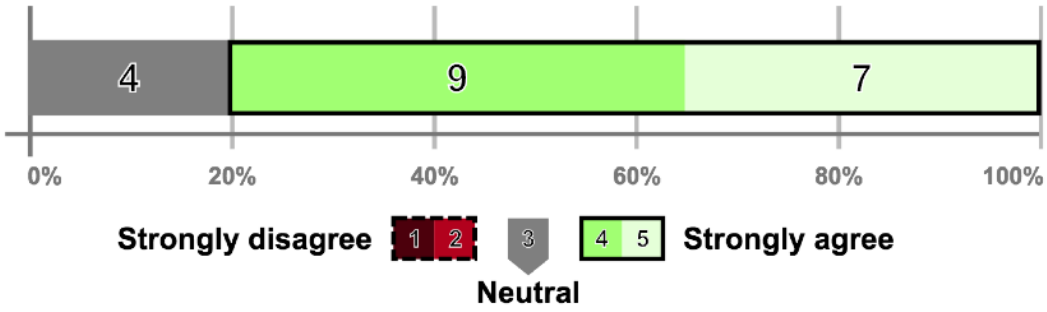


Fig.E.2: Ease of understanding for different visualizations

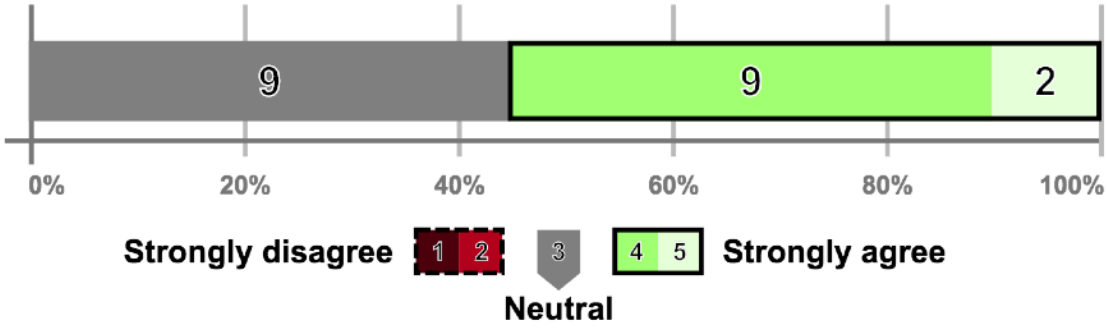


Fig.E.3: Ease of use for the menus

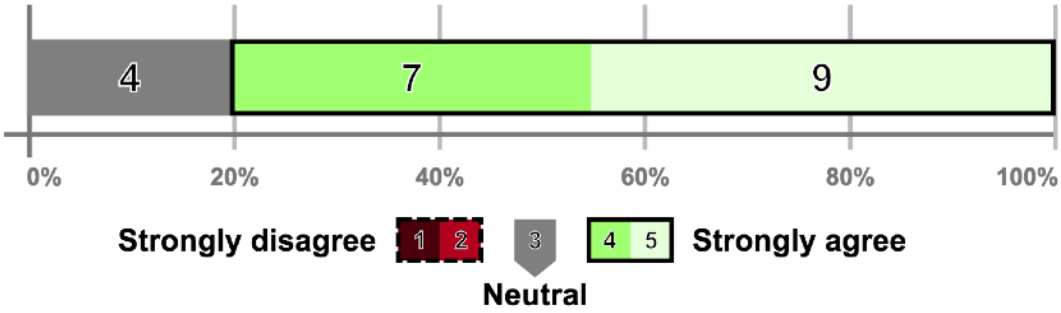


Fig.E.4: Ease of understanding for the menus

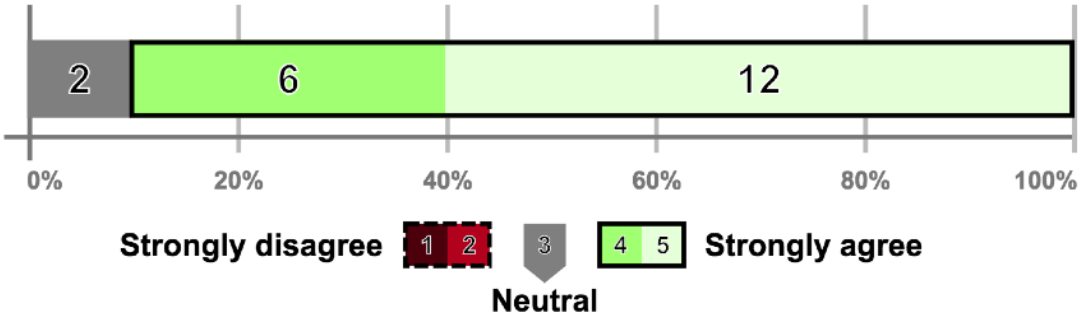


Fig.E.5: Ease of use for the timeline slider

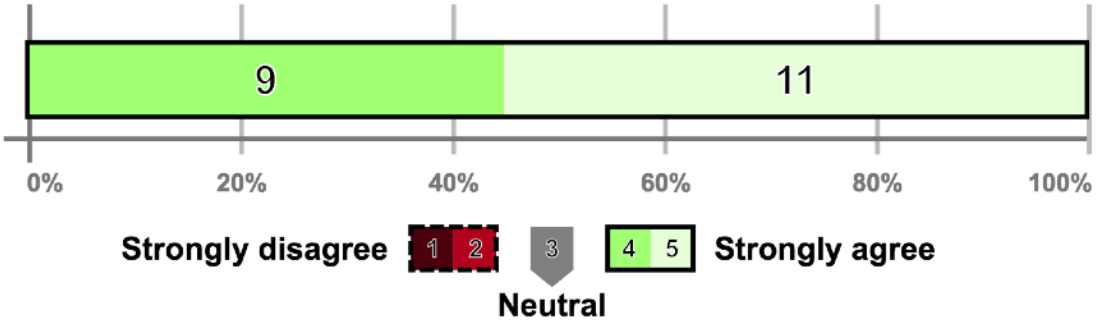


Fig.E.6: Ease of understanding for the timeline slider

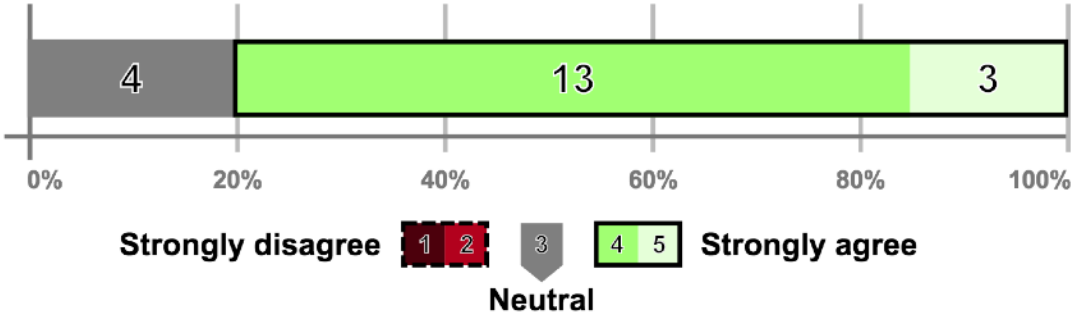


Fig.E.7: Ease of use for the overall application

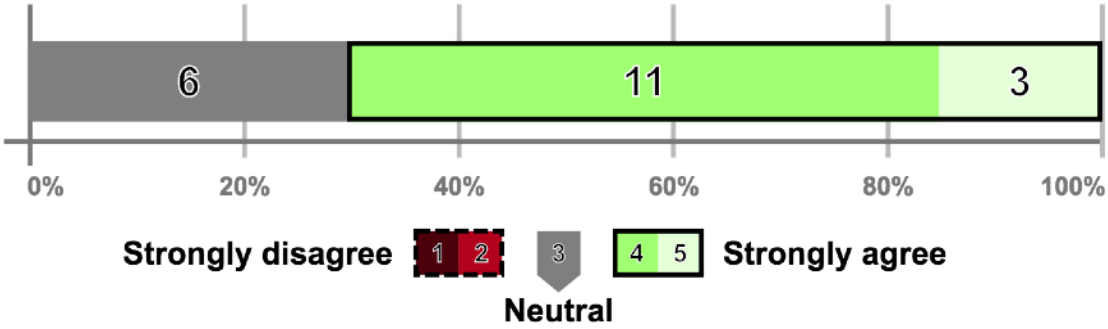


Fig.E.8: Ease of understanding for the overall application

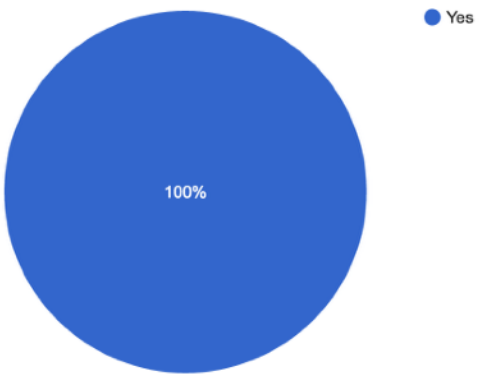


Fig.E.9: Smartphone possession

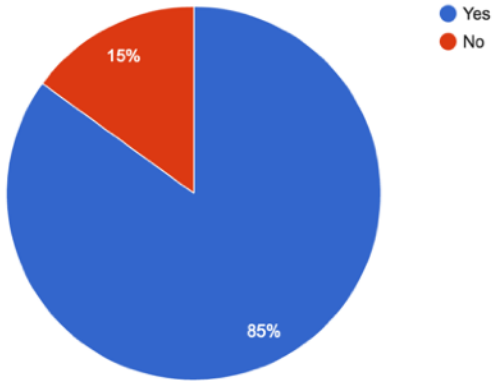


Fig.E.10: GPS use

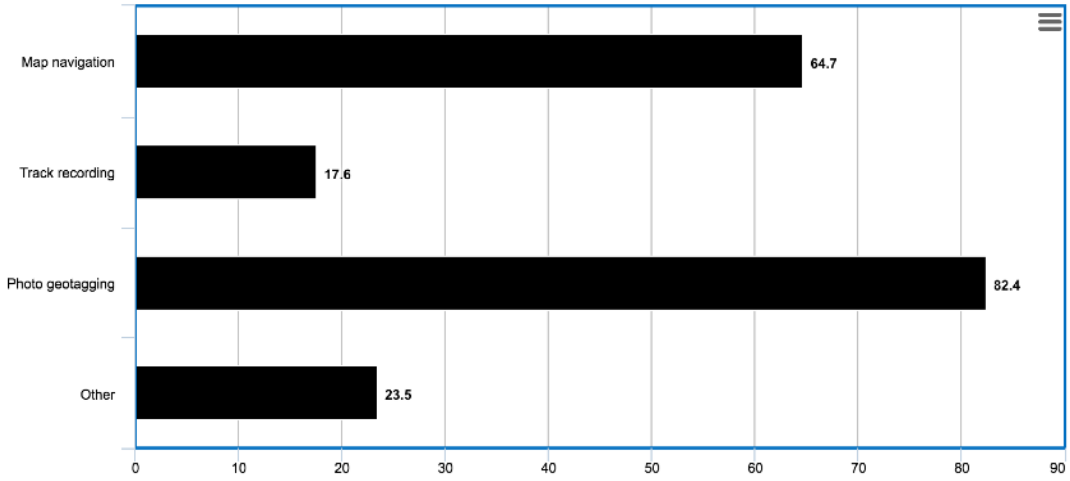


Fig.E.11: GPS use situations (%)

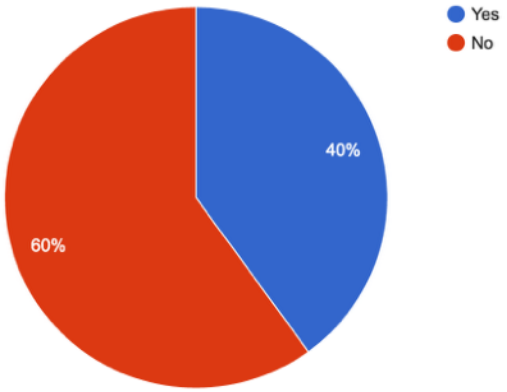


Fig.E.12: Would users track spatio-temporal data

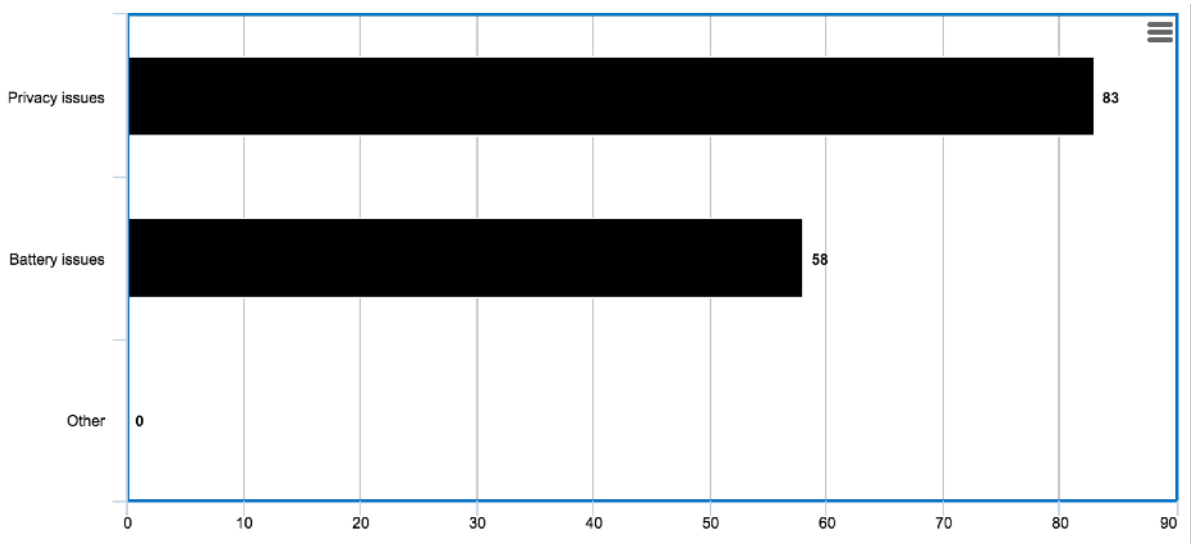


Fig.E.13: Reasons to prevent user tracking (%)

Question / User	1	2	3	4	5	6	7	8	9	10	SUS Score
1	3	1	3	2	4	2	4	1	3	2	72.5
2	4	2	4	1	4	1	3	1	4	1	80
3	4	2	4	2	3	1	2	2	5	2	75
4	3	1	4	2	4	2	3	2	4	2	72.5
5	3	2	4	2	4	1	3	2	3	2	70
6	3	1	4	2	4	2	3	2	5	1	77.5
7	4	1	4	1	2	1	2	1	5	2	77.5
8	3	2	4	2	4	3	3	1	4	2	70
9	3	1	4	2	4	3	4	2	5	1	77.5
10	4	2	3	2	4	2	4	2	4	2	75
11	3	2	4	1	4	1	2	1	3	1	82.5
12	3	2	4	2	5	2	5	1	4	1	80
13	3	1	3	2	3	1	3	1	2	1	70
14	4	2	4	2	3	2	4	1	4	2	75
15	4	2	4	1	4	1	4	1	4	2	82.5
16	3	2	3	2	4	3	3	1	3	1	67.5
17	3	3	4	2	4	1	2	1	4	1	72.5
18	3	2	3	1	4	1	3	1	4	2	75
19	4	1	4	1	4	1	3	1	4	1	85
20	4	2	3	2	4	1	4	2	4	1	77.5
											75.5

Table 5: SUS summary