TÉCNICO LISBOA

Follow My Steps

Rodrigo Capela Ferreira França Martins

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Daniel Jorge Viegas Gonçalves

Examination Committee

Chairperson: José Luís Brinquete Borbinha

Supervisor: Prof. Daniel Jorge Viegas Gonçalves

Member of the Committee: Sandra Pereira Gama

May 2018

Acknowledgments

Throughout these years I have been helped by dearest and closest people, who have been supporting me in this long and complex phase of my life. I would like to express my sincere gratitude to my family, teachers and friends, as without them I would not be able to accomplish this work.

First, I address to my family who had back me up on this journey, especially my parents and sister, with love, perseverance and patience. To them, thank you for supporting my convictions and for providing me with your determination, dedication and inspiration to achieve my goals.

I want, as well, to manifest my gratitude to my supervisor, Professor Daniel Gonçalves, for his admirable teaching, tutoring, guidance and incentive throughout this thesis.

To my friends, thank you for all moments inside and outside the institution, for all the fun, dramas and friendship.

Finally, my special thanks to Mafalda Antunes for all her help, support, patience, incentive and love provided during all the years we have been together.

Abstract

Computer storage has become significantly cheaper and we are facing great developments regarding sensing technologies allowing to efficiently understand personal activities and experiences. These advancements are supported by the growth of the quantified self-movement, in which life activities are tracked by using wearable sensors in the hope of better understanding human performance in a variety of tasks. Unfortunately, despite being easy to gather data, displaying it in a personal relevant way, avoiding scalability issues rising from the large amount of data collected, it has been a predicament for many interface developers.

This work is focused on overcoming these issues and to provide a simple and elegant user interaction. Its main goal is to understand which are the best visualizations that allow people to recall experiences, within a certain period of their lives, in a personal relevant way. The system structure is divided into two versions designed for PC and mobile platforms, which are focused on time and location, respectively.

Follow My Steps browser version implements a customizable interface that enables the addition and the removal of interactive and personalized visualizations, which can be filtered without compromising the scalability and the holistic view of the interface, that displays the submitted user's data.

The Follow My Steps mobile version is a smooth application that provides a variety of information, related to visited places by users, based on their current location.

The results, generated by the usability tests, reveal that the system is easy to use, functionalities were well integrated, without inconsistencies, and the interface presents a simple design. However, a set of older participants felt the need of a technical support while using the web interface for the first time.

Keywords

Lifelogging, Information Visualization, Dashboard, Quantified self

Resumo

O armazenamento digital tem se tornado significativamente mais barato, e estamos a observar grandes avanços nos campos das tecnologias de detecção que captam, eficientemente, as actividades e experiências das pessoas. Estes avanços são suportados pelo crescimento do auto-movimento quantificado, em que as actividades que praticamos durante a vida são monitorizadas através do uso de sensores portáteis com o objectivo de perceber melhor o desempenho dos humanos numa variedade de tarefas. Infelizmente, e apesar da facilidade em recolher dados, exibi-los de uma forma pessoal relevante, evitando problemas de escalabilidade que surgem com o crescimento dos dados recolhidos, tem imposto grandes dificuldades para os programadores de interfaces.

Este trabalho está focado em ultrapassar estes problemas e proporcionar ao utilizador uma interação simples e elegante.

O seu objectivo é entender quais são as visualizações que melhor se adequam a recordar experiências vividas, dentro de um certo período de tempo, de uma forma pessoal e relevante. A estrutura do sistema está dividida em duas versões, desenvolvidas para as plataformas de PC e de mobile, focadas no tempo e na localização, respectivamente.

A versão do browser de Follow My Steps implementa uma interface customizável que permite a adição e remoção de visualizações personalizáveis e interactivas, que se podem filtrar sem comprometer a escalabilidade e a visão holística da interface, que exibe os dados submetidos pelo utilizador.

A versão mobile é uma simples aplicação que oferece uma variedade de informações, relacionadas com os lugares visitados pelos utilizadores, baseadas na localização actual.

Os resultados, gerados pelos testes de usabilidade, revelam que o sistema é fácil de usar, que as funcionalidades estão bem integradas, sem inconsistências, e que a interface apresenta um design simples. No entanto, alguns participantes mais velhos sentiram a necessidade de ter apoio técnico enquanto usavam a interface pela primeira vez.

Contents

List of Tables

List of Images

1. Introduction

Life is a continuous chain of experiences, which constitutes important factors according to which we define ourselves. Our brain turns each experience into a memory, allowing us to encode, store, retain and, subsequently, recall information and past experiences to affect or influence our current behavior. Unfortunately, and although our brain is able to store huge amounts of information, we can only recall a few portions of the memories that we experienced. Consequently, it only selects the most important and relevant ones. Most of these experiences are perceived on important occasions, such as journeys and family reunions, and as the time goes by, we tend to forget them. Therefore, we started to reinforce these memories by collecting souvenirs, taking photos, writing on diaries, among other.

Nowadays, with the exponential growth of the technology that we use in our daily lives, we can collect and store massive amounts of personal data and use it to study patterns and improve our self-awareness. Intelligent devices, like smartphones, are equipped with sophisticated sensors. These sensors, together with advanced computing power and network connectivity, offer opportunities to track everyday experiences, accurately and automatically, generating lifelogs that have great potential in providing valuable information about us.

The process of tracking the data that is generated by our own behavioral activities is called *Lifelogging*.
Lifelogging data can be addressed for three main purposes.
The first is *Self-Reflection,* which allows users to exercise introspection and think carefully about their own behavior and beliefs. The second is *Self-Experimentation,* which is defined by the analysis of the user's own behavior in order to improve or achieve a defined state. The third is *Elicit Memories,* which allows users to see their data in order to help them recall past events.

In this work, we will be driving our attention for the third and last purpose, Elicit Memories, in which we will be addressing visualization techniques that help users remember past memories, giving a specific date or period of time. In the last decade, this area started to capture the interest of many people, due to the increase of the number of websites and applications that offer a variety of solutions and insights about personal development. Most implemented solutions are focused on a specific subject, such as exercises or sleep duration, and can only be used for a certain context.

With the advancement of the sensing technology, users tend to register and capture more personal data and to display it accordingly to their own particular interests.
Nowadays, more than 4 billion people own smartphone devices able of sensing live activities, which means that more people have the means to record their personal information in a simpler and efficient way.

However, this practice of collecting daily information produces huge amounts of data. When visualized, this data must be arranged and treated to avoid issues, such as, in scalability, and requires a massive volume of available space to store it.

To developers, who operate in the area of Information Visualization and similar fields, the urgency of creating new and improved web and mobile applications, focused on the problems that occur from the massive amounts of data, arises from the continuous growth of lifelog members.

To create an interface able of supporting data, generated by daily life activities, it was mandatory to develop methods that store, efficiently, the content uploaded by each user. Furthermore, the visualizations will need to present higher levels of abstraction in order to deploy it.

Presenting visualizations with personal data has been a predicament since its selection relies on the user personal choice. Moreover, the same set of data can be seen in more than one visualization technique.

Attending to this requirement, it was essential to create an interface, which allows to customize the dashboard and to include/remove personalized visualizations, through a set of interactions provided by the system. It also needs to customize several features including all the visualizations techniques and the styles of the entire display.

1.1 Objectives

The main objective of this work is to help people visualize their data, collected throughout a certain period of their lives. This way, they can remember and make choices upon their own experiences. It proposes to the use this data to display and report the memories of a selected period of time, such as, day, week or month, throughout the selected visualization techniques. While developing our solution, it is important to keep in mind the concepts of the lifelogging.

Our focus will be:

To study the best way to present visualizations to users, about their past experiences based on lifelogging data collected over an extended period of time, in a personal relevant way.

The aimed solution needs first to address the problems of scalability, personal semantics and holistic view. Additionally, the solution needs to be configurable, allowing users to have the best experience possible while using our system. It is important to acknowledge, when we reflect about configurability, that users provide different sorts of data, according to several factors that affect their data collection, and they must be able to select the interface visualization content.

From the main goal, we can define three phases, which we will describe below:

The **first phase** will consist of understanding which features are more relevant for users, i.e, what they consider to be more important to be displayed on the visualization interface. For example, they might prefer

to highlight what they have eaten in a certain restaurant instead of how much they paid, or to highlight how many times they have been in that cinema instead of the movies seen in that place.
We need to collect the relevant knowledge and ideas, as much as possible, we can, in order to make the best visualization to help users recall moments of their lives.
This phase will aid us to understand how people visualize our interface and what their expectations are.
The **second phase** will consist of determining what are the visualization methods offering a better fit in the features selected for the interface. The importance of selecting the right visualization techniques lies in how human brain processes the information. Using charts to display and compare the number, frequency or another measure of different discrete categories of data or pie charts to show percentage or proportional data, it helps our brain to process and to visualize large amounts of complex data. Data visualization is a quick and easy way to convey concepts in a universal way.
The **third phase** will consist of developing the system and its visualizations from scratch that deals with visual queries of the information and help users to achieve the main goal of this work.

1.2 Domain Solution

Each user will collect data from different methods and data sources, which mean that data structure will differ for each case. Naturally, data structure produced by users, who collect information using *GPS* based files, is different from data structure produced by users, who collect information using an Excel based file.
Our system was designed to parse a limited group of data structures and store it using a common and similar basis. For different types of data, we create a new parser and a new database table. On the other hand, if the structure is identical the system uses the same resources. For example, it uses the same parser and database table while analyzing and storing csv and xlsx files, decreasing the space used and reusing code.

Follow My Steps browser solution provides several customizable visualizations, based on time, that can be added to a dashboard and manipulated in order to display particular pieces of information.
Depending on the data inputted by users, the system can display the places where they have been, how frequent they were in a certain place, what they have eaten at a certain day, the distance covered, the movies watched, among others. This can be used to enhance knowledge about personal data, helping users to decide and improve their life decisions in the future.
As well as, Follow My Steps web application incorporates an operation that sends an “emailable” report, based on time, which allows to receive, periodically, an email with the information regarding the activities performed within a specific period of time. For instance, the system will send a daily, monthly or yearly report, with information regarding the activities executed on a particular day, month or year, accordingly with the user's preferences.

Besides, our project browser solution, we created a mobile version that provides insights about users' activities based on their current location. This simple app was designed to detect and analyze the position in order to ascertain if users have been at their current or near places. If a place has been visited, the app deploys the information related to the previous stopovers, displaying photos, places nearby, how many times they were in that place, among others.

1.3 Document Structure

The first section of document's structure is the ***Related Work*** section, where we present the analysis of works that are related to concepts and objectives of this system. The next section is the ***Solution,*** where we describe the server-side and client-side of our interface and the frameworks we used to implement them. On the ***Usability Evaluation*** we can observe the results we obtained while testing the interface usability with users. The last section is the ***Conclusions,*** where we reflect on the obtained results and explain the future work.

Summary of the document structure:

- **2. Related Work**: Analysis of the works related to the concepts of this project context;
- **3. Solution**: Detailed description of the system;
- **4. Usability Evaluation**: Description of the usability tests performed with the participants and analysis results;
- **5. Conclusion**: Final conclusions, reflection on the results obtained and future work.

2. Related Work

In this section, we will explore several works in the field of lifelogging visualization to understand the best techniques to represent and display different memories that allow users to recall moments of their lives. These memories are captured by several sensors in order to gather as much data as possible during the user's daily lives. Computer logs, images, location and steps are some examples of the data used in visualizations.

The data collected is, lately, analyzed and homogeneously refined to be used on visual interface. Authors present different approaches, in their works, allowing us to have several solutions to represent the same type of data. For instance, several authors collected GPS coordinate tracks and represented them in a timeline, in chronological order, while others represented them on a map.

This section is divided into five subsections.
The ***Representation using maps***, usually with a timeline, where it is presented GPS data according to the time to give information to users about the activities on a given physical and temporal space. The ***Representation using Calendars*** to provide users with an overview of the self-behavior throughout a selected period of time. The ***Representation using Images*** that allows users to recall and remember specific moments of that day through several photos. The ***Representation using Pie Charts and/or Stacked Graphs*** are used to give information about specific data in an overall perspective. The ***Representation using Personalized Visualization Techniques*** shows different methods to represent and display data.

Some works have more than one type of representation, for example, ***MyLifeBits*** uses both map and image representations to display data.
At the end of this section, we will summarize and discuss the works, with focus on how the visualizations helped users to improve their memory in order to recall past events.

2.1 Representation using Maps

These works provide visualizations where data is represented on maps. Some of them use timelines, although the design and features vary from work to work

2.1.1 Shifted Maps

Shifted Maps [1] offer a new visualization interface based on a map network that represents visited places as scaled circular maps. The time spent in a certain place is correlated with the scale factor in the same way that connection edges, that are connecting circle maps, are correlated with the movements among locations. To represent the number of visits the user made to a certain place, Shifted Maps use the thickness of the ring such as shown in the figure.

Regarding the connections among locations, the prototype offers a feature that allows users to have three different arrangements based on the geographic, temporal and frequential. About the first, geographic, places arrangement is based on their geographic positions. The second, temporal, places are arranged according to the average time it takes to go from one place to another. And the third, frequential, places with frequent connections move closer to each other than places with less frequent connections.

Shifted Maps interface allows users to switch smoothly among views, which can be explored by using a semantic zoom. It also has a time-range slider that enables the selection of a temporal period in order to make it possible to select, observe and compare spans and/or follow the growth of the network over time.

With this prototype, users can observe places and personal movements in a novel way using three different views, which can be helpful to keep track of their own activities. On the other side, there is a lack of interaction and detailed information to notify users about relevant data.

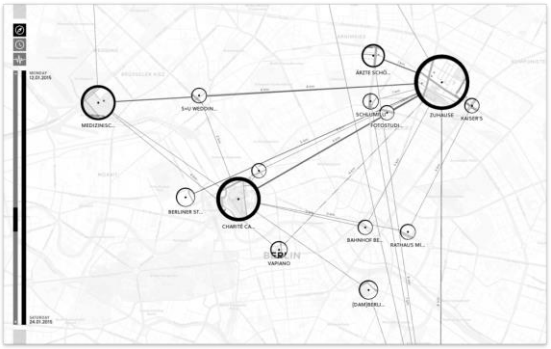


***Figure 1**: Shifted Maps Interface*

2.1.2 MyLifeBits

The main goal of *MyLifeBits* [2] is to storage personal lifetime as digital media and data, including email, calendar events, contacts, documents, audio and video. It is a system that allows any image to open a query window, when clicked, with all items linked to that same image. By combining location, arranged by time, with maps and animations authors create a visualization to tell stories using a much more compelling way. To aggregate the location with the media, they use time correlation between the pocket-size GPS data and the independent media devices data. The program extracts date/time, latitude, longitude, elevation and

precision and correlates them with date/time of the photos. Several data points of the GPS do not have a photo but, for the user, they are memories to be remembered.

The map UI element, in the MyLifeBits interface, represents places where users had taken photos as large red dots and the point location as smaller pink dots. By hovering over one of the red dots, the corresponding thumbnail of the photo pops up in a direction that avoids being partially drawn.

The user is allowed to search location through a text box and zoom in and out through the symbols '+' and '-', respectively. Photos with no location can be dragged and dropped onto the map to set their places. To avoid saturation points on a region, of the map, the interface filters out undesired clusters, using time restrictions. In order to achieve this, pictures and points are divided into collections and each one considered as a "trip", which is displayed has an element of a list, under the map.

To provide information, to the user, about traveling directions and time spent in different places, authors created a trip "replay" visualization that animates the selected trip. Users can manipulate the replay by forwarding, backwarding, stopping and changing animation duration. To avoid the overlapping problem, caused when the user selects several trips, interface uses different colors to represent each route, allowing to compare paths and distances among trips.

With this system, users can easily keep track of their information, regarding places visited throughout their trips around the world, in order to recall experiences of their lives. To select a trip is, however, a very difficult process if the user needs to search through a deep list of trips, which may lead to exhaustion and frustration.

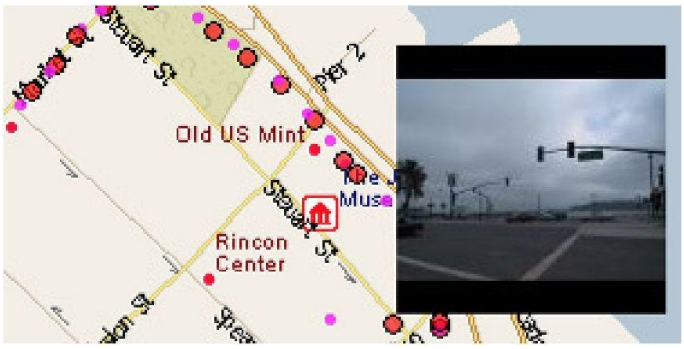


Figure 2: *Popup dots image on mouse over*

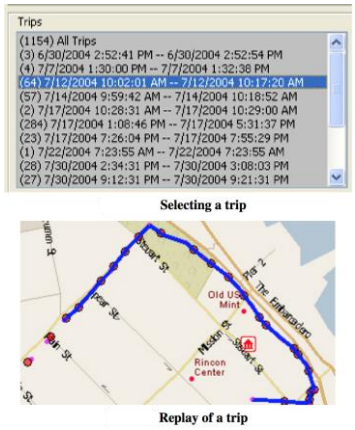


Figure 3: *Selecting and Replay of a Trip*

2.1.3 Visits

Visits [3] is an interface visualization tool developed to reflect people' episodic memories of trips and important moments of their lives. Authors used the spatial and temporal aspects of the gathered data to create a map-timeline that shows the locations as a sequence of visited places.

The goal of this approach is to support the identification, in chronological order, of the stays, the repeated

stays and their duration.

The map-timeline offers a series of circles, each containing a map segment aligned along to a linear timeline. It is represented as circles to prevent connection errors, while interpreting the areas and their sizes correspond to the time span on the timeline, which means that the radius of each circle is proportional to the time spent on each location. Visits are also made of another important feature that authors called of overview-map, which allows users to see each place of the map-timeline connected to its respective location on the world map.

Regarding the interactions, this tool allows the user to see, on the timeline, the beginning and the end of a certain stay by moving the mouse over a circle. In the lower right part of the screen there are two sliders that determine the distance threshold and the controls of the frequency of the location measurements. The user can also zoom to focus the timeline on a specific section.

This solution offers a simple and effective way to represent visited locations over a timeline, allowing users to keep track of the places where they have been, in a certain time. Unfortunately, the biggest problem is based on the scalability, where circles could become too small to interact, if there was a huge number of visited places.

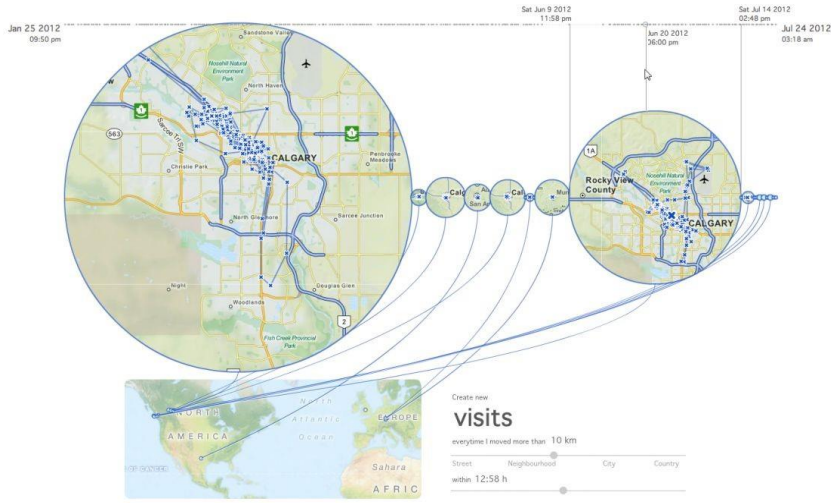


Figure 4: *Visits showing a location history of six months using a map-timeline approach*

2.1.4 Smart Reality Testbed

The main goal of this *work* [4] is to collect huge amounts of daily life data through devices that users carry, such as smartphones, and both designing and implementing an interactive visualization tool helping them to review and analyze their lifelog data.

Authors developed the Smart Reality TestBed, which is a framework that allows to record the logs in smart devices, to collected data such as positioning, social, motion, environment, device, interaction, among

others.

The extracted data was submitted to a clustering algorithm to erase overlapping circles that lower the overall quality of the visualization. The visualization consists of three parts, in which the first is the information panel showing the current data, the currently focused app, the list of apps used on the current day and a calendar control to select the date to view detailed log data. The second is the timeline view that displays white blocks when the participants are using the phone at that time and blue block when they are not. The third part is the map view that shows the user's app usage patterns as black semi-transparent circles with the radius proportional to the app usage duration in that location.

This solution offers a high level of detailed information, which gives a lot of useful and personal information to the user. On the other hand, there are scalability issues, a large number of visited places implies a huge number of black circles and that may cause difficulties when the user is trying to observe the map.

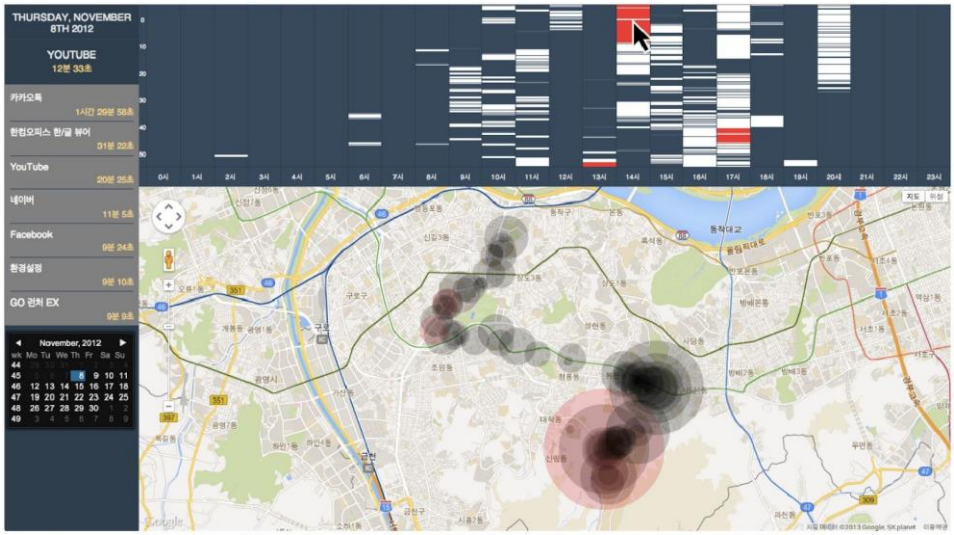


Figure 5: *Visualization design of our exploratory visualization tool*

2.1.5 Footprint Tracker

The *Footprint* Tracker [5] is a tool that helps users recall and review their daily activities and experiences, within a period of time. It was developed to support three kinds of lifelogs: the visual data, represented as video, the location data, represented as location and transitions among places, and the social context, represented by the SMS and calls made and received throughout the day. Authors developed an interface that includes two main sections.

The first section consists of the data pane, which displays the visual and the temporal data, and location pane, which displays both important locations as well as trajectory reminders. The second section consists of a timeline pane, with three interactive bars, allowing users to select the desired period of time, enabling the correspondent images and the respective GPS locations to be loaded on the first section.

The tool strong points are based on the results where authors detect that people were able to improve their recall capabilities, regarding their daily activities, after using the Footprint Tracker. On the other hand, there are results where users suffer from problems such as telescoping, i.e., the incorrect recall of the time when an event occurred.

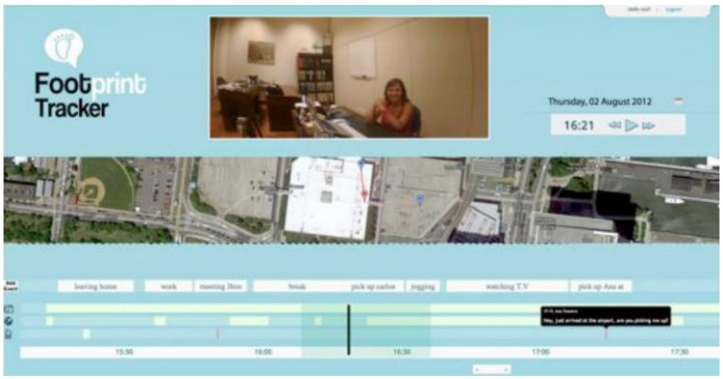


Figure 6: *Footprint Tracker Interface*

2.1.6 SnapTracks

SnapTracks [6] is a visualization interface allowing users to look to their past, in novel ways, to reconstruct where they lived and to help them recall daily life details. Authors increased streams of lifelog images with geographic data to examine how different types of data might affect memory.

The gathered data was collected by 15 participants with a SenseCam, which is a wearable digital camera that captures images at time intervals and when one of the sensors of light, motion or temperature, is activated. SenseCam also has a GPS that capture the coordinates of the current location.
These data were, then, used in three lifelog visualization tools:
The first is snaps, which is a picture viewer embodied in a web browser allowing people to view their pictures sequentially on a day by day basis. The second is tracks, which uses the GPS data to show users their routes in a real-time custom Microsoft Virtual Earth map. Users have the option to display, on the same map, multiple days and they can access timestamps that are shown when they have waypoints, to identify places and remember activities at those locations. The third is the Snaptracks, which combines both GPS coordinates and the pictures on a map visualization in order to display users' routes and waypoints associated with photos, that were taken on the respective places.
The goal of this visualization is to use the map as an overview method, allowing users to get more detailed information using their personal images. They can select more than one day to display several routes and, by hovering over a waypoint, the corresponding photo pops up.

This prototype enables users to recall and reconstruct past events in a simple and easy way. Unfortunately, it lacks on scalability. For example, too many waypoints make it hard to select one of them.

Figure 7: *Sequential images of everyday activity*

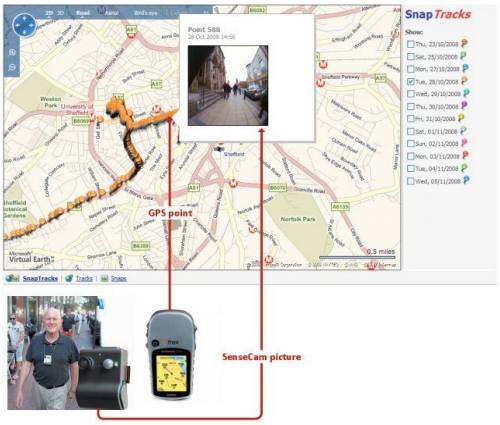


Figure 8: *SnapTracks visualization*

2.1.7 SoundBlogs

This *work* [7] proposes a system that records audio, continuously, through a smartphone or a web recorder. The gathered data is, then, submitted to a segmentation algorithm to convert long audio data into smaller clips using Switching Linear Dynamical Systems that falls under the Dynamic Bayesian Networks Framework.

After segmentation, clips are indexed and marked down with acoustic and semantic tags, in order to achieve them into a long database for future retrieval. To mark automatically down the path from a query sound, all tags are evaluated based on the given weights among nodes. The shortest path is assigned to the tag since it is considered the best match. The implemented interface, an Android Platform, allows the user to record audio for long and short durations. By plugging in the smartphone into the computer application, the program starts the segmentation automatically, as well as the annotation.
Later the app can be used for instant blogging the interface and sharing interesting incidents or sounds. The interface captures the current location of the device, displaying it as an icon and, by selecting it, the user can read a brief description, archive or share the audio.

This approach provides the user with a simple and effective way to share and recall moments through sounds in a specific location. However, if there are a huge number of recorded audio files the data may overlap each other, becoming impossible to listen to some clips.

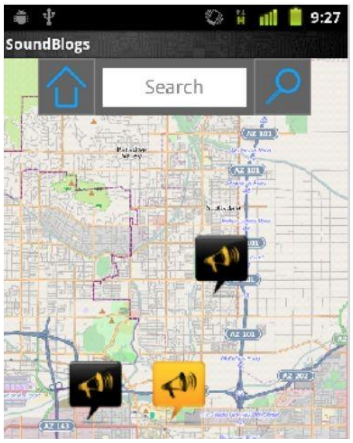


Figure 9: *Interface on an Android-powered smartphone*

2.2 Representation using Calendars

These works provide visualizations where the data is represented by calendars.

2.2.1 Everyday Visualization

In this *paper* [8] authors propose to aggregate and visualize data from users in order to help them have a better understanding of themselves. Since this data, which we will describe later, was collected by a set of sensors, it had different precisions, so authors needed to standardize it. They manage to gather:

The steps, which is the representation of the number of steps taken on a given period of time by the user. The sleep, which represents the number of hours that the user slept each night. The location, which can be seen as a set that contains timestamps, coordinates and the name of the location where the user was. The climate, which can also be seen as a set that contains the temperature and the climate conditions of the location where the user was. The emails sent and received that includes the timestamp, subject and message body of each email sent and received by the user. The tweets that include timestamps, the mentions ("@") and the Twitter messages sent by the user. The appointments, which include the start and end time of the appointments, title and description. Finally, the work, that is composed of the time spent to achieve main goals and all the logging activity, organized by categories.

In order to make the Everyday tool more intuitive, and to represent the data mentioned above, authors used three visualization metaphors that allow users to watch their activities more easily.
The first metaphor is the calendar, which is composed of daily cells containing information regarding a single day (the date, the temperature, the climate condition and the steps taken). The date represents the current day of the month. The temperature, supported by color, represents the average temperature during that day. The climate condition, represented by an icon, gives information about the weather and the steps,

supported by color and width of a bar, represents the number of steps taken. The calendar provides, to users, an overview of their self-behavior along several days. The second metaphor is the clock, which provides detailed information regarding one single day. The clock is divided into two parts, the outer part that has 24 slices, each representing an hour with the number of steps taken in that period of time, and the inner part where we can see bubbles representing the amount of time spent on different projects. The third, and last, metaphor is the map showing circles where the radius and the color saturation is proportional to the number of steps taken, per day, on that respectivelocation.

This work helps users, through three different visualization techniques, to visualize and understand patterns in their daily lives activities in a novel and simple approach.
Unfortunately, there are some problems with scalability on the map technique, where the areas can overlap when the number of steps or the hours spent on projects, during a day, is very high.

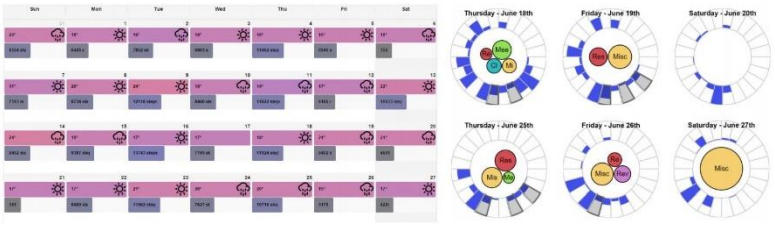


Figure 10: Calendar view and Clock view

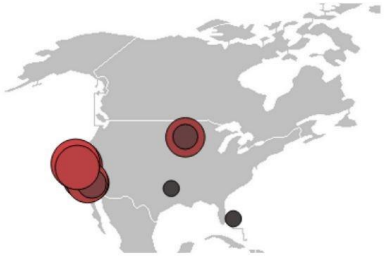


Figure 11: Map showing number of steps

2.2.2 Data in Everyday Life: Visualization time-varying on a Calendar

The main goals of *Data in Everyday Life* [9] visualization are to provide personal context, to help users with data interpretation and to lower the barriers to accessing data without interfering with daily routines.

Authors created a web app that works as an online digital calendar, synchronized with calendar events and also fetching data feeds in real time. The calendar is composed of events, by a data stream visualization layer, which is an approach to use data as a media stream, and by interaction buttons allowing users to personalize it. Users can choose the visual encoding for displaying the data, for example as a line graph or color encoding, and adjust the opacity of the data stream visualization layer to balance between the calendar ambience and the layer. They performed two pilot field studies and, in both cases, the goal was to collect feedback on the interface design approach and to reveal problems within the developed prototype. In the first study, the web app was deployed to people living in an eco-friendly smart home with the data source connected to their electricity meter. In the second study, the calendar was deployed to ten undergraduate students as a part of psychological seminar course.
According to authors the results were encouraging and users found the aggregation between the data

stream and the calendar as an easy approach to see their lives routines, allowing them to keep track of relevant data. Unfortunately, there are some inconsistencies in functionalities and looks.

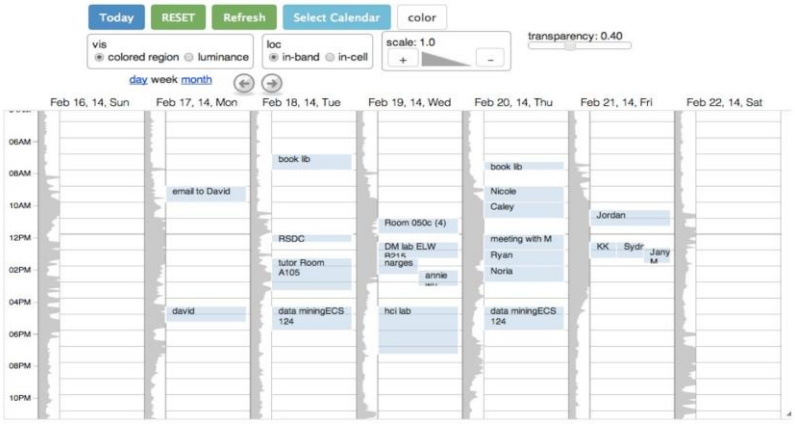


Figure 12: *Data in Everyday Life Interface*

2.3 Representation using Images

These works provide visualizations where data is represented by images.

2.3.1 Personal Lifelog Visualization

The main goal of this *work* [10] is to create visualization tools to support users access to lifelogs.
To gather the data needed to fulfill it, people working into this used an acceleration sensor, which gives the physical position of the user, a GPS providing geographic location, a Bluetooth giving social context, a Wi-Fi providing indoor location and cache location, a camera allowing automatic photo/video capture and a speaker giving environmental sound noise/level.

In order to explore the match between the user's needs and data visualization, three scenarios were created, with UI patterns, wireframes and interface prototypes, to support different user behaviors and contexts.
The first scenario, the visual Diary, provides photographs as a summary of the user's daily log. Each event is represented as a grid and its size provides a visual cue to the event importance. The second scenario, the Social Interaction Radar, is composed of a sensor that captures social activity that uses a coxcomb visualization technique, helping users to understand the whole and its individual simultaneously. The third and final scenario, the Activity Calendar, provides users detailed understanding of their physical activities, encoded as color, where the darker areas indicate more activity involved in a given day. This level is calculated using the data collected from the GPS and the accelerometer.

The advantages of the three visualizations are based on the simplicity of each technique used to represent the user's personal data.
The disadvantages are based on the lack of information and on scalability (events with a small importance won't be big enough to give information about it).

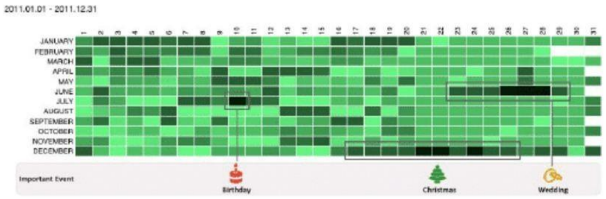


Figure 13: *Heatmap calendar*

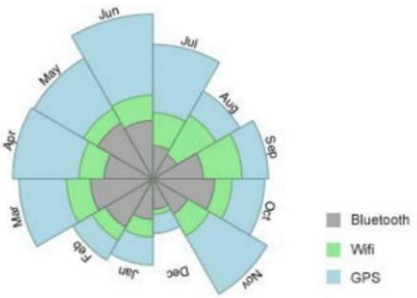


Figure 14: *Social Interaction Radar*

2.3.2 SenseSeer

SenseSeer [11] is a web-base lifelogging application that tracks daily life activities, through their smartphones, to provide users a dynamic overall view and direct feedback of the collected data. In order to accomplish that, this framework is divided into three main parts:
Data collection, which passively captures photos that users are engaged with; data enrichment, which is composed by three processes, reduction, compression and rendering; and data presentation, which is a web-based interface with real-time visualization of users lifelog data. Authors developed two interaction designs both for smartphones and tablets.

On the smartphone a compact view of the whole day's visual lifelog is displayed to fit on the full screen, where the user can see an overview summary of the day through photos and each one is encoded by a color border, representing the type of physical activities associated with the photo when it was captured. By tapping the grid more details are shown.
On the tablet, authors used a timeline metaphor, where the x-axis represents time and the y-axis the event's important score that is divided into two main parts. In the bottom part users can find a summary of the day, week, month or year through a set of representative photos and in the upper part they can find all the photos of each event on a carousel display, where the border color indicates the type of physical activity performed.

This tool allows users to recall past events, through photos, in a simple and effective approach. On the other hand, there is a lack of information, for example, photos do not have any details attached which can harden the process of identifying a certain picture.

Figure 15: *Overview on smartphone*

Figure 16: *Visualization interface for tablet devices*

2.3.3 SenseCam

SenseCam [12] is a wearable device which automatically captures up to 2500 photos per day. In this work, authors identify three stages in the process of capturing and structuring SenseCam images and, then, displaying them to a UI, allowing users to review and recall moments of their life. This system employs several image analysis techniques to automatically structure and index the captured photos. This analysis can be divided into three main processing elements, which are the Event Segmentation. This involves the segmentation of all photos into distinct groups or events, the Landmark Photo Selection, consisting on the selection of a landmark photo for each event, namely a single photo from within an event representing the event's content. And, finally, the Calculating Event Novelty, which calculates how important each event is.

SenseCam interface has a concise overview of all events, occurred in a day, presented on a single page. The number of events shown is 20, by default, but the user can drag the slide bar to adjust it.
Photos have different sizes, depending on the novelty value, and a mini calendar is provided, where the user can select any particular day, week, month or any number of dates to be displayed.
They can also move the mouse cursor over an event to see all photos associated with it in a slideshow, where they are able to mark down, edit or delete if wanted.

This approach provides an interface exploring the events of a user's lifetime, highlighting the more important ones and helping them to recall important moments in our lives.

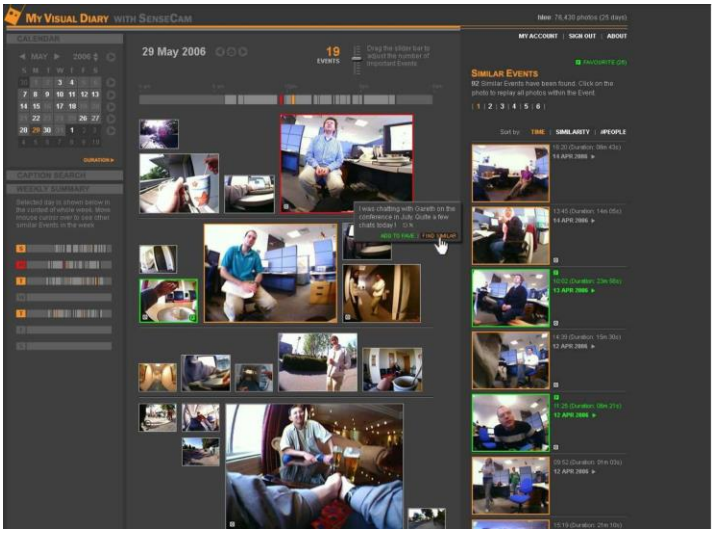


Figure 17: *Interactive SenseCam Photo Browser*

2.4 Representation using Pie Charts or Stacked Graphs

These works provide visualizations where data is represented through pie charts or stacked graphs.

2.4.1 PieTime

This *tool* [13] was developed to help users reflect on their life patterns, through a visualization presenting communication activities aggregated in levels. Authors focused their attention on the temporal patterns and away from the details of specific events. To accomplish their goal, they developed “The Pie”, using the clock metaphor, to emphasize sections as part of a whole. Each section, or slice, represents the number of timestamps corresponding to each incoming and outgoing communication. The radii is scaled in order to have the slice, representing maximum activity for that timescale, extended to the edge of the invisible circle that encloses the pie.

The Media chosen, from several sources, to be represented, was the email, having into account its ubiquitous and the amount of prior research that has used it, and the phone bell, having into account the captured data that is not often used in reflection tools.
This work allows users to keep track of the number of emails sent and received, as well as the text SMS and calls made throughout each hour of the day, to compare their behavior across communication media. Unfortunately, this system offers a very simple visualization that lacks information that could be relevant to the user.

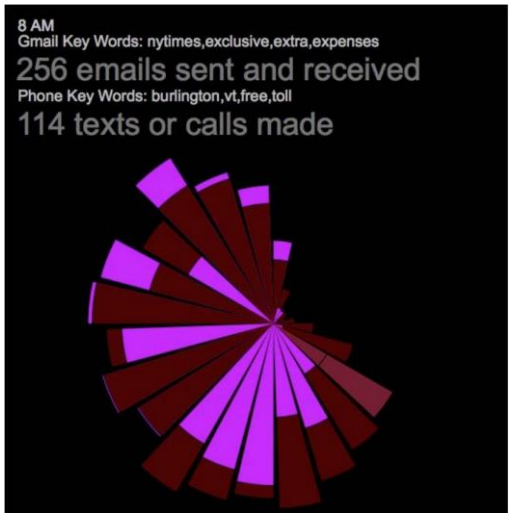


Figure 18: *PieTime, showing hourly email and phone activity*

2.4.2 Muse

Muse (Memories USing Email) [14] is a tool that combines data mining techniques and an interactive interface to help users find and navigate email archives. This tool analyzes each archive and generates one of the following cues: communication activity, recurring name entities, sentimental words and images attachments.

The first cue, communication, employs a group mining algorithm, that satisfies several properties, and it is created automatically by analyzing co-recipiency in messages. The second cue, named entities, is created to carry out rich associations in the user's episodic memory. The best cues are names, including people, places, organizations, and so on. The third cue, sentimental words, is used with a simple sentimental technique to let users quickly browse messages by the sentiment associated with them. The fourth and last cue, images or pictures, are useful because the vividness of images provides strong cues to memory.
Users can activate multiple views in different windows or tabs of the browser, to encourage multiple chains of exploration, and they can follow cues by clicking on a name in the monthly summaries or by selecting a point in a graphic visualization. The system responds with a message view displaying all the content of the actual message (header, text, attachments, etc.).

This approach has a very simple visualization and it is very easy to understand it, but it lacks relevant information.

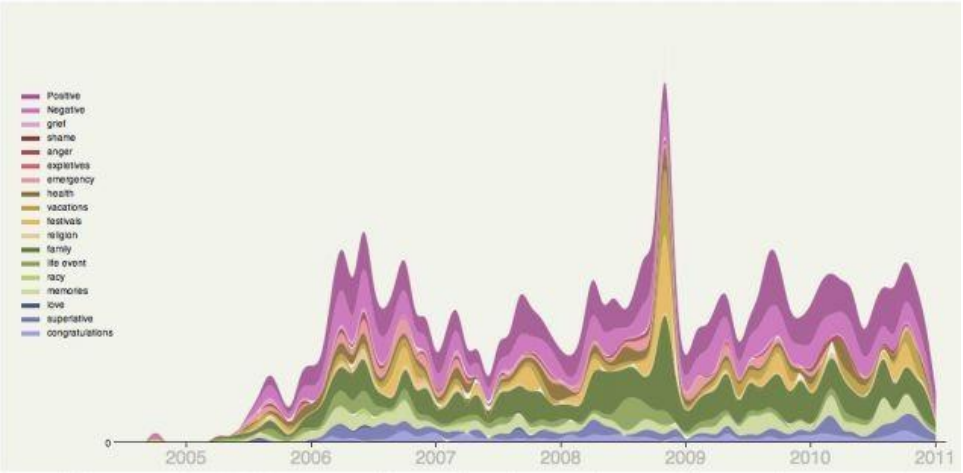


Figure 19: *MUSE Interface*

2.5 Representation using Personalized Visualization Techniques

These works provide visualizations where data are represented through personalized visualization techniques.

2.5.1 Egocentric Visualization Design

This *paper* [15] was developed to visually characterize each individual research career path and to compare them with other's paths.

To achieve it, authors propose an approach that uses a representation of a 2D tree, which includes several features to represent the information. The features include the trunk size giving information about the number of articles authored by this researcher. The trunk sides that can be mapped to some natural binary division of the data. The tree branches used for encoding time information. The leaves that can represent a particular kind of collaborative relationship. The leaves color and size indicating quantitative information, such as, paper length, number of co-authors, or number of citation received. And the fruit that can be used to highlight particular aspects of an article. Finally, the trunk gives a view of a researcher overall career.

Authors propose several ways to show each element in several case studies in order to bring out particular aspects of a research development and to help comparisons.

The advantages of this approach are the effectiveness in presenting multidimensional information and the way that allows individuals to keep track of their work throughout the years.

The disadvantages are based on scalability. Too many articles would result in too many overlaid leaves

that could, at some point, make them impossible to differentiate, and the lack of information displayed to the user in the visualization.

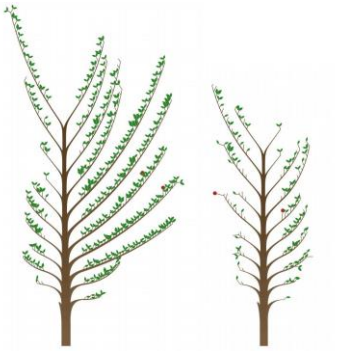


Figure 20: *Two active researchers. Each branch encodes two years of publications*

2.5.2 QS Spiral

QS Spiral [16] emerged as a proposal to an interactive visualization technique that aims to capture the periodic properties of quantified self-data, allowing the user to interact and interpret the data through simple touch-based gestures.

This tool uses a clock-dial metaphor composed of rings representing the present in the outer ring and the past towards the center, which is captured by the continuous spiral timeline with circles corresponding to a time span that could be hours, days, weeks or years.
The arcs are proportionally thicker as their approach the outer parts, allowing most space to be allocated for information in the outer ring, and the color encoding can be shown to represent data points, as well as symbols or even animations. The user can interact with the visualization through a single tap gesture, that allows to highlight specific data elements, a pinch gesture allowing to zoom into a part of the data, a swipe gesture, to pan, such as a zoomable map metaphor, and the double tap gesture that switches different time views.

This solution provides a very simple visualization technique allowing users to understand personal data. Unfortunately, there is a lack of scalability, which can become a problem if the user wants to see long past events.

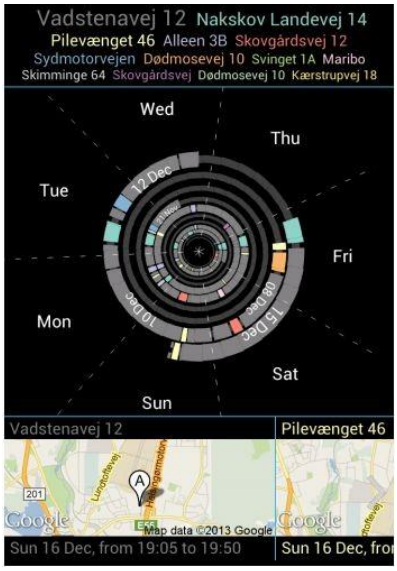


Figure 21: *A week view showing the geolocation data captured over a 4 month duration.*

2.5.3 AffectAura

AffectAura [17] is an emotional tool allowing users to reflect on their emotional states over long periods of time. It continuously logs audio, visual, physiological and contextual data in order to create a classification scheme, to predict the sentimental state and an interface for users' reflection.

The multimodal sensor is a combination of portable and workstation based sensors, developed to collect data for prospective users and to capture their emotions. It is made of a webcam, to capture and analyze facial actions and head motions while users are at their desktop. A Kinect, to capture the posture of users in order to understand their engagement while they were at the desk. A microphone, to capture the speech that can be a strong pointer to emotions. An EDA Sensor, which stands for Electrodermal activity sensor to detect arousal. A GPS, to capture the location, which is an important variable, associated with affective state. A File Activity, to understand the level of engagement as well as the affective states. And a calendar scraper, to extract times and meetings attendees scheduled.

The AffectAura's interface incorporates two components, the AffectAura prediction engine and a timeline allowing users to reflect on their data. This timeline is the main feature of the visualization and captures the movement and flow of AffectAura represented by a series of bubbles. By hovering over these bubbles, users can reveal details of their activities and iterations associated with that time.

Each circle has properties according to three behaviors shown by users.

The Valence, represented by color, uses the "hot/cold" metaphor allowing three possible states, pink for positive emotions, purple for neutral emotions and blue for negative emotions. The second is the Arousal of the user, represented by the shape, where a circle or a "burst" was chosen to represent calm and "pumped up", respectively. The third, and last, is the Engagement, represented by opacity, with higher

opacity corresponding to greater engagement.
The display shows a day divided into hour intervals (columns).

These tool greater features rely on the usability and design allowing users to recall and reflect on their past emotions, in a very simple way.

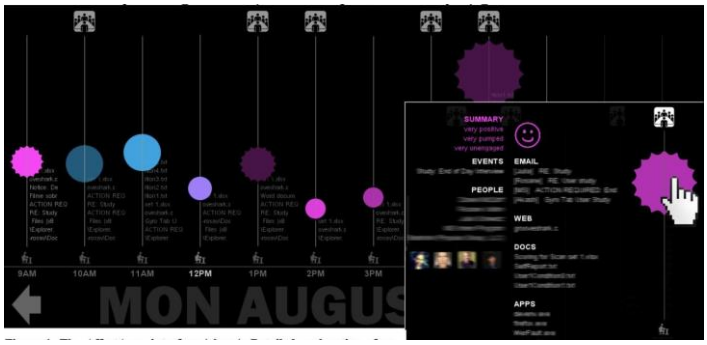


Figure 22: *The AffectAura interface*

2.5.4 LastHistory

LastHistory [18] is an interactive visualization tool that displays music, listening histories as well as contextual information gathered from personal photos and calendar entries. The term listening history describes an account of consumed music by a single person and has become relevant recently, as the digitalization of music.

A perfect listening history is a complete chronological collection of musical items, where each one is a pre-existing piece of music that can be identified based on some of its attributes and has been heard by the user, at least in parts during the recorded time interval.
In order to receive details regarding particular parts of the listening history a user interaction is required. The user can zoom in and out, by scrolling with the mouse wheel or through a dedicated zoom slider, and pan the virtual canvas, by clicking and dragging the canvas.
On the upper right corner there is a textbox where the user is able to enter arbitrary filter terms to cause the system to display only songs, which the metadata contains those terms.
The songs are represented as circles with musical taxonomy of genders encoded with color.
The algorithms on this system are very complex but allow Last History interface to present, to each user, the best songs according to their preferences, and the amount of information available.
This amount of information sometimes gets hard to see and understand.

Figure 23: *LastHistory: Visualizing personal music listening histories, photo and calendar streams for analysis and reminiscin*

2.5.5 LoggerMan

Loggerman [19] is a tool developed to help researchers and users to see activity patterns produced by the interaction with a computer during normal usage. Working in the background, it intercepts usage events and storage to analyze them later.

Its design is made of multiple logging modules, which can be switched on/off, independently.

The first module is the keyboard that is responsible for capturing keystrokes events and storing them in local files. The mouse module is responsible for tracking all mouse actions, such as, scroll, move, button events, etc. The screenshot module is responsible for capturing a screenshot of the current active windows. Users can select the interval among screenshots. The app module was designed to track app transitions. The fifth and last module is the clipboard module that is responsible for tracking copy-paste operations.

The Loggerman has an interface tool allowing users to visualize their computers usage that can be separated into used apps, typed words and window tiles. Mouse clicks and keystrokes are also presented as a graph.

The tool also organizes the screenshots and the app usage data in a zoomable timeline.

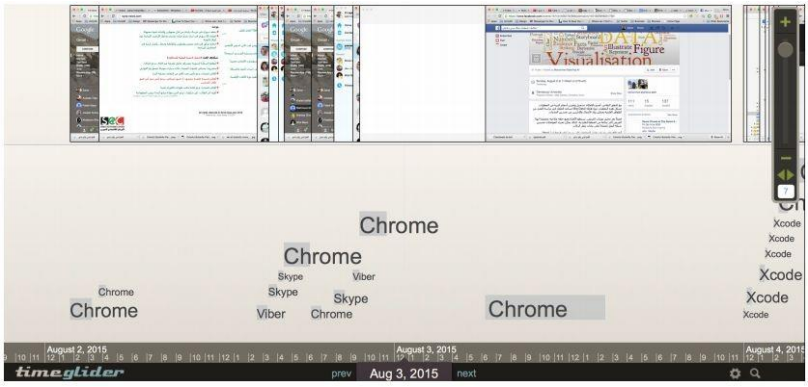


Figure 24: *Apps & Screenshots Timeline*

2.6 Discussion

After reading and analyzing the studies mentioned in this section, we can conclude that lifelogging visualization solutions need considerable improvements. Several studies proposed to solve particular problems but, as a result, they lack on others, such as scalability, memory storage and relevant visualization data. In order to summarize the main topics, we want to address, a table was created to provide an easy view of the differences among each developed work. The first column, "*Scalability*", shows if authors of each study developed a solution having in mind the process of handling a growing amount of data, or its potential to be enlarged in order to accommodate that growth. The second column, "*Data Source*", shows which were the devices or mechanisms used to collect the data used for the visualization. The third column, "*Visualization Method*", presents the visualization techniques the authors selected to display the data gathered by users. The fourth column, "*Preprocessing*", displays the methods used to redefine the gathered data, passed as input, in order to be deployed in the visualization. The fifth column, "*Panels and Menus*", shows whether the visualization uses, or not, menus and/or panels for navigation, for selection and for filtering the information to be displayed. The sixth column, "*Personal Semantics*", can be 'Yes' or 'No' according to the information shown. If it is 'Yes' it means that the visualization data has personal meaning to the user, otherwise, it will be marked as 'No'. The seventh column, "*Holistic View*", provides information about whether the several parts of the visualization are interconnected and how to explain it. It is necessary to reference the whole visualization.

	Scalability	Data Source	Visualization Method	Preprocessing	Panels and Menus	Personal Semantics	Holistic view
Shifted Maps	Some	Collected Dataset	Map	Unknown	Yes	Yes	No
MyLifeBits	Yes	GPS and Media Capturing Devices	Map and Images	Clustering	Yes	Yes	Yes
Visits	No	GPS and Others	Map and Timeline	Clustering and Custom Algorithm	No	Yes	No
Smart Reality Testbed	No	GPS, Smartphone, Wi FI and others	Map and Timeline	Clustering	Yes	Yes	No
FootPrint Tracker	Some	Camera and Google Android Device	Map, Timeline and Images	No	No	Yes	Yes
SnapTracks	No	GPS and Camera	Map and Images	Unknown	Yes	Yes	Yes
SoundBlogs	No	Smartphone or Web Recorder	Map	Annotation and Segmentation Algorithms	No	Yes	No

	Scalability	Data Source	Visualization Method	Preprocessing	Panels and Menus	Personal Semantics	Holistic view
Everyday Visualization	Some	Jawbone Up3 tracker, Foursquare, Openweathermap API, Gmail, TicTac, iCal App, Hours App and RescueTime tool	Calendar, Map and Coxcomb	Data Standardization	No	Yes	No
Data in Everyday Life	Yes	Web App	Calendar	No	No	Yes	No
Personal Lifelog Visualization	Some	Acceleration Sensor, GPS, Bluetooth, WiFi, Camera and Speaker	Images, Calendar and Coxcomb	No	No	Yes	No
SenseSeer	Yes	Smartphones	Images and Timeline	Data Reduction, Compression and Rendering	Yes	Yes	No
SenseCam	Some	Camera	Images	Segmentation and Indexation	Yes	Yes	No
PieTime	No	Email and phone	Coxcomb	No	No	Yes	No
Muse	No	Email	Stacked Area Chart	Email Processing and Cues Generation	No	Yes	No
Egocentric Visualization Design	No	Documents	2D Tree	No	No	Yes	No
QS Spiral	No	WiFi, GPS	Spiral	No	No	Yes	No
AffectAura	Some	Webcam, Kinect, Microphone, EDA Sensor, GPS, File Activity and Calendar Scraper	Timeline	Classification and Clustering	No	Yes	No
LastHistory	Yes	Last.fm	Timeline, Images and custom visualization	No	Yes	Yes	Yes
LoggerMan	Some	Computer Logs	Timeline	No	No	Yes	No

Table 1: *Visualization Summary Table*

To avoid a very common problem in the field of visual analytics, "*scalability*", the major part of the explored works use existing datasets or data that comprises short time intervals. When data volume increases, visualizations tend to suffer from clutter, making it hard for users to absorb all the information.

In order to solve this issue, studies such as ***SenseSeer*** and ***LastHistory*** use timelines to represent chronological events where users can easily follow their past activities effortlessly. The data comes from a great variety of sources and it is related to the type of visualization and visualization techniques used in each approach. For example, when the visualization methods are based on a map the main sources are a portable GPS and a camera. The majority of the maps were developed using Google Maps API with different representations of the tracks and/or locations shown. The timelines also differ from each other. For instance, ***SenseSeer*** presents its timeline with images and ***Data in Everyday Life*** shown in a calendar type view. There are also several studies with personalized techniques, such as coxcomb in ***PieTime*** and ***Personal Lifelog Visualization***. These techniques offer an easy way to understand the interface but provide less information. Another technique is the spiral visualization of ***QS Spiral*** with a serious scalability problem.

A relevant topic of this area is "*preprocessing*", which englobes methods such as data cleaning, track simplification, clustering, removing inconsistencies, repetitions or redundancies. Although, only half of the studies used it, the importance, in order to have a good visualization experience, is unquestionable because it allows data to be previously analyzed and transformed into information that makes sense for a certain context.

Another topic addressed is whether, or not, these studies deal with "*personal semantics*". This is related to the information shown having personal meaning to the user (eg. my house, my work, dinnertime, etc.). All of the explored studies support data, which is important and relevant to the user.
The "*panels and menus*" column refers if the studies show paths instead of tracks, vice versa, or both. Half of the studies rely on the use of menus and/or panels for navigation, selecting and filtering the information to be displayed.
The "*holistic view*" provides information regarding the connectivity of the visualization techniques displayed on the interface and in order to understand one part, the user has to understand the whole. The few studies that have this topic implemented, ***MyLifeBits***, ***Footprint Tracker***, ***Snap Tracks*** and ***LastHistory***, provide a dynamic interface that keeps coherency among all the visualizations.

According to our main goal, in order to study how best we can present users with visualizations of their past lives in a personal relevant way, we need, first, to focus our attention on solving three major issues.

Since our users will collect huge quantities of data from different data sources throughout a certain period of time, the first issue we need to address is *scalability,* in order to have the ability to support massive amounts of data.

The second issue we need to address is the holistic view, which will aid users to see the visualization as a whole instead of individual representations of their data. The user must feel that everything is connected and based on the current context.
The last issue, where we will initially focus our attention, is the personal semantics, which means that the data we will show need to have personal significance for that specific user.

3. Solution

In this section of the document it will be described how each component was developed and the reason we chose the frameworks and tools that we use to achieve our goals.

We will start with the ***Architecture***, where we will illustrate a concept design model representing how the system components are connected and what messages are exchanged to communicate among them.

Secondly, we will approach our ***Backend*** and ***Front-end*** modules and explain several topics surrounding their concepts, their development and their components. During our ***Front-end*** analysis, we will describe our interface, as well as justify the design choices and functionalities developed.

This section will end with some ***Usage Examples*** for Follow My Steps interface.

3.1 Architecture

The designed architecture of our solution adopts a simple *client-server model*, which can be seen in ***Figure 25***. This model is structured with two main components, the client-side, developed to provide users a front-end system to visualize their personal data and the server-side, developed to parse and compute the incoming requests. The communication between these components is achieved by exchanging JSON objects from specific modules within these components.

On the next subsections, we explain how each component works and the functions of their modules.

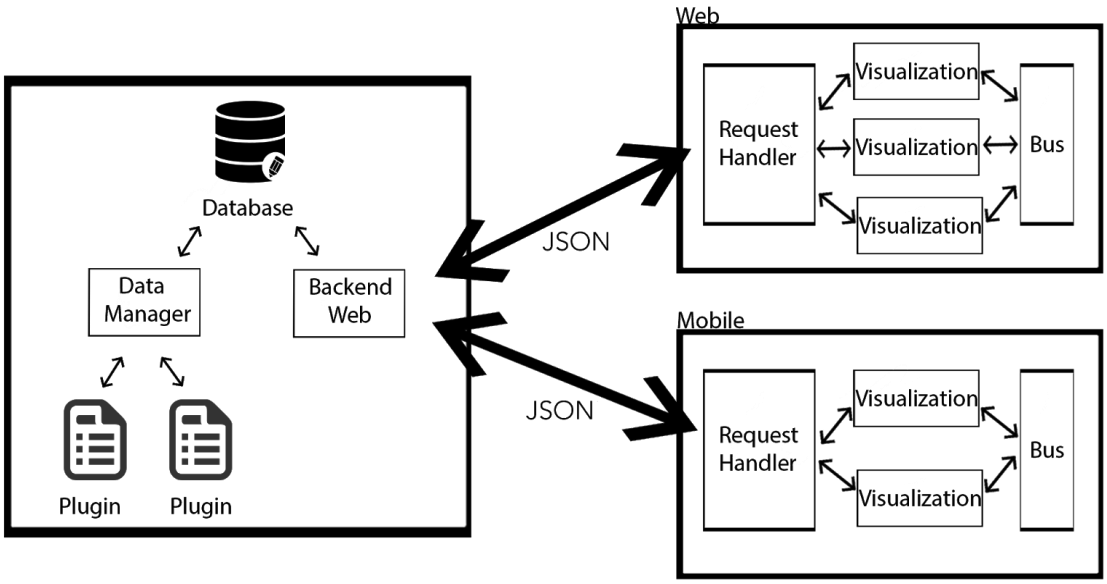


Figure 25: *Architecture of Follow My Steps system*

3.1.1 Server-Side

In our solution the server-side is made of four major modules, the *backend web,* the *database,* the *data manager* and the *plugins*, that process and retrieve data from and to the client-side, respectively.

Our system's primarily core functionalities are guided to a web application. However, as we mentioned before, we will develop additional software to include mobile and report features. Both web and mobile communicate with the server-side and, in order to interpret and analyze all the exchanged requests, it was imperative to create a module able of synthesizing and processing the large number of messages generated by each feature. On the other hand, the reporting service will run entirely on the server, meaning that it will not swap messages on the client side. We developed a module called *backend web,* which can be seen as the server communication module between the client-side and the *database*, responsible for interpreting the client's requests, formulating the correct *database* queries, handle with results from the *database,* and to manage the reporting feature internally.

The data collected will have heterogeneous properties since its sources may differ accordingly to the relevance that each user assigns to different sets of data. Several individuals can gather distinct information from various origins accordingly to their own interests and needs.
To support these data formats, the system is prepared to handle standard structures of data, that could, otherwise, cause consistency issues while storing and displaying it.
To solve this type of problems, we created two elements on our server-side, the *plugins* and the *data manager* or *plugins manager*.

The *plugins* will receive the data collected from the client-side and parse it in order to, homogeneously, store heterogeneous data from different datasets without changing its information. To accomplish that, each *plugin*, in our system, has an algorithm that knows how to arrange a specific type of data and where to send it. This also allows them to detect errors and missing data on the inputted information. Initially, we developed five *plugins* that process and handle data from *Google Spreadsheets, CVS, LIFE, GPS* and *Photos*.

The *data manager* receives the data sent by each *plugin*, computes it and creates the correspondent queries in order to store it on the database.

To store all the information inputted by the user, we created a *database* that not only provides storing operations but also has access to the *backend web* in order to post and get the information requested by the client-side. It is mandatory to manage data efficiently to allow users to perform multiple tasks easily and effortlessly. To accomplish that, the data stored is arranged into seven tables to cluster the information. These tables are used to store the files content, mentioned above, the definitions of the dashboard (to save progress), the variables (to use on the visualizations) and the information file (to provide quick details and update the files uploaded).

Since the system does not know the origin or the format of the inputted data and it needs to store collections of contextual information, based on time, such as tracks, trackpoints and routes with a name, date and time, it does not use a common relational database because it would not handle data like points, edges and rectangles.

To handle the incoming messages, containing information with different data formats generated by the users' uploaded files, the system follows a set of requirements allowing to store, homogeneously, all the content within these messages. The most relevant requirement, that allows to connect data between database entries, is the timestamp. This property must be specified for each record that is stored, providing means to associate all the information in the tables with a specific period of time. Furthermore, and since our goal is to represent personal data based on a time interval, the timestamp attribute is present in most of the dashboard visualizations.

While collecting data, the users must guarantee that this attribute is being recorded.

3.1.2 Client-Side

During the system usage, a huge number of requests will be exchanged between the client and the server side. To handle these requests, the client-side must have a module able to interpret and forward the messages to the proper modules. These messages contain information regarding the users' personal data, which will be displayed accordingly to their own interests and preferences. In order to provide a personalized interface, the system must ensure means to personalize the content displayed and to guarantee time consistency. In other words, the content displayed must be optional, editable, replaceable and be represented within the same period of time throughout the entire interface.

To allow users to manage, analyze and observe the data collected and stored, two interactive interfaces were developed, each one represented by a component on the client-side structure (***Figure 25***).

The first interface was built for web browsers and consists of three modules, the request handler, the visualizations and the bus.

The *request handler* act as an intermediary, which sends requests, made by the user, to the server and, then, receives the information, sending it to the proper visualization module. To avoid some problems such as performance, efficiency and response time, the handler manages the requests from the multiple visualizations, prioritizing and sending the correct information to each one. It uses a cache that stores an object with information about past requests and the correspondent answers, to avoid recent requests repetition made to the server. The *visualization* modules are visual techniques connected among themselves through time, to preserve the interface coherency, which can be represented as Bar Charts, Line Charts, Maps, among others. Similarly, to the *plugins*, these modules are created individually, allowing developers to add or remove new visualization modules to the system. The main goal of the final module, the *bus*, is to ensure coherency among all the visualizations by displaying their respective information

bounded to a given period of time. To link them, the bus is notified by the visualizations each time an update occurs, in order to communicate the changes to the remaining ones, by trigger a time event that each visualization is subscribed to.
The second interface was built for mobile, displaying a simpler design driven by the purpose of providing an “on the go” solution for users interested in analyzing personal information on a portable device. This structure is composed of the same modules found in the first structure; the request handler, the visualizations and the bus. The main goal of this component is to allow users to remember, anywhere and in a lower detailed visualization, their past experiences according to their location.

The only difference among the structures is focused on how the request handler manage the requests, redirecting the data from the server to unique mobile visualization modules.

3.1.3 Technology

Before we developed the Follow My Steps system, we studied and scrutinized several frameworks and tools in order to decide which were the best options to integrate our project. We submit each alternative into an evaluation process composed of three phases.
The first consists of selecting and analyzing all of the free tools on the market, the second we apply performance and efficiency criteria and the third was based on the technologies accessibility.
As a result of this analysis we decide to use, for the backend, the combination of *Node.js* [a.] for server services, and *PostgreSQL* [b.], for database services.
Among several options, such as, *PHP* [c.], *Java* [d.], *React* [e.] and *Angular* [f.], we elect *Node.js* [a.] since it is easily employed as a server-side proxy to handle large amounts of simultaneous connections. It has the great advantage of proxying different services with different response times or collecting data from multiple source points. The *PostgreSQL* [b.] selection, over the *MySQL* [g.] and *MongoDB* [h.], was based on a particular extender called *PostGIS* [i.] that adds support for geographical objects to the *PostgreSQL* [b.] object-relational database, allowing to storage and query information about location and mapping.
In a different perspective, the system frontend was selected based on the frameworks number of features and on personal experience. Nowadays, there are numerous Javascript Libraries that can be easily handled to present information such as *Echarts* [j.], *Tau Chart* [k.], *Chart.js* [l.] and *C3.js* [m.]. We chose *NVD3* [n.] to deploy our visualizations because it provides a large spectrum of graphs that can be effortlessly handled to display different types of data. *NVD3* [n.] build reusable charts and chart components for d3.js without abolishing its features. This library allows developers to bind random data to a Document Object Model (DOM), and then apply data-driven transformations to the document.

3.2 Implementation

In this section, we will discuss the functional purposes of each developed component that adds new features and functionalities to our system. Furthermore, we will explore and reflect the design approaches and explain the choices we made in this work.

3.2.1 Backend

One of the biggest achievements that we endeavor in this project is to represent data that has personal meaning to users accordingly to their own preferences. In order to achieve it we cannot restrict the type or the format of the data that is inputted and inserted throughout our interface usage. As a result of this requirement, we created the plugins that we will scrutinize on the following subsection, which are pieces of software that allow us to transform the incoming data and store it with a predefined format on our database. Since this data scheme changes between different types of information, and depends on the user's personal choices, it was imperative to create distinct tables for dissimilar types of data.

Follow My Steps server creates these tables as the data flow into our system, in order to store the inputted files content (for example the *GPX* table is generated when the user upload a *GPX* file). This approach, consistently, produces tables as they are needed instead of creating them when the server starts to operate. The scope of the generated tables number, created by the database, can fluctuate between zero and seven. These are arranged into two clusters, the common tables, which are generated whenever information disregarding the data format is uploaded, and the file tables, which are produced accordingly to the file structure uploaded by users.

The first cluster is made of the *definitions* table, created to store the progress and all information changed, by the user, in the interface. The *files* table was developed to register all the file paths and names uploaded, as well as the last time that they were modified, allowing to update the interface by keeping on track all the changed files. The *types* table, which was structured to preserve all the data types from the file variables that were uploaded (string, number or boolean), gives access to them every time the system needs to verify whether a column is or not of a certain type. This is particularly useful when we want to plot a new chart on the dashboard, ex: where the y-axis must be an integer. The second cluster is made of tables that store the content of each uploaded file. Within this cluster we generate the *GPX* table, for *GPX* files; the *LIFE* table, for *LIFE* files; the general table, for *CSV* and *XLSX* files and the photos table, for all photos type files (such as *PNG* and *JPG*).

Every time a file is uploaded its content is analyzed by the server and send it to the proper *plugin*, which is prepared to parse it and arrange the data to homogeneously store it in the *database*.

There are a couple of properties that need to be saved that are common to every file. To keep coherency we dedicated the first four columns of every table that stores file content to the row id, file path, file name

and to the row timestamp. The id column is an integer that is automatically assigned by the database every time a new entry is inserted, which allows the client side to upload several new records without concerning about their database identification. The file path, the filename and the timestamp are information properties that depend on the file properties, which, consequently, need to be provided whenever a new file is uploaded.
The following columns diverge and to handle these changes each *plugin* must parse the assigned files content to organize the information according to the proper table specifications and restrictions.

In this subsection, we present the plugins by explaining how each phase of their parser transforms the specific file content data into a format that can be stored in the *database*.
Furthermore, we present backend services that provide a better understanding of how the system behaves and a better user interface experience.

3.2.1.1 Plugins

As mentioned before, each user will collect heterogeneous data with personal relevance, from different procedures and data sources, with origin consistency problems while storing it in the database. To counter this issue, we create fragments of code, called file parsers that allow the system to restructure the file content format in order to match with the database format. In Follow My Steps backend, we commonly designate file parsers as *plugins*. A parser is a program or a method that arrange input data, containing a specific format, so other programming entities can manage it. As mentioned before each *plugin* receives a file content and produces an output that is arranged, with a specific structure, in order to be stored.
We focus our efforts on developing plugins that go towards the most common data formats used to store daily information. Initially, five plugins were developed, the *CSV* plugin, the *XLSX* plugin, the *GPX* plugin, the *Photos* plugin and the *LIFE* plugin.
In this subsection, we explain how the *plugins* work and what are their purposes.

3.2.1.1.1 CSV

The first plugin was developed to read and parse *CSV* files. The popularity of these files has been increasing due to their readability, manually editable simplicity, straightforward information schema, faster handling and easy generation.
To analyze and handle the *CSV* file content, the system interprets the file format as a table. Therefore, we use the first line to identify the columns and the following lines to identify the values. Each value, excluding those from the timestamp column, is arranged into a row on the database, where the system stores, among other information, the value and the value's column name. For each pair of key-value, where the key is the column and the value is its respective value, the system gets the correspondent time from the timestamp column. This column is specified by the user while uploading the file.

3.2.1.1.2 XLSX

Another file that became one of the most popular, around the world, is the Excel (*XLSX* files) due to its capacity to perform, among many functionalities, statistical analyses and spreadsheets allowing users to define the fonts, character attributes and cell appearance in the sheet.
To arrange the data within these files the system uses an external module, called *SheetJS js-xlsx* [o.]**,** converting every filled cell of each row in a concatenated comma-separated value (CSV format). Once converted the system uses the same methods applied by the csv plugin to parse and store the information in the database. Similarly, users must specify the column that corresponds to the timestamp.

3.2.1.1.3 GPX

The *GPX* plugin was developed to parse *GPS* data stored on *GPX (GPS eXchange*) files. These files' format is an XML schema conceived to be exchanged among applications, containing geolocation data, such as tracks, track segments, track names, track points and other information. The importance and popularity of the GPS is a continuous growing over time as it allows users to locate themselves anywhere on the planet, freely, without technical details. On Follow My Steps system, it was essential to include positional information, in order to provide an insight of the places where users have been.

To understand this plugin, it is fundamental to learn the logic behind its modules. The first module is the *gpx-parse* that uses an algorithm to, basically, convert the *GPX* tracks information into a *GPX* object. This *GPX* object provides a synthesized version of the information within our file, which is subsequently used to create the arrangements needed to be inserted into the *database*. By analyzing this object, we can obtain all the data needed to support our visualizations based on location, such as the *Map*.

The problem of this approach resides on the huge number of information that is generated while collecting GPS data. In order to save all the information, the GPX files save the points, tracks, routes and all specified above. A simple walk of 1.4 km can produce over 115 track points and a 2 km walk can produce over 215 track points. For longer distances, the amount of information that is produced requires a lot of space and memory. In order to reduce this information quantity, we use another external module called *simplify-path* [p.], that uses the Ramer–Douglas–Peucker algorithm that reduces the number of track points of each track allowing to create similar curves and lines with fewer points. At ***Figure 26***, we can observe how this algorithm works given a set of points

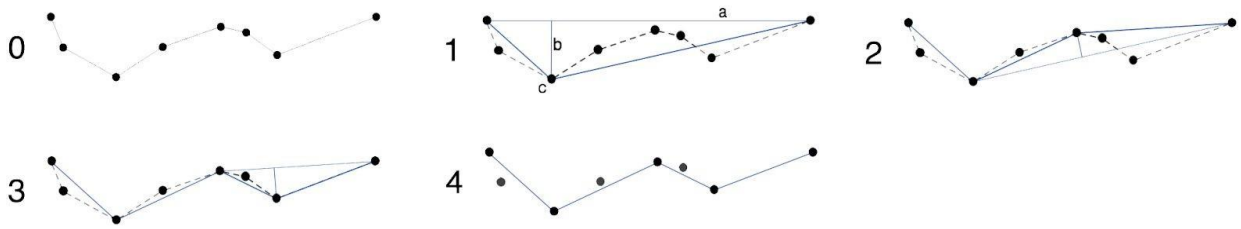


Figure 26: *Ramer–Douglas–Peucker algorithm representation [ff.]*

This method reduces, on a large scale, the information we send to the database. From the previous examples, it is feasible to diminish the 115 track points to 12 without losing too much information from the real track. From the 215 it is possible to store only 24 track points.

3.2.1.1.4 LIFE

The *LIFE* [q.] plugin was created to parse an uncommon type of file format, developed by Professor Daniel Gonçalves from Instituto Superior Técnico, which provides a standard procedure to record information that the user collects throughout the day in a simple, expressive and detailed approach.

This format is, especially, useful because it is focused on structuring, through time, the activities performed by the user and offers syntax configurations to cluster them in three types, which are the spans, that represent the location or places that the user was performing during a particular time interval. The subspan, that provides additional activities about the places within a particular span and the trips, that represent the user displacements. Each of these activities is bounded to a timestamp, which must be manually inserted by users.

The file format also provides additional information about places and locations. With this data the system is able to register place inclusions, which represent places inside other places (for example, a shop inside a mall); canonical locations containing geographically information about a certain place; place categories, to arrange places in categories and name changes, which inform that a certain place has changed its name. This plugin was developed to analyze the syntax of each activity and additional information and to handle them in order to generate the appropriate arrangements for the *database*.

3.2.1.1.5 Photos

Photographs are one of the most popular hobbies for many people all around the world. They help us to remember the past by preserving memories and reminding us about the people, places and activities we love.

The *Photos* plugin, unlike the others, is the only that does not require to parse the file content to store the information in the *database*. Instead of saving the photos content, this plugin only saves their file path, filename, street, city, country, timestamp, latitude, longitude and descriptions. This approach allows the

system to store large amounts of photos without consuming a massive volume of space and to obtain the images content, through their file path, every time the user requests them.
To return successfully their content, the plugin uses an external module designated as *base64-img* [r.].

To populate the database with all the information, we analyze the image metadata, using another external module called *exif* [s.], which contains several details, such as the place where the image was taken, providing the latitude and longitude coordinates and the day the image was created. To ascertain the street, city and country we use the same external module used in the *GPX plugin,* the *reverse-geocoding* [t.].
Finally, to rotate the upside-down images we use the *exif-image-auto-rotation* [u.] module that rectifies the image orientation. Unfortunately, this module does not work in every image, it is required to have exif properties in order to be able to access and adjust the image orientation.

3.2.1.2 Services

Follow My Steps backend provides several services to the front-end, improving the interface quality and the user experience. These services englobe the add, update and remove files, the save interface definitions, the send report, the send feedback and the mobile access.
Since every method on the server side runs asynchronously the user keeps the capacity to interact with the interface while the server synthesizes and execute all requests.
In this subsection, we are going to explore the processes that support the mentioned services and explain how they operate.

3.2.1.2.1 Add Files

The goal of Follow My Steps is to allow users to display and analyze visualizations with personal information. To accomplish this, the system's data source must rely on the content inputted by users, instead of a static set of data.
Every time the user submits a new file to the interface, and send its content to the server, the post handler manages the request and infer which is the proper *plugin* to process the incoming data.
After the *plugin* operates and parses this data, the handler adds the new file to a subscription list that triggers an event every time a change occurs. The process of detecting the file content changes is implemented on an external module, named *file-changed* [v.], which returns an array with all the paths of the files that were modified. In the ***Update Files*** subsection, we will explain the utility of this module.
After the plugin concludes its operations, the parsed data is sent to a query converter, where a set of methods arrange queries to populate three tables of the *database*. The population processes begin by adding an entry to the files table with its path, name and last time modified (at this stage it remains with the equivalent date of its creation). On the second table, the query specifies and stores the variable name and type; and in the third table, which represents the proper table to store specific information about this type

of files, it saves the content result, shown in the subsection ***Plugins***, generated by the proper plugin.

3.2.1.2.2 Update Files

With the upload of a file into the system, a user is able to display its content on a visualization, which could be a chart, a map, among others. Since this system is dedicated to present daily live information, it was necessary to develop a feature that provides a database update whenever users change the content of an inputted file, avoiding the upload process over and over again. To achieve this, the file system implemented on Follow My Steps allows updating the files content stored without specific requests from the user or the client-side. As we had already mentioned before, we added to our project an external module named *file-changed* [v.] that provides the list of the paths of every file that was modified after being submitted and stored in the *database*. Each time that the user adds or removes content from a certain file, that has been uploaded, the module will notify the update handler, which will delete all of its previously stored information and reinsert its entire content.

This approach has, undoubtedly, several issues. For example, if a user was able to write and successfully store one million lines of data, collected over the past 10 years, and updates one single line, the system will delete every record linked to this file and reinsert all its content.

Due to the complexity of the project, this issue could not be solved at the moment the document is delivered, but it is one of the biggest priorities to handle in a future work.

3.2.1.2.3 Delete Files

The upload of several files can sometimes lead to mistakes or unwanted information on the system. Furthermore, the data that users desire to observe today may not be the same data they want to display tomorrow. In order to provide full control over the submitted files, we developed a backend service allowing the user to specify a file, through its path, in order to remove it from the *database*.

When the server receives an *HTTP* delete request, with a file path as the parameter, the system runs through tree tables that incorporate data from this file and expunge every record related to it. These three tables are the same we populate while adding a new file to the database.

3.2.1.2.4 Save

As a result of the interface daily usage, the Save operation became a mandatory service and one of the most important provided by the backend.

If for some reason, if the user closes the Follow My Steps tab or the browser window, what consequences could happen to the interface usage if the system was not able to store the user's progress?

How could a user work and benefit from an interface that needs to be rebuilt each time the application is relaunched?

Deprive users to save changes could mean the obsolescence of this system. So we developed a service that receives data from all the visualizations inserted on the dashboard, as well as all of the settings, and saves them on a table designated as *definitions*.
Not all of the columns, of this table, will be filled when the row represents data from settings. For example, when the user updates the cache values, on the client-side, the data stored will contain less information than a visualization, since it is necessary to store fewer data in order to save all the content changed.

3.2.1.2.5 Report

Nowadays almost every individual has, at least, an email account, either for professional or personal reasons, in which they verify, daily, their new mailing. For most of users, the burden of accessing their interface every day might decrease the interest in the system. To avoid this course, we developed the *Report* service.
This *Report* service is the unique backend service that interacts with the user outside the Follow My Steps project. It uses an email account, created solely for this purpose, to deliver a report, with data from a certain day, month or year, to an email address that is passed as an argumentation the client-side. This email is dispatched through the use of a resourceful external module, designated as *nodemailer* [x.]**,** which handles all the processes needed to structure and send the email.
To complete this module we installed the *node-schedule* [y.], which allows users to program an email delivery to a certain date and time. Whenever an email is dispatched, the reporting process leads this function to create a new schedule.
To illustrate how this schedule operation works, imagine we settled everything to receive an email every day, at 9.00 AM, with data related to what users have done two weeks before. If today is Monday, they will receive information regarding their activities done on Monday from two weeks ago. By the time they receive this data the reporting process schedules a new email for Tuesday, which will contain data of Tuesday from two weeks ago.

We will analyze the report content and design further in this document.

3.2.1.2.6 User Feedback

All the processes presented above run methods that communicate with the *Backend Web* server module, notifying whether or not an operation was successfully concluded. Unfortunately, the user does not have any feedback regarding these processes. To contradict this information absence, we installed a module designated as *socket.io* [w.], which is able to create a virtual bridge that enables real-time bidirectional event-based communication, allowing to send and receive information both on the client and server side. The system uses this wonderful module to send information to the user about all the processes running on the backend.

3.2.1.2.7 Mobile Access

Mobile Access is the unique operation dedicated to the Follow My Steps mobile app, from the list of processes presented above. The mobile app was developed to display the user's data regarding a certain position, providing information about the number of times the user has been in a specific location, the places nearby ordered by distance, the activities performed on the last day in that location and the photos taken.
In order to connect a mobile phone with a local server, the user has to ascertain the machine external IP, which is the representation of the machine address on a computer network, and the port, which represents the endpoint of communication. Only after obtaining both, the mobile is able to exchange messages with the server. However, this operation does not imply that the user has access to the *database* data. There is a restriction where the maximum number of mobile devices connected, at the same time, is lesser or equal to a given number and a simple security measure where the Mobile Access service produces a six random digit code. After generated, the code is sent to the browser, through the *socket.io* [w.] module. This method restricts the number of accesses to the user's data by guaranteeing that only individuals with contact with the browser version are able to receive it, excluding third parties and inconvenient guests that might find the user's IP address and try to visualize user's personal information.

3.2.1.3 Security and Privacy

As we know, the internet is a global computer network, which provides large amounts of information, using standardized communication protocols. To store and share this information in a huge global network, users must be certain that only they can see and grant access to their data. This approach would require complex methods to avoid security issues and privacy information leakage.
The Follow My Steps system uses a local server, which means that the messages exchanged between the front-end, of the web browser, and the backend are processed internally, without leaving the user's machine. However, the mobile app needs to be connected to the network, so we create the security measures that were mentioned in the section above (***Mobile Access***).

3.2.2 Front-end

The front-end of a software, application or website, is the interactive interface displayed to the users. In our system, we developed two interfaces, one for the web browser and other for the mobile app.

In this section, we are going to explore these interfaces and how they work, as well as, the libraries and frameworks used to create them.

3.2.2.1 Web Browser Interface

Follow My Steps is a project developed to support users to visualize their data, collected throughout a certain period of their lives, allowing them to remember their own memories.
To achieve its goal, this system transforms the raw information, inserted in the uploaded files, into customizable and elegant visualizations, that can be inserted into the interface to fulfill the user's intentions.
In this subsection, we examine and discuss the interface elements, such as the menus and the visualizations, and their design options to provide an insight of the choices we made while programming this interface.

3.2.2.1.1 Menus

When the user initializes the application, for the first time, the dashboard is empty and the only interactive elements presented on the interface are four lateral buttons, on the right upper corner, that represent the four different menus provided by the front-end.

The first button, represented by three gears, opens the settings menu. The selection of the gears metaphor resides on how the traditional operation mechanism used to be performed, where the mechanics had to manually make changings to the gears. The number three represents the three different categories of the settings menu.
The second button, represented as a calendar, opens the timeline menu. It is often to associate the calendar icon with a date picker.
The third button, represented as a plus icon, opens the visualizations menu. This icon is associated with the addition of two or more numbers on an equation. In the case of Follow My Steps interface, the plus represents the possibility to include or add a new visualization to the dashboard.
The fourth and final button is used to open the help menu, which is symbolized as a question mark. This metaphor is used to represent questions that users might have during the use of specific features of the interface.

The different menus share common properties that allow to drag them, on the interface, and to close them, using the cross icon or clicking outside the container.
Each menu has specific keyboard codes to open its particular features. Every time that the ctrl key is pressed simultaneously to another key the settings menu open (for example, Ctrl+A opens the Add File menu and the ctrl+R opens the Report menu). The same process is observed when the user, simultaneously, press the shift+T keys, to open the Timeline Menu, and the alt+H, to open the Help Menu. The visualizations are shown by clicking on the initial character of the visualization name (for example, the A key to open the Area Chart Menu and the B to the Bar Chart Menu).

Bellow, we will observe all of the mentioned menus and scrutinize the features behind each one.

3.2.2.1.1.1 Settings Menu

As we can observe in ***Figure 27*** the *Settings* Menu provide three submenus, the *Files* submenu containing all the possible file procedures, the *Style* submenu which incorporates the available operations over the interface styles and the *Advanced* submenu that encloses all the extra editable features.

Figure 27: *Settings Menu*

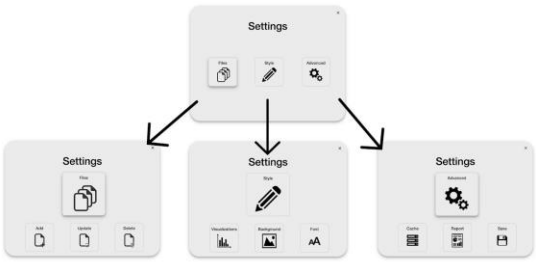


Figure 28: *Settings Menu Hierarchy*

- Files

By expanding the *Files* submenu the user has three new options, which are the *Add Files*, *Update Files* and the *Delete Files*.

Since the number of submitted files is distinct for all users the *Add Files* container encloses only one input field to preserve the container simple design, where the user can select a set of files to submit towards the server-side. After each selection, the container creates another input field allowing to provide constantly an empty file input. On Chrome and Firefox browser we can toggle between adding a set of files and a set of folders by clicking on a specific button, on the upper left corner of the container. The function that allows this toggle is called *webkitdirectory* and it only works on the mentioned browsers due to compatibility issues.

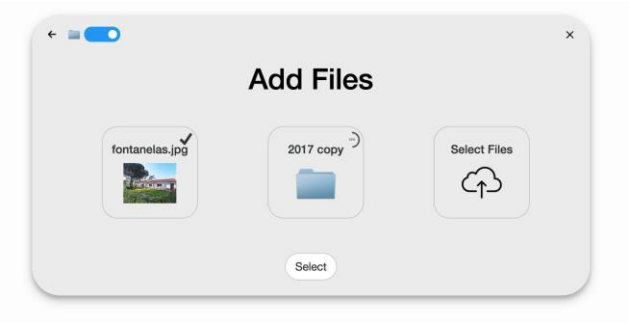


Figure 29: *Add Files*

Figure 30: *Images Modal*

If the files selected have the extension of a photo file, the program calls a modal, from the *bootstrap library* [z.], to get a description, given by the user, for the image inserted (***Figure 30***). However, if the files selected have the extension of both an excel and a *CSV* file, the program calls a modal that requests the file

timestamp column. While the folders and files are being processed by the interface, the user can observe the progress through a radial progress bar placed on the right upper corner of the correspondent input field (see ***Figure 29*** inside the '2017 copy' input). Once this process is complete, it switches to a checkmark to signalize that the content is ready to be submitted. The user can remove a file input, after being added, by clicking on the close icon that emerges by hovering the checkmark icon.

As we mentioned in the ***Update File*** subsection, whenever an uploaded file is modified, the system, automatically, updates the changes on the *database,* without requiring any user request.
The code fragment that runs periodically the function, that verifies if there has been a change, can be set on the client-side. The user is able of selecting the updates time interval, as shown in ***Figure 31*** and if the system should or not process the update method.

Figure 31: *Update Files*

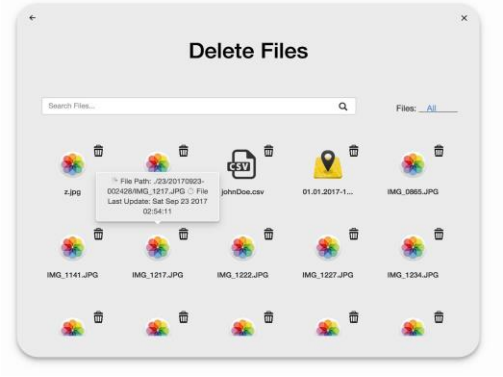


Figure 32: *Delete Files*

The *Delete Files* container lists all the files submitted to the *database*, in order to remove them whenever desired. It provides a search mechanism to find files by name, an extension selector to isolate the selected type and a hover property that presents the file path and the file last update.
To delete a file the user has to click on the bin icon attached to the respective file.

- *Styles*

The next submenu, the *Styles* submenu, is related to the visual personification of the interface and it is made of the *Visualizations,* the *Font* and the *Background* options.

On the *Visualizations* container, the user can change several style aspects of the dashboard visualizations. This container provides several methods that can handle the background, the font, the updates and the dragging property of each element added to the interface. The operations based in the background, allow users to add or remove it and to change its color and border-radius.
The operations related to the visualizations font include the color, font family, font shadow and font shadow color changing.

When the update input corresponds to ‘Automatically’ the system enables all updates in the dashboard visualizations, otherwise it disables all receptors of the time events.

Similar to the update, the draggable property has also two options, whether enable or not the drag attribute on every visualization.

The final element of this container is the bin icon adjacent to the bar chart graph image. When clicked, an alert is shown with the following message ‘This action will remove all visualizations from your dashboard. Are you sure you want to proceed?’. If the user confirms, all visualizations are removed from the interface.

Figure 33: *Visualizations Style*

Figure 34: *Font Style*

The *Font* container elements have analogous processes as the methods handling the font properties in the *Visualizations* container. It is possible to update several font properties of the containers headers and texts.

The *Background* container is responsible for enclosure all the operations related to the interface background. To change it the system offers two options, the first is to obtain an image link and insert it on a specific text input, the second is to use one of the photos files uploaded to the *database*.

Figure 35: *Bkg Image Link*

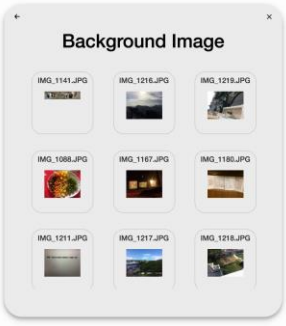


Figure 36: *Bkg Image Server*

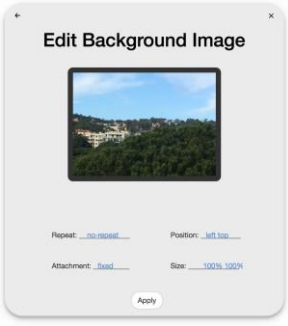


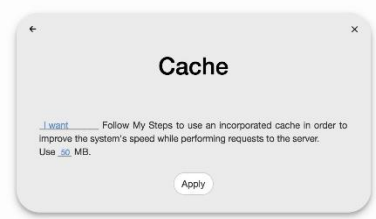
Figure 37: *Bkg Properties*

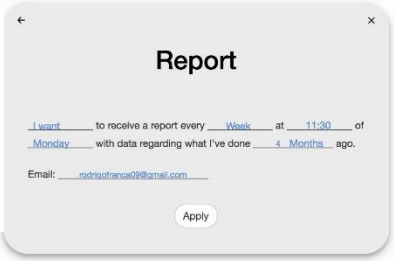
Once the user selects an image, the *Background* container displays a small preview of all the background attribute inputs that are editable. These attributes are the repeat, the position, the attachment and the size.

- *Advanced*

The last submenu of the *Settings* is the *Advanced* submenu, which is related to the remaining features provided by Follow My Steps interface. These features are the *Cache*, the *Report* and the *Save.*

During the usage of the interface, the user, involuntarily, generates and dispatches, to the backend, the same request several times. If the server was not running in the localhost, the user could start to detect same issues due to the waiting time and to the requests overflow. To avoid this situation we developed a facultative cache system that stores the requests and their respective responses in order to respond promptly to the incoming repeated messages made on the client-side. When a request is sent to the server, the *Cache* algorithm ascertains if this request was previously performed. If it was not, the algorithm waits for the server reply and verifies if the size of that request added to the response size is lower than the available space on the cache object (the system provides an input to change the *Cache* size). If the size is smaller, the cache insert both on the *Cache* object. However, if the size is greater than the space available and lower than the *Cache* size, the algorithm will erase the older requests, until the space size is equal or greater to the input size and stores the request and the response.





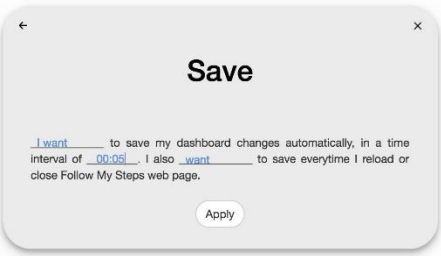


Figure 38: *Cache*

Figure 39: *Report*

Figure 40: *Save*

The *Report* feature was first introduced in the subsection ***Report*** to explain how the backend operates to send a report to the user email account. On the client-side, the *Report* feature defines the parameters used on the server-side. The container provides seven input fields that are used to describe the email type which could be a Daily, Monthly or Yearly report, and to schedule the periodic emails by passing the time of the day (ex:11:30 AM), the day of the week (ex: Monday) and a specific moment (ex: 4 Months ago). The last input is reserved for the user's email address.

The final feature from the *Settings* menu is the *Save* component. This component is particularly useful because it allows to automatically save the dashboard visualizations, at regular intervals of time, and every time that the user reloads or closes the Follow My Steps tab. These properties can be modified on the *Save* container.

The saving process begins by analyzing all the elements inserted into the interface and pushing, inside an array, all the information required to reinsert the same visualizations without losing data. When the user

reloads or opens the page, the stored data is parsed and sent to the proper visualization module that handles the insertion of the previous visualizations.

3.2.2.1.1.2 Timeline Menu

Follow My Steps supplies mechanisms allowing to select a time interval that restricts the quantity of data displayed on the visualizations. On the *Timeline* Menu, the user is able to change the values of this interval.

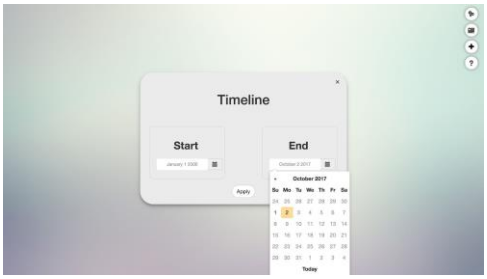


Figure 41: *Timeline Menu*

The menu is made of two input fields that represent the starting and the ending date, each with a clickable calendar icon setting the input date to the default and a button to apply these time changes to the visualizations. The input fields provide a date picker, which is a digital representation of the calendar month page (observed in ***Figure 41***), to support users while selecting the required date in a faster and efficient approach.

3.2.2.1.1.3 Visualizations Menu

The main goal of Follow My Steps dashboard is to present users the data that was uploaded to the system's database. This would not add any additional value if presented with the same format found on the submitted files. To invert this situation we developed a set of nine visualization categories, which are *Images*, *Bar Charts*, *Area Charts*, *Line Charts*, *Pie Charts*, *Map*, *Timeline*, *Heatmap Calendar* and *Text*, that provide friendly approaches to analyze and interpret personal information.

Figure 42: *Visualizations Menu*

After selecting a visualization category, the user is able to adjust several variables to personalize the information that will be inserted. The properties of each category will be described in the next paragraphs.

- *Images*

The *Images* category is divided into two sections that can be accessible by clicking on the icons located on the right upper corner of the current container. The selected section is represented by a blue circle around the respective icon.

The first of these two sections displays a list containing the images/photos uploaded on the system and provides a search mechanism that returns all of the images with the name matching the text provided in the input field. The user can select up to nine photos that are automatically arranged in a div container. When the ninth is selected, all the other images on the list disinherit their clickable attribute and their opacity is reduced to 1/5 of their original opacity.

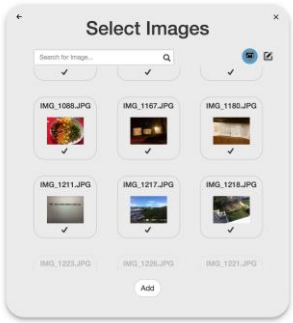


Figure 43: *Image Selection*

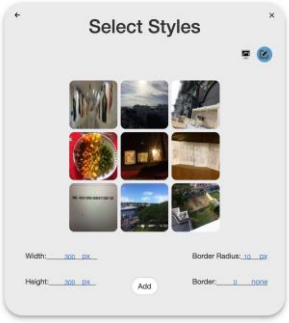


Figure 44: *Image Properties*

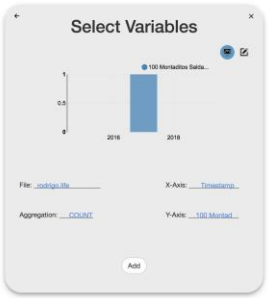
The second section contains the image div preview, which incorporates the selected images, and the properties that customize them. The preview is updated whenever the system detects a change in the input properties, except when those properties are the *Width* and the *Height*. Additionally, to these properties, the system allows the user to change the *Border Radius* and the *Border* of each image on the container div.

- *Area, Bar, Line and Pie Charts*

Similarly to the *Images,* the *Area, Bar, Line and Pie Chart* categories have two sections with an interactive preview that updates whenever an input field changes.

In the first section, the user is able to select from which file the system should obtain the data to display, the x-axis description, the y-axis values and the aggregation type, which could be SUM, MAX, MIN and COUNT. Since our application is based on lifelogging data the x-axis values of every chart represent the timestamp. In ***Figure 45*** we observe an input that is attached to the label *'X-Axis'*, this label represents the information displayed based on the x-axis value and present, for example, the description of an activity

performed on that date.



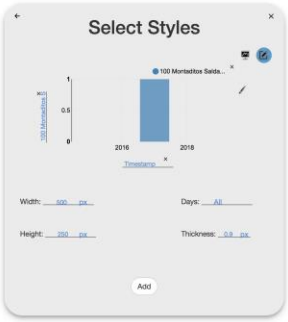


Figure 45: *Bar Chart Selection* ***Figure 46:*** *Bar Chart Properties* ***Figure 47:*** *Pie Chart Properties*

The second section was developed to provide methods to change the axis labels, the graph color, the width, the height, the days and the thickness. The ***days'*** input implements an algorithm that is designed to update charts by introducing all the days, including the ones that do not have data (inserted with the value zero), or only the days that contain information. For example, if we upload a file with two values for two discontinuous days, the system is prepared to display only those two days or with the in-between dates with the value zero assigned.

From the set of properties mentioned the thickness is the only that is restricted to the Bar Chart category. This property, as the name suggests, change the thickness of each bar within the current Bar Chart.

The *Pie Chart* category does not contain the *days* input but, instead, it includes the *Label Type* property that defines the type of information that is shown on the graph labels (could be percentage, key or value). A donut/pie chart button, which allows to alternate the graph between a pie chart and a donut chart. And a Donut Radio to define the inner space radius of the donut.

- *Map*

The *Map* category design diverges from the developed solution applied to the graphs containers. Although both contain a preview, the map encloses a list of possible map tiles, to customize the visualization, and three checkboxes that add or remove the respective markers from the map visualization. These three markers represent the *Photos*, *Locations* and the *Routes*.

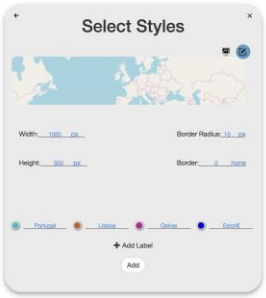


Figure 48: *Map Selection* ***Figure 49:*** *Map Properties*

On the second section of the map, the user is able to change the width and height, such as on the charts container, the border, border radius and the labels. The user can change up to sixteen labels that can be added by clicking on the '*Add Label*' and inserting the name of the location. The color is generated automatically. However, it can be modified by selecting the respective circle and choosing a different value. This feature was developed to provide a method that changes the color of the markers popup that opens whenever the user clicks on a location marker. We will address this subject in this document.

- Timeline

The *Timeline* container is the simplest from the set of visualizations created. Analogously to every other visualization category, this container provides a real-time preview that displays all the preferred properties changed by the user.

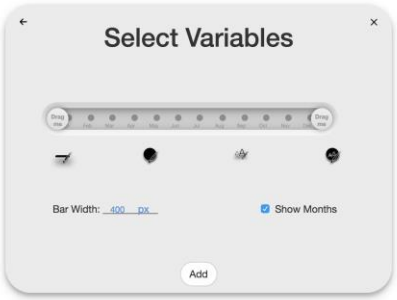


Figure 50: *Timeline Selection*

The *Timeline* category encloses four icons that personalize the timeline visualization. The first icon changes the bar color, the second the draggable circles color, the third the month's font color and the fourth the draggable circle's font color. Width and months display can also be changed.

- Heatmap Calendar

The *Heatmap Calendar* first section is equal to the charts first section. It contains the calendar preview, the filename input, the data aggregation type, the timestamp input (similar to the x-axis on the charts) and the variable input (similar to the y-axis of the charts).

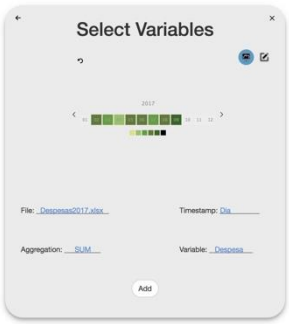


Figure 51: *Heatmap Calendar Selection*

Figure 52: *Heatmap Calendar Properties*

The second section provides two interactive buttons, near to the calendar's legend, that change the color that represents the lowest and the highest values of the calendar. When the user updates one of them, the system runs an algorithm that interpolates these colors and creates four intermediate ones. These are used to paint the days on the calendar according to its respective value. The user can also select if the calendar should highlight the today's date, the year or month label position (top, bottom, right and left), if the legend is visible or hidden and its position (left, center and right).

- *Text*

The *Text* container was developed to provide a method to insert random text on the interface. This could be used to add graph descriptions, titles, dates or any other pieces of information that the user might find useful to fulfil the other visualizations.

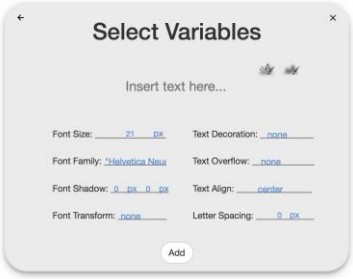


Figure 53: *Text Selection*

The *Text* unique section has a text input preview, where the user is able to type the text to insert and several options that change its design. We provide operations that modify the font size, font family, font shadow, font transform, font color, font color shadow, text decoration, text overflow, text-align and letter spacing.

3.2.2.1.1.4 Help Menu

While using the interface, the user might encounter some difficulties or have some doubts on how to perform certain actions. This problem will potentially be more frequent at the beginning of the usage of the interface, where the user is not used to all the system features.

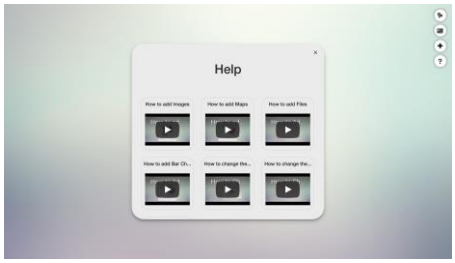


Figure 54: *Help Menu*

The *Help Menu* is a collection of short videos, created and uploaded on YouTube exclusively for this purpose, that explain how to add visualizations, how to change their design, how to use the features from the *Settings Menu,* and other relevant subjects that might help the users to overcome several obstacles that might appear.

3.2.2.1.2 Visualizations

The *Visualizations* are replicas of the previews that we referenced in the previous subsection, inserted on the dashboard with the properties selected by the user. Every time that a visualization is created the system inserts its data on a specific array that is, posteriorly, used to save it in the *database*.
After inserted, each visualization contains a draggable settings bar, collapsed on a menu icon, that allows to expand and display several buttons performing the following operations, in real time. The first button, on this bar, allows the user to drag and drop the visualizations on the interface dashboard (using *interact.js* [aa.]), while the second, which is a resizable button, allows to increase or decrease their size (*jQuery resizable* [bb.]). The third brings the visualization to the front and the fourth sends it to the back, both control what visualizations stay above or under other visualizations. The fifth is used to enable and disable updates, whenever the user triggers a time event (could be sent by the timeline or by interacting with any other visualization). The final button removes the respective visualization from the interface dashboard.

In this subsection, we will explore the interactions that we provide on each visualization and talk about the frameworks that we relied on to accomplish them.

3.2.2.1.2.1 Images

One of the most usual visualizations that human brain associate to our memories and past experiences are the photos. They can illustrate vast volumes of information that cannot be expressed by words "*a picture is worth more than a thousand words*".

Figure 55: *Image Visualization*

To build an interface that is focused on showing past experiences it was mandatory to include photos on the visualizations set. Each visualization can contain up to nine photos and all of them are independent, allowing to drag and resize a picture without affecting the others.

With these features, we can arrange and personalize our photos visualizations to highlight important or gorgeous images with simplicity and we can insert more than one picture without creating different visualizations.

3.2.2.1.2.2 Area, Bar, Line and Pie/Donut Chart

The Area, Bar, Line and Pie/Donut charts share a range of properties that are common to every chart on the interface. To develop them we used a library, named *NVD3* [n.], that build charts and their components for *d3.js* [cc.], a "*JavaScript library for manipulating documents based on data*", without removing the features that this script provides to users.

Additionally, to these features, the system includes, for every chart that the user inserts on the dashboard, a mechanism to ascertain if the data timestamp corresponds to days, months or years. The opacity is the visual key that is modified accordingly with the date level, for example, the opacity is assigned to 0.5 if the date level is equal to the days, 0.75 to the months and 1 to the years.

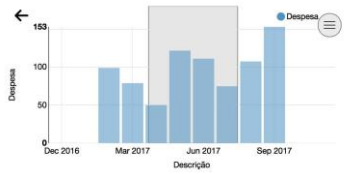


Figure 56: *Bar Chart Brush*

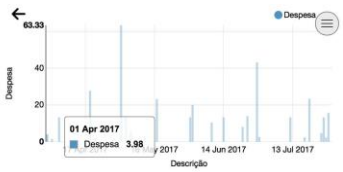


Figure 57: *Bar Chart Brush result*

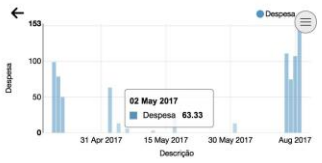


Figure 58: *Bar Chart Click result*

The Area, Bar and Line Charts share a unique interaction called brush, that provides the ability of selecting a time interval, through clicking and dragging inside the graph, as we can observe in ***Figure 56***. After this interaction, the system verifies the x-axis values regarding the initial and the final position of the brush section and dispatches a time event, with the selected time period, to every visualization. In ***Figure 57***, we can observe the result of the process, denominated as "zoom in", generated by brushing the Bar Chart from the ***Figure 56***. Whenever this feature is used a back arrow is shown, on the left corner of the graph, and its function is to trigger a time event that returns the date level immediately above to the current one. We created the date levels to each one of these charts and developed a set of conditions to guarantee that if the current date level is the days, the operation of "zoom out" would produce a result containing the date level of the months. The same process is applied to the transition between the months level and the years level.

The *Bar Chart* also includes another interactive feature, analogously to the only interaction of the *Pie Chart*, which allows to expand a year or a month by clicking on the correspondent bar. If the user triggers a click event on a bar that represents a year, only this bar will be expanded to the months level and the remaining ones will maintain the year level format. The same process, represented in ***Figure 58***, is observed when the user clicks on a month bar, which is expanded to the days level.

All of the charts display a tooltip whenever the mouse hovers one of their values.

3.2.2.1.2.3 Map

Another usual visualization that our human brain associates to the location memories and past experiences is the map. The *Map* is probably the most complex visualization on Follow My Steps system and, therefore, the most complete of them. To create it we use a JavaScript library for interactive maps, designated as *Leaflet* [dd.].

Figure 59: *Map Visualization*

Each map inserted on the interface dashboard has four different layers that represent interactive elements, which can be hidden or set to visible by unchecking/checking the checkboxes present in the right bottom corner.

The first interactive layer encloses the photo markers, which correspond to the photos taken at 50 meters or less radius distance from the marker location. When the user zooms into a certain level, the map adds circles corresponding to the area covered by each photo marker.

If the number of images, inside a certain location, is bigger than one, the photos are displayed within a slider that also presents the time, name and location of the current picture.

The second interactive layer encloses the location markers, which are the places where users have been during a specific time of their lives. Each location marker has a click event that opens the correspondent popup, containing information about how many time users have been there. The time range (for example, from the 23rd January, 2017, to the 14th February, 2017). The sum of all the days, hours and minutes and a calendar icon, that shows a heatmap calendar with the days painted accordingly with the number of times that users have been there, on that specific day (we will address this visualization further in this document).

The third interactive layer corresponds to the user current location marker on the map. This service uses the navigator object, containing information regarding the browser, that allows to obtain the current location through the geolocation method. The popup associated with the marker provides the place, street, city and country of the user's actual position, as well as the nearest places.

The final interactive layer is designated as routes, which correspond to the tracks where the user has been, and it is represented as blue lines that connect two or more locations. Similarly to the markers, these lines have a clicked event property that opens a bound popup. The content of this popup includes

the name of the track, the addresses of the origin and destination, the distance that separates both and the transportation method (for example, car, walk, plane, etc).

3.2.2.1.2.4 Timeline

In order to analyze their personal data, within a certain period of time, users have to change constantly the period of time on the *Timeline Menu* (subsection ***Timeline Menu***). To complete this procedure, the user has to open the menu, focus the date text input and browse the date picker to the desired date. For some users, this process can become monotonous after being executed for many times.

To create an alternative option to the *Timeline Menu,* the system provides a visualization that returns the same results but in an efficient and faster approach.

Figure 60: *Timeline Visualization*

The *Timeline* visualization is made of a horizontal bar with the month names, to guide the user while selecting a new date, and two circles, that represents the starting and ending dates of the interface time interval. These circles provide two interactive events that allow the user to drag and drop them on the desired date, and to double click them, to open a popup with a date text input. This second event was an important requirement, especially, to present a method that would allow the user to change the year inside the circles.

3.2.2.1.2.5 Heatmap Calendar

To provide a simple visualization that englobes the daily information, from an uploaded file, we developed a heatmap calendar generator that uses the JavaScript module *Cal-Heatmap* [ee.], which contains a simple API to create and customize calendar heatmap based on the *d3.js* [cc.] library.

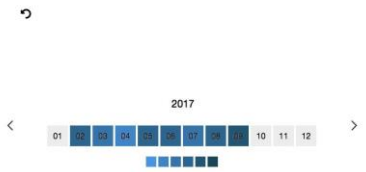


Figure 61: *Heatmap Calendar Visualization (Years)*

Figure 62: *Heatmap Calendar Visualization (Months)*

The *Calendar Heatmap* can be arranged in months, as we can observe in ***Figure 61***, or in days, ***Figure 62***. By clicking on the back icon, situated on the left upper corner, the user can switch the days view to the month's view, and the reverse process can be achieved by clicking on the respective month square. Whenever a day or month is hovered by the mouse, the visualization presents a tooltip with information regarding that specific date. The colors, which can be selected by the user, are arranged on a scale. The first represents the lowest and the last one the greatest value found on the visualization file. For example, if the calendar would be filled by a file with charges from 1€ to 60€, the lighter blue would correspond to the value from 1-10€ and the darker blue to the value from 51€-60€).

The side arrows allow to navigate back and forward on the calendar year or months.

3.2.2.1.2.6 Text

The *Text* visualization, as mentioned above, is a string that can be used, for example, to add descriptions, notes, ideas and any other complementary information on the interface dashboard. This string can be a word or a sentence that the user writes on the input field presented in the *Text* container.

3.2.2.2 Server Feedback

Follow My Steps architecture is based on a client-server model, as we mentioned in the ***Architecture*** section, that uses an external module, named *socket.io*[25], to exchange messages regarding the operations processed on the server-side. These messages are received by the client-side and arranged, based on its content, in order to display the correct information to the user. Each feed message is displayed during 15 seconds, except the mobile code messages that disappear after one minute, and they are composed of an image and a string, that present an illustration and a description of the operations performed by the server.

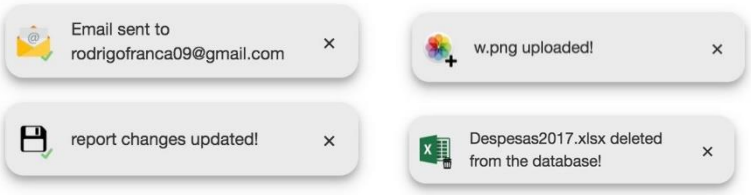


Figure 63: *Feedback Messages Examples*

3.2.2.3 Mobile Interface

The system browser version is focused on showing personal data within a certain period of time. To focus on position, we developed a mobile app that shows information about where users were and their current locations. The app is simpler than the browser version because it contains predefined visualizations that cannot be moved or erased.

When the user accesses the app, for the first time, the system launches an alert requiring the IP and Port of the machine that is running the server. The alert is composed of two text input fields, associated with these variables that will only dismiss if the user types a connectable address. After this step, the backend produces a code that is sent to the client-side, which is displayed on the web interface, allowing the user to type it on the mobile app to successfully connect to the server-side.

Once the code is verified and accepted, the app ascertains its current position, requests the nearest locations, the photos taken and the activities performed on the last day that the user was in the current place. The system also runs a period of time to update the interface accordingly with the mobile current position.

The map visualization, present on the app, is a similar and simpler version of the map created for the web interface. This map only provides two interactive elements, which are the location markers that represent the places where the user has been and the “my location” marker that corresponds to the user’s current position. Additionally, the map has a visual element, which is a circle that shows the covered area selected. The user’s “my location” popup content is identical to the information shown by the same popup developed on the browser version. On the other hand, the locations popup has the same differences. They keep the time, days, hours and minutes at a certain place, but remove the heatmap calendar and the time range.

The map also includes a fixed button that provides an interaction that locates and navigate towards the user’s “my location” marker. Since this visualization fills the entire window, the app contains an arrow button, on the right bottom corner, allowing the user to scroll down to the following sections. The first of the four sections that complement the map gives an insight about the user’s current location, with information about the street, city and country; the second displays a list of fifteen or less items that correspond to the closest locations; the third and four shows the schedule of the last day in the current location and the photos taken on the current place;

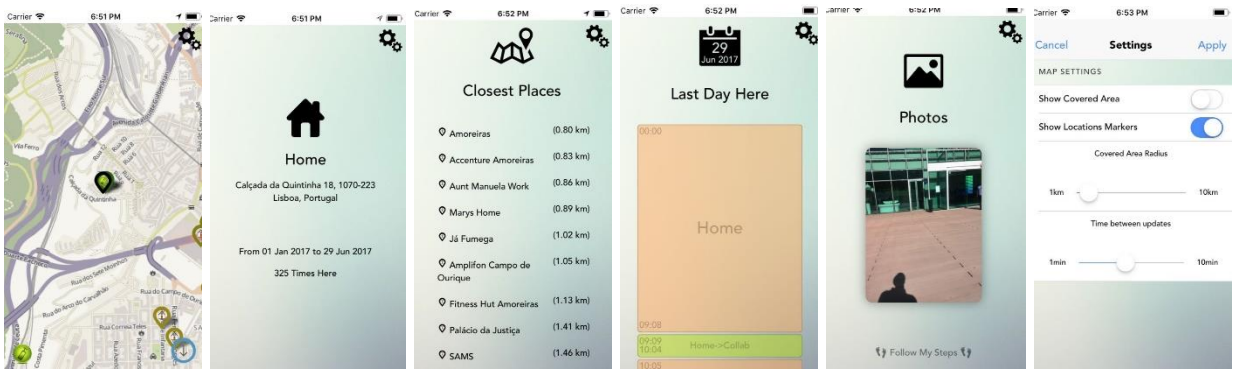


Figure 64: *Mobile Interface*

The mobile app settings allow users to change several default values accordingly with their own needs and necessities. It handles the visibility of the covered area and locations markers, the size of the covered area and the app update time.

3.2.3 Usage Examples

In this section, we are going to present some examples of possible arrangements for Follow My Steps visualizations. Since one of goals of this program is focused on the customization, users can build incredible interfaces with their imagination and their personal ambitions.

After a great year traveling around the world, Michael Brown and his wife looking forward to use the Follow My Steps system to register some of their travels details, in order to help them recall their amazing moments. They want representative photos of the best moments to appear on the top of the page with a title showing the correspondent year. They decide to create a simple visualization dashboard, so they only include the times they have been at their hotels, represented as a heatmap, the number of concerts they have attended per month, as a bar chart, and the times they have eaten their favorite meal, the “Super Steak”, as an area chart.
Michel also keeps all their expenses, represented as a line chart, to study the best solutions for the next year trip.

Figure 65: *Travelling Interface Example*

Follow My Steps Inc. is a company that develops software solutions for several web and mobile platforms. On the last decade, they started to increase their clients and, consequently, the income, the number of employees, among many other aspects. The CEO is a very organized individual, so he decided to use a framework, developed internally, allowing to analyse the company's information, with frequents automatic updates. These updates were set to run every minute, which generates several requests to the server. However, the young CEO is not worried about as during the development phase the team implemented a cache feature that reduces the messages exchanged with the server-side.

Finally, he wanted to receive a daily report of the company's updates. This way, he was able to define a report service to send automatically an email every day at 9 am.

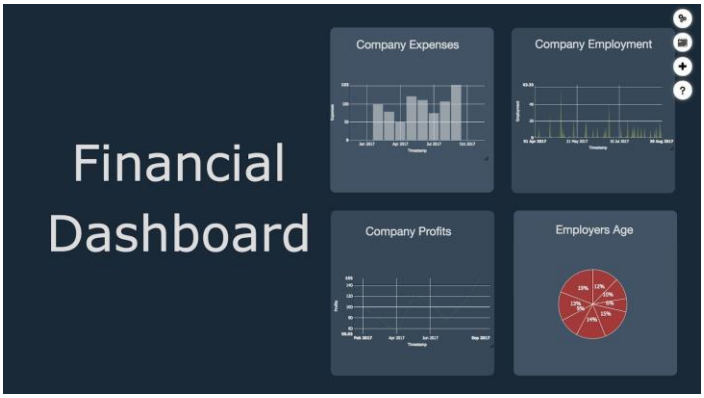


Figure 66: *Financial Interface Example*

After long days fighting, Jon Snow decided to review the previous seasons and to register the places where he has been, along with the routes, photos and white walkers killed. He added a map to his dashboard and included the locations he already visited on Westeros. To add a different design to the interface, he chose to change the font color and the font-family present on the interface, as well as the background image. To vaunt himself he inserted a bar chart where he displays the number of dead walkers killed per season.

With all is set, Snow only have to worry about one thing "Winter is coming"!

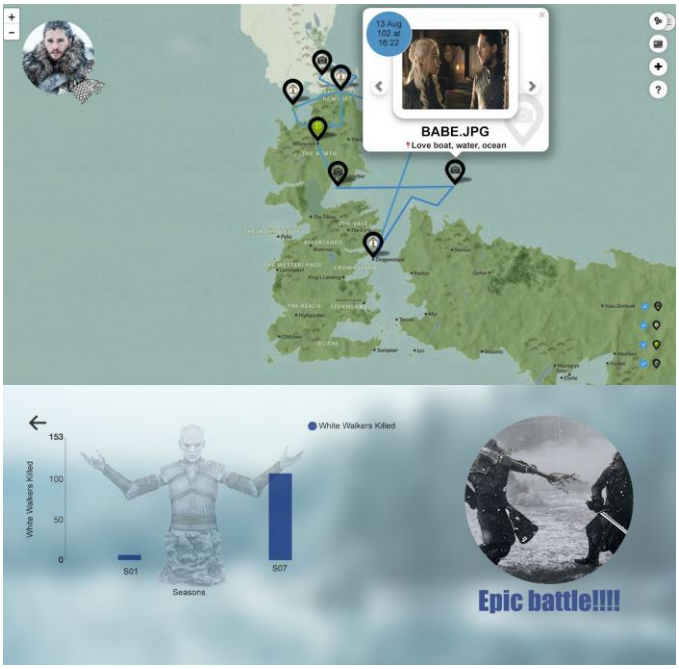


Figure 67: *Jon Snow Interface Example*

Mariah Peterson is a young lady that loves to visit her grandmother when she is lonely on her summer house. She often decides to spend one or two weeks during the summer vacations, enjoying the beach and the company of her friends. Unfortunately, last year, her grandmother was very sick and felt the need for some specific medicine. Since Mariah only went to the pharmacy two times, she acknowledges its existence in a nearby location, however she is not sure where. So she grabs her phone from her pocket and turns on the Follow My Steps application in order to get all the nearby places that she went on the past. She knows that since she only went there two times the color of the correspondent marker will be more orange than green. She quickly discovers the correct marker and runs to the pharmacy.

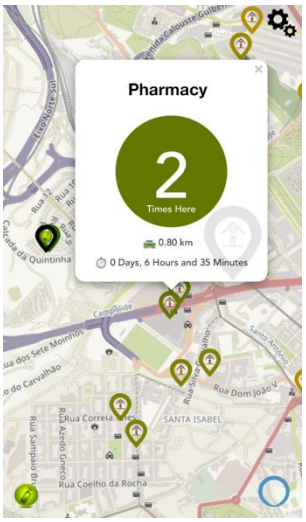


Figure 68: *Follow My Steps Mobile App*

4. Usability Evaluation

The concept of *Usability Evaluation* refers to evaluating a product with representative users, allowing to identify several issues on the earlier stages of the product development.

To evaluate the Follow My Steps system we conducted usability tests with 21 participants. Each usability test was divided into two sections; the web tests, containing 12 tasks, and the mobile tests, containing 6 tasks. During the tests procedures we monitored the number of clicks, the time spent and studied the user's choices used to accomplish the required tasks.

From the 21 participants that underwent the experimental evaluation, 14 were male and 7 were female. Through the results observation we infer that the gender did not affect the performance or the efficiency of the tasks required on this evaluation process. However, the age had a major influence on understanding how the interface works. The worst performances are connected to users aged between 46 and 60 years old.

The selected participants to evaluate this system knew how to interact with computers and mobile phones, and were used to work with other web interfaces and mobile apps. This knowledge was a major requirement to guarantee that we observe and study the problems faced throughout the interface interactions, instead of the problems faced throughout the computer or mobile phone usage.

After concluding the Follow My Steps web version and mobile app tests, it was requested to fill three questionnaires to ascertain the participants' opinion about the usability and complexity of the interface. The first questionnaire encloses questions about the user's feelings regarding the usage of the web version of this system. The second questionnaire is similar to the first but inquires about the mobile version. The final questionnaire encompasses several areas, such as application usefulness, easiness to use, motivation level, among others. In all of the questionnaires, users had the possibility to provide a written feedback.

4.1 Experimental Protocol

The main goal of this thesis is to discern what are the best visualizations that represent the user's memories and past experiences. To assess this system it was necessary to produce a set of predefined tasks, both for web and mobile that would enforce the interaction between the user and specific interface features.

The web version evaluation starts with a brief explanation of the concept of liffeloging, where users learn the ideology of this practice and its main purpose. The explanation extends to the interface goals and usage, where the user acquires a basic insight of the available interactions and operations provided by the system.

Once the explanation is over, the user is allowed to test and explore the interface features, without being observed, within a period of time of 5 minutes. The tasks process that begins after the experimentation time is composed of the following assignments:

1 → Add file 'Despesas2017.xslx' to the Follow My Steps database

2 → Find how much cash I spent in September, 2017

3 → Find how many times I've been on IST;

4 → Find in what Days I've been in Amoreiras;

5 → Find how many times I've been in Fitness Hut between 1st Feb., 2017 and 1st June, 2017;

6 → Find the photos I took in the Railway Station of Pragal;

7 → Find the places I've been on the 17th Jan., 2017;

8 → Change the Background Image

9 → Send Monthly report to followmysteps@gmail.com

10 → Find the transportation method that I used to go to Alegro

11 → Remove all the visualizations from the Dashboard

12 → Delete file 'Despesas2017.xslx'

These tasks go through the main functionalities of Follow My Steps' web version and allow to appraise the difficulties on understanding how the system's features works. All of the tasks, excluding the first, the tenth and the eleventh, were randomly arranged to guarantee that the tasks' results are independent from their order, therefore the results ensure that the difficulties on completing one task aren't affected by the tasks sequence. During each evaluation, we measure the time to perform each assignment and the number of clicks made by the user. Since the interface does not provide unique methods to achieve the tasks aim, we did not measure the number of errors.

Once this phase is complete, we requested users to fill a questionnaire related to the user's experience while using the web version (***Appendix A – Browser Questionnaire***).

This questionnaire is composed of 12 linear scale questions that can be rated on a 5-point scale (1-'Very Easy'; 5-'Very Hard).

Similarly to the web, on the mobile app, the evaluation begins with a short explanation on how the system operates on this platform. Due to its simplicity, the experimental period is discarded, and we advanced to the tasks phase. The assignments for the mobile app are:

1 → Find my current position (street, city and country)

2 → Verify how long I have been here the last time I was in this place

3 → Did I take any pictures on this place?

4 → Check if I've been in any restaurant near my location

5 → Change the radius of the max distance

6 → Set the update time to 5 minutes

These tasks englobe the main functionalities, of the Follow My Steps app, and provide an insight on how the user handles the system. The order is also randomly arranged for the same reasons mentioned in the browser version and we measure the performance by counting the time and the number of taps.
Analogously, at the end of the mobile evaluation phase, we request to fill a questionnaire related to the mobile app usage experience (***Appendix B – Mobile Questionnaire***).
It encloses 6 linear scale questions that can be rated on a 5-point scale (1-'Very Easy'; 5-'Very Hard).

The evaluation process ends with a satisfaction questionnaire (***Appendix C – Satisfaction Questionnaire***), in order to assess the efficiency, effectiveness and satisfaction regarding the Follow My Steps' system. It contains a similar approach to the System Usability Scale (SUS) form that allows to collect qualitative data to measure, within the usability criteria, the degree of satisfaction of users while testing the browser and mobile versions. The SUS is a 10-item questionnaire, where 8 are related to the structure usability and 2 with the learnability, used to assess the subjective perception of interaction with a system. This form does not contain questions, instead, it presents statements that can be rated on a 5-point linear scale (from 1-'Strongly Disagree' to 5-'Strongly Agree'). This scale has been widely adopted within the engineering community since it integrates the three basic concepts of usability criteria:
The first is effectiveness, which is the ability to successfully achieve the system objectives; the second is efficiency, that measures the efforts required to complete the system's tasks; and the third is satisfaction, that can be described as the positive and negative reactions emerged from the system usage.

4.2 Results

During the usability tests, we measure the number of clicks/taps and time required to, successfully, complete the tasks. The output data from this procedure was, subsequently, processed and analyzed to examine what were the major difficulties and the facilities while performing each task.
In this section, we will scrutinize the data collected from our direct observation of the tests and from the questionnaire responses. Furthermore, we will elaborate our conclusions based on all the data gathered and study the comments and suggestions provided by users throughout the usability tests.

4.2.1 Web Version

As mentioned earlier, we measure the number of clicks performed by the user during each task. We decided to display the results in a *Multiple Bar Chart* due to its ability to provide an insight allowing viewers to easily make comparisons among values.

Through the analysis of ***Figure 69,*** we can infer that the average number of clicks per task is, on the majority of the cases, greater than two times the number of minimum clicks.

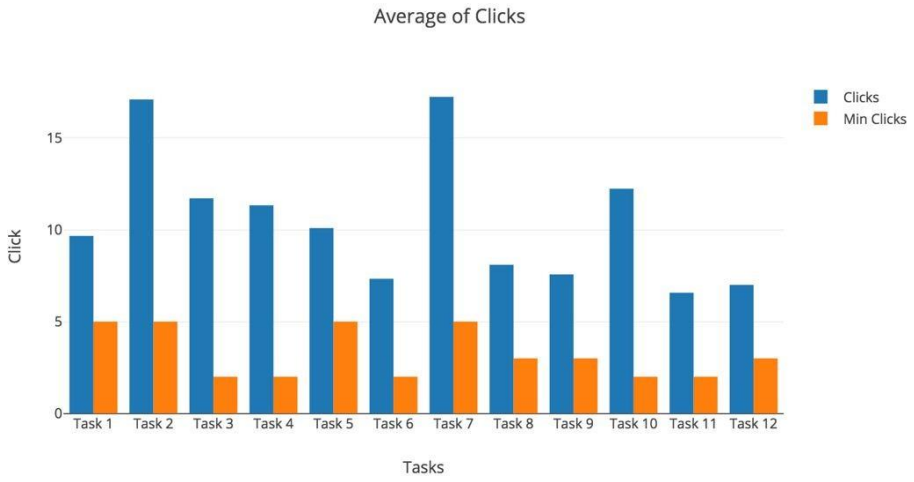


Figure 69: *Browser Clicks*

There are distinct reasons that can be related to this behavior, considering the fact that users did not had contact with the interface until the beginning of the usability tests.

The first, and the major disadvantage of users, was their unawareness regarding the shortcuts provided by the system. As we quoted, on the subsection ***Menus***, each category of the menus can be accessed through keyboard interactions. This information was not revealed to the user, in order to study their reaction towards the keyboard. From what we observed, only 3 out of 21 users successfully used the keyboard as shortcuts to perform operations that supported the tasks performances.

Another disadvantage that influenced these results was their experience absence. Before every usability test, we granted 5 minutes where the user was allowed to freely explore the interface, to understand and examine the system functionalities. Follow My Steps provides an interface that contains several features, as we observed in the ***Solution*** section that requires more than 5 minutes to learn and perceive all of their abilities. Since we only provide this small period of time, the learning process decreases its efficiency and effectiveness, allowing us to explore not only how the user achieves a task solution but also, which were the difficulties found while performing them.

Similarly, to the keyboard shortcuts, we omit the visualization interactions to simulate the starting point of the interface usage. With this approach, we can observe how users perceive the possible interactions that handle each visualization.

Through the examination of the values, from the '*Average of Clicks'* chart, we can identify two major columns that go above the 15 clicks. These columns correspond to task 2 (*Find how much cash I spent in September, 2017*) and the task 7 (*Find the places I've been on the 17th Jan., 2017*). To accomplish these tasks users often used the timeline, to change the period of time of the interface, which provides a navigable date picker that, usually, requires several clicks to insert a new date.

On task 5 (*Find how many times I have been in Fitness Hut between the 1st Feb., 2017 and the 1st June, 2017*), users did not show the same behavior, which justifies the fewer number of clicks observed. The majority decided to count the number of days through the map calendar presented on each location popup, as we mentioned in the ***Map*** subsection, instead of changing the interface time interval. This procedure reduces the number of clicks but increases the number of errors and the time spent during the task.

To represent the task's execution time we created a graph, which can be observed in ***Figure 70***, containing a *Box Plot* and a *Bar Chart*. The *Box Plot* allow us to observe important details to measure statistical dispersion. With this chart, we can ascertain the time's full range variation (from the minimum value to the maximum value), the probable variation range (interquartile range) and the typical value (the median). The *Bar Chart* provide us an insight of the minimal time, recorded while performing the usability tests with an expert, accomplished on each task.

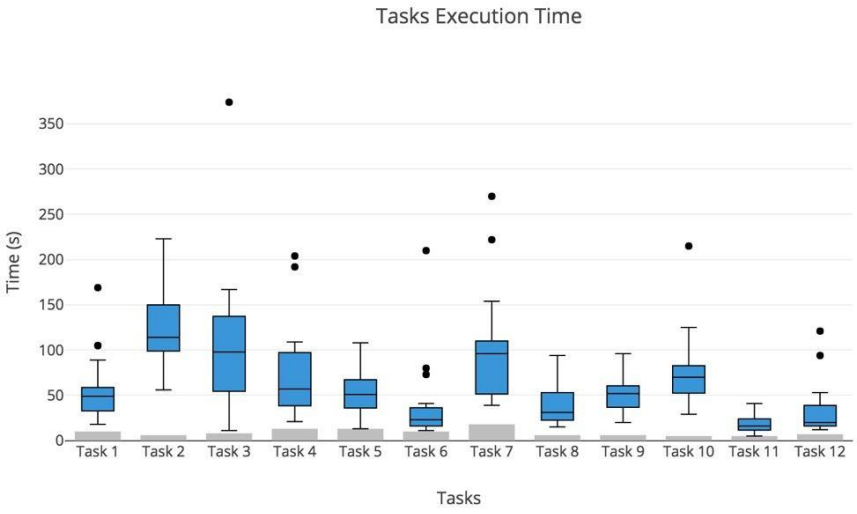


Figure 70: *Task's Execution Time (Browser)*

Accordingly to ***Figure 70***, task 2 (*Find how much cash I spent in September, 2017*) was, on the overall, the assignment that took more time. This observation is obtained by analyzing the median of each task (in this case is 1 minute and 54 seconds). However, the higher time to perform an assignment was registered by a user that required 6 minutes and 14 seconds to, successfully, complete task 3 (*Find how many times I've been on IST*).

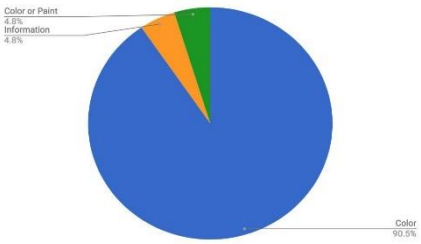
During the usability tests, we observed a higher difficulty from the participants aged between 45 and 60 years old. The set of results, produced by their tests, contained values that diverged on a large scale from

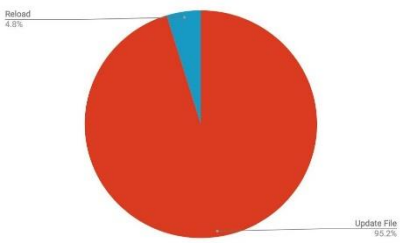
the values produced by other results. The values that are detached from the variation range are considered as outliers, which are, commonly, ignored by analysts on statistical operations.
In contrast, we observed faster results from users that began to feel acquainted with the interface usage. We registered, on two tasks, similar execution times between the user performance and the minimum value obtained. On task 6 (*Find the photos I took on 'Pragal Railway Station'*) we observed an execution time equal to 11 seconds, 1 second more than the 10 obtained by the expert, and on the task 11 (*Remove all the visualizations from the Dashboard*) the fastest result was 7 seconds, more 2 than the expert execution time. These two tasks were also the fastest tasks that were performed, where the median of task 6 is 23 seconds and the median of task 11 is 16 seconds.

At the end of the usability tests, for the web version, we requested users to fill a questionnaire, in order to understand their opinion regarding the interface usage, and, in overall, their results showed a satisfying feedback. To have a real idea about what users thought regarding the system's usage, we included a set of questions related to the tasks they performed. In these questions, users had to select the difficulty level felt while performing the tasks, using a numerical linear scale from 0 (very easy) to 5 (very difficult). ***Figures 71***, ***72*** and ***73***, helped by ***Table 2***, present the questions' results.

Questions	***Letter***
How difficult was it to add a new file to the system?	A
How difficult was it to insert a new chart?	B
To which label would you associate thefollowing image? 	C
To which label would you associate thefollowing image? 	D
To which label would you associate thefollowing image? 	E
How difficult was it to see where I've been?	F
How difficult was it to find the day where I was in a certain place?	G
How difficult was it to change the interface background?	H
How difficult was it to send Daily report?	I
How difficult was it to add a map?	J
How difficult was it to remove all visualisations from the interface?	K
How difficult was it to remove a file from the database?	L

Table 2: *Survey Questions (Browser)*





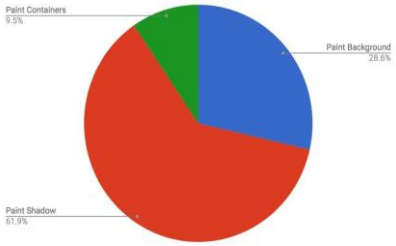


Figure 71: *Answers to C* ***Figure 72:*** *Answers to D* ***Figure 73:*** *Answers to E*

The aim of these questions was to ascertain if users's perception towards the icons inserted on the interface was identical to the operation that they perform. We only selected these three representative icons, to include on the survey, because we believe they represent the overall difficulty of the icons inserted.

As we can observe, in ***Figure 71***, 19 out of 21 users associated the paint-brush to the color label, which correspond to positive feedback since, on our system, this icon is used to change the color of the available visualizations. Through the analysis of ***Figure 72,*** we conclude that 20 out of 21 users selected the update file option, which represents the system capacity of performing this operation. So far we collected positive results, and intuitive, from the icons selected to represent several features. However, on the final icon the user had more difficulties to understand its purpose, ***Figure 73***. 13 out of 21 selected the label 'Paint Shadow', 6 out of 21, the 'Paint Background' and 2 out of 21, the 'Paint Containers. The icon is used to change the color of the font shadow, and we can observe that more than 38% of the participants were not able to understand this icon purpose.

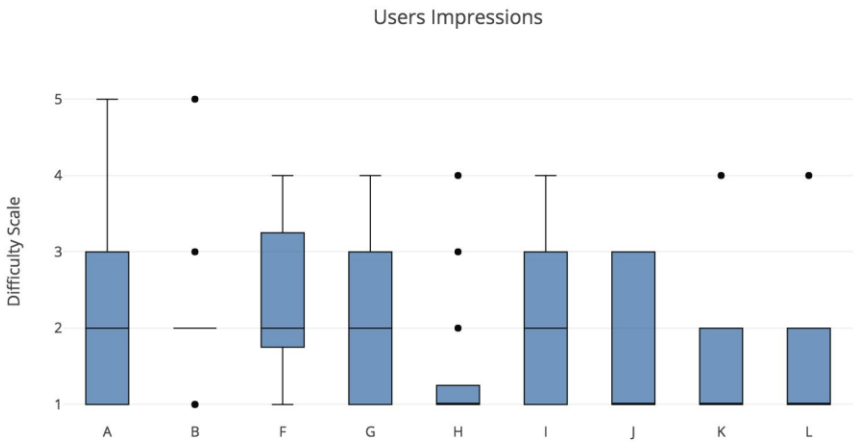


Figure 74: *User Impressions (Browser)*

Once again, we decided to use a box plot to show the variants of the difficulties felt by the participants during the usability tests. This chart is similar to the chart found in ***Figure 70***, but in this case our range is discrete and only takes values from 1 to 5 as we mentioned earlier.

To explain some values, on the second, representing the difficulty of insert a new chart, we have 3 outliers

and the reason resides in the 18 users, out of 21, that selected the option 2. When the majority of users select an option the and few users select other different options the column adopts a format similar to the one found in the B column. The H, J, K and L columns have their variant at 1, reason why we do not see a line separating the rectangle.
As we can observe, the majority of users felt that the assignments were easy to perform, since the variant of all tasks is under the value 3.

4.2.2 Mobile Version

Analogously to the browser version, the mobile users taped more times than required to perform the mobile tasks. The representation will also be on a Multiple bar chart to easily compare the values between the average number of taps required by users to perform the tasks and the minimum taps required to perform them.

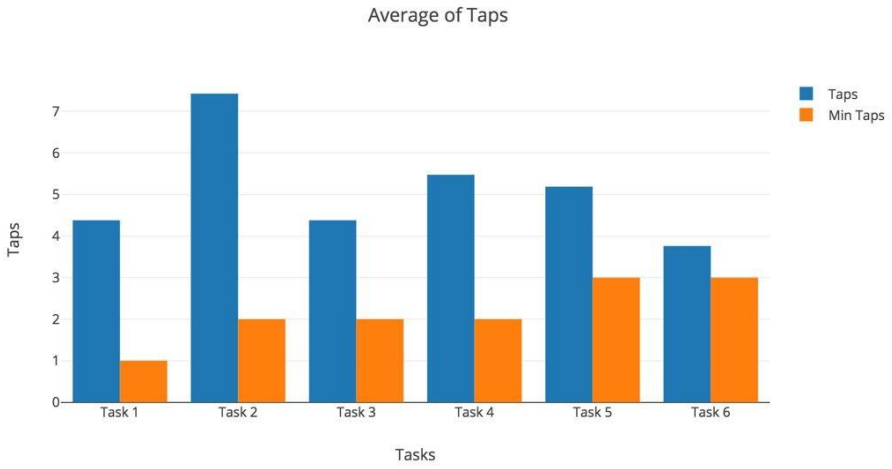


Figure 75: *Mobile Taps*

On the mobile version, the system is considerably less complex and it requires less interaction to perform the tasks with success. Unlike the browser version, the mobile does not have any shortcuts to help users to perform the tasks faster and with fewer taps.
On the usability tests for this version we do not provide an experimental period at the beginning of the test, instead, we start directly with the tasks. The reason why we skip the experimental period resides in our intention on ascertain how intuitive the icons and the interaction on the mobile are.

If we observe the graph, ***Figure 75***, we can infer that the user required more taps to perform the task 2 (*Verify how long I have been here the last time I was in this place*). An explanation for this behavior can be the time necessary to perceive that the calendar of the last day, on the current location, does not provide clickable interactions and to find the hours the user need to subtract the final hour and the initial hour (something that we will have to change in future work).

On the other hand, the task that demonstrates lesser clicks was task 6 (*Set the update time to 5 minutes*), and one of the reasons is because these actions are associated with the settings menu, so users know exactly what to do and where to find it.

The mobile execution time will, also, be presented on a box plot with a bar chart to provide an insight of the variation time, while performing each task, based on the minimal time to complete them.

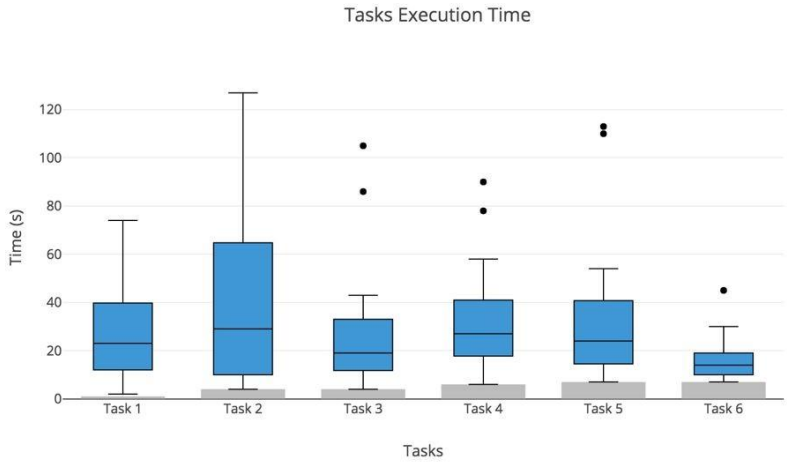


Figure 76: *Task's Execution Time (Mobile)*

As we can observe the task that took more time, for users, to accomplish was the task 2 (*Verify how long I have been here the last time I was in this place*). Its median value is 29 seconds and the max value registered was 2 minutes and 7 seconds. The faster task is the task 6 (*Set the update time to 5 minutes*) with the median equal to 14 seconds and the higher value registered was 45 seconds.

After the usability tests with the mobile version, we asked users to fill another questionnaire, but this time regarding the usage of the mobile app. These tests showed better performances than the browser tasks, due to its simplicity. Similar to the browser questionnaire we created a set of questions, to understand the user's opinion while using the app.

Questions	***Letter***
How difficult was it to locate my position?	A
How difficult was it to know how long I've been in here, the last time I came to this place?	B
How difficult was it to know if I took pictures in this location?	C
How difficult was it to check if I've been in a near restaurant?	D
How difficult was it to change the locations maximum distance radius?	E
How difficult was it to change the update time?	F

Table 3: *Survey Questions (Mobile)*

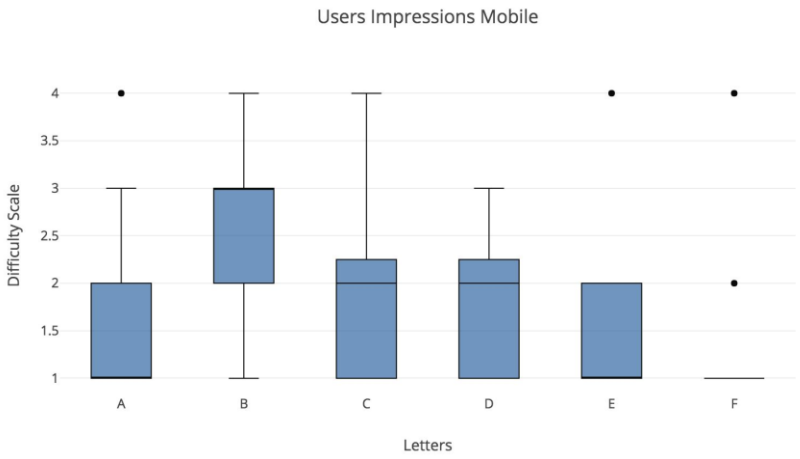


Figure 77: *User Impressions (Mobile)*

By observing the box plot that represents the users' impressions to the mobile app usage, we can infer that the task that they felt more difficulties was the task where they spent more time and made a bigger number of taps (task 2). More than half of the participants selected the difficulty equal to three.
The easiest tasks to perform, according to the users's opinion is task 6, which we also observed that was the task with fewer taps and with less time spent. From the 21 participants, 19 voted on value 1 (Very Easy).

4.3 Discussion

These usability tests provided important metrics to infer the major difficulties and facilities encountered while using the Follow My Steps interface.
As we mentioned earlier, at the beginning of the tests, the dashboard of the interface was empty, allowing users to build their own interfaces accordingly with their preferences. Each test started with the experimental period and we noticed that the visualizations absence was not emotionally positive for a set of users. On the suggestions section, from the web questionnaire, we read an opinion stating that the system should provide default themes, contained predefined visualizations, in order to display random information at the beginning of the usage. The problem with this approach consists on the fact that the system does not possess any data at that time, and the information needs to be uploaded by the user. A simple solution would be to redirect the user to the *Add Files* menu, in order to impose the requirement of inserting files on the *database*.

During the interface usage, the participants felt confident while performing the majority of the tasks.

We observed that most of users chose the map visualization to perform the tasks presented during the usability tests. All the tasks, except task 6 (*Find the photos I took on 'Pragal Railway Station'*) and task 10

(*Find the transportation method that I used to go to Alegro*), related with the visualizations, had more than one method to be successfully concluded. For example, users can use the map, the bar chart, the line chart, the area chart or the pie chart visualizations to ascertain the number of times that they had been on a certain place.

One of the possible reasons for this behavior is the association among locations and maps. When we are submitted to questions or tasks regarding specific places (such as, find how many times you have been on Lisbon) we tend to think on maps to search for the information requested.

A similar behavior was recorded while observing the user's performance on task 2 (*Find how much cash I spent in September, 2017*). However, in this specific task, users preferred the *Bar Chart* visualization technique in order to ascertain the value for this specific month. This option may be related to the fact that we associate the *Bar Chart* visualization to sets of values arranged over time.

To provide support to users we developed a *Help Menu*, which presents quick feedback on how the interface works, through a set of videos conceived only for this purpose. Throughout the tests, we observed that only 6 out of 21 users used the help to verify how to perform a certain task.

We conclude that the complexity of certain actions can involve some pre-learning experience. Despite this complexity, in overall, users were pleased with the simplicity of the interface and ease while executing the system's operations. However, there is still plenty of work that can be done to improve this system, considering the comments and suggestions provided by users collected throughout the usability tests and the final questionnaires. These improvements are related to a couple of small corrections on the interface and in mechanisms to turn things easier for users to handle with system's features.

5. Conclusions

The practice of Lifelogging started to increase, over the years, with the expansion of lifeloggers tracking their personal activities, by collecting their daily data, such as exercising, sleeping, or eating.
With the growth of smartphones in our lives, we can gather and store large amounts of data that can be used to help us remember past experiences through several visualization techniques.
This extraordinary increase of computer storage power generated several issues when developers try to display this quantity of information. One of these issues is scalability, which is the system's ability of handling the growth of the upcoming data inputted by users.

On this report, we describe the developed solution that allows users to easily recall personal experiences, in a simple and effective way, addressing problems such as scalability, as we mentioned before, and holistic view. As we were able to explore, on the ***Related Work*** section, there is a lack of solutions that provide efficient results for all of these problems. However, they offered us new perspectives and new insights on how to design an interface that could solve them without compromising the information that the user wants to display.

Our approach design was structured to implement a customizable interface, developed to allow users to personalize it according to their own preferences and interests. This particularity enabled us to understand which are the visualization techniques users associate to a certain set of data (map, bar chart, etc...), and how they choose to visualize it.
To evaluate this system we submitted it into several usability tests, which were focused on usability and utility, to identify some major and minor issues, in terms of design and functionalities, and to study the best visualization techniques used to represent specific personal information.
From the results obtained, we infer that users adopt the map visualization while trying to observe information related to a specific location. For example, the number of times or the days on that location. We, also, infer that data with measurable values are commonly represented in a bar chart. For example, the daily expenses or the number of travels.

To conclude this work, we deduce, from the questionnaires and users' feedback, that Follow My Steps is a helpful system that provides insights about the users' past memories and experiences.

5.1 Future Work

Follow My Steps has several aspects that could be improved to enrich not only its user's experience but also some of its backend mechanisms. To simplify this section, we are going to focus on the major improvements that we believe that are the most relevant ones and beneficial to include in the future.

During the usability evaluation, we observed several difficulties faced by the majority of users, while starting the interface tests. This set of initial difficulties have a considerable influence on users' willingness to learn the interface features and on their motivation state.
To smooth this problem, we could implement an initial help system to guide users through a set of predefined actions, such as uploading new files and/or the inserting new visualizations to the dashboard, in order to aid them to learn basic operations provided by the system.

Another issue that we want to highlight is the updating process of the file update feature. This process, as mentioned earlier, runs on the server side of this application and deletes specific previous stored information from the database to insert the changed file full content. The time spent to execute each update operation depends on the file size context instead of the changes performed. This is the most important feature, on the server-side, that needs to be updated in order to reduce the amount of information that needs to be managed.

Follow My Steps system is focused on presenting and displaying information through a set of visualizations techniques, however it does not provide any operation to collect that data on its own. Users are compelled to use other applications or to manually adjust the changes on the uploaded files. To avoid this monotonous cycle, we pretend to develop a data collectable module able of store information based on a given filename. The system would need to be able to recognize this file and adjust the information collected to that file format, (for example, if the user inputted the file type as a life file the system should collect temporal and geographical information).
This mechanism would save a considerable amount of time each time, to users, each time that a file is created or updated.

Bibliography

[1] Heike Otten, Lennart Hildebrandt, Till Nagel, Marian Dörk, and Boris Müller: "Are there networks in maps? An experimental visualization of personal movement data". *In IEEE VIS 2015 Workshop*.

[2] Gemmell J., Aris A., Lueder R.: "Telling stories with MyLifeBits*". In Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on (2005), IEEE*, pp. 1536–1539. 2

[3] Alice Thudt, Dominikus Baur and Sheelagh Carpendale et al. 2013. "Visits: A spatiotemporal visualization of location histories*". In Proceedings of the eurographics conference on visualization*. p. 79-83.

[4] Jae Ho Jeon, Jongheum Yeon, Sang-goo Lee, and Jinwook Seo. "Exploratory Visualization of Smartphone-based Lifelogging Data using Smart Reality Testbed". *In Proc. Big Data and Smart Computing*, pp. 29-33, 2014

[5] Rúben Gouveia and Evangelos Karapanos: "Footprint tracker: Supporting diary studies with lifelogging*". In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13*, pages 2921–2930, New York, NY, USA, 2013.

[6] Vaiva Kalnikait, Abigail Sellen, Steve Whittaker and David Kirk: "Now Let Me See Where I Was: Understanding How Lifelogs Mediate Memory". *CHI 2010*, April 10–15, 2010.

[7] Mohit Shah, Brian Mears, Chaitali Chakrabarti and Andreas Spanias: "LIFELOGGING: ARCHIVAL AND RETRIEVAL OF CONTINUOUSLY RECORDED AUDIO USING WEARABLE DEVICES*". In IEEE International Conference on Emerging Signal Processing Applications, ESPA*, 2012

[8] Bruno Pagno and Luciana Nedel: "Everyday Visualization". *In IEEE VIS 2015 Workshop*.

[9] D. Huang, M. Tory, and L. Bartram: "Data in everyday life: Visualizing time-varying data on a calendar". *Proc. Poster Compendium IEEE VIS*, 2014.

[10] Yang Yang and Cathal Gurrin: "Personal lifelog visualization". *Proceeding SenseCam 2013 Extended Abstracts, San Diego, USA*.

[11] Yang Yang, Hyowon Lee and Cathal Gurrin: "Lifelogging: New Challenges for Information Visualization on Mobile Platforms". *CHI'13, April 27 – May 2, 2013, Paris, France*.

[12] Hyowon Lee, Alan F. Smeaton, Noel O'Connor, Gareth Jones, Michael Blighe, Daragh Byrne, Aiden Doherty, and Cathal Gurrin: "Constructing a SenseCam Visual Diary as a Media Process". *Multimedia Systems 341-349, 2008*

[13] Tiffany Ng, Ou Jie Zhao and Dan Cosley: "pieTime: Visualizing Communication Patterns". 2011 IEEE

[14] Sudheendra Hangal, Monica S. Lam and Jeffrey Heer: "MUSE: Reviving Memories Using Email Archives". *UIST'11, October 16-19, 2011*

[15] Tsai Ling Fung and Kwan-Liu Ma: "Visual Characterization of Personal Bibliographic Data using A Botanical Tree Design*". In IEEE VIS 2015 Workshop*.

[16] 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.*

[17] D. McDuff, A. Karlson, A. Kapoor, A. Roseway, and M. Czerwinski: "AffectAura: an intelligent system for emotional memory," *in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, 2012,* pp. 849–858.

[18] Dominikus Baur,, Frederik Seiffert, Michael Sedlmair, and Sebastian Borin: "The Streams of Our Lives: Visualizing Listening Histories in Context". *IEEE Trans Vis Comput Graph*, 2010.

[19] Zaher Hinbarji, Rami Albatal, Noel O'Connor and Cathal Gurrin: "LoggerMan, A Comprehensive Logging and Visualization Tool to Capture Computer Usage". *In 22st International Conference on MultiMedia Modelling (MMM 2016), 4-6 Jan, 2016, Miami*

Web References

Last visited on July 8, 2018

a. *https://nodejs.org/*

b. *https://www.postgresql.org/*

c. *http://www.php.net/*

d. *https://www.java.com*

e. *https://reactjs.org/*

f. *https://angular.io/*

g. *https://www.mysql.com/*

h. *https://mongodb.com*

i. *https://postgis.net/*

j. *http://echarts.baidu.com/*

k. *https://taucharts.com/*

l. *https://chartjs.org*

m. *https://c3js.org/*

n. *http://nvd3.org/*

o. *http://cal-heatmap.com/*

p. *http://leafletjs.com/*

q. *https://d3js.org/*

r. *https://jqueryui.com/resizable/*

s. *http://interactjs.io/*

t. *https://getbootstrap.com/*

u. *https://www.npmjs.com/package/socket.io*

v. *https://www.npmjs.com/package/node-schedule*

w. *https://www.npmjs.com/package/nodemailer*

x. *https://www.npmjs.com/package/file-changed*

y. *https://www.npmjs.com/package/exif-image-auto-rotation*

z. *https://www.npmjs.com/package/reverse-geocoding*

aa. *https://www.npmjs.com/package/exif*

bb. *https://www.npmjs.com/package/base64-i mg*

cc. *https://github.com/domiriel/LIFE*

dd. *https://www.npmjs.com/package/simplify- path*

ee. *https://github.com/SheetJS/js-xlsx*

ff. *https://upload.wikimedia.org/wikipedia/commons/9/91/Douglas_Peucker.png*

gg. *https://plot.ly/create/?fid=PythonPlotBot:24*

Appendixes

Appendix A – Browser Questionnaire

Figure 78: *Browser Interface Questionnaire*

Appendix B – Mobile Questionnaire

Follow My Steps Mobile App

This section is related to the mobile version of Follow My Steps. It has 6 questions.

How difficult it was to enter on the app? *

	1	2	3	4	5	
Very Easy	○	○	○	○	○	Very Hard

How difficult it was to locate my position? *

	1	2	3	4	5	
Very Easy	○	○	○	○	○	Very Hard

How difficult it was to know how long I've been in here, the last time I came to this place? *

	1	2	3	4	5	
Very Easy	○	○	○	○	○	Very Hard

How difficult it was to know if I took pictures in this location? *

	1	2	3	4	5	
Very Easy	○	○	○	○	○	Very Hard

How difficult it was to check if I've been in a near restaurant? *

	1	2	3	4	5	
Very Easy	○	○	○	○	○	Very Hard

How difficult it was to change the locations maximum distance radius? *

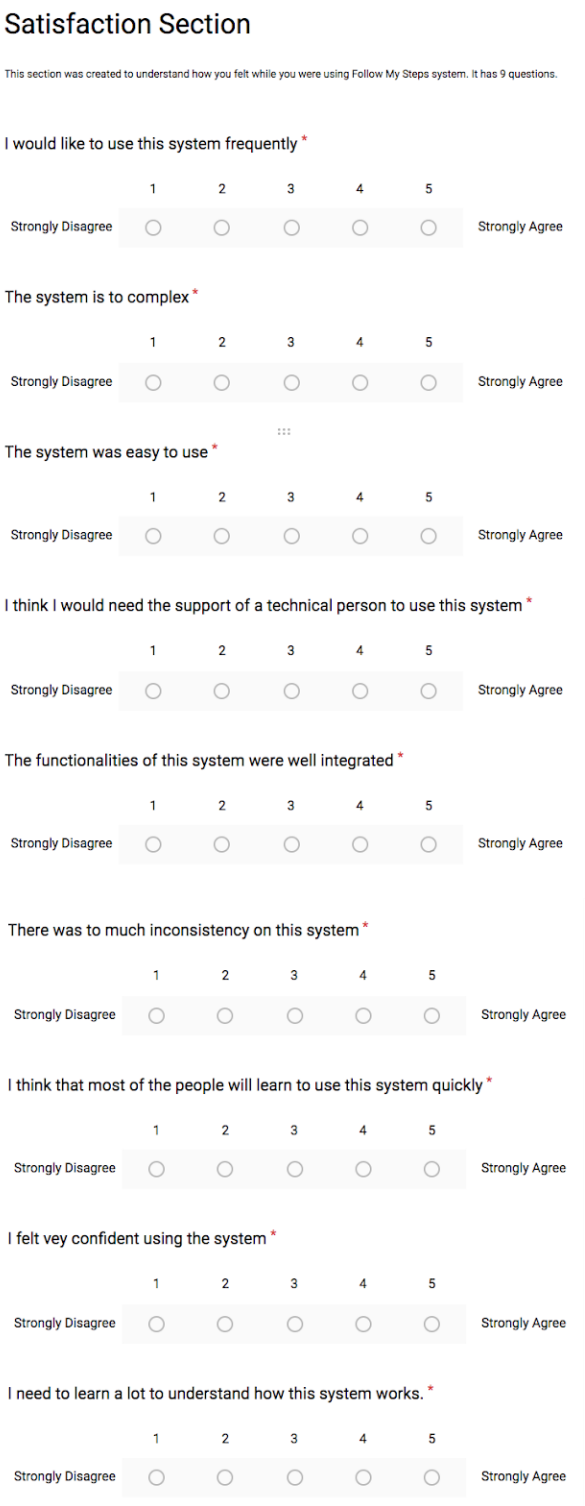
	1	2	3	4	5	
Very Easy	○	○	○	○	○	Very Hard

How difficult it was to change the update time? *

	1	2	3	4	5	
Very Easy	○	○	○	○	○	Very Hard

Figure 79: *Mobile App Questionnaire*

Appendix C - Satisfaction Questionnaire



Satisfaction Section

This section was created to understand how you felt while you were using Follow My Steps system. It has 9 questions.

I would like to use this system frequently *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

The system is to complex *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

The system was easy to use *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I think I would need the support of a technical person to use this system *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

The functionalities of this system were well integrated *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

There was to much inconsistency on this system *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I think that most of the people will learn to use this system quickly *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I felt vey confident using the system *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I need to learn a lot to understand how this system works. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Figure 80: *Satisfaction Questionnaire*

Appendix D – Tasks Clicks Results (Browser)

	Task Description	Top User	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
Task 1	Add Files	5	8	6	6	18	18	7	7	10	7
Task 2	Despesas Set 2017	5	19	20	13	28	18	13	17	13	13
Task 3	IST	2	25	12	9	14	4	7	13	17	5
Task 4	Amoreiras	2	25	7	5	11	27	6	11	7	8
Task 5	Fitness Hut	5	5	14	17	8	6	5	9	10	17
Task 6	Photos Pragal	2	7	3	3	27	4	3	9	6	4
Task 7	Onde Estive	5	8	18	12	32	27	13	15	14	16
Task 8	Bkg	3	9	7	7	8	8	12	11	5	6
Task 9	Report	3	11	14	10	6	6	7	4	5	5
Task 10	Transportation method	2	20	8	11	17	9	18	14	6	7
Task 11	Remove vis	2	11	5	12	4	10	4	6	5	3
Task 12	Delete File	3	7	5	7	18	6	4	5	5	6

User 10	User 11	User 12	User 13	User 14	User 15	User 16	User 17	User 18	User 19	User 20	User 21
12	8	9	12	11	6	8	11	6	9	12	12
17	15	16	17	20	14	12	21	17	18	20	18
13	8	13	9	17	10	14	9	12	7	15	13
9	14	13	8	14	9	9	7	11	13	9	15
11	8	11	7	5	14	13	6	14	9	11	12
5	6	4	8	6	7	9	9	10	7	6	11
16	19	21	15	14	21	17	18	14	19	17	16
10	6	8	7	9	11	6	9	7	6	10	8
11	5	8	6	7	9	8	7	5	10	6	9
13	7	11	8	15	9	10	14	17	18	8	17
6	3	8	5	10	4	9	9	8	4	6	6
8	5	9	7	6	10	7	5	4	6	8	9

Figure 81: *Tasks Clicks Results (Browser)*

Appendix E – Time Results (Browser)

	Task Description		User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
Task 1	Add Files	0:10	0:20	0:57	0:20	2:49	1:45	0:33	0:51	0:18	0:21
Task 2	Despesas Set 2017	0:06	3:43	1:51	1:43	2:45	1:32	2:04	2:30	0:56	1:02
Task 3	IST	0:08	6:14	1:35	1:38	1:56	0:11	0:47	1:16	2:15	0:25
Task 4	Amoreiras	0:13	3:24	0:37	0:21	1:05	3:12	0:32	0:39	0:24	0:37
Task 5	Fitness Hut	0:13	1:13	0:58	0:43	0:51	0:31	0:18	1:48	0:48	1:40
Task 6	Photos Pragal	0:10	0:40	0:15	0:16	3:30	0:15	0:11	1:20	0:30	0:23
Task 7	Onde Estive	0:18	1:50	1:21	1:27	4:30	3:42	1:50	0:40	0:50	0:39
Task 8	Bkg	0:06	0:56	0:23	0:21	0:34	0:30	0:56	0:52	0:15	0:23
Task 9	Report	0:06	1:36	0:57	1:22	0:51	0:30	0:24	0:20	0:26	0:52
Task 10	Transportation method	0:05	1:53	1:02	1:19	2:05	0:57	3:25	1:08	0:39	1:10
Task 11	Remove vis	0:05	0:35	0:08	0:24	0:16	0:07	0:21	0:10	0:09	0:05
Task 12	Delete File	0:07	1:34	0:16	0:20	2:01	0:38	0:28	0:31	0:18	0:18

User 10	User 11	User 12	User 13	User 14	User 15	User 16	User 17	User 18	User 19	User 20	User 21
0:51	0:49	0:57	0:34	1:29	1:45	0:39	0:53	0:32	0:41	0:45	1:04
1:33	2:47	1:51	1:53	2:35	1:54	2:04	2:30	1:41	1:22	2:04	1:54
0:36	2:24	1:13	2:08	1:49	2:35	0:57	1:36	0:18	2:15	2:42	2:27
0:47	1:49	0:57	0:50	0:47	1:52	1:33	0:59	1:37	1:38	0:57	0:43
1:23	0:59	0:38	0:37	0:27	0:13	0:33	0:52	1:32	0:38	0:56	1:02
0:21	0:32	0:23	0:41	0:16	0:12	1:13	0:35	0:26	0:22	0:17	0:16
1:34	0:52	1:56	2:34	1:40	1:38	1:42	0:41	0:41	1:42	1:36	1:12
1:26	0:18	0:41	1:34	0:33	0:57	0:23	0:17	0:31	0:25	0:15	0:36
1:33	0:45	1:03	0:56	0:42	0:37	1:02	0:36	0:52	1:00	0:38	0:59
1:12	1:15	1:09	1:34	0:37	2:05	0:53	0:29	1:15	0:51	0:31	1:12
0:36	0:16	0:12	0:24	0:13	0:41	0:26	0:12	0:19	0:24	0:15	0:22
0:14	0:12	0:16	0:53	0:41	0:18	0:46	0:12	0:21	0:12	0:28	0:18

Figure 82: *Time Results (Browser)*

Appendix F – Tasks Taps Results (Mobile)

	Task Description		User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
Task 1	Current Location	1	12	2	4	8	2	2	8	4	2
Task 2	Last Time Here	2	2	7	22	19	7	7	2	2	5
Task 3	Photos	2	8	3	6	2	7	4	5	3	3
Task 4	Restaurant	2	6	5	6	4	5	5	7	3	3
Task 5	Max Distance	3	12	3	17	4	4	3	3	2	7
Task 6	Update Time	3	5	3	3	4	3	3	4	3	3

User 10	User 11	User 12	User 13	User 14	User 15	User 16	User 17	User 18	User 19	User 20	User 21
4	3	5	6	3	4	3	4	2	7	3	4
5	3	4	3	6	5	12	9	6	6	15	9
4	5	4	2	4	5	3	6	3	4	4	7
6	4	7	5	8	6	5	7	8	4	6	5
4	7	4	5	3	4	3	5	7	5	3	4
4	3	3	3	5	6	3	4	5	3	4	5

Figure 83: *Tasks Taps Results (Mobile)*

Appendix G – Time Results (Mobile)

	Task Description		User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
Task 1	Current Location	0:01	0:45	0:16	0:20	1:03	0:04	0:23	1:04	0:23	0:02
Task 2	Last Time Here	0:04	0:10	0:31	1:37	2:07	1:10	0:49	0:04	0:05	0:09
Task 3	Photos	0:04	1:45	0:11	0:12	0:07	0:33	0:21	0:20	0:15	0:04
Task 4	Restaurant	0:06	1:30	0:18	0:17	0:16	0:23	0:39	0:40	0:58	0:06
Task 5	Max Distance	0:07	1:50	0:07	0:23	0:18	0:24	0:13	0:18	0:07	0:48
Task 6	Update Time	0:07	0:30	0:07	0:07	0:12	0:13	0:18	0:15	0:10	0:07

User 10	User 11	User 12	User 13	User 14	User 15	User 16	User 17	User 18	User 19	User 20	User 21
0:38	0:23	0:32	1:14	0:09	0:19	1:05	0:28	0:07	0:12	0:20	0:12
0:14	0:31	0:58	1:10	1:14	1:03	0:24	0:13	0:08	0:13	0:10	0:29
1:26	0:11	0:22	0:19	0:28	0:34	0:43	0:18	0:07	0:33	0:13	0:14
1:18	0:34	0:19	0:27	0:45	0:28	0:23	0:38	0:11	0:44	0:22	0:09
1:53	0:09	0:32	0:33	0:54	0:15	0:43	0:12	0:40	0:19	0:32	0:37
0:45	0:12	0:08	0:24	0:27	0:15	0:22	0:10	0:10	0:14	0:17	0:16

Figure 84: *Time Results (Mobile)*